UNIVERSIDAD MIGUEL HERNÁNDEZ
Programa de Doctorado en Tecnologías Industriales y de Telecomunicación

Kinematic Analysis and Design of the HyReCRo Robot: a Serial-Parallel and Redundant Structure-Climbing Robot

Autor: Adrián Peidró Vidal
Director: Dr. Ing. Óscar Reinoso García
Codirector: Dr. Ing. Arturo Gil Aparicio

Tesis Doctoral presentada en la Universidad Miguel Hernández para la obtención del título de Doctor del Programa de Doctorado en Tecnologías Industriales y de Telecomunicación

2018

La presente Tesis Doctoral está sustentada por un compendio de trabajos previamente publicados en revistas de impacto, indexadas según JCR Science Edition. El cuerpo de dicha tesis queda constituido por los siguientes artículos, cuyas referencias bibliográficas completas se indican a continuación:

- *An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots.* [145]
  A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
  Eng. Appl. of Artificial Intelligence. Vol 64, pp. 197-207 (2017)
  ISSN: 0952-1976. Ed. Elsevier.
  JCR-SCI Impact Factor: 2.819, Quartile **Q1**
  Web: `https://doi.org/10.1016/j.engappai.2017.06.009`
  DOI: 10.1016/j.engappai.2017.06.009

- *A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots.* [146]
  A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
  Mechanism and Machine Theory. Vol 128, pp. 84-109 (2018)
  ISSN: 0094-114X. Ed. Elsevier.
  JCR-SCI Impact Factor: 2.796, Quartile **Q1**
  Web: `https://doi.org/10.1016/j.mechmachtheory.2018.05.013`
  DOI: 10.1016/j.mechmachtheory.2018.05.013

UNIVERSITAS
Miguel Hernández

AUTORIZACIÓN DE PRESENTACIÓN DE TESIS DOCTORAL POR UN CONJUNTO DE PUBLICACIONES

Director: Dr. Ing. Óscar Reinoso García
Codirector: Dr. Ing. Arturo Gil Aparicio

Título de la tesis: ***Kinematic Analysis and Design of the HyReCRo Robot: a Serial-Parallel and Redundant Structure-Climbing Robot***

Autor: Adrián Peidró Vidal

Departamento de Ingeniería de Sistemas y Automática
Universidad Miguel Hernández de Elche

El director y codirector de la tesis reseñada AUTORIZAN SU PRESENTACIÓN EN LA MODALIDAD DE CONJUNTO DE PUBLICACIONES.

En Elche, a de de 2018.

Fdo: Dr. D. Óscar Reinoso García

Fdo: Dr. D. Arturo Gil Aparicio

UNIVERSITAS
Miguel Hernández

PROGRAMA DE DOCTORADO EN TECNOLOGÍAS INDUSTRIALES
Y DE TELECOMUNICACIÓN

Dr. D. Óscar Reinoso García, Coordinador del Programa de Doctorado en Tecnologías Industriales y de Telecomunicación de la Universidad Miguel Hernández de Elche.

Certifica

Que el trabajo realizado por D. Adrián Peidró Vidal titulado ***Kinematic Analysis and Design of the HyReCRo Robot: a Serial-Parallel and Redundant Structure-Climbing Robot*** ha sido dirigido por el Dr. D. Óscar Reinoso García y codirigido por el Dr. D. Arturo Gil Aparicio y se encuentra en condiciones de ser leído y defendido como Tesis Doctoral ante el correspondiente tribunal en la Universidad Miguel Hernández de Elche.

Lo que firmo para los efectos oportunos en Elche, a de de 2018.

Fdo.: Dr. D. Óscar Reinoso García
Coordinador del Programa de Doctorado en Tecnologías Industriales y de Telecomunicación

DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y AUTOMÁTICA
Universidad Miguel Hernández de Elche. Ed. Innova. Avda. de la Universidad s/n, 03202 Elche
Telf: 96 665 8616

Abstract

Three-dimensional vertical structures such as bridges, skeletons of buildings in the construction industry, or electrical and telecommunication towers, require inspection and maintenance tasks. These tasks imply important risks for the human operators that usually perform them, such as falling from heights. A solution to this problem consists in using climbing robots for performing such dangerous tasks.

This thesis presents the kinematic analysis and design of the HyReCRo robot (Hybrid Redundant Climbing Robot), a robot designed for climbing three-dimensional metallic structures. This robot is redundant and has a hybrid architecture, since it is composed of four 2R$\underline{\text{P}}$R-PR parallel mechanisms serially connected. The main characteristic of this robot is that it can be driven by binary actuators in order to get closer to the basic postures necessary for exploring three-dimensional structures (namely: convex and concave plane transitions), finely adjusting later the pose of its grippers by means of continuous actuation, in order to adhere its grippers to the climbed structure. This mixed binary-continuous strategy facilitates the control and movement planning of the robot, which usually are very complex tasks in structure-climbing robots. The present thesis is focused on the study of the HyReCRo robot as a continuously-actuated robot, with the purpose of using mixed actuation strategies in the future.

In the first place, this thesis presents a complete kinematic analysis of the HyReCRo robot and of the parallel mechanisms that make up its legs. Regarding the kinematic analysis of these parallel mechanisms, this thesis demonstrates that these mechanisms can enlarge their workspace by switching between different assembly modes without traversing singularities. This occurs when enclosing special isolated singularities which are fourfold solutions of the forward kinematic problem of these parallel mechanisms. Regarding the kinematic analysis of the complete HyReCRo robot, this thesis solves both its forward and inverse kinematic problems, obtaining simple parameterizations of the 4- and 5-dimensional self-motion manifolds of this robot. These kinematic analyses are performed with the help of PaRoLa, a collection of graphical simulators developed specifically in the present thesis with the purpose of facilitating the kinematic analysis of parallel robots.

The present thesis also presents the study of the workspace of the HyReCRo robot, with the purpose of designing this robot so that it can perform convex and concave plane transitions required for exploring structures. As a result of this workspace analysis, this thesis proposes two new methods for obtaining the workspace of redundant robots, like the HyReCRo robot.

The first proposed method is an improved Monte Carlo method, which is able to obtain the boundaries of the workspace much more accurately than previously existing Monte Carlo methods, requiring the same or less computation time than them. This method consists in uniformly growing the workspace by means of normal random distributions, until the boundaries of the workspace are attained.

The second method proposed in this thesis is a method for obtaining the boundaries and interior barriers of the workspace of redundant robots induced by general collision constraints, which are difficult to obtain using previously existing methods due to their difficulties to handle such collision constraints. The proposed method identifies the vanishing of self-motion manifolds of the robot with the occurrence of interior barriers, and it consists of three stages. First, these self-motion manifolds are densely sampled, discarding samples that do not satisfy collision constraints. Next, non-discarded samples are clustered using kd-trees, with the purpose of identifying disjoint components of these manifolds. Finally, the disjoint manifolds identified at pairs of neighboring points of the workspace are compared, in order to determine if any of these manifolds vanishes when traveling from one point to its neighbor. The application of this method demonstrates that collision constraints drastically alter the distribution of interior barriers within the workspace.

Finally, this thesis concludes with the development of a totally functional prototype of the HyReCRo robot. This prototype, which weighs 2.19 kg, uses magnetic grippers based on the technology of switchable permanent magnets, achieving an adhesion force of 33 kg per gripper. This prototype has been successfully validated on real steel structures.

Resumen

Las estructuras verticales tridimensionales, como los puentes, los esqueletos de edificios en la industria de la construcción, o las torres eléctricas y de telecomunicación, requieren tareas de inspección y mantenimiento. Dichas tareas suponen riesgos importantes para los operarios humanos que las realizan habitualmente, como la caída desde altura. Una solución a este problema pasa por utilizar robots trepadores para desarrollar tales peligrosas tareas.

Esta tesis presenta el análisis cinemático y diseño del robot HyReCRo (*Hybrid Redundant Climbing Robot*), un robot diseñado para trepar por estructuras metálicas tridimensionales. Se trata de un robot redundante y de arquitectura híbrida, formado por la conexión en serie de cuatro mecanismos paralelos de tipo 2RPR-PR. La principal característica de este robot es que puede ser accionado de forma binaria para aproximarse a las posturas básicas necesarias para explorar estructuras tridimensionales (a saber: transiciones convexas y cóncavas entre distintos planos de trabajo), ajustando posteriormente de forma fina la pose de sus garras mediante actuación continua para fijar sus garras a la estructura. Esta estrategia mixta binaria-continua facilita el control y la planificación de movimientos del robot, tareas que suelen ser muy complejas en robots trepadores de estructuras. Esta tesis se focaliza en el estudio del robot HyReCRo como robot accionado de forma continua, con el fin de poder operarlo mediante estrategias mixtas en el futuro.

En primer lugar, esta tesis presenta un análisis cinemático completo del robot HyReCRo y de los mecanismos paralelos que forman sus patas. En cuanto al análisis cinemático de dichos mecanismos paralelos, en esta tesis se demuestra que éstos pueden ampliar su espacio de trabajo alternando entre distintos modos de ensamblado sin cruzar singularidades. Esto ocurre al rodear singularidades aisladas especiales que son soluciones cuádruples del problema cinemático directo de dichos mecanismos paralelos. En cuanto al análisis cinemático del robot HyReCRo completo, se resuelven en esta tesis sus problemas cinemáticos directo e inverso, obteniéndose parametrizaciones sencillas de las variedades de auto-movimiento 4- y 5-dimensionales de este robot. Estos análisis cinemáticos se realizan con la ayuda de PaRoLa, una colección de simuladores gráficos desarrollados específicamente en la presente tesis para facilitar el análisis cinemático de robots paralelos.

La presente tesis también presenta el estudio del espacio de trabajo del robot HyReCRo, con el fin de diseñar este robot para que pueda realizar las transiciones convexas y cóncavas requeridas para explorar estructuras. Como resultado de este análisis del espacio de trabajo, se proponen en esta tesis dos nuevos métodos para determinar el espacio de trabajo de robots redundantes, como el robot HyReCRo.

El primer método propuesto se trata de un método Monte Carlo mejorado, capaz de obtener las fronteras del espacio de trabajo con mucha más precisión que los métodos Monte Carlo preexistentes, requiriendo el mismo o menos tiempo de cálculo que dichos métodos. Dicho método se basa en hacer crecer de forma uniforme el espacio de trabajo mediante distribuciones aleatorias normales, hasta alcanzar las fronteras del mismo.

El segundo método propuesto en esta tesis es un método capaz de obtener las fronteras y barreras interiores del espacio de trabajo de robots redundantes debidas a restricciones de colisión generales, que son difíciles de obtener mediante los métodos preexistentes, los cuales presentan dificultades notables para manejar tales restricciones de colisión. El método propuesto identifica el desvanecimiento de variedades de auto-movimiento del robot con la ocurrencia de barreras interiores, y consta de tres etapas. Primero, se muestrean densamente dichas variedades de auto-movimiento, descartando las muestras que no cumplen las restricciones de colisión. Seguidamente, se agrupan las muestras no descartadas mediante kd-trees, con el fin de identificar componentes disjuntas de dichas variedades. Por último, se comparan las variedades disjuntas identificadas en pares de puntos vecinos del espacio de trabajo, para determinar si alguna de dichas variedades se desvanece al viajar de un punto a su vecino. La aplicación de este método demuestra que las restricciones de colisión alteran drásticamente la distribución de barreras interiores dentro del espacio de trabajo.

Finalmente, esta tesis concluye con el desarrollo de un prototipo totalmente funcional del robot HyReCRo. Dicho prototipo, que pesa 2.19 kg, utiliza garras magnéticas basadas en la tecnología de imanes permanentes alternables (*switchable magnets*), logrando una fuerza de adhesión de 33 kg por garra. Este prototipo ha sido validado con éxito en estructuras de acero reales.

Acknowledgments

First of all, I wish to thank my supervisors Óscar Reinoso and Arturo Gil for their valuable guidance and support, their trust placed on me, and, especially, their patience. I am also very grateful to José María Marín, whose devotion, kindness, and know-how have helped me during many stages of the present thesis.

I also wish to thank Luis Miguel Jiménez, Luis Payá, and all other members and colleagues of the Automation, Robotics and Computer Vision lab of the Miguel Hernández University for welcoming me into their research group many years ago.

I am also very grateful to Mahmoud Tavakoli from the University of Coimbra for welcoming me to work with them during three months in the fall of 2016. I also very much appreciate the kindness and help received from my friend Carlos Viegas, also from the University of Coimbra, during this three-month stay.

Last but not least, I wish to thank my family, my girlfriend Mariasa, and my friends for their support and encouragement during these years.

This thesis has been supported by the Spanish Ministry of Education, Culture, and Sport through two grants with reference numbers: FPU13/00413 and EST15/00483.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

This thesis belongs to the field of robots designed for climbing three-dimensional human-made structures, like those present in industrial pipelines [178], in metallic bridges, and in the skeletons of buildings in the construction industry, among many other [14]. These structures require inspection and maintenance tasks to guarantee their safety and correct performance, tasks which are typically performed by teams of human operators. However, these tasks are very dangerous for these operators due to several risks, being the most evident and important risk the fall from a considerable height.

With the purpose of avoiding risking the lives of human operators, several researchers from all over the world have been investigating for more than thirty years the possibility of employing climbing robots to perform these dangerous tasks. Dozens of different climbing robots have been proposed during this period, yet despite this effort, the vast majority of these dangerous industrial inspection tasks are still performed by human workforce in practice. While robotic manipulators have become very popular and their use in modern factories and production lines has become completely generalized and normalized, this is not certainly the case of climbing robots for industrial inspection tasks, which rarely go beyond research laboratories.

Structure-climbing robots typically have an inchworm-like design, for they are composed of a multi-degrees-of-freedom manipulator whose ends carry grippers with which the robot adheres to the structure and climbs it. This design effectively turns these robots into mobile manipulators, in which the base of the robot is no longer fixed

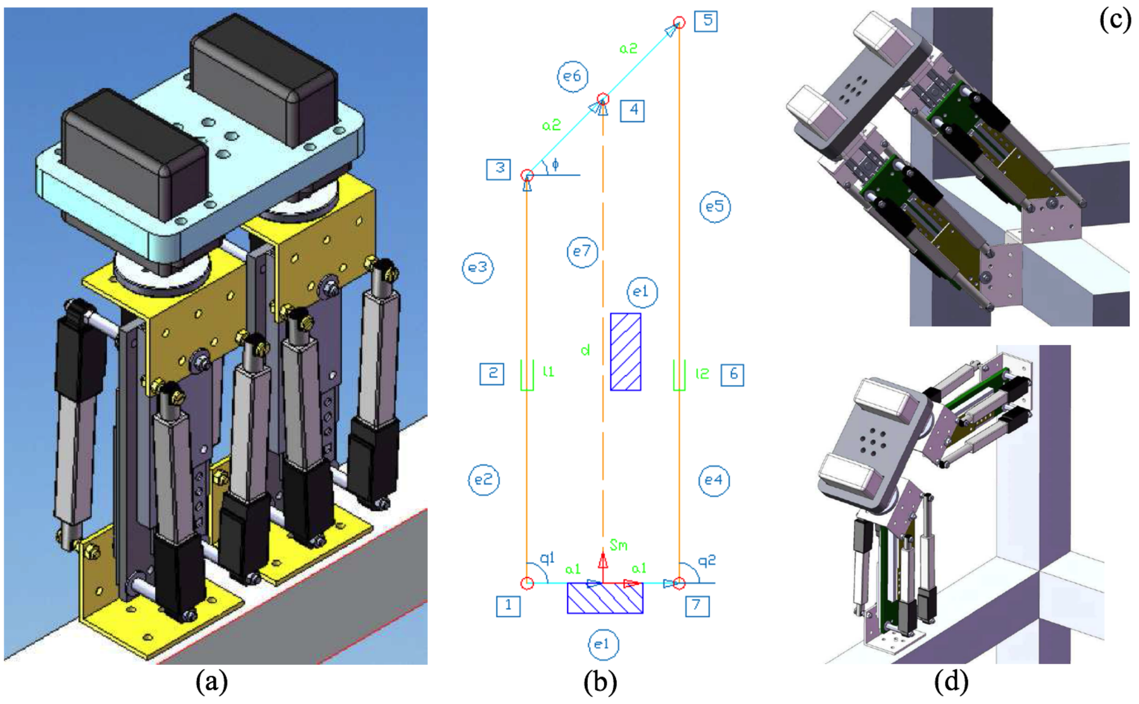


Figure 1.1: The binary HyReCRo robot as originally proposed in [183].

but changes as the robot explores the structure. This manipulator-like architecture grants these robots a high maneuverability that is necessary for exploring complicated three-dimensional structures, which are typically crowded by obstacles [178]. However, their manipulator-like architecture also complicates the development and use of these robots in practice, since they are slow and it is much more difficult to control them and plan their movements when compared to other much simpler wheeled climbing robots [177]. Indeed, the high complexity of inchworm-like structure-climbing robots seems to be one of the major reasons that prevent the normalized and generalized use of these robots in industrial inspection [177].

With the purpose of alleviating this complexity problem and simplifying the architecture and operation of structure-climbing robots, Úbeda et al. [183] proposed the biped robot shown in Figure 1.1a. The legs of this biped robot have a **hybrid** serial-parallel architecture, since each such leg is composed of a serial combination of two identical 2R$\underline{\text{P}}$R-PR parallel mechanisms (this parallel mechanism is shown in Figure 1.1b). Moreover, this robot has ten degrees of freedom (DOF), which makes it kinematically **redundant**. Therefore, in the present thesis, this robot will be named "HyReCRo", which stands for Hybrid Redundant Climbing Robot.

Although the hybrid and redundant architecture of the HyReCRo robot initially seems quite complex, the original advantage and novelty of this proposal was that, by using only binary actuators, this robot could attain the basic postures necessary for exploring three-dimensional structures, namely: convex plane transitions to change between different faces of a beam (Figure 1.1c) and concave plane transitions to change between different beams (Figure 1.1d). A structure-climbing robot with purely binary actuation is very attractive from the points of view of control and trajectory planning, since both these problems are greatly simplified when considering binary actuation. In principle, this may contribute to simplify the operation of structure-climbing robots and encourage their use in industrial inspection tasks.

Nevertheless, it is argued here that a purely binary actuation scheme is not completely feasible for structure-climbing robots. This is because, in order for these robots to effectively climb and explore structures, they must place their grippers on the surface of the structure or sufficiently close to it (so that they can be adhered to the structure). However, this may not be possible in general if all actuators are binary, since binary actuators provide only a finite and discrete number of different postures, and none of these postures may be able to place the grippers at the necessary position for adhering to the structure. Therefore, it is necessary to continuously actuate at least some (or all) of the actuators of the HyReCRo robot in order for this robot to be able to finely place its grippers on the climbed structure and effectively perform plane transitions like those shown in Figure 1.1c-d.

In fact, a mixed continuous-binary actuation scheme may be the most promising solution. According to this mixed actuation scheme, purely binary actuators may be initially used for coarsely placing the robot near the posture necessary for performing a desired plane transition. After this, some of the actuators of the robot would be continuously actuated in order to correct the discrete posture of the robot (obtained using purely binary actuation) and finely place its grippers at the necessary position for performing the desired plane transition. However, this requires performing first a kinematic analysis of the HyReCRo robot as a robot with continuous actuators, i.e., a robot whose actuated joint coordinates can take any value between two joint limits. This continuous kinematic analysis of the HyReCRo robot is the main purpose of the present thesis.

Therefore, in this thesis, the binary actuation condition of the HyReCRo robot is relaxed, and a comprehensive continuous kinematic analysis of this climbing robot is presented. This analysis is a prerequisite for exploring mixed continuous-binary actuation schemes in the future, like the proposal sketched in the previous paragraph. The kinematic analysis presented in this thesis covers the forward and inverse kinematic problems of the HyReCRo robot, as well as studies of its workspace. On the basis of these kinematic analyses, a prototype of the HyReCRo robot using magnetic grippers to adhere and climb real steel structures is also presented in this work.

1.1.1 Topics Covered by this Thesis

The next list summarizes the topics covered by the present thesis. All these topics are explored in this thesis with a shared main objective: developing a functional prototype of the HyReCRo robot for climbing real steel structures.

- **Parallel kinematics.** The HyReCRo robot has hybrid architecture, since it is made of the serial combination of several 2R$\underline{P}$R-PR parallel mechanisms of the type shown in Figure 1.1b. In order to solve the forward and inverse kinematic problems of the complete HyReCRo robot, first it is necessary to solve these problems for the 2R$\underline{\text{P}}$R-PR parallel mechanisms that make up the legs of this robot. As explained in Chapter 3 of the present thesis, the forward and inverse kinematic problems of these parallel mechanisms were already solved by other researchers in the past. However, the singularities of these mechanisms were not investigated yet, so it becomes necessary to study them in the present thesis, in order to avoid them in the design of the HyReCRo robot. When studying the singularities of 2R$\underline{\text{P}}$R-PR parallel mechanisms in the present thesis, it is found that these mechanisms exhibit special singularities which are fourfold solutions of the forward kinematic problem of these mechanisms, such that encircling these special singularities allows the mechanism to switch between different solutions of the forward kinematics without crossing singularities (*nonsingular transitions*).

- **Simulation of parallel robots.** The kinematic analysis of the HyReCRo robot and of its parallel mechanisms requires the development of new graphical simulation tools that allow us to visualize the different kinematic solutions of these robot, as well as their singularities. The lack of existing tools that can satisfy the specific functionalities required by the present thesis (namely: visualizing all the solutions of the forward kinematics, and visualizing the continuous deformation of singularity loci as the design of the robot is altered) has led to the development of new simulation tools presented in Chapter 4 of the present thesis.

- **Kinematic analysis of the HyReCRo robot.** In this thesis, the forward and inverse kinematic problems of the HyReCRo robot are solved, considering that all its actuators are continuously actuated, instead of being binary. These problems, which require solving first the kinematics of the 2R$\underline{\text{P}}$R-PR parallel mechanisms, have not been solved yet for the HyReCRo robot. Worthy of special mention is the inverse kinematic problem of the HyReCRo robot, which is complicated due to its serial-parallel architecture and its kinematic redundancy. Since the HyReCRo robot has 10 degrees of freedom, and since 6 degrees of freedom are sufficient for arbitrarily positioning and orienting a free body in three-dimensional space, the solution sets (self-motion manifolds) of the inverse kinematic problem of this robot generically are (10-6=)4-dimensional. This high dimension complicates the resolution of the inverse kinematics problem, since 4-dimensional sets cannot be represented graphically. However, Chapter 5 of the present thesis proposes simple parameterizations and a compact representation of the solution sets of the inverse kinematics of the HyReCRo robot, which consists in projecting these 4-dimensional solution sets on some 2- and 3-dimensional subspaces, without losing relevant information due to this projection.

- **Workspace analysis.** The workspace of any robotic manipulator is the set of positions and orientations that its end-effector can reach. The workspace is very important for planning the movements of the robot, as well as for optimally designing it. In the particular case of the HyReCRo robot, it is necessary to study its workspace in order to ascertain whether a given design of this robot will be able to perform the typical plane transitions required for exploring three-dimensional structures, which are convex transitions between different faces of a beam (Figure 1.1c) and concave transitions between different beams (Figure 1.1d). Roughly speaking, the workspace typically is a volume containing all the poses reachable by the end-effector of the robot. When analyzing workspaces, we can focus on two levels:

– **External level.** This means that we focus on the outermost boundaries of the workspace, which define its shape. This analysis gives an idea of the volume of the workspace, i.e., of the amplitude of movements of the robot. Most research works on the workspace of robotic manipulators are limited only to the external level of the workspace.

– **Internal level.** Inside the workspaces of robot manipulators, normally there exist interior barriers due to singularities, joint limits, or collision constraints. These are kinematic barriers which block the motion of the robot. They are very important for planning the trajectories of a robot since trajectories crossing these barriers may be unfeasible in practice, depending on the posture of the robot when approaching them.

The external level of the workspace of the HyReCRo robot (i.e., its boundaries) is studied in Chapter 6 of the present thesis. Due to the complexity of the architecture of the HyReCRo robot, Monte Carlo methods turn out to be the most suitable ones for obtaining the boundaries of this robot. However, these methods usually yield imprecise results, in which the boundaries of the workspace are not clearly defined. In order to avoid this, Chapter 6 proposes a new improved Monte Carlo method for obtaining the boundaries of the workspace of robotic manipulators with higher precision than previous Monte Carlo methods, without increasing the computation time.

The internal level of the workspace (i.e., interior barriers) is addressed in Chapter 7. After analyzing the state of the art of existing methods for computing interior barriers, it is concluded in this thesis that these methods find difficulties for obtaining these barriers under collision constraints (i.e., interior barriers generated due to the condition that different bodies cannot mechanically interfere). In the HyReCRo robot, collision constraints are very important, since the robot must not collide with itself or with the climbed structure. Chapter 7 presents a new method for obtaining interior barriers of the workspace of redundant robots under collision constraints. Unlike previously existing methods, the method proposed in this thesis can easily accommodate arbitrarily complex collision constraints (i.e., collisions between arbitrarily-shaped bodies with arbitrary relative pose). To that end, the method developed in Chapter 7 identifies the interior barriers of the workspace with the vanishing of disjoint components of the self-motion manifolds.

- **Development of a prototype with magnetic grippers.** To conclude this thesis, a completely functional prototype of the HyReCRo robot is built. The design of this prototype is based on the kinematic analyses and simulations developed in this thesis, which allow the designer to determine the dimensions that the

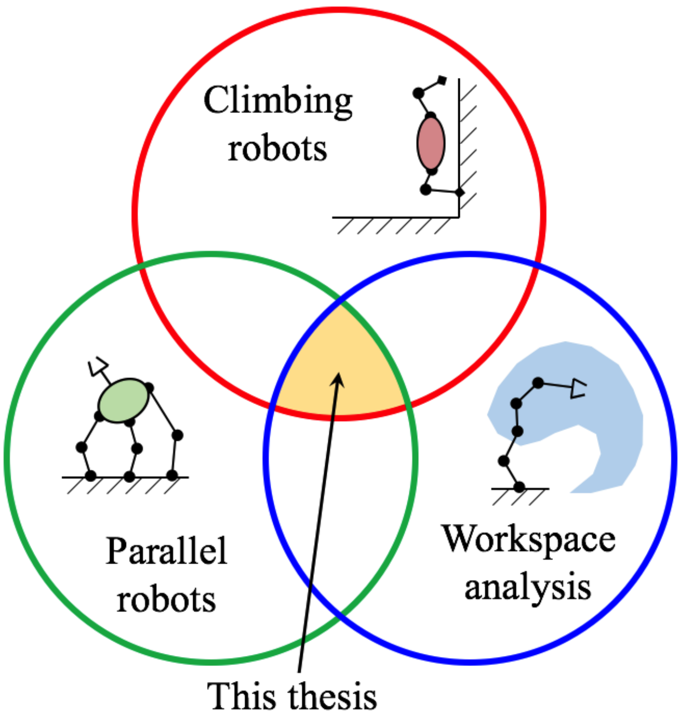


Figure 1.2: Venn diagram illustrating the main topics covered by the present thesis.

HyReCRo robot should have in order for this robot to be able to perform convex and concave plane transitions. Chaper 8 presents the detailed design of magnetic grippers for this prototype, so that it can adhere to real steel structures and climb them. The developed magnetic grippers are based on the technology of *switchable magnets*, which are safer and more energy-efficient than traditional electromagnets. Moreover, these grippers are very compact, which makes them especially suitable for adhering to the narrow beams that make up three-dimensional steel structures.

Figure 1.2 presents a Venn diagram illustrating all these topics studied in the present thesis.

1.2 Objectives

The main objective of the present thesis is the continuous kinematic analysis of the HyReCRo climbing robot, as well as the development of a functional prototype of this robot to climb real steel structures. To that end, several goals were established:

- **Analyzing the kinematics of 2R$\underline{\text{P}}$R-PR parallel mechanisms.** These parallel mechanisms compose the legs of the HyReCRo robot. Therefore, analyzing their kinematics is a necessary step for solving the kinematics of the complete HyReCRo robot. Chapter 3 presents a comprehensive analysis on the kinematics and singularities of these parallel mechanisms.

- **Developing graphical simulation tools for studying parallel robots.** In order to perform the kinematic analyses required by the present thesis, it is necessary to develop new graphical simulation tools to visualize the solutions of the forward kinematics of parallel robots and their singularities. The developed simulation tools are presented in Chapter 4.

- **Solving the forward and inverse kinematic problems of the complete HyReCRo robot.** These two problems are solved in Chapter 5, departing from the kinematic analysis of the 2R$\underline{P}$R-PR parallel mechanisms.

- **Computing the workspace of the HyReCRo robot.** In order to aid in the optimal design of the HyReCRo robot and in the planning of its trajectories, it is necessary to obtain and visualize its workspace. Chapter 6 develops a new Monte Carlo method to obtain the workspace of robot manipulators more accurately than with previous methods.

- **Obtaining interior workspace barriers under collision constraints.** In order to effectively plan the trajectories of redundant robots like the HyReCRo robot, which explores environments that include obstacles, it is necessary to obtain the interior barriers of the workspace originated by collision constraints. Chapter 7 develops a method able to easily deal with arbitrarily complex collision constraints when computing the interior barriers of the workspace of redundant manipulators.

- **Developing a functional prototype of the HyReCRo robot with magnetic grippers.** To conclude this thesis, it is necessary to develop a functional prototype of the HyReCRo robot using magnetic grippers to adhere to real steel structures and climb them. Chapter 8 presents such a prototype, whose grippers employ the technology of switchable magnets in order to firmly adhere to ferromagnetic surfaces even in the most critical climbing situations.

1.3 Framework of this Thesis

This thesis has been developed under a wider framework supported by different grants, research projects, and collaborations, as detailed next.

1.3.1 Grants and Awards

The development of thesis has been mainly supported by an FPU grant from the Spanish Ministry of Education, Culture, and Sport. This grant, whose reference number is FPU13/00413, has supported financially the author of the present thesis during four years (from September 2014 to September 2018), in order to develop this thesis during this period.

Also, another grant from the Spanish Ministry of Education was conceded to the author of this thesis in 2016 in order to do a short research stay at a foreign university, as described in the next subsection.

Finally, it is worth mentioning that a 6-pages communication summarizing this thesis [139] was presented at the 2018 Spanish Robotics Conference in Valladolid (June 14-15, 2018). This communication was awarded the *Best PhD Thesis Communication Award* of this conference.

1.3.2 Research Stays and Collaborations

From September to November 2016, the author of this thesis spent three months collaborating at the *Institute of Systems and Robotics* of the University of Coimbra (Portugal). The objective of this research stay, which was supervised by Prof. Dr. Mahmoud Tavakoli, was to investigate the kinematic calibration of the climbing robot studied in the present thesis, as well as to investigate the use of switchable magnets to develop magnetic grippers for this climbing robot (see chapter 8). This research stay was supported by the Spanish Ministry of Education, through grant with reference number EST15/00483.

1.3.3 Projects

The present thesis has been developed under the PhD project entitled "Planificación de movimientos de robots en entornos no estructurados" (Planning movements of robots in unstructured environments), which was the original name of the PhD project granted by the Spanish Ministry of Education with the purpose of analyzing and designing the HyReCRo climbing robot. This project has been supported by the FPU grant referred above.

1.4 Publications

The research work conducted in the present thesis has produced 7 journal papers ranked in JCR Science Edition, 1 book chapter, 10 international conference papers, and 8 national conference papers.

Section 1.4.1 presents the two main JCR publications achieved under this thesis, which support much of the developed research work. Section 1.4.2 summarizes the rest of the publications, which are also intimately related to this thesis.

The last section of each chapter of this thesis includes a list indicating which of these publications are related to the research work presented in the corresponding chapter.

1.4.1 Journal Publications Supporting this Thesis

The two main contributions of this thesis are supported by a couple of articles published in JCR-indexed journals belonging to the first quartile (Q1) of their respective categories. These two articles, whose metadata are provided next, are directly aligned with the motivation and objectives of the present PhD thesis.

- *An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots.* [145]
  A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
  Eng. Appl. of Artificial Intelligence. Vol 64, pp. 197-207 (2017)
  ISSN: 0952-1976. Ed. Elsevier.
  JCR-SCI Impact Factor: 2.819, Quartile **Q1**
  Web: https://doi.org/10.1016/j.engappai.2017.06.009
  DOI: 10.1016/j.engappai.2017.06.009

- *A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots.* [146]
  A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
  Mechanism and Machine Theory. Vol 128, pp. 84-109 (2018)
  ISSN: 0094-114X. Ed. Elsevier.
  JCR-SCI Impact Factor: 2.796, Quartile **Q1**
  Web: https://doi.org/10.1016/j.mechmachtheory.2018.05.013
  DOI: 10.1016/j.mechmachtheory.2018.05.013

The first article proposes a new improved Monte Carlo method for accurately computing the workspaces of robot manipulators with complex architecture, like the HyReCRo robot. This method is more efficient than previously existing Monte Carlo methods since it can attain much higher accuracy than these methods, requiring the same or less computation time than them. This method, which is developed in Chapter 6, consists in uniformly "growing" the workspace using normal distributions, until the boundaries of the workspace are reached.

The second article proposes a new sampling and clustering method for obtaining the interior barriers of the workspace of redundant manipulators, like the HyReCRo robot. The main contribution of this method is that it is able to handle arbitrarily-complex collision constraints very easily, whereas previous methods find it difficult to handle these constraints effectively. The proposed method identifies the interior barriers of the workspace of redundant manipulators with the vanishing of disjoint components of their self-motion manifolds. This method is developed in Chapter 7.

These two articles are appended in Appendix A.

1.4.2 Other Publications Related to this Thesis

In addition to the two main publications presented above, during this thesis many other articles were published in JCR-indexed journals and conferences. These additional publications are also related to much of the research work presented in the next chapters of this thesis.

1.4.2.1 JCR-indexed journals

- A. Peidró, A. Gil, J.M. Marín, L. Payá, and Ó. Reinoso. On the stability of the quadruple solutions of the forward kinematic problem in analytic parallel robots.

Journal of Intelligent & Robotic Systems, 86(3):381–396, 2017 [135] **(SCI-JCR Impact Factor: 1.583, Q3)**.

- A. Peidró, A. Gil, J.M. Marín, and Ó. Reinoso. A web-based tool to analyze the kinematics and singularities of parallel robots. *Journal of Intelligent & Robotic Systems*, 81(1):145–163, 2016 [137] **(SCI-JCR Impact Factor: 1.512, Q3)**.
- A. Peidró, J.M. Marín, A. Gil, and O. Reinoso. Performing nonsingular transitions between assembly modes in analytic parallel manipulators by enclosing quadruple solutions. *ASME Journal of Mechanical Design*, 137(12):122302, 2015 [140] **(SCI-JCR Impact Factor: 1.444, Q2)**.
- A. Peidro, A. Gil, J.M. Marin, and O. Reinoso. Inverse kinematic analysis of a redundant hybrid climbing robot. *International Journal of Advanced Robotic Systems*, 12(11):163, 2015 [136] **(SCI-JCR Impact Factor: 0.615, Q4)**.
- A. Gil, A. Peidró, Ó. Reinoso, and J.M. Marín. Implementation and assessment of a virtual laboratory of parallel robots developed for engineering students. *IEEE Transactions on Education*, 57(2):92–98, 2014 [60] **(SCI-JCR Impact Factor: 0.842, Q3)**.

1.4.2.2 Book chapters

- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematics, simulation, and analysis of the planar and symmetric postures of a serial-parallel climbing robot. In Joaquim Filipe, Kurosh Madani, Oleg Gusikhin, and Jurek Sasiadek, editors, *Informatics in Control, Automation and Robotics 12th International Conference, ICINCO 2015 Colmar, France, July 21-23, 2015 Revised Selected Papers*, pages 115–135. Springer International Publishing, 2016 [133].

1.4.2.3 International conferences

- A. Peidró, J.M. Marín, Ó. Reinoso, L. Payá, and A. Gil. Parallelisms between planar and spatial tricept-like parallel robots. In Vigen Arakelian and Philippe Wenger, editors, *ROMANSY 22 – Robot Design, Dynamics and Control*, pages 155–162. Springer International Publishing, 2019 [141].
- A. Peidró, C. Tendero, J.M. Marín, A. Gil, L. Payá, and Ó. Reinoso. m-PaRoLa: a mobile virtual laboratory for studying the kinematics of five-bar and 3RRR planar parallel robots. *IFAC-PapersOnLine*, 51(4):178 – 183, 2018. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID 2018) [153].
- A. Peidró, A. Belando, D. Valiente, O. Reinoso, and L. Payá. A multi-perspective simulator for visualizing and analyzing the kinematics and singularities of 2UPS/U parallel mechanisms. In *INTED2018 Proceedings*, 12th International Technology, Education and Development Conference, pages 3785–3793. IATED, 2018 [130].

- A. Peidró, Ó. Reinoso, J.M. Marín, A. Gil, L. Payá, and Y. Berenguer. A simulation tool for visualizing the assembly modes and singularity locus of 3RPR planar parallel robots. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau, and Carlos Cardeira, editors, *ROBOT 2017: Third Iberian Robotics Conference: Volume 1*, pages 516–528. Springer International Publishing, 2018 [150].

- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Second-order Taylor stability analysis of isolated kinematic singularities of closed-chain mechanisms. In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics - Volume 2*, pages 351–358. INSTICC, SciTePress, 2017 [148].

- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Calculation of the boundaries and barriers of the workspace of a redundant serial-parallel robot using the inverse kinematics. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 412–420, 2016 [147].

- A. Peidró, Ó. Reinoso, A. Gil J.M. Marín, and L. Payá. A simulation tool to study the kinematics and control of 2RPR-PR parallel robots. *IFAC-PapersOnLine*, 49(6):268–273, 2016. 11th IFAC Symposium on Advances in Control Education (ACE 2016) [149].

- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, L. Payá, and O. Reinoso. Montecarlo workspace calculation of a serial-parallel biped robot. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 157–169, 2016. Springer International Publishing [131].

- A. Peidró, O. Reinoso, A. Gil, J.M. Marín, and L. Payá. A virtual laboratory to simulate the control of parallel robots. *IFAC-PapersOnLine*, 48(29):19 – 24, 2015 [143].

- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematic analysis and simulation of a hybrid biped climbing robot. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 24–34, 2015 [132].

1.4.2.4 National conferences

- A. Peidró, J.M. Marín, A. Gil, Y. Berenguer, and Ó. Reinoso. Analysis and design of a serial-parallel redundant biped climbing robot. In *Libro de Actas de las Jornadas Nacionales de Robótica 2018*, 2018 [139].

- A. Peidró, L. Payá, V. Román, J.M. Marín, A. Gil, and O. Reinoso. Laboratorio virtual móvil de robots paralelos. *Aceptado, a ser publicado en las Actas de las XXXIX Jornadas de Automática*. CEA, 2018 [142].

- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. Análisis de estabilidad de singularidades aisladas en robots paralelos mediante desarrollos de Taylor de segundo orden. In *Actas de las XXXVIII Jornadas de Automática*, pages 821–828. CEA, 2017 [144].
- A. Peidró, A. Hortal, A. Gil, J.M. Marín, D. Úbeda, and Ó. Reinoso. Modelado dinámico y simulación de un robot trepador tipo serie con 4 grados de libertad. In *Actas de las XXXVII Jornadas de Automática*, pages 1067–1074, 2016 [138].
- A. Peidró, O. Reinoso, L. Payá, Y. Berenguer, A. Gil, and J.M. Marín. Análisis cinemático y simulación de un robot trepador con arquitectura serie-paralela. In *Actas de las XXXVI Jornadas de Automática*, pages 400–407. CEA, 2015 [151].
- A. Peidró, A. Gil, J.M. Marín, L. Payá, and O. Reinoso. Control de un ascensor como caso práctico para la docencia de control avanzado. In *Actas de las XXXV Jornadas de Automática*, page Capítulo 2. CEA, 2014 [134].
- A. Peidró, J.J. Rodríguez, J.M. Azorín, and O. Reinoso. Implementación de una maqueta de control bilateral de 1 GDL con Arduino para telerrobótica. In *Actas de las XXXV Jornadas de Automática*, page Capítulo 68. CEA, 2014 [152].
- A. Gil, A. Peidró, J.M. Marín, O. Reinoso, D. Valiente, L.M. Jiménez, and M. Juliá. Laboratorio virtual y remoto de robots paralelos. In *Actas de las XXXIV Jornadas de Automática*, pages 235–241. CEA, 2013 [59].

1.4.2.5 Pending publications

In addition to all the journal and conference papers listed above, which are already published, there are some unpublished results of the present thesis that will be submitted for publication in other journals in the near future. These pending publications are:

- A paper on the state of the art of climbing robots, related to Chapter 2.
- A paper on an updated and improved version of the virtual laboratory PaRoLa presented in Chapter 4, with additional robots and functionalities.
- A paper on the design of the magnetic grippers presented in Chapter 8.

1.5 Structure of this Thesis

This document has been organized as follows:

- Chapter 2 presents an up-to-date literature review of the wide and rich field of climbing robots, with the purpose of better contextualizing the HyReCRo climbing robot studied in the present thesis. In this chapter, climbing robots are classified in terms of the environments that they explore, in order to better identify the class of structure-climbing robots to which the HyReCRo robot belongs.

After reviewing the main classes of climbing robots identified in the scientific literature, the origins of the HyReCRo robot as a robot with purely binary actuators are revisited, discussing the advantages and shortcomings of using purely binary actuation schemes in climbing robots.

- Chapter 3 presents a comprehensive kinematic analysis of the 2R$\underline{\text{P}}$R-PR parallel mechanisms that make up the legs of the HyReCRo robot. This kinematic analysis is necessary for analyzing the kinematics of the complete HyReCRo robot in later chapters. To that end, this chapter solves the forward kinematics of these 2R$\underline{\text{P}}$R-PR parallel mechanisms, analyzing also their singularities.

- Chapter 4 presents PaRoLa, a collection of Java graphical simulation tools developed within the framework of this thesis with the main purpose of aiding in the diverse kinematic analyses performed for the 2R$\underline{\text{P}}$R-PR mechanisms, for the HyReCRo robot, and for many more parallel robots.

- Chapter 5 presents the kinematic analysis of the complete HyReCRo robot, departing from the kinematic analysis of the 2R$\underline{\text{P}}$R-PR parallel mechanisms performed in Chapter 3. Accordingly, the forward and inverse kinematic problems of the HyReCRo robot are solved in this chapter.

- Chapter 6 deals with the computation of the boundaries of the workspace of the HyReCRo robot. To that end, methods for computing the workspace of robot manipulators are first reviewed, in order to choose the method that best fits the characteristics of the HyReCRo robot. Using Monte Carlo methods, the effects of changes of the design of the HyReCRo robot on its workspace are investigated. Also, some accuracy problems of Monte Carlo methods are discussed in this chapter, and a new Monte Carlo method for obtaining the workspaces of robot manipulators more accurately is developed.

- Chapter 7 copes with the calculation of the interior barriers of the workspace of the HyReCRo robot. These kinematic barriers are important when planning trajectories, since they constitute motion impediments for the robot. First, existing methods for obtaining the interior barriers of the workspace of robot manipulators are reviewed, analyzing their ability to handle collision constraints. Then, a novel method for effectively obtaining the interior barriers of the workspaces of redundant robots considering collision constraints is developed.

- Chapter 8 presents a lightweight prototype of the HyReCRo robot, mainly made of 3D-printed parts. The geometric design of this prototype is based on the kinematic and workspace analyses performed in previous chapters, which are used to determine the necessary dimensions of the robot so that it can perform the necessary motions for exploring three-dimensional structures, namely: concave and convex plane transitions. After briefly presenting this prototype, this chapter presents the detailed design of magnetic grippers based on the technology of switchable magnets, so that the developed prototype can use these grippers to adhere to real steel structures and climb them.

- Finally, Chapter 9 summarizes the main contributions of this thesis, and hints possible future research works derived from these contributions.

1.6 Summary of Materials, Methods, and Discussion of Results

This section presents a summary of the main materials and methods used for developing the research work presented in this thesis. Also, the main results obtained in each chapter will be briefly exposed and discussed.

1.6.1 Materials

The following list summarizes the main materials and tools used during the development of this thesis.

- **Easy Java Simulations (EJS).** This authoring tool, originally designed for aiding in the development of educational simulators, has proven to be extremely useful in the present thesis for developing simulation tools that support the kinematic analysis of the HyReCRo robot and of the 2R$\underline{\text{P}}$R-PR parallel mechanisms that make up its legs. Most of the simulations presented in the next chapters have been performed through Easy Java Simulations; in fact, Chapter 4 is devoted to introducing these developed simulations. The reason for choosing this tool over other options like Matlab (which has also been used in some parts of this thesis for performing symbolic calculations and doing other minor simulations) is that EJS allows one to easily build graphical representations of the simulated robots, which are of great support when analyzing the kinematics, workspace, and singularities of parallel robots. A brief introduction to Easy Java Simulations is presented in Section 4.1.

- **3D printer.** A custom-made 3D printer, developed by Prof. Dr. José María Marín, has been used for printing some PLA parts of the prototype of the HyReCRo robot presented in Chapter 8.

- **Prototype of the HyReCRo robot.** For the last part of this thesis, a prototype of the HyReCRo robot was built, with the purpose of testing this climbing robot on real steel structures. This prototype is described in Chapter 8, it weighs 2.19 kg, it is mainly made of aluminum and 3D-printed PLA parts, it is controlled by an Arduino MEGA 2560 board, and it is driven by Maxon and Actuonix electric DC actuators.

- **Magnetic grippers.** In Chapter 8, the development of novel magnetic grippers for the HyReCRo robot is described. These magnetic grippers are made of PLA 3D-printed parts, steel housings, and neodymium permanent magnets. They also carry Pololu DC micromotors, as well as Vytaflex rubber pads for increasing friction and preventing slippage (see Chapter 8 for more details).

- **MPJExpress** [170]. This Java library has been used to parallelize over eight processors the execution of the method proposed in Chapter 7 for obtaining the interior barriers of the workspace of redundant robots under collision constraints.
- **SOLID** [188]. This C library has been used in Chapter 7 to detect collisions between different cylindrical bodies of the Stewart platform.

1.6.2 Methods

The following list summarizes the main methods used during the development of this thesis.

- **Elimination methods.** For solving the position problem of robots and mechanisms in this thesis (i.e., their forward and inverse kinematic problems), the preferred methods have been *elimination methods*. These methods consist in *carefully* eliminating unknowns from the system of equations in successive steps, until the problem is reduced to finding the roots of a univariate polynomial in one of these unknowns. The reasons for preferring elimination methods over other methods are their low CPU times and the possibility of obtaining all solutions, including the non-real ones. In this thesis, elimination methods have been used for:
  - Solving the forward kinematic problem of 2R$\underline{\text{P}}$R-PR and 3R$\underline{\text{P}}$R parallel manipulators in Chapter 3.
  - Implementing the simulation of the forward and inverse kinematics of parallel robots in the virtual laboratory PaRoLa presented in Chapter 4.
  - Solving the forward and inverse kinematic problems of the complete HyReCRo robot in Chapter 5.
  - Solving the forward kinematics to generate random workspace points in the Monte-Carlo methods described in Chapter 6.
  - Densely sampling the self-motion manifolds of redundant robots in the method proposed in Chapter 7 for obtaining the interior barriers of their workspace under collision constraints.
  - Determining the minimum static friction coefficient necessary to prevent slippage of the grippers, in Chapter 8 (part of this problem is solved graphically, as explained next).

  However, elimination methods may have some limitations. On the one hand, these methods may miss solutions or may introduce spurious solutions. On the other hand, in some cases elimination methods become impractical since they lead to univariate polynomials whose degree is too high, and even computer algebra systems find it difficult to obtain the final univariate polynomial obtained after eliminating all other unknowns. For these reasons, at some points of this thesis, graphical methods have been used for solving nonlinear systems of equations instead of elimination methods, as explained next.

- **Graphical methods.** For nonlinear systems of two equations in two unknowns, these two equations can be regarded as defining two curves in the plane of these unknowns. Thus, one can plot these curves and solve the original system by finding the points where these two curves intersect. This graphical method has been used at two points of this thesis:

  – For solving the forward kinematics of 2-degrees-of-freedom degenerate parallel robots. The forward kinematics of the degenerate 2U$\underline{\text{P}}$S-U parallel robot studied in Section 3.6.4 has a positive-dimensional solution set that can be missed by elimination methods. Alternatively, the simulators presented in Chapter 4 solve graphically this problem as the intersection of two planar curves, and this graphical resolution does not miss solutions (see Section 4.4.3.2).

  – For determining the coordinates of the Instantaneous Center of Rotation (ICR) of the fixed gripper of the HyReCRo robot when it is on the verge of slipping due to insufficient friction. As shown in Section 8.7.2, the coordinates of the ICR must be solved from a system of two highly nonlinear equations that are too complicated to be solved via elimination. Therefore, it is much more practical to solve these equations graphically.

  Other graphical methods have been used to solve the inverse kinematics of the HyReCRo robot in Chapter 5. Since this is a kinematically redundant robot, the solution to its inverse kinematic problem is a positive-dimensional set (generically, a positive-dimensional manifold). As explained in Chapter 5, one possible way to solve the inverse kinematics of the HyReCRo robot consists in plotting projections of these positive-dimensional solution sets on two- and three-dimensional spaces (e.g., see Figures 5.4, 5.12, and 5.14).

- **Monte Carlo methods.** Classical Monte Carlo methods have been compared in Chapter 6 with a new Monte Carlo method developed in that chapter to obtain the boundaries of the workspace more accurately. As explained in that chapter, beta and Gaussian random numbers were generated from uniformly distributed numbers in $(0, 1)$ using the methods of Johnk ([49], p. 432) and Box-Muller ([49], p. 235), respectively.

- **kd-trees.** kd-trees are used by the method proposed in Chapter 7 to obtain the interior barriers of the workspace of redundant robots under collision constraints. More precisely, kd-trees are used in that chapter for clustering and matching discrete approximations of disjoint components of self-motion manifolds.

1.6.3 Results and Discussion

To conclude this introductory chapter, this subsection summarizes and discusses the main results derived from the research work described in each chapter. Most of these results have been published in JCR-indexed journals, as well as in peer-reviewed national and international conferences. All these publications have been listed in Section 1.4.

- Chapter 2: the review of the state of the art of climbing robots suggests that step-by-step structure-climbing robots can be very versatile, but they are usually too complex and this may contribute to the fact that the use of these robots in industrial inspection has not managed to generalize in practice. As an attempt to simplify these robots, the HyReCRo robot was originally proposed in the past as a robot with purely binary actuators, but purely binary actuation may prevent the robot from finely adjusting the position of its grippers to adhere them to the structure and perform plane transitions. It will be necessary to continuously actuate some of the actuators of the robot to finely position and attach its grippers to the climbed structure. Thus, a continuous kinematic analysis of the HyReCRo robot is necessary, and this analysis is conducted in the next chapters of the present thesis. This continuous analysis will be a necessary preliminary step to propose mixed continuous-binary motion schemes in the future.

- Chapter 3: this chapter mainly focuses on the kinematic analysis of the 2R$\underline{\text{P}}$R-PR parallel mechanisms that make up the legs of the HyReCRo robot. First, it is found that the kinematics of these mechanisms was solved by other researchers in the past. In this chapter, however, the forward kinematics and singularities are revisited and studied in more depth, and it is demonstrated for the first time that this mechanism always exhibits two special singularities which are fourfold solutions of the forward kinematic problem. One of these two singularities is isolated, and this chapter demonstrates that encircling this isolated singularity allows this mechanism to perform transitions between different solutions of the forward kinematics (also known as *assembly modes*) without traversing singularities. Then, this chapter extends this analysis to well-known 3R$\underline{\text{P}}$R planar parallel robots with flat platforms, which are shown to exhibit analogous fourfold singularities and the ability to perform nonsingular transitions by encircling these singularities. Despite all of this, the HyReCRo robot (whose legs are composed of these 2R$\underline{\text{P}}$R-PR parallel mechanisms) cannot perform such nonsingular transitions since its joint limits completely separate the different assembly modes of its 2R$\underline{\text{P}}$R-PR mechanisms and do not allow for encircling the aforementioned fourfold singularities. Actually, this separation of assembly modes is beneficial for simplifying the kinematic analysis of the complete HyReCRo robot in Chapter 5.

- Chapter 4: the authoring tool Easy Java Simulations turned out to be very useful for developing graphical simulators of parallel robots in this thesis. The simulators developed using this tool, and presented in Chapter 4, allow the user to visualize and graphically analyze the solutions of the forward and inverse kinematic problems of parallel robots, as well as their workspace and singularities. In particular, the developed tools have proven to be especially useful and powerful when visualizing how the singularities and workspace of parallel robots deform under changes in the design of the robot.

- Chapter 5: in this chapter, the forward and inverse kinematic problems of the HyReCRo robot are solved.

– Regarding the forward kinematic problem, this problem is quite straightforward once the forward kinematics of the 2R$\underline{\text{P}}$R-PR parallel modules of the legs of the robot is solved. In principle, since the forward kinematics of each of these modules has four real possible solutions, one should expect $4^4 = 256$ different solutions to the forward kinematics of the complete HyReCRo robot (since this robot is made of the serial combination of four 2R$\underline{\text{P}}$R-PR parallel modules). However, due to joint limits and collision constraints, only one of these four solutions of the parallel modules is valid, which means that the forward kinematics of the complete HyReCRo robot has only one collision-free solution.

– After solving the forward kinematics, a simplified version of the inverse kinematics is solved before addressing the general inverse kinematics of the HyReCRo robot, which is a more complex problem. This simplified version considers planar and symmetric postures of this robot. Although these postures may seem limited, it its found that they are very useful for analyzing the basic movements necessary for exploring structures, such as longitudinal movements along beams, convex transitions between different faces of the same beam, or concave transitions between different beams. After solving the Planar and Symmetric Inverse Kinematic (PSIK) problem, the PSIK-workspace (i.e., the reachable space attainable by the HyReCRo robot using only planar and symmetric postures) is analyzed in order to determine the sensitivity of this workspace with respect to its geometric design parameters. From this analysis, it is found that the workspace is most sensitive to changes in the width of the feet, which is one of the main geometric parameters of the 2R$\underline{\text{P}}$R-PR parallel modules, and in the stroke of the linear actuators used in these parallel modules.

– After solving the simplified inverse kinematic problem that considers only planar and symmetric postures, the general inverse kinematic problem of the HyReCRo robot is solved. Due to the kinematic redundancy of the HyReCRo robot, this problem has infinitely many different solutions lying on the so-called *self-motion manifolds* of redundant robots. For the HyReCRo robot, these manifolds generically are four dimensional. However, it is found that these manifolds become five-dimensional for special singularities of this robot in which one of the faces of both its feet become parallel, which turns out to be a very frequent case when performing plane transitions in a structure. In both cases, with four- or five-dimensional manifolds, the high dimensions of these self-motion manifolds impedes plotting them and performing a graphical analysis of the solutions of the inverse kinematics of this robot. Nevertheless, it is found that, for the HyReCRo robot, these four-dimensional manifolds can be projected to two-dimensional subspaces without losing relevant information regarding the overall posture of the robot, which allows for an intuitive and compact graphical visualization of the solutions of the inverse kinematics of this complex robot (analogously, in the case of five-dimensional manifolds, they can be projected to three-dimensional subspaces). Using these lower-dimensional projections, compact graphical representations of the solutions to the inverse kinematic problem of the HyReCRo robot can be visualized, which is of great use for planning the movements of this robot (i.e., for determining how to reach a desired position and orientation).

- Chapter 6: in this chapter, conventional Monte Carlo methods are first used for obtaining the workspace of the HyReCRo robot and study the sensitivity of its shape with respect to the design parameters of this robot, supporting the results found in Chapter 5 for the PSIK-workspace: the workspace of this robot is most sensitive to the width of its feet and the stroke of its linear actuators. Then, after discussing the limitations of existing Monte Carlo methods, which cannot obtain accurately the boundaries of the workspace even when largely increasing the number of randomly sampled points, a new Monte Carlo method is proposed in this chapter. Through several experiments performed with the workspace of the HyReCRo robot, it is demonstrated that the new proposed method can compute the boundaries of the workspace of robot manipulators more accurately than previous methods, requiring the same or less computation time than them, which makes the proposed method more efficient. The proposed method consists in using normal distributions to uniformly "grow" the workspace until its boundaries are reached.

- Chapter 7: this chapter begins by analyzing the limitations of existing methods for obtaining the interior barriers of the workspace of redundant manipulators under collision constraints. It is argued in this chapter that, with the exception of some simple cases, collision constraints generally are too complex for existing methods to accommodate them. Thus, in order to compute the interior workspace barriers in redundant robots under general collision constraints (which are important for the HyReCRo robot), a new method is developed in this chapter which is able to easily cope with such constraints. The proposed method consists in densely sampling the self-motion manifolds of redundant robots (discarding samples that do not satisfy collision constraints), clustering these samples to identify disjoint self-motion manifolds, and detecting when one of such manifolds vanishes, which denotes the occurrence of an interior barrier. Several experiments with redundant parallel robots are performed in Chapter 7 in order to demonstrate the feasibility of the proposed method and the importance of collision constraints, which drastically alter the distribution of interior barriers inside the workspace of redundant robots. In practice, however, this method is only feasible for robots with one- and two-dimensional self-motion manifolds, since applying it to robots with higher-dimensional manifolds would require prohibitive computational times. Thus, in order to apply this method to the HyReCRo robot, whose self-motion manifolds are four- and five-dimensional, an approximate method is suggested, which consists in detecting the vanishing of lower-dimensional projections of these manifolds. However, this approximate method may miss some interior barriers due to some information lost on account of these projections.

- Chapter 8: in this chapter, a lightweight prototype of the HyReCRo robot is built. This prototype is mainly made of PLA 3D-printed parts, is driven by electric DC motors and linear actuators, and is controlled by an Arduino and a custom-made power board. The dimensioning of this prototype is done on the basis of the kinematic analyses performed in the previous chapters of this thesis, and with the help of the developed simulation tools, in order to find a design of the HyReCRo robot which is able to perform concave and convex plane transitions necessary for exploring three-dimensional structures. After building this prototype, which weighs 1.55 kg (excluding grippers), magnetic grippers are developed, which employ the technology of switchable magnets. Each of the developed grippers offers a holding force of 33 kg on 3mm-thick steel plates, and their design is based on two safety criteria. Firstly, the grippers should not detach from the structure, which would produce the tip-over and fall of the robot. Secondly, the grippers should not slip due to insufficient friction. By imposing these criteria, design conditions are obtained, and magnetic grippers satisfying these conditions are built. The prototype carrying the developed grippers is tested on a real steel structure, demonstrating that it can firmly adhere to the structure and climb it even in the most demanding climbing scenarios, in which the detaching torque due to the weight of the robot is maximal. The developed grippers have reduced dimensions, which make them especially useful for structure-climbing robots, since these robots must move along the narrow beams of structures, in which there is little room available for robust (yet bulky) grippers.

Finally, the last chapter of this thesis presents further discussion regarding possible future research work derived from the results summarized in the previous list.

2 Literature Review of Climbing Robots

This chapter presents an up-to-date literature review of climbing robots, in order to contextualize the HyReCRo robot, which is the structure-climbing robot studied in the present thesis. To that end, climbing robots are first classified in section 2.1 according to the geometric characteristics of the environment explored by the different types of climbing robots. Then, section 2.2 introduces the original binary HyReCRo robot, places this robot in context considering the previous literature review, and analyzes the shortcomings of using a purely binary actuation scheme.

2.1 Climbing Robots

The framework of this thesis is the field of climbing robots. Generally speaking, the objective of climbing robots is to perform tasks at height, which are too dangerous to be performed by human operators due to several risks such as falling from height, electrocution, or working in difficult-to-access areas. Some of the typical applications of climbing robots are:

- cleaning facades of tall buildings,
- inspecting and maintaining bridges and other large structures in the construction industry (Figure 2.1),
- inspecting large vessels and storage tanks,
- inspecting electrical power lines,
- inspecting pipeline networks in industrial plants,

Figure 2.1: Vertical structures require inspection and maintenance tasks which are dangerous for human operators due to several risks, including falling from considerable heights. Examples of structures requiring operators working at heights: (a) Bimillenary's Hanger Bridge of Elche, (b) expansion works at the Quorum III building of the Miguel Hernández University.

- fumigating and harvesting in tall trees,
- exploring rough natural terrains and unstructured areas after disasters,
- and many more.

Nowadays, the field of climbing robots is quite mature and fertile, as it has been enriched by the work of numerous researchers from all over the world during the last thirty years. For example, as an indicator of the general interest of the scientific community in this field, one can check that searching the string "climbing robot" in the Web of Science returns more than 2000 results. Due to this large number of research works performed during the last decades, a wide variety of climbing robots have been developed, such that the term "climbing robot" has become a too broad term that includes from insect-like wall-climbing robots to rover-like robots for space exploration.

In essence, and roughly speaking, the design of any climbing robot encompasses two main parts [168]:

- Design of the **adhesion system** used by the robot to adhere to the climbed surfaces. Typical adhesion technologies used in climbing robots are [168]: magnetic, pneumatic, mechanical prehension, electrostatic adhesion, and chemical adhesion.
- Design of the **locomotion mechanism** that allows the climbing robot to move along the environment. Typical locomotion mechanisms used by climbing robots are [168]: arms and legs, wheels and chains, sliding frames, wires, and rails.

Based on these two dimensions (adhesion and locomotion technologies), Schmidt and Berns [168] present an excellent and quite exhaustive review of climbing robots until year 2013, which the reader is invited to read in order to get an overview of the field of climbing robots.

In this chapter, we present a different review of climbing robots, with the purpose of better contextualizing the present thesis. The review presented in the next subsections will classify climbing robots depending on the geometrical characteristics of their environments, since this classification will allow us to better contextualize the climbing robot studied in this thesis, distinguishing it from other climbing robots which are more suitable for other environments. The classification here presented is not perfect, since in some cases, a robot classified into a group may also fit into another group if other classification criteria are considered. Also, the review presented next is obviously not exhausting, since there exist over 2000 bibliographic references of climbing robots. Only the most relevant robots inside each group have been included in the following review, or those robots which are more strongly related to the present thesis.

Taking this into account, next we will present a literature review of climbing robots, in which robots will be classified into one of the following four groups:

- Wall-climbing robots
- Structure-climbing robots
- Cable-climbing robots
- Robots for traversing rough terrains

Most of the climbing robots found in the scientific literature fall into one of these four groups, although there are a few exceptions which will not be included in the next discussion.

In the next subsections, each of these four groups of climbing robots will be reviewed, identifying some of the most representative robots of each group, and highlighting their most characteristic features.

2.1.1 Wall-climbing Robots

This group of robots encompasses those robots that typically climb vertical walls, or surfaces with small curvature (almost flat walls), with the possibility of performing transitions between different working planes. This group of robots is characterized by the fact that their dimensions are much smaller than the size of the climbed surface. The case in which the climbed surface has similar dimensions to the robot will be discussed in next subsection 2.1.2 (structure-climbing robots). The robots considered in the present subsection usually are too bulky to climb structures.

Wall-climbing robots are used for performing diverse tasks at heights. For example, these robots are used for cleaning the large facades of tall skyscrapers, which is

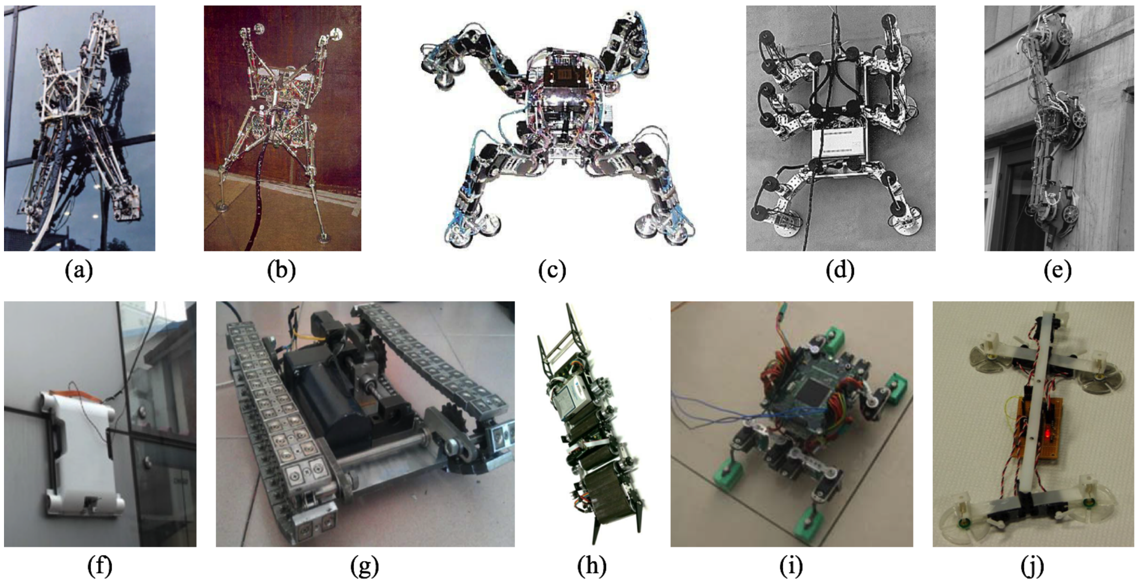


Figure 2.2: Some wall-climbing robots.

a very dangerous task for human cleaners due to the risk of falling from a considerable height [94]. Other robots which can be considered as wall-climbing robots are those that climb large pressure vessels in nuclear reactors [103], those that climb ship hulls in the maritime industry for inspecting and welding [8], or robots that climb large concrete bridges and other civil infrastructures [98, 168].

While the other classes of climbing robots that will be discussed in the next subsections have fairly homogeneous architectures (i.e., the overall design of the robot does not change drastically between different robots), the architectures of wall-climbing robots are quite heterogeneous. This is because the walls climbed by these robots, which have large sizes compared to the robots, provide enough space for the climbing robots to freely maneuver on them, which offers some flexibility for choosing a locomotion technology. Thus, one can find wall-climbing robots using legs, such as the NINJA quadruped robots ([74], Figure 2.2a), the RobugIIs robot ([103], Figure 2.2b), the MRWALLSPECT robot ([82], Figure 2.2c), or the REST robots ([8], Figure 2.2d). It is also possible to find wall-climbing robots using wheels, such as the Alicia3 robot ([98], Figure 2.2e) or the robot presented in [94] for cleaning glass facades (Figure 2.2f). Other robots use tracks, like those presented in [76] and [90] for the ship-building industry (Figures 2.2g and 2.2h). Finally, other researchers have designed bio-inspired gecko robots for climbing and maneuvering on vertical walls [184, 114] (Figures 2.2i and 2.2j).

As for the adhesion technologies used in wall-climbing robots, these will depend on the characteristics of the walls to be climbed. Thus, ferromagnetic walls present in ship hulls and storage tanks can be climbed using electromagnets [8] or permanent magnets [90], but also suction cups [103], which are also the preferred adhesion technologies for climbing concrete and glass walls [98, 94]. Another adhesion technology used in wall-climbing robots is dry adhesion [184]. Regarding mechanical prehension, this technology is not suitable for wall-climbing robots since these walls usually lack bulky protuberances that can be grasped, and the robot can usually access only one of the faces of the wall, i.e., it is not possible to "enclose" the wall using some claw, unlike in structure-climbing robots, in which this adhesion technology is very appropriate (see next subsection and Figure 2.4).

Sometimes, wall-climbing robots must negotiate diverse obstacles present on the climbed walls. The most frequent obstacles are small protuberances whose height is just a few milimeters, like small slots and ledges present on the facades of buildings [98]. Wheeled and tracked robots can usually overcome these small obstacles without any problem [76, 98]. However, for climbing higher obstacles it is necessary to provide the robot with specific mechanisms, like the arms of the Alicia3 robot [98], which lift the mobile modules of this robot (one at a time) to overcome obstacles present on concrete walls. Obviously, legged climbing robots can easily negotiate any obstacle present on walls, thanks to the high maneuverability offered by legs.

Another typical obstacle encountered by wall-climbing robots is a corner formed between two intersecting walls, which the robot must overcome in order to switch between different working planes (e.g., passing from floor to a vertical wall, or from wall to ceiling, or from one wall to another wall). Many wall-climbing robots are designed without the ability to perform these plane transitions, since many applications do not require them (for example: this is not usually required when cleaning the large glass facades of a building, or when inspecting large ship hulls). However, other wall climbing robots do have the ability to change between different working planes [103] (Figure 2.2b). Nevertheless, we may consider that, usually, wall-climbing robots are especially designed for working on large vertical planes and surfaces, with few transitions between different working planes. On the contrary, the structure-climbing robots analyzed in the next subsection are characterized by having to explore complex three-dimensional structures, in which transitions between different working planes are very frequent in order to explore all faces of all beams of the structure.

2.1.2 Structure-climbing Robots

This class refers to robots that climb and explore three-dimensional structures, composed of a network of beams and bars interconnected through structural nodes. In this case, we consider that the dimensions of the robot are similar to the width of the beams of the structure, for if the width of these beams was much larger than the dimensions of the robot, then it may be practically considered as a wall-climbing robot like those analyzed in the previous subsection.

Structure-climbing robots find applications in a number of inspection and maintenance tasks of human-made structures. For example: in the construction industry [14], these robots can be used for inspecting and searching structural defects such as corrosion or cracks of the protective coating of the metallic beams, both in the skeletons of buildings and in large metallic bridges [126]. In industrial plants, there are complex pipeline networks that constitute true three-dimensional structures that

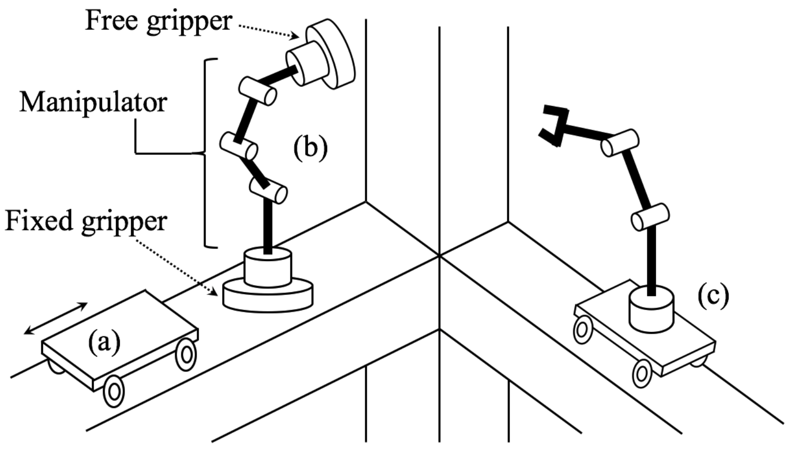


Figure 2.3: Types of structure-climbing robots.

include horizontal and vertical pipes, bends and T-junctions, pipes with different sections, valves and other obstacles. These pipelines require inspection and maintenance tasks such as NDT (non-destructive testing) for assessing the degradation of material or detecting welding defects [178]. The maintenance and change of light bulbs of street lights is another dangerous task especially suitable for climbing robots, which can climb the light pole, negotiating such obstacles as bends, changes of the diameter of the pole, and boxes [67, 181]. Other truss-like structures in which these robots can be useful are present in electric and telecommunication towers, stadiums, airports, and in general any man-made truss-like structure composed of many interconnected metallic bars. It is also worth to mention that structure-climbing robots are not limited to maintaining or inspecting existing structures, but they can also participate actively in their construction [125].

While wall-climbing robots analyzed in the previous subsection usually have heterogeneous architectures (i.e.: tracked, wheeled, legged, bio-inspired, gecko-like robots, etc.), structure-climbing robots usually have much more homogeneous architectures and can be classified into two main groups [178]: continuous-motion robots and step-by-step robots. Continuous-motion robots (Figure 2.3a) typically are mobile robots that employ wheels to slide along the structure [176, 11], whereas step-by-step robots (Figure 2.3b) are inchworm-like robots consisting of two grippers interconnected through a multi-Degrees-of-Freedom (DOF) manipulator. One of these grippers is attached to the structure through some adhesion technology (magnetic, suction cups, etc.), whereas the other gripper, which is free, is moved and placed by the manipulator at the next attachment point of the environment. Next, the grippers swap their roles, so that the previously free gripper adheres to the structure whereas the previously attached gripper is released, and a new motion cycle begins.

Generally, continuous-motion climbing robots are faster and simpler than step-by-step ones, which are slower, complex, and difficult to build and control. However, step-by-step robots offer a higher maneuverability due to their manipulator-like structure, which allow them to negotiate obstacles and move between different working

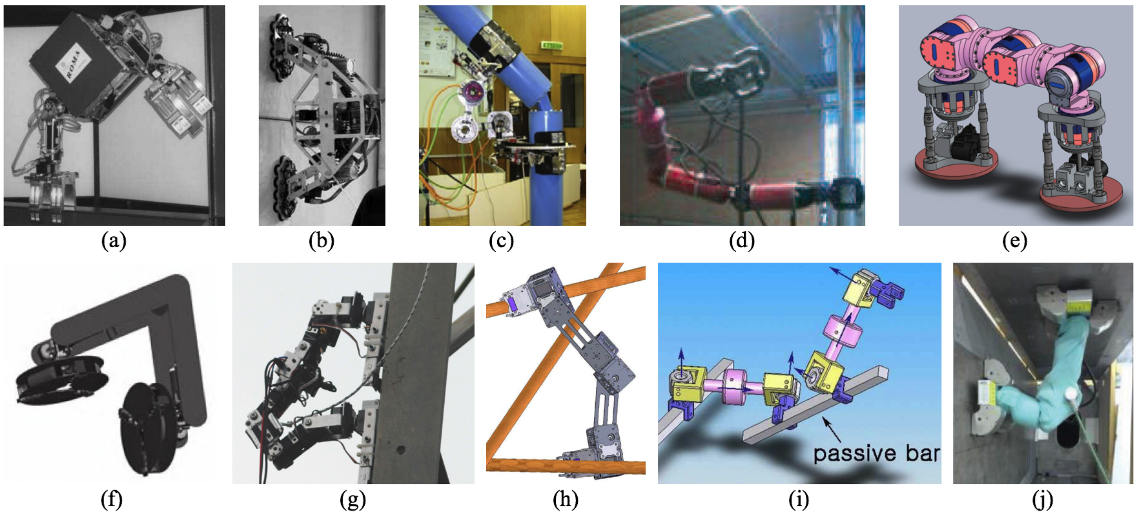


Figure 2.4: Serial step-by-step structure-climbing robots.

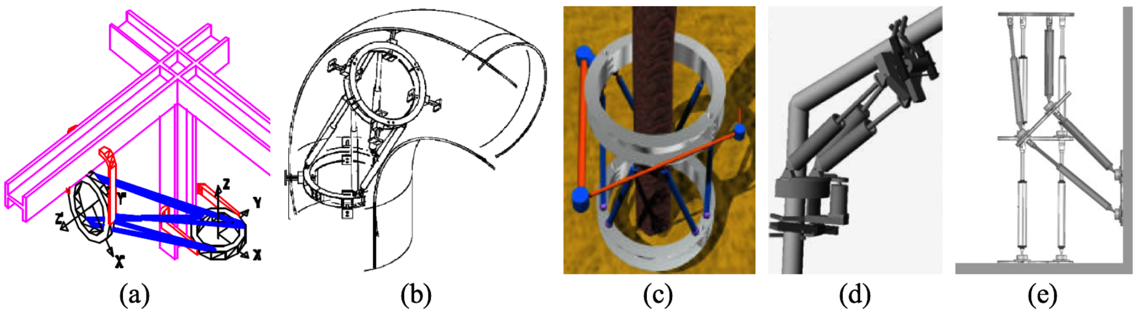


Figure 2.5: Parallel and hybrid step-by-step structure-climbing robots.

planes more easily. There is also the possibility of building hybrid robots (Figure 2.3c), which typically consist of wheeled continuous-motion robots equipped with articulated arms that increase their maneuverability and allow them to easily perform transitions between different working planes [189, 35].

The design of step-by-step climbing robots comprises two main parts: design of the grippers that provide adhesion to the climbed structure, and design of the multi-DOF manipulator that connects these grippers. Regarding the manipulator connecting both grippers, it can have a serial, parallel, or hybird architecture:

- **Serial climbing robots:** Balaguer et al. [14] developed the ROMA1 robot (Figure 2.4a), which was a serial climbing robot with six degrees of freedom (DOF), for climbing and inspecting metallic structures. Then, the ROMA2 robot was developed, which was a lighter improved version of the previous robot [62]. The ROMA2 robot (Figure 2.4b) only had four DOF and, for movements like convex transitions between different faces of a beam, it requires performing more maneuvers in order to perform the same plane transition as the ROMA1 robot [62]. Another 4-DOF serial climbing robot is the 3DCLIMBER ([178], Figure

2.4c), which was able to perform all the basic movements necessary for exploring three-dimensional structures with just four DOF. Guan et al. [67] developed the Climbot, a serial robot with five DOF for climbing poles and other vertical structures (Figure 2.4d). This climbing robot shares its architecture with other two serial climbers: a robot presented in [126] for inspecting steel bridges (Figure 2.4e), and the TREMO robot ([160], Figure 2.4f). Shvalb et al. [173] presented an inchworm-like 8-degrees-of-freedom redundant robot for climbing steel structures (Figure 2.4g). Mampel et al. [110] presented modular robots for climbing structures, they developed locomotion and gripping modules that provided the robots with a number of DOF between three and six (Figure 2.4h). Other modular robots for exploring three-dimensional structures are the Shady3D robots [204], which are 3-DOF robots that can individually explore three-dimensional structures, but they can also combine with other identical modules for building robots with higher maneuverability (Figure 2.4i). Finally, [193] presents a 7-DOF serial robot for climbing steel bridges (Figure 2.4j).

- **Parallel climbing robots:** the 6-DOF Stewart parallel platform has also been proposed as the manipulator of step-by-step structure-climbing robots, demonstrating its ability to negotiate structural nodes [165]. This climbing robot consists of a Stewart platform in which the mobile and fixed platforms carry grippers for laterally grasping the beams of the climbed structure (Figure 2.5a). By modifying this adhesion system, it is possible to use this robot also for outer and inner inspection of pipes (Figure 2.5b) [7], as well as for climbing and fumigating palm trees (Figure 2.5c) [166].

- **Hybrid Climbing robots:** these robots are combinations of serial and parallel mechanisms. For example: Tavakoli et al. [181] developed a hybrid pole climbing robot for replacing light bulbs of streetlights (Figure 2.5d). This robot consisted of a 3R$\underline{\text{P}}$R parallel robot serially connected to an actuated revolute joint. Figliolini et al. [58] developed a biped robot, in which each leg is composed of two identical 3R$\underline{\text{P}}$S parallel mechanisms serially arranged (Figure 2.5e). Although this biped robot was not originally presented as a structure-climbing robot but as a wall-climbing robot, we consider that its architecture seems capable of easily performing both concave and convex plane transitions and, therefore, it would be also appropriate for climbing structures.

Hybrid architectures seem to be especially suitable for structure-climbing robots, since they exploit both the wide workspace of serial manipulators (which is necessary for maneuvering and negotiating the obstacles found in these structures) and the high payload-to-weight ratio of parallel manipulators (which is desirable for general climbing robots, since they must carry their own weight as they climb).

As for the design of the grippers, the most frequently used adhesion technologies in structure-climbing robots are magnetic adhesion [173, 126, 193, 160] since climbed structures typically are ferromagnetic, and also mechanical prehension [178, 181, 14, 165, 67, 110], since the beams of the climbed stuctures usually have small width, which

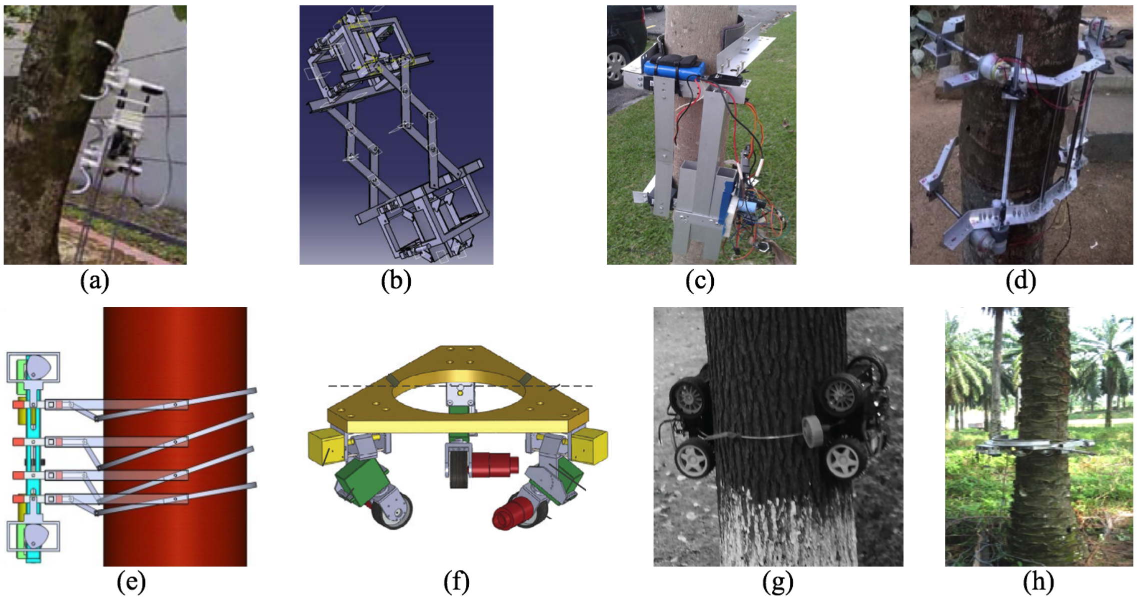


Figure 2.6: Tree-climbing robots.

allows the robot to grasp and embrace them using claws with appropriate size. It is also possible to find structure-climbing robots making use of suction cups [62].

The typical obstacles found in three-dimensional structures are T-joints, structural nodes (like the one illustrated in Figure 2.3, at which several beams meet), changes in the cross-section of the beams, valves and other control elements of pipelines, concave and convex transitions between different planes, etc. Continuous-motion structure-climbing robots usually find it difficult to negotiate these obstacles, although they can carry additional articulated arms for evading them, as explained above. On the contrary, step-by-step robots can easily and naturally evade such obstacles, since their manipulator-like architecture provides them with a high maneuverability for negotiating obstacles.

2.1.2.1 Tree-climbing robots

In this subsection, we have focused on robots for climbing artificial structures, composed of interconnected bars and beams. However, structure-climbing robots are closely related to tree-climbing robots, since trees can be regarded as natural structures, which are composed of a trunk and several interconnected branches. Tree-climbing robots are very similar to the structure-climbing robots discussed above: in fact, some of the discussed robots, like the parallel robot based on the Stewart parallel platform, were designed also with the purpose of climbing trees [166]. The objective of using robots for climbing trees is to perform dangerous tasks at them, such as fruit harvesting or pruning, but also fumigation, which is doubly dangerous for human operators since they may fall from the top of the tree or breathe the toxic pesticides.

Besides the parallel climbing robot mentioned earlier, we can find other tree-climbing robots in the literature. For example, the Treebot [104], shown in Figure

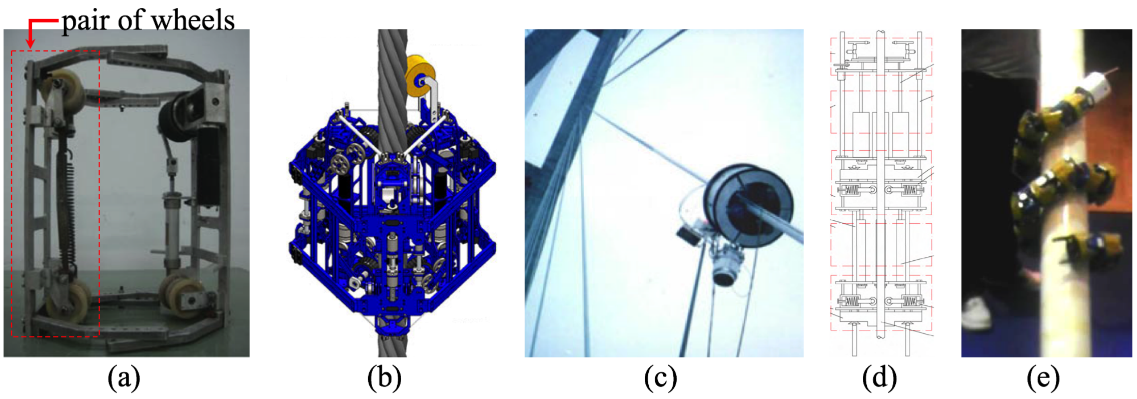


Figure 2.7: Cable-climbing robots.

2.6a, is an insect-like climbing robot with a tendon-driven continuum body and two claws at both ends of this body. These claws have spines which are used for adhering to the rugged surface of the tree. Unlike most tree-climbing robots, which usually wrap completely the trunk of the climbed tree and have limited ability to negotiate obstacles, the Treebot can perform transitions between different branches and the trunk. Examples of robots that completely wrap the trunk are: a robot for harvesting areca nut ([119], Figure 2.6b), the robot by Nor Faizal et al. ([57], Figure 2.6c), a coconut harvester ([51], Figure 2.6d), and the ingenious robot by Li et al. [92], which uses a single motor and some cams to climb trees (Figure 2.6e). All these robots have a common architecture, which consists of two grippers interconnected through some articulated or extensible mechanism that allows the robot to climb a tree following an inchworm-like gait, similar to step-by-step structure-climbing robots. However, we can also find single-body wheeled tree-climbing robots with simpler structure, like the robots presented in [69, 158, 172] (Figure 2.6f-h), which are very compact.

2.1.3 Cable-climbing Robots

The third group of robots considered in this chapter are those for climbing cables. Actually, the separation between structure- or pole-climbing robots and cable-climbing robots may not be clear in some cases, since a pole with a sufficiently small diameter may be regarded as a cable. However, we will consider here that cable-climbing robots refer to those robots that slide along cylindrical elements whose diameter is negligible compared to their length, and with much fewer obstacles than those encountered when climbing structures (in which structural nodes and plane transitions are very frequent). These cables are typically found in large hanger bridges, like the one shown in Figure 2.1a. In this subsection, we will also analyze the robots that move along electrical power lines, in order to inspect them.

2.1.3.1 Robots for climbing cables of hanger bridges

First, we will discuss robots that move along steel cables of hanger bridges. These cables, which are very tall, must be inspected in order to determine their defects and prevent further deterioration that may compromise the stability of the bridge [198].

As shown in Figure 2.7a, many cable-climbing robots consist of pairs of wheels (some of them motorized) or tracks oriented along the cables and pressed against them by means of springs that provide the necessary friction to climb. Following this philosophy, for example, the robot by Fengyu et al. [198, 199] has two pairs of wheels placed on opposite sides of the climbed cable (Figure 2.7a). Similarly, Cho et al. [34] present a robot composed of three pairs of wheels symmetrically distributed along a circle, placed every 120° (see Figure 2.7b). A similar robot with three pairs of wheels is presented in [105] (Figure 2.7c).

Besides wheeled and tracked robots, it is also possible to find step-by-step robots for climbing cables, in which one of the ends of the robot is attached to the cable whereas the other end is extended in order to grasp the upper part of the cable and climb, like the pneumatic robot presented in [91] (Figure 2.7d), which is similar to the parallel climbing robot shown in Figure 2.5c. Finally, snake-like robots that wrap around the cables may also be useful for climbing them ([194], Figure 2.7e).

The obstacles encountered in cables typically consist in progressive or abrupt changes of the diameter of the cable, which are due to defects [198]. Usually, when the height of these obstacles is moderate, they can be overcome directly by the wheels of cable-climbing robots.

One of the central topics in relation with cable-climbing robots is the safe landing mechanism. If the robot experiences some failure while climbing the cable at a considerable height, it is necessary to make the robot descend along the cable in order to recover it on the ground. However, this descent should be controlled in order to limit excessive velocities that may damage the robot or the cable. To control the descent of cable-climbing robots, they are usually provided with safe landing mechanisms which can be active or passive, such as brakes [33], pneumatic damping [199], or regenerative devices which transform kinetic energy into electrical energy that may be used for increasing the power autonomy of the robot [199].

2.1.3.2 Robots for climbing electric power lines and ropes

The second class of cable-climbing robots that we will consider are those that slide along high-voltage power lines in order to inspect their conductors, which are dangerous tasks for human operators due to the additional risk of electrocution. While the robots that climb steel cables of hanger bridges usually have to climb cables with large slopes (almost vertical cables), the robots that slide along power lines usually slide along cables whose slope does not exceed 30° [122].

Robots for power lines must be able to avoid some obstacles present on those lines, such as [85]: insulators, dampers, spacers, and warning spheres for preventing collisions between these lines and aircrafts.

Most climbing robots for inspecting power lines have a gondola-like design, in which a hanging heavy central body (the gondola), which usually carries control and power elements of the robot, acts as a stabilizer, since it lowers and balances the center

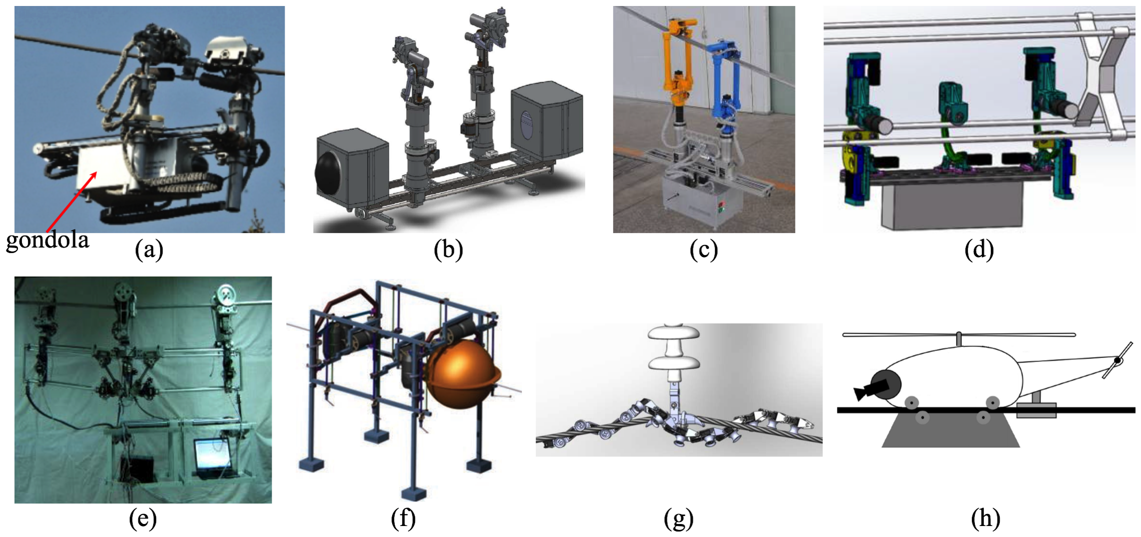


Figure 2.8: Climbing robots for inspecting high-voltage power lines.

of gravity of the whole robot. This central body hangs below the power lines through two or three articulated arms, which have wheels at their tops for sliding along the conductors. For example, the robot presented in [205] (Figure 2.8a) uses two arms with PRP architecture for hanging from the conductors, and it avoids obstacles by using inchworm-like or brachiation maneuvers with these arms, displacing conveniently the gondola to reduce the inertia during these maneuvers. A similar robot is presented in [93], which also has two RPR arms for avoiding obstacles through brachiation maneuvers (Figure 2.8b). Another similar robot with two arms is presented in [202], which also balances its center of mass by conveniently displacing the gondola, but the arms of this robot completely wrap the conductors (Figure 2.8c), which may hinder the avoidance of some obstacles.

Gondola-like robots with three arms can avoid obstacles more safely, since these robots can have their gondolas stably suspended from the conductors through two of the arms at all times, whereas the third arm is used for maneuvering and negotiating obstacles. For example: the robot presented in [197] has two locomotion arms and one support arm. When encountering an obstacle, these arms are sequentially disconnected from the conductor in order to overcome the obstacle, such that two of the arms are grasping the conductor at all times (Figure 2.8d). The three-arm robot presented in [200] follows a similar strategy, although this robot has a fairly complicated architecture by means of which it balances the center of gravity of its gondola in order to increase stability (Figure 2.8e). Finally, the robot presented in [122] does not have arms, but its gondola slides directly along the cables through three rollers (Figure 2.8f). This robot has counterweights for lowering the center of gravity and increasing stability, and its rollers are actively and sequentially shifted upwards in order to overcome obstacles such as the warning spheres found in power lines [122].

Although most climbing robots for power lines have gondola-like designs, it is possible to find also other less-conventional designs, such as snake-like hyper-redundant

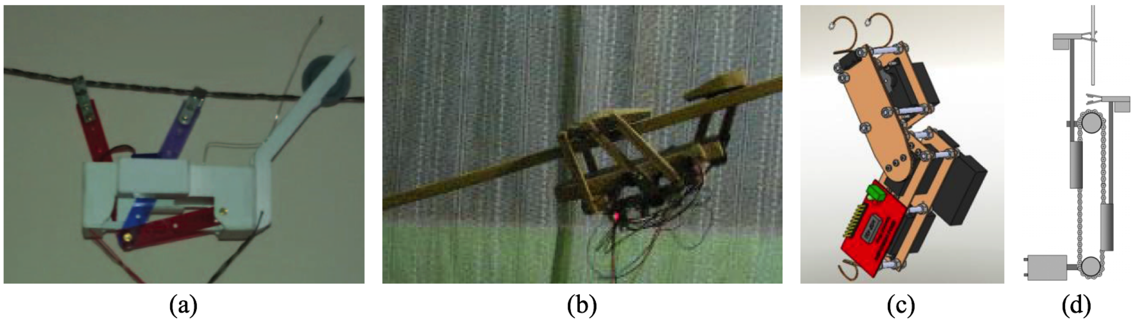


Figure 2.9: Rope-climbing robots.

robots which wrap around the conductors ([192], Figure 2.8g) or hybrid robots like the one proposed in [84], which consists in a small helicopter with wheels (Figure 2.8h). This hybrid flying-wheeled robot uses its wheels to slide along the conductors until an obstacle is found, which is overcome by flying.

Besides having to cope with the negotiation of obstacles, other topics of interest in the field of power-line-climbing robots are [85]: the possibility of powering the robot directly from these power lines (instead of carrying batteries), the need to shield the electronics of the robot from electromagnetic interferences due to the proximity of the high-voltage conductors, and the difficulty of detecting the aforementioned obstacles through computer vision systems.

To conclude this subsection, it is worth discussing rope-climbing robots, which are similar to those that slide along power lines. These robots climb either vertical or almost-horizontal ropes (with slopes generally not exceeding 30°). Rope-climbing robots may be useful in military contexts, in order to transport tools or supplies across mountains and other natural obstacles [83]. In [83], an innovative robot based on a four-bar mechanism was proposed for traversing ropes with slopes under 30° (Figure 2.9a). Another two-legged robot for traversing low-slope ropes was presented in [73] (Figure 2.9b). Köse et al. [88] presented an articulated miniature robot for climbing vertical ropes using clips (Figure 2.9c). Finally, Schober [169] presented another robot for climbing vertical ropes using two parallel claws driven by a single motor and a chain mechanism (Figure 2.9d).

2.1.4 Rough-terrain Climbers

To conclude this review of the state of the art of climbing robots, we will discuss the class of mobile robots designed for traversing rough terrains and unstructured environments, such as natural terrains or post-disaster scenarios (e.g., after an earthquake). This class of robots includes rescue robots and robots for space exploration. Actually, this class of robots cannot be strictly regarded as robots for climbing vertical structures (unlike most of the robots studied earlier in this chapter), but these robots that traverse rough terrains certainly have the ability to *climb* and overcome some obstacles of the environment, such as stones, steps, stairs, slopes, and other objects that would hinder the motion of conventional wheeled mobile robots.

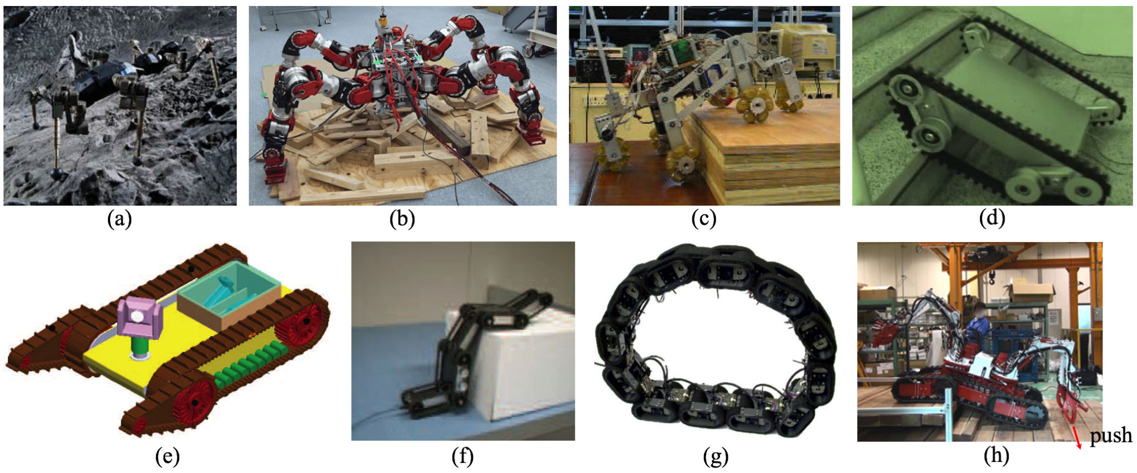


Figure 2.10: Robots for traversing rough terrains and climb their irregularities.

Obviously, and omitting issues related to gait stability and tip-over, legged robots generally find it easy to traverse rough terrains full of obstacles, since these robots have an excellent maneuverability for negotiating them. For example, the SpaceClimber robot [16] is a six-legged insect-like robot designed for planetary exploration (Figure 2.10a). This robot is able to climb slopes below 25° with fine-grained soils. The WAREC-1 robot [111] (Figure 2.10b) is a crawler quadruped robot for rescue and exploration tasks in unstructured post-disaster environments. This robot can successfully climb inclines covered by debris, which hinder the motion of the robot since these debris tend to collapse when the robot walks on them. To decrease the amount of slippage of the robot due to the debris collapsing under its weight, the belly of this robot has spikes which act as "hooks" that interfere mechanically with these debris.

Besides legged robots, rover-like all-terrain autonomous vehicles are also quite suitable to traverse rough terrains. These robots may have four [157] or six [56] wheels. These robots can passively climb obstacles whose height is twice the diameter of their wheels [56], and may employ omnidirectional wheels for increasing their maneuverability in unstructured environments [99] (Figure 2.10c).

Other type of robots especially suitable to traverse rough terrains are those using tracked locomotion. For example, Lim et al. [95] present a single-track rescue robot able to modify the shape of its tracks by reorienting its wheels, with the purpose of facilitating the climbing of steps and stairs (Figure 2.10d). Reference [50] presents an articulated robot, composed of two serially-connected tracked segments that facilitate the climbing of stairs (Figure 2.10e). Obviously, one may continue adding more and more tracked segments in order to increase the mobility of the robot and its ability to negotiate obstacles. An example of this is presented in [203], which presents a snake-like robot consisting of five tracked segments that allow it to climb higher steps (Figure 2.10f). Another interesting tracked robot is the OUROBOT by Paskarbeit et al. [128], which is composed of twelve tracked segments forming a closed loop (Figure 2.10g). This robot moves by rolling as if the whole robot was a wheel, and since it is composed of so many segments, it can adapt its shape to accurately match the obstacles of the terrain.

Finally, it may be interesting to combine different locomotion technologies to build more flexible robots for traversing rough terrains. For example, Kamekazi et al. [81] developed the OCTOPUS robot, a disaster-response robot consisting of a tracked mobile base that carries four articulated arms. These arms can be used to grasp objects of the environment, but they can also help the robot to traverse the environment and negotiate obstacles more easily. For example, these arms can push against the ground while the tracked base attempts to climb a step, in order to provide an additional impulse (Figure 2.10h).

2.2 The HyReCRo Robot: a Redundant Serial-parallel Structure-climbing Robot with Binary Actuation

After the previous review of the state of the art of climbing robots, we are in the position of better contextualizing the robot studied in the present thesis. The HyReCRo robot studied in this thesis belongs to the class of step-by-step structure-climbing robots discussed in section 2.1.2 of the present chapter. This robot, as shown in Figure 2.11a, was proposed by Ubeda et al. [183] for climbing and exploring three-dimensional steel structures. This robot is biped, with each of its legs being composed of two parallel mechanisms of type 2R$\underline{\text{P}}$R-PR, which is the two-DOF mechanism illustrated in Figure 2.11b. This robot has ten degrees of freedom, which makes it kinematically redundant. In chapter 5, a more detailed description of the architecture of this robot will be provided.

Originally, the HyReCRo robot was conceived as a robot with purely binary actuators, i.e., all its actuators could adopt only two extreme states: the linear actuators could be either completely retracted or completely extended. The advantages of a purely binary actuation scheme are related to simpler control and motion planning algorithms. This idea is very attractive for step-by-step climbing robots, since these robots usually are quite difficult to control and operate, which prevents these robots from finding their way out of laboratories and generalizing their use in industrial inspection [177]. However, if all actuators of the robot are binary, it can only attain a finite subset of discrete postures, which may impede the execution of some movements which are essential for exploring a three-dimensional structure, such as convex plane transitions between adjacent faces of the same beam (Figure 2.11c) or concave transitions between adjacent beams (Figure 2.11d). This is because these essential movements can only be performed if the gripper of the robot is placed on the surface to which it must be adhered (or sufficiently close to it), which may not be possible in general if purely binary actuation is used, since binary actuation only yields a finite set of discrete postures and it may occur that none of these postures places the gripper sufficiently close to the adhesion surface (actually, this would depend on the initial position of the robot, before starting to perform the plane transition).

Therefore, it becomes evident that at least some actuators of the HyReCRo robot should be continuously operated, i.e., the actuated joint coordinates of this robot should

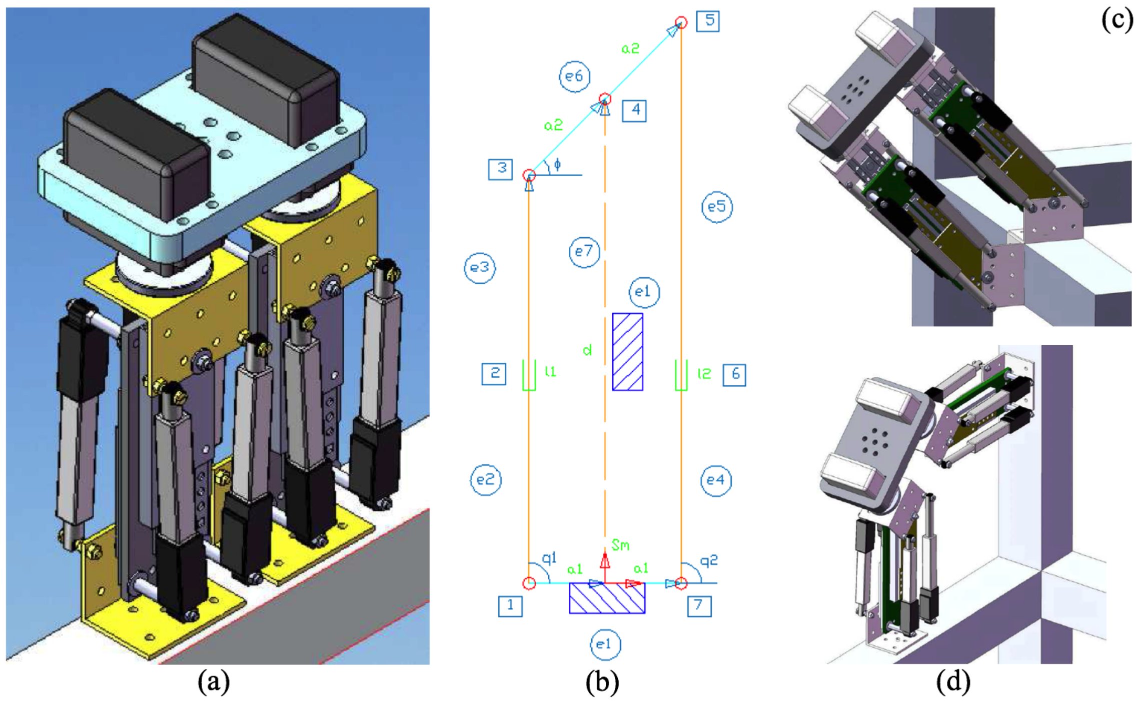


Figure 2.11: The binary HyReCRo robot as originally proposed in [183] (Repeated from Figure 1.1).

be allowed to take any value between two joint limits. Taking this into account, this thesis relaxes the purely binary actuation constraint of the original HyReCRo climbing robot, and presents an in-depth kinematic analysis of the continuously-actuated version of this robot. As it will be discussed in chapter 9, this continuous kinematic analysis is the first step towards a mixed binary-continuous version of this robot, in which the robot may be operated following a two-steps procedure. According to this procedure, first the robot would be coarsely moved to a configuration near the desired posture using a purely binary actuation, and then some (or all) of its joint coordinates would be continuously actuated around the coarse posture attained through binary actuation, in order to finely reach the precise posture necessary for exploring the structure (e.g., for performing a plane transition).

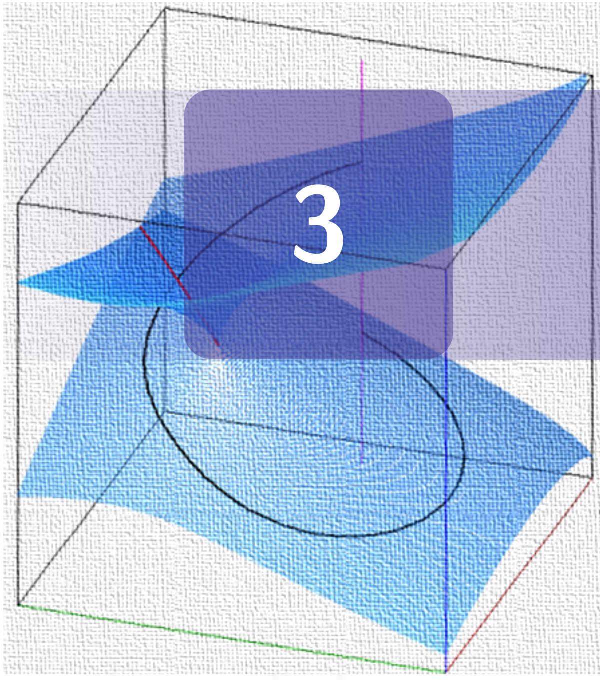
The continuous kinematic analysis of the HyReCRo robot is not easy, due to two reasons: this robot is kinematically redundant (it has ten degrees of freedom) and has a hybrid serial-parallel architecture. In order to perform a comprehensive kinematic analysis of the HyReCRo robot, the following analyses are presented in the next chapters of this thesis:

- The next chapter presents a detailed kinematic analysis of the 2R$\underline{P}$R-PR parallel mechanisms that make up the legs of the HyReCRo robot. This kinematic analysis is aided by the graphical simulation tools presented in chapter 4. The kinematic analysis of these parallel mechanisms is a prerequisite for the study of the forward and inverse kinematic problems of the complete HyReCRo robot, which are solved in chapter 5.

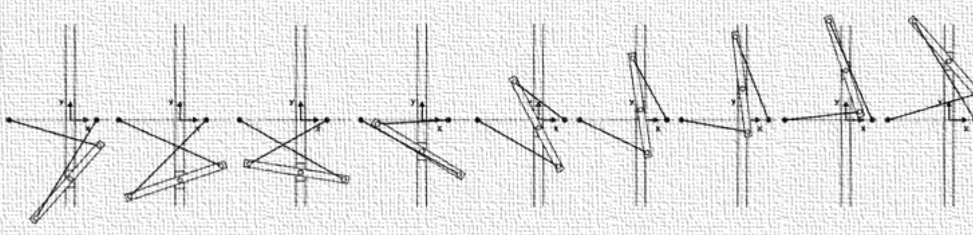
- Chapter 6 analyzes the boundaries of the workspace of the HyReCRo robot, with the purpose of investigating the influence of the geometric design parameters of this robot on its workspace. These boundaries are computed using a new Monte Carlo method developed in this thesis to obtain more accurately the boundaries of the workspace of robot manipulators.

- Chapter 7 investigates the interior kinematic barriers existing within the boundaries of the workspace of this robot. These interior barriers are of uttermost importance for planning the movements of general robot manipulators. To obtain these interior barriers respecting the restriction that mechanical interferences should not occur (i.e., collisions between different parts of the robot or with the climbed structure are forbidden), a new sampling method is proposed in Chapter 7.

- Finally, based on the kinematic analyses performed in the previous chapters, chapter 8 presents a prototype of the HyReCRo robot, as well as the detailed design of novel magnetic grippers developed so that this prototype can adhere to real steel structures and climb them.



3 Parallel Modules and Nonsingular Transitions



This chapter presents the kinematic analysis of the 2R$\underline{\text{P}}$R-PR parallel mechanism shown in Figure 3.1. Each leg of the HyReCRo climbing robot studied in this thesis is composed of two parallel mechanisms of this type connected in series. This chapter begins by analyzing the kinematics and singularities of this 2R$\underline{\text{P}}$R-PR mechanism (Section 3.1). This analysis is necessary for analyzing the complete climbing robot in later chapters. During this analysis, it is found that the 2R$\underline{\text{P}}$R-PR parallel mechanism has the ability to enlarge its range of operation by re-configuring itself without crossing singularities (Section 3.2). This behavior is achieved by encircling isolated singularities at which the forward kinematic problem admits solutions with multiplicity four. Later, in Section 3.4, this behavior is demonstrated also in 3R$\underline{\text{P}}$R parallel robots with flat platforms, which were thought to be unable to exhibit such behavior until now. Then, Section 3.5 studies the stability of this behavior (reconfiguration by enclosing isolated quadruple singularities) under small perturbations in the design of the robot. Finally, Section 3.6 presents a method for determining how isolated singularities transform under these perturbations, complementing and completing in a more formal way the analysis of Section 3.5.

3.1 2R$\underline{\text{P}}$R-PR Parallel Mechanisms

Each leg of the HyReCRo biped climbing robot studied in the present thesis, is composed of the serial combination of two parallel mechanisms of the type 2R$\underline{\text{P}}$R-PR, which is shown in Figure 3.1a. This 2R$\underline{\text{P}}$R-PR mechanism is composed of a mobile platform B-B_1-B_2 and a fixed platform A_2-A_1. Point B of the mobile platform is constrained to move along a passive slider OB which forms a constant angle α with segment A_2-A_1.

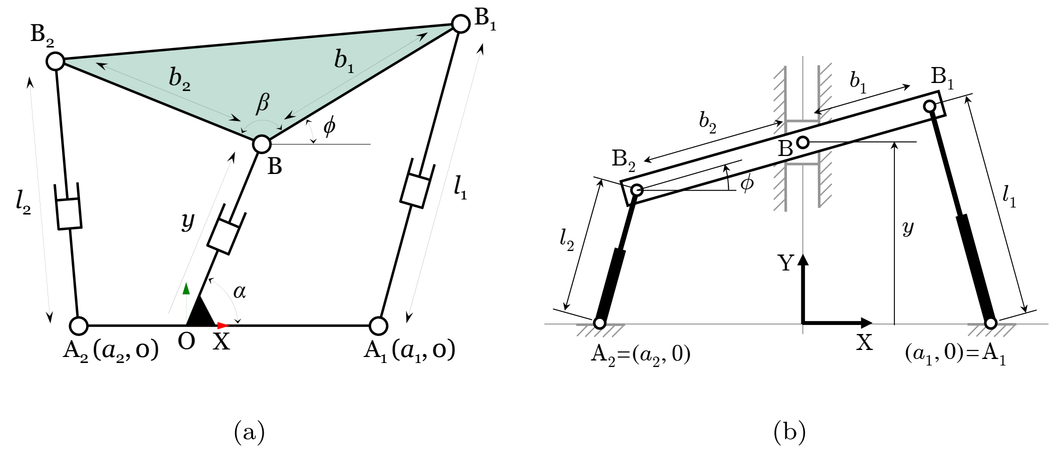


Figure 3.1: General (a) and analytic (b) 2R$\underline{\text{P}}$R-PR mechanisms.

Furthermore, the mobile platform is connected to the fixed platform A_2-A_1 through two linear actuators A_1B_1 and A_2B_2, whose lengths are l_1 and l_2, respectively. By controlling these lengths l_1 and l_2, one can control the orientation ϕ of the mobile platform, as well as its position y along the passive slider OB.

Actually, the parallel mechanisms that compose the legs of the HyReCRo robot are a variant of the general 2R$\underline{\text{P}}$R-PR mechanism shown in Figure 3.1a. Such a variant is shown in Figure 3.1b, and is characterized by $\alpha = \pi/2$ rad (i.e., the passive slider OB is perpendicular to segment A_2A_1) and $\beta = \pi$ rad (i.e., the three joints B-B_1-B_2 of the mobile platform are aligned). This modified 2R$\underline{\text{P}}$R-PR mechanism is *analytic*, which means that its forward kinematic problem (i.e., the problem which consists in solving y and ϕ in terms of l_1 and l_2) can be solved analytically or in closed form.

A symmetric version (i.e., with $a_1 = -a_2$ and $b_1 = b_2$) of the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b was studied by Ridgeway et al. [159], who combined several mechanisms of this type in series to form a snake-like articulated robot for inspecting nuclear facilities. Later, Kong and Gosselin [87] studied the forward kinematic problem of the general 2R$\underline{\text{P}}$R-PR parallel mechanism of Figure 3.1a, describing some variants (or particular cases) of this general mechanism, for which the forward kinematic problem becomes more simple and admits analytical solutions (among these variants was the analytic mechanism of Figure 3.1b). Furthermore, using Sturm's theorem, Kong and Gosselin [87] also demonstrated that the maximum number of real solutions of the forward kinematic problem of the analytic mechanism of Figure 3.1b is four.

Although the forward kinematic problem of the analytic parallel mechanism of Figure 3.1b (i.e., the mechanism used in the legs of the HyReCRo robot) was already solved by previous researchers [159, 87], this problem will be revisited in the following subsection 3.1.1 with the purpose of studying in more depth the multiplicity of its solutions, as well as the relationship between this multiplicity and the parallel (or Type-2) singularities (subsection 3.1.2) of this analytic mechanism. It is well known (subsection 3.2) that the solutions (of the forward kinematic problem) with multiplicity three or more are closely related to the ability of a parallel mechanism to reconfigure itself and perform transitions between different solutions of its forward kinematic problem without crossing undesirable singularities (*nonsingular transitions*), which is useful for enlarging the workspace of the mechanism [185] without losing control of the robot.

3.1.1 Forward Kinematics

This subsection revisits the resolution of the forward kinematic problem of the analytic mechanism of Figure 3.1b, a problem which was already solved by other researchers in the past [159, 87]. The objective of revisiting this problem is to gain further knowledge about the multiplicities of its solutions, as well as about the relationship between these multiplicities and the singularities of this mechanism. Obviously, the resolution of the forward kinematic problem of this mechanism will be also necessary for solving the forward kinematic problem of the complete HyReCRo robot in chapter 5.

In general, for any parallel mechanism or parallel robot, the forward kinematic problem consists in determining the *pose* ($=$ position $+$ orientation) of the mobile platform of the mechanism in terms of its active joint coordinates (i.e., the coordinates associated to the joints driven by actuators). In general, for fixed active joint coordinates, there are different valid poses for the mobile platform, i.e., the forward kinematic problem admits several different solutions. Traditionally, these different solutions have been called *assembly modes*.

For the analytic parallel mechanism of Figure 3.1b, the forward kinematic problem consists in determining the position y and orientation ϕ of the mobile platform in terms of the lengths l_1 and l_2 of the linear actuators. These four variables are related through the two following restrictions, which can be easily derived from Figure 3.1b:

$$(b_1 \cos\phi - a_1)^2 + (y + b_1 \sin\phi)^2 = l_1^2 \tag{3.1}$$

$$(-b_2 \cos\phi - a_2)^2 + (y - b_2 \sin\phi)^2 = l_2^2 \tag{3.2}$$

where it is assumed that the geometric parameters of the mechanism satisfy $b_1 > 0$, $b_2 > 0$, and $a_1 \neq a_2$. Equation (3.1) imposes the condition that the distance between A_1 and B_1 must be l_1, whereas (3.2) imposes the condition that the distance between A_2 and B_2 must be l_2. The objective of the present subsection is to obtain solutions (ϕ, y) to the previous system of equations with multiplicity higher than two (i.e., solution pairs (ϕ, y) which are repeated three or more times), since these solutions may enable the phenomenon of nonsingular transitions.

Eliminating y between Equations (3.1) and (3.2) by means of resultants, one arrives at a cubic equation in $\psi = \cos\phi$:

$$P(\psi) = p_3\psi^3 + p_2\psi^2 + p_1\psi + p_0 = 0 \tag{3.3}$$

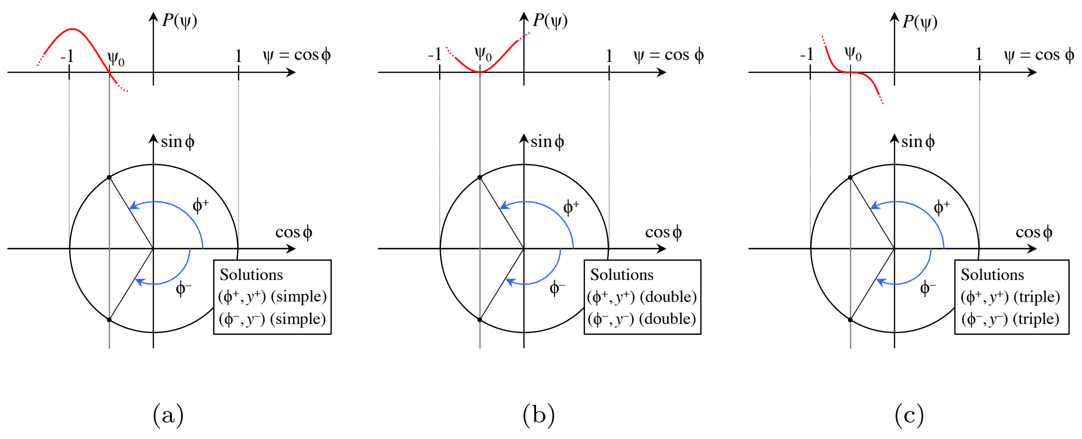


Figure 3.2: Zeros of $P(\psi)$ in $(-1, 1)$ with different multiplicities. (a) Simple zero. (b) Double zero (singularity). (c) Triple zero (cusp point).

where the expressions of the coefficients (p_3 ... p_0) are:

$$p_3 = 8b_1b_2(a_2 - a_1)(b_1 + b_2) \tag{3.4}$$

$$p_2 = 4\left[(b_1 + b_2)(b_1s_2 + b_2s_1) - (a_1b_1 + a_2b_2)^2\right] \tag{3.5}$$

$$p_1 = 4(s_1 - s_2)(a_1b_1 + a_2b_2) - p_3 \tag{3.6}$$

$$p_0 = -4(b_1 + b_2)(b_1s_2 + b_2s_1) - (s_1 - s_2)^2 \tag{3.7}$$

and where $s_i = a_i^2 + b_i^2 - l_i^2$ $(i = 1, 2)$. $P(\psi)$ is the *characteristic polynomial* of this robot, and it is always cubic since $b_1, b_2 > 0$ and $a_1 \neq a_2$. Thus, Equation (3.3) always yields three solutions for $\psi = \cos\phi$, two of which may be complex. After obtaining ϕ from Equation (3.3), y must be computed from Equations (3.1) and (3.2), completing the solution pair (ϕ, y). To study the possible multiplicities of a given solution pair (ϕ, y), we will analyze how the graph of polynomial $P(\psi)$ intersects the horizontal ψ axis in $[-1, 1]$ (note that intersections out of this interval do not result in real solutions, since real angles have cosines between -1 and 1). Three cases will be considered because the equations involved are different for each case.

3.1.1.1 Case 1: intersection in the open interval $(-1, 1)$

The upper part of Figure 3.2 shows the graph of $P(\psi)$ intersecting the abscissa axis at some $\psi_0 \in (-1, 1)$. Since $P(\psi)$ is cubic, three types of intersection are possible: simple, double, or triple.

If $P(\psi)$ has a simple zero at $\cos\phi = \psi_0$ (Figure 3.2a), two different[1] solutions are possible for ϕ:

$$\phi^+ = \text{acos}(\psi_0), \quad \phi^- = -\text{acos}(\psi_0) \tag{3.8}$$

[1]Angles differing by an integer multiple of 2π rad are considered equal.

The calculation of these two solutions is shown graphically in Figure 3.2a, where a vertical line passing through $\cos\phi = \psi_0$ intersects the unit circle at two points, yielding two possible values for $\sin\phi$ and hence two values for ϕ. For each value of ϕ, y is computed from the following equation, which is obtained by subtracting (3.2) from (3.1)

$$y = \frac{s_2 - s_1 + 2(a_1 b_1 + a_2 b_2)\cos\phi}{2\sin\phi(b_1 + b_2)} \tag{3.9}$$

which is valid because $\sin\phi \neq 0$. Thus, a simple zero of $P(\psi)$ in $(-1, 1)$ gives two different real solutions to the forward kinematics: (ϕ^+, y^+) and (ϕ^-, y^-), where y^+ and y^- are the values obtained after substituting ϕ^+ and ϕ^- into Equation (3.9), respectively.

If the intersection between the graph of $P(\psi)$ and the ψ axis in $(-1, 1)$ is double (Figure 3.2b), the solutions to forward kinematics are calculated as in the simple case, obtaining (ϕ^+, y^+) and (ϕ^-, y^-). However, these two solutions are now double because $\cos\phi = \psi_0$ is a double zero of $P(\psi)$. This case corresponds to a singularity.

Finally, $P(\psi)$ may have a triple zero at ψ_0 (Figure 3.2c). In that case the two solutions (ϕ^+, y^+) and (ϕ^-, y^-) are obtained as in the previous cases but have multiplicity three. To make this case possible, it is necessary to solve the unknowns (l_1^*, l_2^*, ψ_0) that satisfy the following system (condition of triple zero of a polynomial):

$$P(\psi) = 0, \quad P'(\psi) = \frac{\partial P}{\partial \psi} = 0, \quad \frac{\partial^2 P}{\partial \psi^2} = 0 \tag{3.10}$$

with $l_1^*, l_2^* > 0$ and $\psi_0 \in (-1, 1)$. In case such a solution exists, a trajectory enclosing the cusp points (l_1^*, l_2^*) in the joint space may produce nonsingular transitions. The triple zeros of $P(\psi)$ will be studied later for a particular geometry because it is difficult to solve the system (3.10) when generic design parameters are considered.

Next, it will be shown that the forward kinematics can have solutions with multiplicity higher than two if $\sin\phi = 0$, even if $P(\psi)$ has no triple zeros.

3.1.1.2 Case 2: intersection at $\psi = -1$

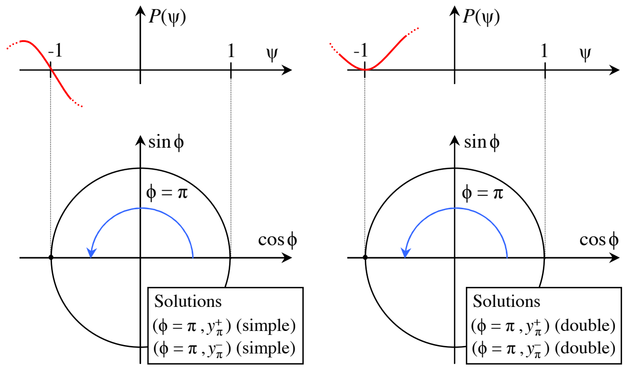
Figure 3.3 (left) shows the graph of $P(\psi)$ intersecting the ψ axis at $\psi = -1$. The condition $P(-1) = 0$ is equivalent to:

$$l_1^2 = l_2^2 + (a_1 + b_1)^2 - (a_2 - b_2)^2 \tag{3.11}$$

which defines a hyperbola in the joint space (l_1, l_2). When the joint coordinates belong to it, $P(\psi)$ vanishes at $\psi = -1$, which yields a single solution for the orientation: $\phi = \pi$. Thus, the curve defined by Equation (3.11) will be called hereafter the π-*hyperbola*.

For $\phi = \pi$, y cannot be computed using Equation (3.9) because $\sin\phi = 0$ (the denominator vanishes). To solve y in this case, we proceed as follows. First, we perform the following linear combination of Equations (3.1) and (3.2):

$$b_2\,[\text{Equation (3.1)}] + b_1\,[\text{Equation (3.2)}] \longrightarrow [\text{Equation (3.13)}] \tag{3.12}$$



$P(\psi)$
-1
1
ψ
$\sin\phi$
$\phi = \pi$
$\cos\phi$
Solutions
$(\phi = \pi, y_\pi^+)$ (simple)
$(\phi = \pi, y_\pi^-)$ (simple)
$P(\psi)$
-1
1
ψ
$\sin\phi$
$\phi = \pi$
$\cos\phi$
Solutions
$(\phi = \pi, y_\pi^+)$ (double)
$(\phi = \pi, y_\pi^-)$ (double)

Figure 3.3: Simple (left) and double (right) zeros of $P(\psi)$ at $\psi = -1$

note that this linear combination is never problematic because we assume that both b_1 and b_2 are strictly positive. The previous linear combination cancels the linear term of y, so that the resulting equation only involves y quadratically. The resulting equation is:

$$2b_1b_2(a_2 - a_1)\cos\phi + (b_1 + b_2)y^2 + b_1s_2 + b_2s_1 = 0 \tag{3.13}$$

solving y from Equation (3.13) yields the following two solutions:

$$y = \pm\sqrt{\frac{2b_1b_2(a_1 - a_2)\cos\phi - b_1s_2 - b_2s_1}{b_1 + b_2}} \tag{3.14}$$

Now we can obtain the solution corresponding to $\phi = \pi$ rad. Inserting $\phi = \pi$ into Equation (3.14) gives:

$$y = y_\pi^\pm = \pm\sqrt{\frac{2b_1b_2(a_2 - a_1) - b_1s_2 - b_2s_1}{b_1 + b_2}} \tag{3.15}$$

Hence, when $P(\psi)$ vanishes at $\psi = -1$, two solutions are obtained: $(\phi = \pi, y = y_\pi^+)$ and $(\phi = \pi, y = y_\pi^-)$, where y_π^+ and y_π^- denote the results of choosing the positive and negative signs of the radical in Equation (3.15), respectively. Note that, although ϕ is real, y will be imaginary if the radicand in Equation (3.15) is negative. If the radicand is positive, these two solutions to forward kinematics are real and distinct.

If $P(\psi)$ has a double zero at $\psi = -1$, the situation is the same as in Figure 3.2b and two double solutions are obtained, as shown in Figure 3.3 (right). However, it will be shown next that these two double solutions actually correspond to a single quadruple solution. Since the joint coordinates belong to the π-hyperbola, $P(\psi)$ has a zero at $\psi = -1$. To make that zero double, the derivative $P'(\psi)$ must also vanish at $\psi = -1$, which translates into the following condition:

$$3p_3 - 2p_2 + p_1 = 0 \tag{3.16}$$

Substituting the value of l_1^2 of Equation (3.11) into Equation (3.16) yields:

$$(b_1 + b_2)^2((a_2 - b_2)^2 - l_2^2) = 0 \tag{3.17}$$

which is satisfied only if $l_2 = |a_2 - b_2|$. The substitution of this value of l_2 into Equation (3.11) yields $l_1 = |a_1 + b_1|$. For these values of l_1 and l_2, Equation (3.15) gives $y = 0$. Hence: $y_\pi^+ = y_\pi^- = 0$, and the two double solutions of Figure 3.3 (right) coalesce to a single quadruple solution $(\phi = \pi, y = 0)$. The point of the π-hyperbola where this quadruple solution occurs will be denoted by $\lambda_\pi = (|a_1 + b_1|, |a_2 - b_2|)$.

3.1.1.3 Case 3: intersection at $\psi = 1$

This case admits the same analysis as the previous one. Imposing $P(1) = 0$ yields the following equation:

$$l_1^2 = l_2^2 + (a_1 - b_1)^2 - (a_2 + b_2)^2 \tag{3.18}$$

which defines another hyperbola. If the joint coordinates belong to it, $P(\psi)$ has a zero at $\psi = 1$, which corresponds to a unique angle: $\phi = 0$. Hence, from now on, the curve defined by Equation (3.18) will be called the *0-hyperbola*. To compute y, we use Equation (3.14) again since Equation (3.9) is not valid, obtaining two values:

$$y = y_0^\pm = \pm\sqrt{\frac{2b_1b_2(a_1 - a_2) - b_1s_2 - b_2s_1}{b_1 + b_2}} \tag{3.19}$$

Again, two real and different solutions $(\phi = 0, y = y_0^+)$ and $(\phi = 0, y = y_0^-)$ will be obtained if the radicand in Equation (3.19) is positive. Imposing $P'(1) = 0$ to make these two solutions double:

$$3p_3 + 2p_2 + p_1 = 0 \tag{3.20}$$

Substituting l_1^2 from Equation (3.18) into Equation (3.20) results in the following equation:

$$(b_1 + b_2)^2((a_2 + b_2)^2 - l_2^2) = 0 \tag{3.21}$$

The previous equation holds true only if $l_2 = |a_2 + b_2|$, which substituted into Equation (3.18) gives: $l_1 = |a_1 - b_1|$. Finally, inserting these values of l_1 and l_2 into Equation (3.19) yields $y_0^+ = y_0^- = 0$. Therefore, the two solutions to forward kinematics at $\psi = 1$ do not become two different double solutions but a single quadruple solution $(\phi = 0, y = 0)$. The point of the 0-hyperbola where this quadruple solution occurs will be denoted by $\lambda_0 = (|a_1 - b_1|, |a_2 + b_2|)$.

3.1.2 Singularities

Singularities are special configurations at which a parallel mechanism gains or loses instantaneous degrees of freedom (DOF). Singularities of parallel mechanisms can be classified into two main types [65]: serial (or Type-1) singularities, and parallel (or Type-2) singularities. The effects of these types of singularities are:

- **Loss of dexterity:** Serial singularities occur when $\det(\mathbf{J}_{joints}) = 0$, where $\mathbf{J}_{joints}$ denotes the Jacobian matrix of partial derivatives of the kinematic restrictions of the mechanism with respect to the active joint coordinates. At serial singularities, the mechanism loses instantaneous degrees of freedom: velocities along some directions are not possible.

- **Loss of control:** Parallel singularities occur when $\det(\mathbf{J}_{pose}) = 0$, where $\mathbf{J}_{pose}$ denotes the Jacobian matrix of partial derivatives of the kinematic restrictions of the mechanism with respect to the variables that parameterize the pose of its mobile platform. At parallel singularities, the mechanism gains instantaneous degrees of freedom: the mobile platform of the robot may be locally movable along some directions even if all actuators are locked (the mechanism loses its stiffness).

At parallel singularities, the velocity of the mobile platform of the mechanism is not uniquely determined by the feasible velocities of the active joints, so one cannot completely control the motion of the mobile platform using the actuators. Furthermore, when a parallel singularity is approached in the active joint space, at least two solutions of the forward kinematic problem coalesce.

In order to take full advantage of its complete workspace, a parallel mechanism typically needs to reconfigure itself and switch between different solutions of the forward kinematic problem since a single assembly mode cannot cover the whole workspace (see Section 4.4.7), which may require the mechanism to cross parallel singularities. This is complicated because, as we have just said, the motion of the mobile platform is not completely controllable when crossing parallel singularities. However, for many parallel mechanisms, it is also possible to switch between different solutions of the forward kinematic problem without crossing parallel singularities (i.e., keeping under control the motion of the mobile platform at all times), as we will discuss later in Section 3.2.

In the remaining of this chapter, we will focus only on parallel singularities. Therefore, from now on, when we speak about "singularities", it will be understood that we are referring to "parallel singularities".

For now, let us analyze the singularities of the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b. To obtain the singularities of this mechanism, first we compute the Jacobian matrix of the left-hand side (LHS) of Equations (3.1) and (3.2) with respect to the parameters that define the pose of the mobile platform, i.e., ϕ and y:

$$\mathbf{J}_{pose} = \begin{bmatrix} \frac{\partial \text{LHS of Equation (3.1)}}{\partial \phi} & \frac{\partial \text{LHS of Equation (3.1)}}{\partial y} \\ \frac{\partial \text{LHS of Equation (3.2)}}{\partial \phi} & \frac{\partial \text{LHS of Equation (3.2)}}{\partial y} \end{bmatrix} =$$

$$= 2 \begin{bmatrix} b_1(y\cos\phi + a_1 \sin\phi) & b_1 \sin\phi + y \\ -b_2(y\cos\phi + a_2\sin\phi) & -b_2\sin\phi + y \end{bmatrix} \tag{3.22}$$

Singularities occur when the determinant of the previous matrix vanishes, which yields the following condition:

$$\det(\mathbf{J}_{pose}) = 4[y^2(b_1 + b_2)\cos\phi + b_1 b_2(a_2 - a_1)(\sin\phi)^2 + y(a_1 b_1 + a_2 b_2)\sin\phi] = 0 \tag{3.23}$$

For given design parameters (a_1, a_2, b_1, b_2), the previous equation typically defines a set of disjoint or self-intersecting curves in the (ϕ, y) plane. These curves are the

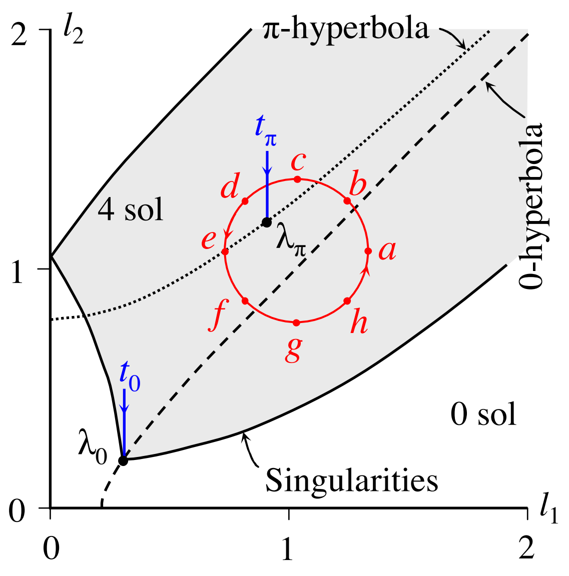


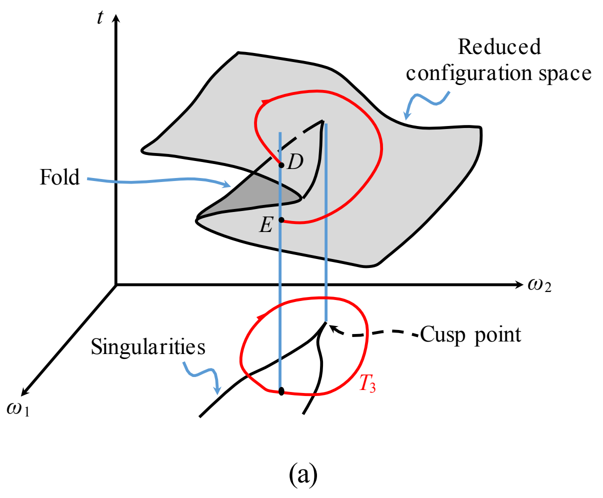
Figure 3.4: Joint space of an analytic 2RPR-PR robot

singularity locus, i.e., the geometric locus of all the parallel singularities of the mechanism. Equation (3.23) is quadratic in y and quartic in $\tan(\phi/2)$, which allows us to use a simple algorithm to plot the singularity curves in the (ϕ, y) plane: one only needs to sweep ϕ (or y) and compute analytically y (respectively, ϕ) from Equation (3.23), following a procedure similar to the one proposed in Algorithm 4 for plotting one-dimensional self-motion manifolds. Once we have plotted the singularity curves in the (ϕ, y) plane, we can numerically map these curves to the active joint space (l_1, l_2) using Equations (3.1) and (3.2). Since parallel mechanisms are controlled in the active joint space, it is important to know the location of parallel singularities in that space.

Let us analyze the singularity locus of an analytic 2RPR-PR parallel mechanism with the following example geometry: $a_1 = 0.3$, $a_2 = -0.7$, $b_1 = 0.6$, and $b_2 = 0.5$ (arbitrary units). The singularity locus of such a mechanism is shown in Figure 3.4 in black continuous line. In this example, the singularity locus divides the joint space into two regions with zero (white region) and four (gray region) real solutions to the forward kinematics in each region, which agrees with [87]. The π- and 0-hyperbolas are represented in dotted and dashed lines, respectively, along with their points λ_π and λ_0. Some trajectories are also shown, but these will be explained in the next section 3.2.

3.2 Nonsingular Transitions

It is well known that many parallel robots and mechanisms can switch between different solutions to the forward kinematics (or *assembly modes*) without crossing singularities, which can be exploited to enlarge the workspace [107]. This ability was observed first in planar 3-RPR robots [78], and it was related to the existence of points in the joint space where three solutions to the forward kinematics coalesce [113]. These points appear



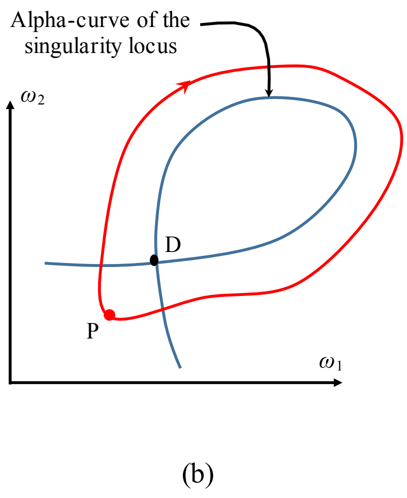


Figure 3.5: (a) Encircling cusps of the singularity curves in the active joint space of a 2-DOF parallel mechanism allows the mechanism to switch its assembly mode without crossing singularities. This can be easily understood in a *reduced configuration space* of a 2-DOF mechanism, which is the projection of the configuration space surface (which is the common solution set of all kinematic constraints of the mechanism) on a three dimensional space (ω_1, ω_2, t), where (ω_1, ω_2) are active joint coordinates and t is a pose parameter (i.e., a position or orientation coordinate) of the mobile platform. (b) Similarly, encircling a so-called α-loop can also produce nonsingular transitions.

as cusps of the singularity curves in planar slices of the joint space (see Figure 3.5a), and encircling them in two [207] or three dimensions [186] can produce nonsingular transitions. The ability to execute nonsingular transitions of the 3-R$\underline{\text{P}}$R robot has been extensively studied, see for example [206, 121, 36, 37].

Nonsingular transitions do not occur only in 3-R$\underline{\text{P}}$R robots. They have been observed also in spatial robots with three [77] and six [26] degrees of freedom (DOF), and in planar 2-DOF robots with a passive leg [43]. Moreover, encircling cusps is not the only method to achieve nonsingular transitions: for 3-R$\underline{\text{P}}$R and 3-$\underline{\text{P}}$RR robots these transitions can also occur when encircling α-curves (loops of the singularity curves) in planar sections of the joint space (see Figure 3.5b), which look like helical ramps when visualized in the reduced configuration space [15, 108]. Normally, the α-curves appear jointly with cusps, but [39] presents a 2-DOF robot that is able to execute nonsingular transitions by encircling α-curves without any cusp.

Parallel robots whose characteristic polynomial is of fourth degree or less are called *analytic* because their forward kinematics can be solved in closed form [87]. They are important in robotics since their mathematical analysis is simpler, which permits one to use them as examples or benchmarks to study concepts, test algorithms, and solve problems difficult to handle in the general case, providing useful information for the later analysis of more complex robots.

The forward kinematic problem of many analytic robots is too simple to exhibit the phenomenon of nonsingular transitions, but some examples can be found. For instance, the R$\underline{\text{P}}$R-2P$\underline{\text{R}}$R robot has been used to illustrate the analytic calculation of the curves of cusps [72], which can be encircled in the 3D joint space to perform nonsingular transitions. The characteristic surfaces and uniqueness domains of this robot have also been studied [28], as well as the relationship between its design parameters and the number of cusps [120].

Four classes of analytic 3-R$\underline{\text{P}}$R robots were studied in [86], but it is widely accepted that their forward kinematic problems cannot have triple solutions and, as a consequence, that they cannot perform nonsingular transitions. This is because the characteristic polynomial of three of these classes is quadratic. The remaining class consists of 3-R$\underline{\text{P}}$R robots with non-similar flat base and platform, and although its characteristic polynomial is cubic in $\cos\phi$ (being ϕ the orientation angle of the mobile platform), it has been shown that it has two roots in $(-1,1)$ at most [66]. On the contrary, the analytic 3-R$\underline{\text{P}}$R robot with congruent platform and base [196] can change different between solutions to the forward kinematics without crossing singularities. The points of the three-dimensional joint space of this robot that can be encircled to execute nonsingular transitions were calculated analytically in [187].

In the following subsection 3.2.1, we will demonstrate that the analytic 2R$\underline{\text{P}}$R-PR parallel mechanism of Figure 3.1b can perform nonsingular transitions between different assembly modes, in the *apparent* absence of both cusps and α-curves. As it will be shown next, this analytic mechanism is one of the simplest and least intuitive examples of mechanisms with the ability to perform nonsingular transtisions found in the literature, and suggests that the assembly modes with multiplicity higher than three should also be analyzed when assessing the ability to perform nonsingular transition in a mechanism.

3.2.1 Nonsingular Transitions in Analytic 2R$\underline{\text{P}}$R-PR Mechanisms

In section 3.1.2, we have studied the singularity locus of a particular design of the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b. This particular design was defined by the following geometric parameters: $a_1 = 0.3$, $a_2 = -0.7$, $b_1 = 0.6$, and $b_2 = 0.5$ (arbitrary units), and the corresponding singularity locus was shown in Figure 3.4. As this Figure 3.4 shows, the singularity curves apparently lack both the α-curves and cusp points that make nonsingular transitions possible in other parallel mechanisms. The absence of *ordinary* or *generic* cusps can be demonstrated more formally by showing that the characteristic polynomial $P(\psi)$ of Equation (3.3) cannot have triple zeros, as it will be shown next.

In order to find triple zeros of polynomial $P(\psi)$, we must solve the system of equations (3.10). This system can be solved as follows: first, we solve ψ from the third equation of the system (3.10) (which is linear in ψ), and substitute the obtained solution into the second equation of the system (3.10). This yields the equation of a parabola in $x_1 = l_1^2$ and $x_2 = l_2^2$:

$$\frac{11x_1^2}{18} + \frac{22x_1x_2}{15} + \frac{22x_2^2}{25} - \frac{401x_1}{450} - \frac{962x_2}{375} + \frac{1904263}{495000} = 0 \qquad (3.24)$$

This parabola can be parameterized as follows:

$$\begin{cases} x_1 = -\frac{61}{51}\left(\omega - \frac{255\sqrt{61}}{7442}\right)^2 - \frac{326273}{205700} \\ x_2 = \frac{305}{306}\left(\omega + \frac{918\sqrt{61}}{18605}\right)^2 + \frac{96859}{41140} \end{cases}, \quad \omega \in \mathbb{R} \tag{3.25}$$

According to Equation (3.25), x_1 is always strictly negative, thus no real value of $l_1 = \sqrt{x_1}$ satisfies Equation (3.24) and $P(\psi)$ cannot have triple zeros. Hence, the singularity locus in the joint space will lack ordinary cusps.

In Section 3.1.1, it has been shown that the joint space of the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b always has two special points λ_π and λ_0, where the forward kinematic problem admits quadruple solutions. These two special points can be observed in Figure 3.4: the point λ_0 lies on the singularity curve that separates the regions with zero (white region) and four (gray region) real solutions to the forward kinematic problem, whereas the point λ_π lies inside the gray region. Although we cannot observe cusps in the singularity curves of Figure 3.4, which may suggest that this mechanism cannot perform nonsingular transitions, it will be demonstrated next that these transitions are indeed possible in general for this parallel mechanism by encircling the special points λ_π or λ_0.

To demonstrate that nonsingular transitions are possible in the analyzed mechanism, we will analyze next the evolution of the solutions to the forward kinematics when describing some trajectories in the joint space. Given a pair (l_1, l_2), solving Equation (3.3) yields three values of ψ, each of which results in two angles ϕ according to Equation (3.8). Since ψ may be outside the interval $[-1, 1]$ or even be complex, ϕ may be complex. For a general complex ψ, ϕ is obtained as follows (see [21], Chapter 3):

$$\phi = \text{acos}(\psi) = \arg\left(\psi + i\sqrt{1-\psi^2}\right) - i\ln\left|\psi + i\sqrt{1-\psi^2}\right| \tag{3.26}$$

where i is the imaginary unit. Since ϕ can be complex, it may be useful to represent the trajectories followed by ϕ in the complex plane when (l_1, l_2) describe some trajectories in the joint space. For example, Figure 3.6a shows the graphical representation of different values of ϕ in the complex plane, where the real Re(ϕ) and imaginary Im(ϕ) parts of ϕ are represented in rectangular axes. The solution σ_1 is real, whereas σ_2 and σ_3 are complex conjugates.

According to Equation (3.26), Re(ϕ) is the argument of a complex number, i.e., an angle undetermined by an integer multiple of 2π. Thus, angles must be restricted to an interval with length 2π, but this produces discontinuous trajectories if a rectangular representation of ϕ is used. For example, in Figure 3.6a the real part of ϕ is restricted to the interval $[-\pi, \pi]$, and a trajectory connecting the solutions σ_4 and σ_5 is shown. Note that the trajectory is discontinuous because when a point crosses the vertical line at Re$(\phi) = \pi$, it reappears at Re$(\phi) = -\pi$ since angles are wrapped to the interval $[-\pi, \pi]$.

In what follows, it will be more convenient to use a representation that avoids discontinuities. The solutions will be represented using modified polar coordinates, as

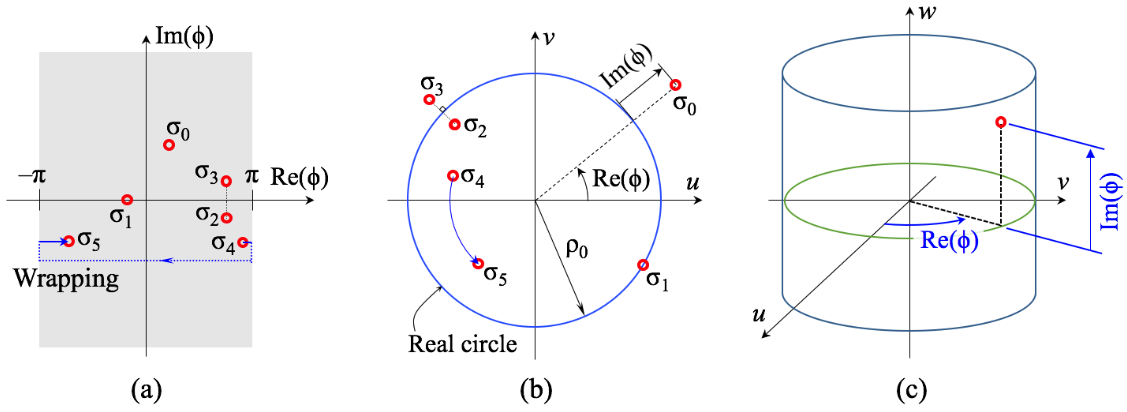


Figure 3.6: Rectangular (a), polar (b), and cylindrical (c) representations of ϕ

shown in Figure 3.6b. An arbitrary solution σ_0 will be represented as a point with coordinates (u, v):

$$u = \rho \cos\left(\mathrm{Re}(\phi)\right), \quad v = \rho \sin\left(\mathrm{Re}(\phi)\right) \tag{3.27}$$

where $\rho = \rho_0 + \mathrm{Im}(\phi)$. In this representation, the circle of radius ρ_0 centered at the origin represents the real solutions (e.g., σ_1). Complex solutions are radially separated from this "real circle" a distance equal to the imaginary part, and complex conjugates are placed symmetrically with respect to the circle (e.g., σ_2 and σ_3). Since one solution in each conjugate pair has negative imaginary part, ρ_0 must be sufficiently high to guarantee $\rho > 0$. For all the trajectories that will be analyzed in this chapter, $\rho_0 = 4$ will suffice.

Alternatively, we can also represent the complex solutions as points on a cylinder with fixed radius using cylindrical coordinates (see Figure 3.6c), taking $\mathrm{Re}(\phi)$ as the angular coordinate and $\mathrm{Im}(\phi)$ (which can be positive or negative) as the height or axial coordinate. This representation will be used in the simulators presented in the next chapter. However, a cylindrical representation requires three-dimensional plots and is more suitable for interactive visualization in a 3D plotting software on a computer, in which the user can rotate the cylinder to properly visualize all the solutions from different perspectives. For its part, the polar representation of Figure 3.6b is a two-dimensional plot, which is more suitable for planar representations on a sheet of paper, like the present document. Therefore, in this section, we will prefer to analyze the trajectories using the 2D polar representation explained in Figure 3.6b, instead of the 3D cylindrical representation of Figure 3.6c.

Using the proposed polar representation, some trajectories will be analyzed next. Figure 3.4 shows a vertical trajectory t_π approaching λ_π in the joint space. Figure 3.7 (top) shows three polar plots with the solutions σ_i $(i = 1, \ldots, 6)$ to the forward kinematics at three points of t_π. To sort the solutions, the labels σ_i are assigned arbitrarily at the beginning of the trajectory. Then, the trajectory is approximated by a set of discrete points (l_1^k, l_2^k) $(k = 1, \ldots, N)$ separated by small steps. Out of singularities, small variations in the joint coordinates produce small variations of the position and orientation of the platform, which in general (when complex solutions are

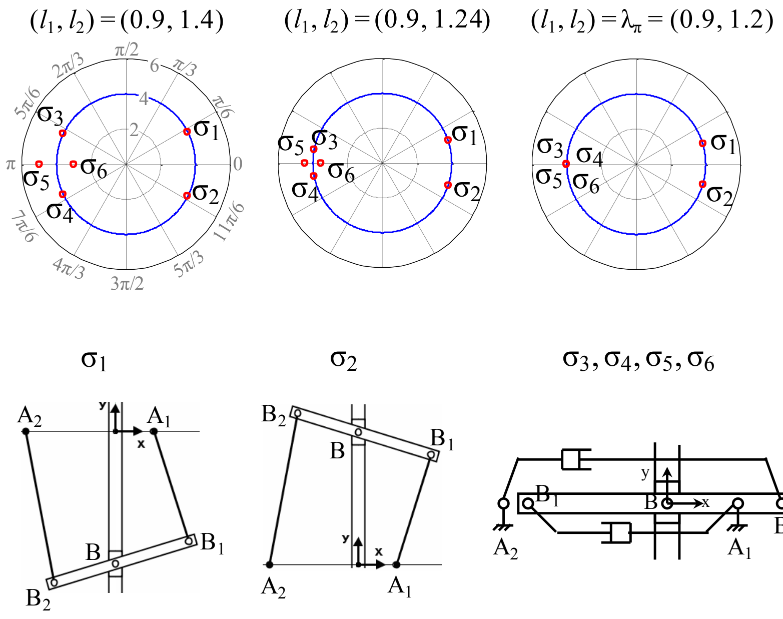

$(l_1, l_2) = (0.9, 1.4)$
$(l_1, l_2) = (0.9, 1.24)$
$(l_1, l_2) = \lambda_\pi = (0.9, 1.2)$
σ_1
σ_2
$\sigma_3, \sigma_4, \sigma_5, \sigma_6$

Figure 3.7: Polar plots of the solutions to the forward kinematics when approaching the point λ_π along t_π (top row). Solutions at λ_π (bottom row).

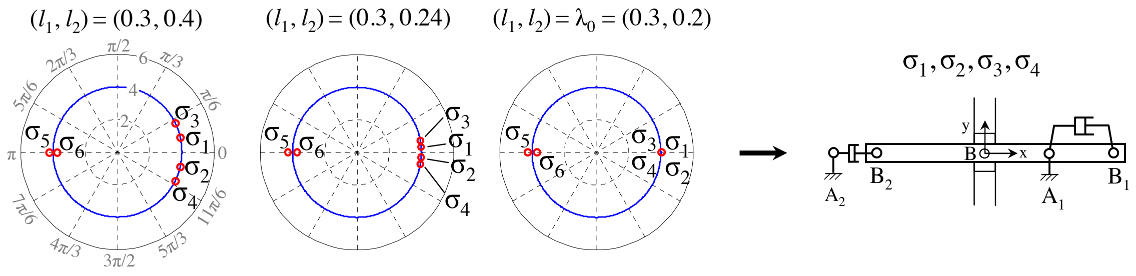

$(l_1, l_2) = (0.3, 0.4)$
$(l_1, l_2) = (0.3, 0.24)$
$(l_1, l_2) = \lambda_0 = (0.3, 0.2)$
$\sigma_1, \sigma_2, \sigma_3, \sigma_4$

Figure 3.8: Polar plots of the solutions to the forward kinematics when approaching the point λ_0 along t_0 (left). Real quadruple solution at λ_0 (right).

obtained) can be encoded in the 4-tuple $\xi = (u, v, \mathrm{Re}(y), \mathrm{Im}(y))$. Thus, the rule to sort the solutions along the trajectories consists in identifying each solution obtained for (l_1^{k+1}, l_2^{k+1}) with the label of the solution obtained for (l_1^k, l_2^k) that has the closest position and orientation ξ, measured using the Euclidean distance.

Using this rule, Figure 3.7 (top) shows the coalescence of the four solutions σ_3, σ_4, σ_5, and σ_6 at $\phi = \pi$ as the joint coordinates approach $\lambda_\pi = (0.9, 1.2)$. For $(l_1, l_2) = \lambda_\pi$, the forward kinematic problem has three different real solutions: σ_1 and σ_2, which are simple, and σ_3, which is quadruple. Figure 3.7 (bottom) shows the configuration of the robot for each of these solutions.

Similarly, Figure 3.8 (left) shows three polar plots of the solutions to the forward kinematics at different points of the vertical trajectory t_0 that approaches $\lambda_0 = (0.3, 0.2)$, shown in Figure 3.4. In this case, the coalescence of four solutions occurs at $\phi = 0$, and the remaining solutions (σ_5 and σ_6) are complex. The configuration of the robot for the quadruple solution is shown in Figure 3.8 (right).

Finally, after the quadruple solutions at λ_π and λ_0 have been illustrated, it will be shown that encircling these points can produce nonsingular transitions. According to Figure 3.4, λ_π can be encircled for the chosen geometry but λ_0 lies on the limit of the feasible region (gray area) of the joint space. Trajectories enclosing λ_0 are not possible in practice since they cross singularities and invade the region where the robot cannot be assembled. This situation changes for other geometries. For example, if the positions of A$_1$ and A$_2$ are swapped, λ_0 lies inside the feasible region and can be encircled without crossing singularities, whereas λ_π shifts to the boundary of this region.

Figure 3.4 shows a circular trajectory $abcdefgha$ that encloses λ_π. The trajectory is defined by the following parametric equations:

$$\begin{cases} l_1 = 1.025 + 0.3\cos\theta \\ l_2 = 1.075 + 0.3\sin\theta \end{cases}, \quad 0 \leq \theta \leq 2\pi \tag{3.28}$$

Figure 3.9 shows some polar plots with the evolution of the six solutions to the forward kinematics along this trajectory. Between a and b, the 0-hyperbola is crossed, which produces the coincidence of σ_1 and σ_2 at $\phi = 0$. Between b and c, the trajectory crosses the π-hyperbola, when σ_3 and σ_4 meet at $\phi = \pi$. As the trajectory evolves from c to e, the solutions σ_5 and σ_6 approach the real circle and coincide at $\phi = \pi$ for $\theta = \pi$, which occurs when the trajectory crosses again the π-hyperbola at e. Although σ_5 and σ_6 share the angle $\phi = \pi$, each solution has a different (imaginary) value for y obtained from Equation (3.15). Between f and g, the 0-hyperbola is crossed once more, with σ_1 and σ_2 meeting again at $\phi = 0$. Finally, the trajectory continues from g to a without crossing any hyperbola and ends at a. According to Figure 3.9, the solutions at the beginning ($\theta = 0$) and at the end ($\theta = 2\pi$) coincide, but σ_3 and σ_5 have interchanged their positions with σ_4 and σ_6, respectively. Thus, a change of assembly mode has occurred.

It is important to remark that, although crossing the hyperbolas produces the coincidence of solutions on the real circle, these coincidences are not singularities in general. In these crossings, two solutions share the same ϕ but have different sign for y, which is computed from Equation (3.14). These crossings become singularities only if $y = 0$, and this only happens at λ_0 and λ_π.

Figure 3.10 shows the trajectory followed by the solution σ_4 in the reduced configuration space (l_1, l_2, ϕ), where the nonsingular transition can be easily understood. The configuration surface intersects itself along the curve η_π, whose projection onto the joint space is the part of the π-hyperbola that lies to the right of λ_π. The projection of the point Λ_π onto the joint space is λ_π.

Finally, Figure 3.11 shows the evolution of the configuration of the robot for the solution σ_4 along the circular trajectory. This figure clearly shows the nonsingular change of the solution to the forward kinematics.

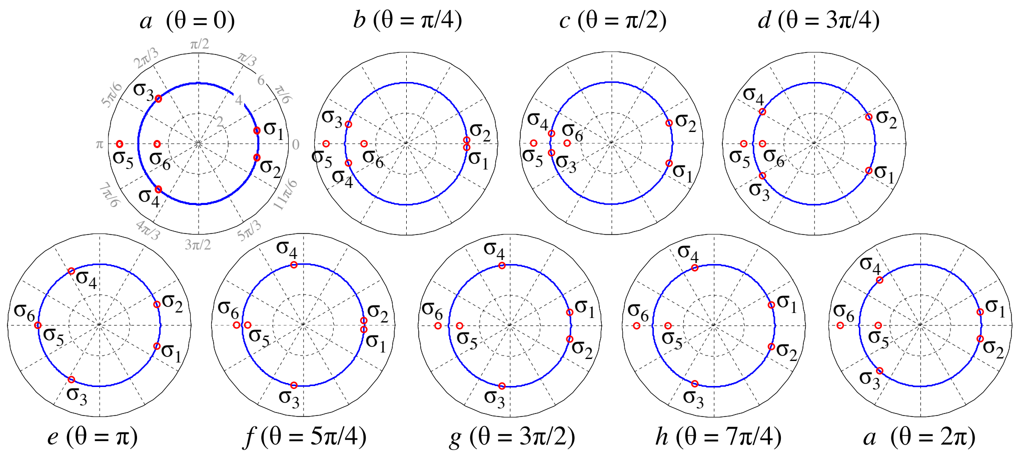


Figure 3.9: Polar plots of the evolution of the six solutions σ_i ($i = 1, \ldots, 6$) to the forward kinematics along the circular trajectory *abcdefgha* in the joint space, shown in Figure 3.4.

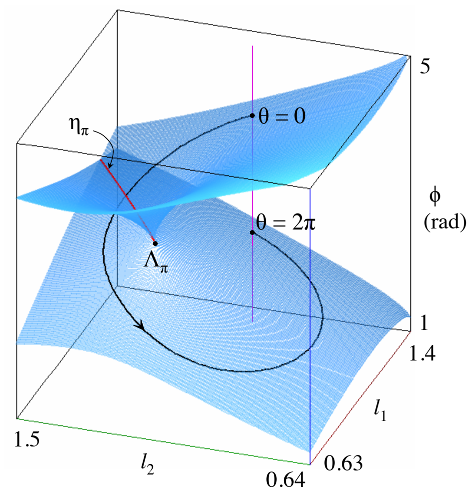


Figure 3.10: Trajectory of the solution σ_4 in the reduced configuration space

3.3 Practical Considerations and Implications for the HyReCRo Climbing Robot

The previous section has shown that the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b, which is used in the legs of the HyReCRo climbing robot studied in this thesis, can switch between different configurations of the forward kinematic problem without crossing singularities. Therefore, the next question is: does this phenomenon affect the operation of the studied climbing robot in any way? In this section, we will show that, due to joint limits, both singular a nonsingular transitions will be impossible for the HyReCRo robot, which notably simplifies its kinematic analysis and operation.

First of all, as we will see in chapter 5, the analytic 2R$\underline{\text{P}}$R-PR parallel mechanisms used in the HyReCRo climbing robot are symmetric, i.e.: $a_1 = b$, $a_2 = -b$, $b_1 = b_2 = p$, where $2b$ is the size of the base of the mechanism, whereas $2p$ is the size of its mobile platform (see Figure 3.12a). The precise design used in the prototype of the HyReCRo

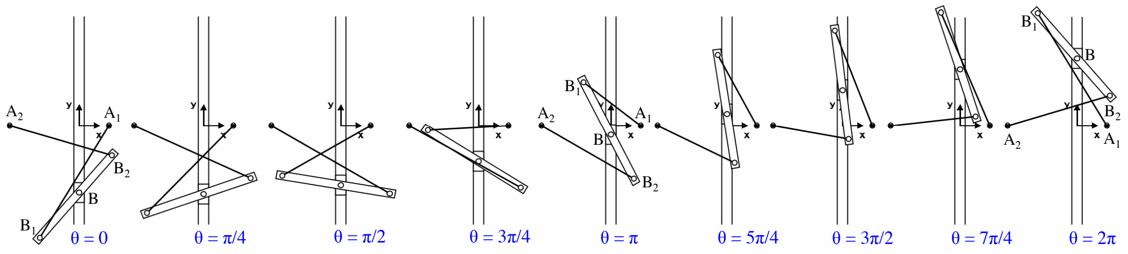


Figure 3.11: Configuration of the analytic 2R$\underline{\text{P}}$R-PR robot for the solution σ_4 along the circular trajectory *abcdefgha* of Figure 3.4

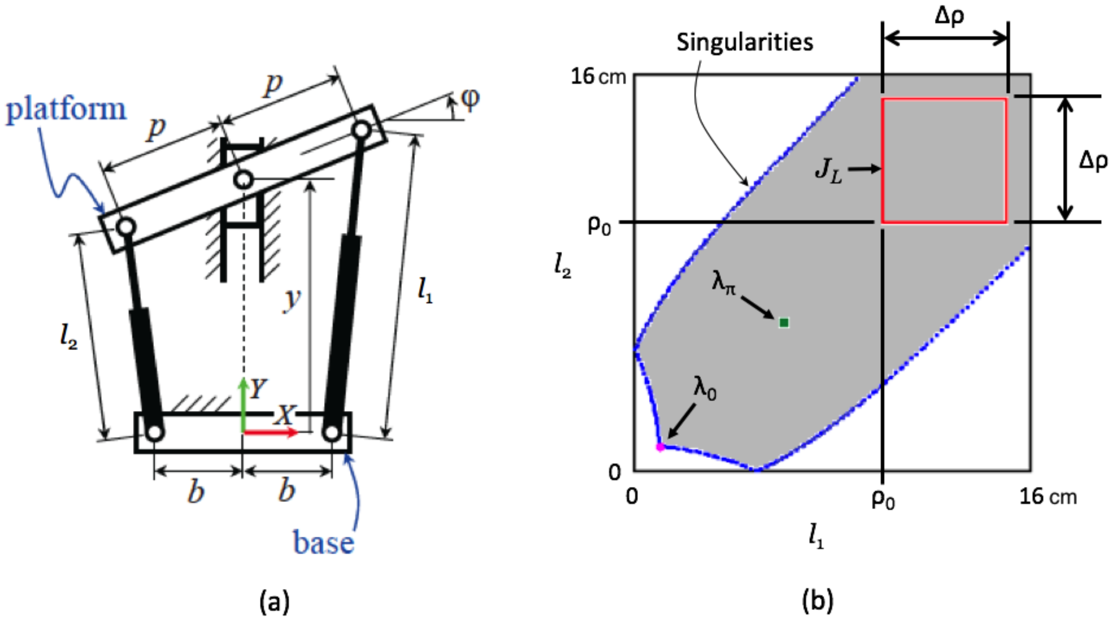


Figure 3.12: (a) A symmetric analytic 2R$\underline{\text{P}}$R-PR parallel mechanism, like those used in the legs of the HyReCRo climbing robot. (b) Active joint space of a mechanism with $b = 2.5$ cm and $p = 3.15$ cm. J_L is the square region inside the joint limits.

robot that will be presented in Chapter 8 is given by the following values: $b = 2.5$ cm and $p = 3.15$ cm.

Besides, the lengths l_1 and l_2 (controlled variables of the mechanism) will have to be between a minimum value ρ_0 and a maximum value $\rho_0 + \Delta\rho$, due to obvious physical limitations of the linear actuators to be used in the implementation of the mentioned prototype (the linear actuators cannot be arbitrarily short or long!). ρ_0 is the length of the linear actuators when they are completely retracted, whereas $\Delta\rho$ is their stroke. In the prototype that will be presented in Chapter 8, we will use linear electric actuators that have the following minimum length and stroke: $\rho_0 = 10$ cm and $\Delta\rho = 5$ cm.

The active joint space (l_1, l_2) of a symmetric analytic 2R$\underline{\text{P}}$R-PR mechanism with $b = 2.5$ cm and $p = 3.15$ cm is represented in Figure 3.12b. Like in Figure 3.4, the singularity locus divides the (l_1, l_2) plane into two regions: a gray region where the forward kinematic problem has 4 real solutions, and a white region with 0 real solutions. Figure 3.12b also represents the special points λ_π and λ_0, which, as we have seen in the previous Section 3.2.1, can be encircled to change the assembly mode of the mechanism without crossing singularities. Note that the singularity locus of the mechanism of Figure 3.12b has essentially the same structure as the example of Figure 3.4. Thus, what we observed in the example of Section 3.2.1 (regarding the ability to perform nonsingular transitions) is perfectly extensible to the mechanisms used in the legs of the HyReCRo robot (i.e., if the mechanism encloses λ_π, it can switch between different assembly modes).

However, unlike in the example of Section 3.2.1, now we are considering joint limits: Figure 3.12b represents a square J_L defined as: $J_L = \{(l_1, l_2) : \rho_0 \leq l_1 \leq \rho_0 + \Delta\rho, \rho_0 \leq l_2 \leq \rho_0 + \Delta\rho\}$. Due to joint limits, the mechanism will be constrained to move inside J_L. This has two important consequences: 1) on the one hand, the mechanism will be unable to enclose the special point λ_π, since it is outside J_L. 2) On the other hand, the square J_L does not intersect the singularity locus, which means that the mechanism will never encounter singularities and lose the control of its mobile platform.

In other words: joint limits will impede both singular and nonsingular transitions in the mechanisms of the legs of the HyReCRo robot. The mechanism will always operate in the assembly mode in which it was originally "assembled", without the possibility to switch to other assembly modes. This will greatly simplify the kinematic analysis of the complete climbing robot in the next chapters.

Next, let us analyze the reduced configuration space of the symmetric mechanism of Figure 3.12 considering joint limits, in order to identify more clearly the different assembly modes of this mechanism. In Section 3.2.1, we analyzed the nonsingular transitions of the analytic 2R$\underline{P}$R-PR mechanism in the reduced configuration space (l_1, l_2, φ) (i.e., we focused on the orientation of the mobile platform). However, in this section, it will be more convenient to analyze the reduced configuration space (l_1, l_2, y) (i.e., we will focus on the position of the mobile platform) because, as we will see next, the position y allows us to clearly distinguish and identify the different assembly modes under joint limits.

If we consider joint limits, then we only need to represent the reduced configuration space for values of the joint coordinates inside the square region J_L, which yields the surfaces shown in Figure 3.13. Note that, above the region J_L of the (l_1, l_2) plane, the reduced configuration space (l_1, l_2, y) is composed of four non-intersecting surfaces or sheets, which are identified in Figure 3.13 as $H+$, $H-$, $X+$, and $X-$. Each sheet corresponds to a different assembly mode. As Figure 3.13 illustrates, the linear actuators intersect for assembly modes $X+$ and $X-$, whereas they do not intersect for assembly modes $H+$ and $H-$. Actually, for the HyReCRo robot, the only feasible assembly mode is $H+$, since the other three assembly modes imply collisions between different parts of the robot (not necessarily between the linear actuators, which can be easily avoided by placing these actuators in different parallel planes). Therefore, the

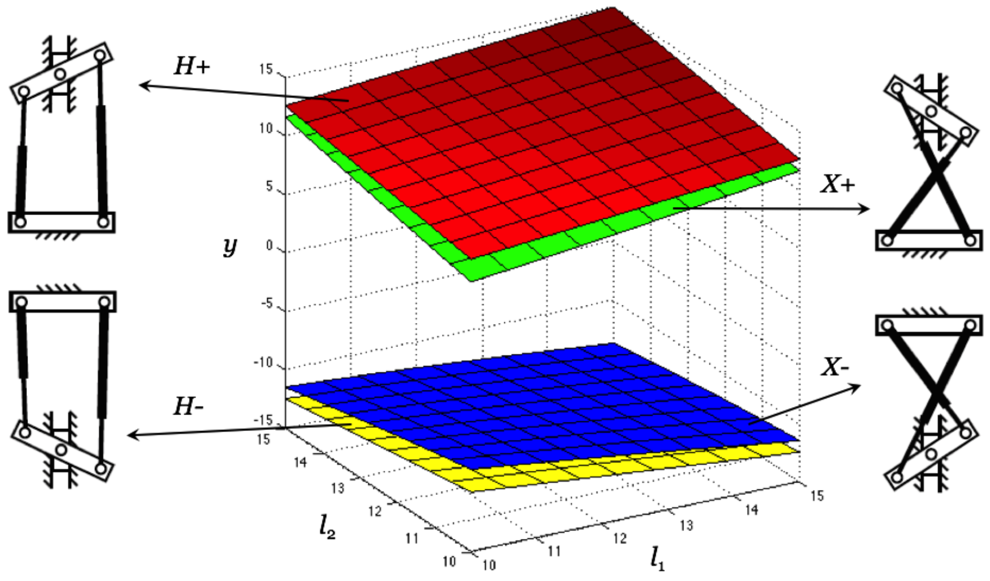


Figure 3.13: Reduced configuration space (l_1, l_2, y) of the parallel mechanism of Figure 3.12, for values of l_1 and l_2 inside the square region J_L delimited by the joint limits. Each surface corresponds to a different assembly mode. The typical posture of the mechanism in each assembly mode is represented.

parallel mechanisms used in the legs of the HyReCRo robot will always move along sheet $H+$.

Finally, to conclude this section, let us analyze the following practical problem: for later chapters, we will need to use the solution of the forward kinematic problem of the parallel mechanisms of Figure 3.12a. This problem has four different real solutions, but only one of them $(H+)$ is feasible in practice, as we have just said. How can we know which one is the right solution? Figure 3.13 provides a simple answer to this question: the valid solution $(H+)$ is always the one with the highest value for y (the red sheet $H+$ is always above the other three sheets). Thus, after solving the forward kinematic problem of the 2R$\underline{P}$R-PR parallel mechanisms in later chapters, we will retain only the solution with the highest y, discarding the other three solutions.

3.4 3R$\underline{P}$R Parallel Robots

In this section, we will analyze the forward kinematic problem and nonsingular transitions of 3R$\underline{P}$R planar parallel robots. Figure 3.14a shows a general 3R$\underline{P}$R robot: this robot is composed of a fixed triangular platform (or base) ACF, and another mobile triangular platform BDE. The geometries of the fixed and mobile platforms are defined by the parameters $\{c_2, c_3, d_3\}$ and $\{l_1, l_3, \beta\}$, respectively. Both platforms are interconnected through three legs AB, CD, and EF, whose lengths are ρ_1, ρ_2, and ρ_3, respectively. By controlling these lengths, it is possible to control the position and orientation of the mobile platform in the plane. In this section, we will parameterize the position of the mobile platform by the Cartesian coordinates (x, y) of its joint B,

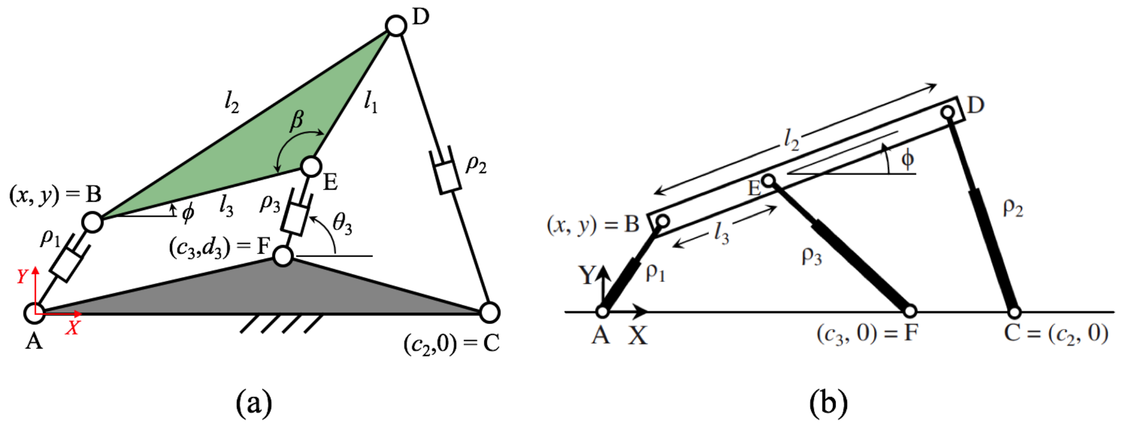


Figure 3.14: General (a) and analytic (b) 3RPR planar parallel robots.

whereas its orientation will be parameterized by the angle ϕ between segments BE and AC.

The general 3RPR planar parallel robot of Figure 3.14a is one of the most widely studied parallel robots in the scientific literature, especially regarding its ability to perform nonsingular transitions (see the references cited in Section 3.2). As we have also indicated in Section 3.2, diverse analytic variants of this general robot have also been studied [86], but earlier studies on the number of real roots of the characteristic polynomials of these variants suggested that such analytic variants could not execute nonsingular transitions. One of these well-known analytic variants is shown in Figure 3.14b: this variant of the 3RPR robot is characterized by the fact that both triangular platforms have degenerated into segments: i.e., joints ACF are aligned ($d_3 = 0$), and joints BDE are also aligned ($\beta = \pi$ rad). The robot of Figure 3.14b assumes that the mobile platform is not a scaled version of the fixed platform, i.e.: $c_3 l_2 - c_2 l_3 \neq 0$. We will also assume that $l_2, l_3 > 0$, $l_2 \neq l_3$, $c_2 \neq 0$, $c_3 \neq 0$, and $c_2 \neq c_3$, which means that different legs cannot be attached to the same point of the fixed or mobile platform.

Why study the 3RPR robot in this chapter? What is the importance of this robot and how is it related to the 2RPR-PR parallel mechanism analyzed in Section 3.1? Firstly, there is an obvious relationship: the general 2RPR-PR parallel mechanism of Figure 3.1a can be obtained by fixing the orientation angle θ_3 of leg EF of the general 3RPR robot of Figure 3.14a, such that the length ρ_3 is no longer a controlled variable but becomes a passive variable y (see Figure 3.1a), which defines the position of the mobile platform along the passive guide OB. However, there is a more important relationship: as we will see next, the forward kinematic problem of the analytic 3RPR robot of Figure 3.14b is essentially the same as the forward kinematic problem of the analytic 2RPR-PR mechanism of Figure 3.1b. Considering the results obtained in Sections 3.1.1 and 3.2.1 relative to the 2RPR-PR mechanism, this similitude suggests that the analytic 3RPR robot of Figure 3.14b may also be able to execute nonsingular transitions, contrary to what earlier studies suggested.

Indeed, in the next subsections we will show that the analytic 3RPR parallel robot of Figure 3.14b can perform nonsingular transitions in a similar way to the analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b. For this purpose, first we will obtain the solutions of its forward kinematic problem, paying special attention to the multiplicity of these solutions.

3.4.1 Forward Kinematics of Analytic 3R$\underline{\text{P}}$R Parallel Robots

The forward kinematic problem of the analytic 3R$\underline{\text{P}}$R robot of Figure 3.14b was solved in [66]. This problem consists in calculating the position and orientation (ϕ, x, y) of the platform given the lengths $\rho_1, \rho_2, \rho_3 > 0$. Next, this problem will be analyzed again paying special attention to the multiplicity of the solutions. The relationship between (ϕ, x, y) and (ρ_1, ρ_2, ρ_3) is:

$$\rho_1^2 = x^2 + y^2 \tag{3.29}$$
$$\rho_2^2 = (x + l_2 \cos\phi - c_2)^2 + (y + l_2 \sin\phi)^2 \tag{3.30}$$
$$\rho_3^2 = (x + l_3 \cos\phi - c_3)^2 + (y + l_3 \sin\phi)^2 \tag{3.31}$$

which states that the distances A-B, F-E, and C-D must be ρ_1, ρ_3, and ρ_2, respectively. Equations (3.30) and (3.31) can be replaced by the following linear combinations, which are independent since $l_{23} = l_2 - l_3 \neq 0$:

$$l_{23}\,[\text{Equation (3.29)}] + l_3\,[\text{Equation (3.30)}] - l_2\,[\text{Equation (3.31)}] \longrightarrow [\text{Equation (3.32)}]$$
$$[\text{Equation (3.30)}] - [\text{Equation (3.31)}] \longrightarrow [\text{Equation (3.33)}]$$

The new equations are:

$$(c_3 l_2 - c_2 l_3)x + l_2 l_3 c_{32} \cos\phi = \frac{l_2(s_1 - s_3) + l_3(s_2 - s_1)}{2} \tag{3.32}$$

$$2y l_{23} \sin\phi + 2(c_{32} + l_{23}\cos\phi)x = s_2 - s_3 + 2(c_2 l_2 - c_3 l_3)\cos\phi \tag{3.33}$$

where $s_1 = \rho_1^2$, $s_i = \rho_i^2 - c_i^2 - l_i^2$ $(i = 2, 3)$, and $c_{32} = c_3 - c_2$. Solving x from Equation (3.32) yields:

$$x = \frac{-2 l_2 l_3 c_{32} \cos\phi + l_2(s_1 - s_3) + l_3(s_2 - s_1)}{2(c_3 l_2 - c_2 l_3)} \tag{3.34}$$

Next, y is solved from Equation (3.29), obtaining:

$$y = \pm\sqrt{s_1 - x^2} \tag{3.35}$$

Inserting this value of y into Equation (3.33) and squaring the resulting equation to eliminate the radical yields an equation that involves x and ϕ. In that equation, x can be substituted by Equation (3.34), obtaining the following cubic equation in $\psi = \cos\phi$ after arranging the terms:

$$P(\psi) = K_3\psi^3 + K_2\psi^2 + K_1\psi + K_0 = 0 \tag{3.36}$$

where the coefficients have the following expressions:

$$K_3 = 8c_2c_3l_2l_3c_{32}l_{23} \tag{3.37}$$

$$K_2 = 4\left[c_2^2c_3^2l_{23}^2 + l_2^2l_3^2c_{32}^2 + (c_3l_2 + c_2l_3)c_{32}l_{23}s_1 + (c_{32}l_2 - c_2l_{23})c_3l_3s_2 + (l_{23}c_3 - l_3c_{32})c_2l_2s_3\right] \tag{3.38}$$

$$\begin{aligned} K_1 = 2\big[&c_{32}l_{23}s_1^2 - c_3l_3s_2^2 - c_2l_2s_3^2 + (l_3c_{32} - c_3l_{23})s_1s_2 + \\ &+(c_2l_{23} - l_2c_{32})s_1s_3 + (c_3l_2 + c_2l_3)s_2s_3 - 2l_{23}c_{32}(c_2c_3 + l_2l_3)s_1 + \\ &+2(c_3^2c_2l_{23} - l_3^2l_2c_{32})s_2 + 2(l_2^2l_3c_{32} - c_2^2c_3l_{23})s_3\big] \end{aligned} \tag{3.39}$$

$$\begin{aligned} K_0 = {}&(c_{32}^2 + l_{23}^2)s_1^2 + (c_3^2 + l_3^2)s_2^2 + (c_2^2 + l_2^2)s_3^2 + 2(l_3l_{23} - c_3c_{32})s_1s_2 + \\ &+ 2(c_2c_{32} - l_2l_{23})s_1s_3 - 2(c_2c_3 + l_2l_3)s_2s_3 - 4(c_3l_2 - c_2l_3)^2s_1 \end{aligned} \tag{3.40}$$

Since $K_3 \neq 0$, Equation (3.36) always yields three values for $\psi = \cos\phi$. For each value, the x and y coordinates must be computed, completing the solution (ϕ, x, y) to the forward kinematics. Next, we will study the multiplicity of the solutions depending on the type of intersection between the graph of $P(\psi)$ and the horizontal ψ axis in $[-1, 1]$, following the same procedure as in Section 3.1.1.

If Equation (3.36) has a solution at $\psi_0 \in (-1, 1)$, two angles ϕ^+ and ϕ^- are obtained according to Equation (3.8). For each of these angles, x is computed from Equation (3.34), whereas y is calculated from Equation (3.33), obtaining:

$$y = \frac{s_2 - s_3 - 2(l_{23}x - c_2l_2 + c_3l_3)\cos\phi - 2xc_{32}}{2l_{23}\sin\phi} \tag{3.41}$$

which is valid because $\sin\phi \neq 0$. Thus, an intersection in $(-1, 1)$ yields two real solutions to the forward kinematics: (ϕ^+, x^+, y^+) and (ϕ^-, x^-, y^-), where (x^+, y^+) and (x^-, y^-) denote the values obtained using ϕ^+ and ϕ^- respectively. As discussed in Section 3.1.1, these solutions to the forward kinematics can be triple only if ψ_0 is a triple zero of $P(\psi)$, which requires searching for a feasible solution to Equation (3.10), if it exists. Solving Equation (3.10) for a general geometry is not easy, hence this case will be studied later for a particular example geometry. Next, the solutions at $\psi = \pm 1$ are analyzed.

3.4.1.1 $P(\psi)$ vanishes at -1

Imposing $P(-1) = 0$ results in the following condition:

$$\rho_1^2(c_2 - c_3 + l_2 - l_3) + \rho_2^2(c_3 + l_3) - \rho_3^2(c_2 + l_2) = (c_2 - c_3 + l_2 - l_3)(c_3 + l_3)(c_2 + l_2) \tag{3.42}$$

which defines a hyperboloid in the joint space. If the joint coordinates lie on this hyperboloid, $P(\psi)$ vanishes at $\psi = -1$, which translates into $\phi = \pi$. Hence, the surface defined by Equation (3.42) will be called the *π-hyperboloid*. For $\phi = \pi$, Equation (3.34) is used to compute $x = x_\pi$, but y cannot be calculated using Equation (3.41) since the denominator vanishes. Instead, y is computed from Equation (3.35), which gives two values y_π^+ and y_π^- when picking respectively the positive and negative sign of the radical.

If the radicand in Equation (3.35) is positive, this case yields two different real solutions to the forward kinematics: $(\phi = \pi, x = x_\pi, y = y_\pi^+)$ and $(\phi = \pi, x = x_\pi, y = y_\pi^-)$. If $P(\psi)$ is forced to have a double zero at $\psi = -1$, these two solutions do not become two different double solutions but a quadruple one, as we will show next. The condition of double zero requires $P'(-1) = 0$:

$$3K_3 - 2K_2 + K_1 = 0 \tag{3.43}$$

Equation (3.43) defines a surface in the joint space. The intersection of this surface with the π-hyperboloid gives the points of the joint space where $P(\psi)$ has a double zero at $\psi = -1$. It can be shown that the intersection of these two surfaces results in two curves γ_π^+ and γ_π^- defined by the following parametric equations:

$$\gamma_\pi^\pm \equiv \begin{cases} \rho_1 = |c_2 + l_2 \pm \omega| \\ \rho_2 = \omega \\ \rho_3 = |c_2 - c_3 + l_2 - l_3 \pm \omega| \end{cases} \quad , \quad \omega > 0 \tag{3.44}$$

When the joint coordinates belong to any of these curves, the substitution of Equation (3.44) and $\phi = \pi$ into Equations (3.34) and (3.35) yields: $y_\pi^+ = y_\pi^- = 0$. Thus, the two double solutions become a quadruple solution.

3.4.1.2 $P(\psi)$ vanishes at 1

Imposing $P(1) = 0$ yields the following condition:

$$\rho_1^2(c_2-c_3-l_2+l_3)+\rho_2^2(c_3-l_3)-\rho_3^2(c_2-l_2) = (c_2-c_3-l_2+l_3)(c_3-l_3)(c_2-l_2) \tag{3.45}$$

which defines another hyperboloid in the joint space. When the joint coordinates belong to it, $\psi = 1$ is a zero of $P(\psi)$, which results in $\phi = 0$. Thus, the surface defined by Equation (3.45) will be called the 0-*hyperboloid*. For $\phi = 0$, x is computed from Equation (3.34), obtaining $x = x_0$, and y is calculated from Equation (3.35), obtaining two values depending on the sign of the radical: y_0^+ and y_0^-. If the radicand in Equation (3.35) is positive, two real solutions are obtained: $(\phi = 0, x = x_0, y = y_0^+)$ and $(\phi = 0, x = x_0, y = y_0^-)$. Repeating the procedure of the previous subsection, these two solutions are forced to become double by imposing $P'(1) = 0$:

$$3K_3 + 2K_2 + K_1 = 0 \tag{3.46}$$

Next, the surface defined by Equation (3.46) is intersected with the 0-hyperboloid, obtaining the locus of joint coordinates for which $\psi = 1$ is a double zero of $P(\psi)$. The intersection of these two surfaces leads to two curves γ_0^+ and γ_0^- parameterized as follows:

$$\gamma_0^\pm \equiv \begin{cases} \rho_1 = |c_2 - l_2 \pm \omega| \\ \rho_2 = \omega \\ \rho_3 = |c_2 - c_3 - l_2 + l_3 \pm \omega| \end{cases} \quad , \quad \omega > 0 \tag{3.47}$$

Substituting Equation (3.47) and $\phi = 0$ into Equations (3.34) and (3.35) gives $y_0^+ = y_0^- = 0$, i.e., the two double solutions become a single quadruple solution.

3.4.2 Nonsingular Transitions

In Section 3.4.1, it has been shown that the joint space of this robot presents in general two hyperboloids that contain four polygonal curves (the γ-curves) where the forward kinematics admits quadruple solutions. Next, we present an example where encircling one of these curves produces a nonsingular change of assembly mode.

For the example, the following geometry will be used: $c_2 = 1.1$, $c_3 = 0.3$, $l_2 = 0.7$, and $l_3 = 0.4$ (arbitrary units). Figure 3.15a shows the joint space for this geometry, for $\rho_2 \leq 1$. The surface shown is the singularity locus, whose shape resembles a hood. There are four real solutions to the forward kinematics inside this "hood" (feasible region) and zero outside it, which agrees with [66]. To compute the singularity surface, the determinant of the Jacobian matrix of Equations (3.29), (3.30), and (3.31) with respect to (ϕ, x, y) has been equated to zero. The resulting equation is linear in x, quadratic in y, and quartic in $\tan(\phi/2)$, so it is easy to vary two of these variables and compute the remaining one to obtain the points that belong to the singularity surface in the workspace. Then, these points are mapped numerically to the joint space using Equations (3.29), (3.30), and (3.31), obtaining the surface shown in Figure 3.15a.

To facilitate the visualization, the hyperboloids are not shown in Figure 3.15a, but the γ-curves are represented: $abcd$ is γ_0^-, ae is γ_0^+, fg is γ_π^+, and fh is γ_π^-. The points $\{a, f\}$ are the origins of the γ-curves and lie in the plane $\rho_2 = 0$, whereas $\{d, e, g, h\}$ are in the plane $\rho_2 = 1$. For $\rho_2 \leq 1$, the curves γ_0^+, γ_π^+, and γ_π^- lie along some sharp edges of the singularity surface that delimits the feasible region, so they cannot be encircled. The part abc of γ_0^- belongs also to the limit of the feasible region, but the segment cd is inside it and can be encircled.

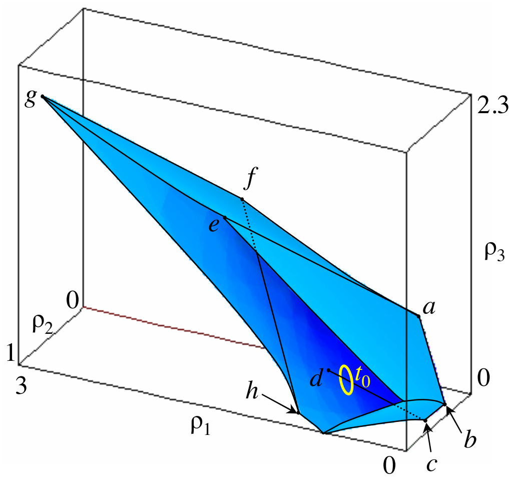
Next, a trajectory that encircles the segment cd will be analyzed. The trajectory is given by the following parametric equations:

$$t_0 \equiv \begin{cases} \rho_1 = 0.5 + 0.8 \cdot 0.1 \cos\theta \\ \rho_2 = 0.9 - 0.6 \cdot 0.1 \cos\theta \\ \rho_3 = 0.4 + 0.1 \sin\theta \end{cases} , \quad 0 \leq \theta \leq 2\pi \tag{3.48}$$

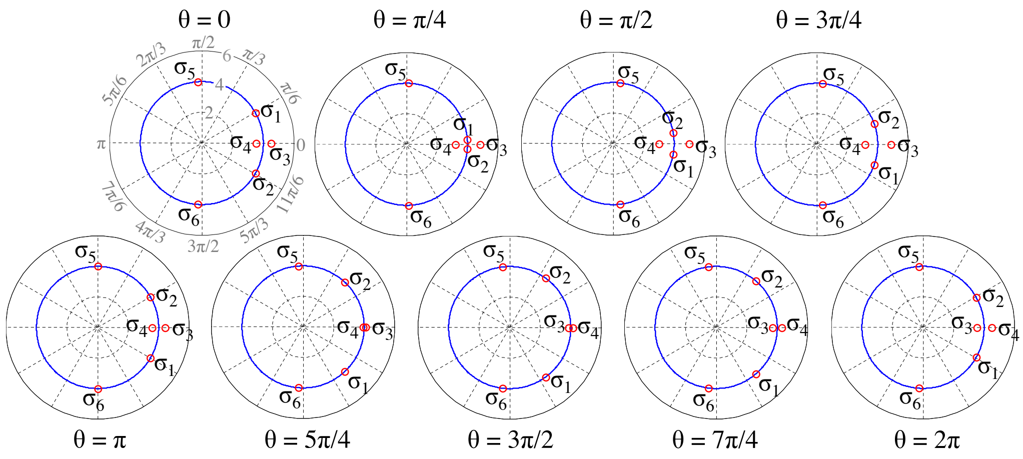
This trajectory, denoted by t_0 in Figure 3.15a, is a circle of radius 0.1 contained in the vertical plane $3\rho_1 + 4\rho_2 = 5.1$ and centered at the point where that plane intersects the curve γ_0^-.

Figure 3.15b shows the evolution of the six solutions to the forward kinematics along trajectory t_0. The rule described in Section 3.2.1 is used to sort the solutions along the trajectory: the continuity of the 6-tuple $\xi = (u, v, \mathrm{Re}(x), \mathrm{Im}(x), \mathrm{Re}(y), \mathrm{Im}(y))$ is imposed for each of the six solutions.

As θ increases from 0 to $\pi/4$, the solutions σ_1 and σ_2 move along the real circle towards $\phi = 0$. Between $\theta = \pi/4$ and $\theta = \pi/2$, these two solutions meet at $\phi = 0$, which corresponds to a crossing of the 0-hyperboloid in the joint space. The



(a)



(b)

Figure 3.15: (a) Joint space of the analytic 3-RPR robot, with the singularity locus and the γ-curves. Dotted lines indicate lines hidden by the singularity surface. (b) Polar diagrams with the evolution of the six solutions σ_i $(i = 1, \ldots, 6)$ to forward kinematics along the trajectory t_0 that encircles the curve γ_0^-.

0-hyperboloid is crossed again between $\theta = 5\pi/4$ and $\theta = 3\pi/2$, which produces the coincidence of the solutions σ_3 and σ_4 at $\phi = 0$. When that crossing occurs, ϕ is real for all the six solutions, but σ_3 and σ_4 are complex solutions since their y coordinate is calculated using Equation (3.35), which gives two imaginary values. At the end of the trajectory ($\theta = 2\pi$), the solutions are the same as at the beginning ($\theta = 0$), but a change of assembly mode has occurred since two pairs of solutions have swapped their positions in the polar diagram: σ_1 and σ_3 occupy the original places of σ_2 and σ_4, respectively.

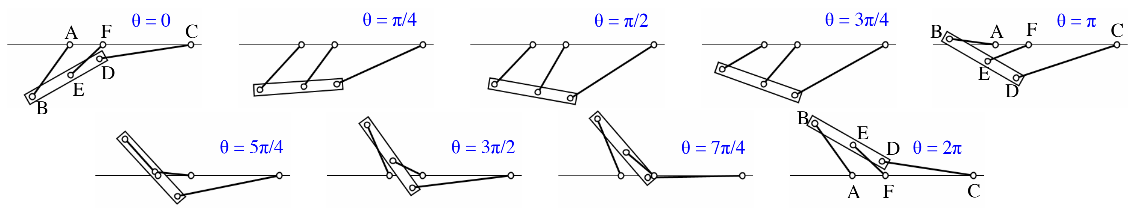


Figure 3.16: Evolution of the configuration of the 3-R<u>P</u>R robot for the solution σ_1 along the nonsingular assembly-mode-changing trajectory t_0

As remarked for the 2R<u>P</u>R-PR robot in Section 3.2.1, two solutions crossing at $\phi = 0$ or $\phi = \pi$ are not necessarily singularities, since the y coordinate for both solutions is computed using Equation (3.35), which gives two real or pure imaginary numbers with opposite sign.

Finally, Figure 3.16 shows the evolution of the configuration of the robot for the solution σ_1 along the trajectory, where the nonsingular change of assembly mode is evident.

3.4.2.1 Trajectory with constant ρ_2

Although Figures 3.15b and 3.16 demonstrate that this analytic 3-R<u>P</u>R robot can perform nonsingular transitions, the best way to understand why such transitions are possible is to visualize the trajectories in the reduced configuration space, as it has been done in Figure 3.10 for the 2R<u>P</u>R-PR robot. However, the representation of trajectories in the reduced configuration space requires varying only two joint coordinates, so that an output variable (e.g., the orientation ϕ of the mobile platform) can be plotted versus these joint coordinates in a 3D space. Hence, for this robot, it is necessary to fix the value of one of the three joint coordinates in order to visualize the trajectories in the reduced configuration space.

Next, a trajectory in which the length of the leg CD is kept constant at $\rho_2 = 1$ will be simulated. The intersection of the plane $\rho_2 = 1$ with the singularity locus of Figure 3.15a yields the curves of Figure 3.17a, where the points d, e, g, and h are the same as in Figure 3.15a (these points are the intersections between the γ-curves and the plane $\rho_2 = 1$). Keeping $\rho_2 = 1$ constant, the joint coordinates ρ_1 and ρ_3 will be varied along the following circular trajectory to encircle the curve γ_0^-:

$$\begin{cases} \rho_1 = 0.6 + 0.1\cos\theta \\ \rho_3 = 0.5 + 0.1\sin\theta \end{cases}, \quad 0 \le \theta \le 2\pi \tag{3.49}$$

which is a circle of radius 0.1 centered at the point d. This trajectory is shown in Figure 3.17a, together with the intersection between the π- and 0-hyperboloids and the plane $\rho_2 = 1$.

Figure 3.17b shows the polar plots with the evolution of the six solutions to the forward kinematics along the trajectory of Equation (3.49). Since this trajectory is similar to the trajectory t_0 of Figure 3.15a, the evolution of the solutions is very similar to the previous example. First, for $0 \leq \theta \leq \pi/4$, the solutions σ_1 and σ_2 move along the real circle, towards $\phi = 0$. Between $\theta = \pi/4$ and $\theta = \pi/2$, these two solutions coincide at $\phi = 0$ in the real circle, which occurs when the 0-hyperboloid is crossed in the joint space. Later, between $\theta = 5\pi/4$ and $\theta = 3\pi/2$, the 0-hyperboloid is crossed again when σ_3 and σ_4 meet in the real circle ($\phi = 0$ for both solutions). As in the previous examples, after completing the trajectory, two pairs of solutions have swapped their positions in the polar diagram: σ_1 and σ_3 occupy the original places of σ_2 and σ_4, respectively. Thus, a nonsingular transition has occurred.

As discussed for the previous examples, the crossings in the real circle are not singularities in this case, because the solutions that coincide in the real circle only share the angle ϕ, having different values for the x and y coordinates.

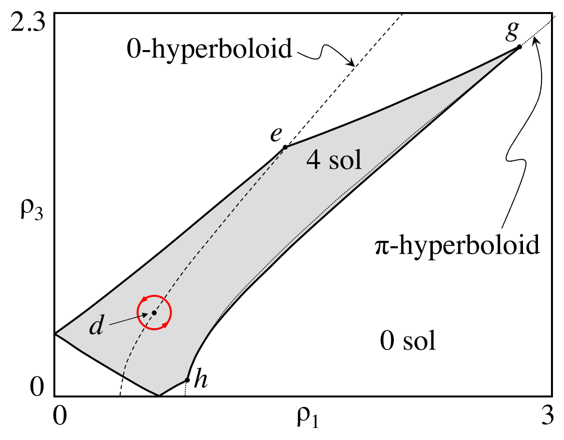
Figure 3.18 shows the trajectory described by the solution σ_1 in the reduced configuration space (ρ_1, ρ_3, ϕ) for this example. Note that the reduced configuration space, in the neighborhood of the quadruple solution D, has the same form as in the example shown in Figure 3.10 for the 2R$\underline{\text{P}}$R-PR robot. In this case, the configuration surface intersects itself along the curve η_0, whose projection onto the (ρ_1, ρ_3) plane yields the part of the 0-hyperboloid which lies to the right of the point d in Figure 3.17a. The point D of Figure 3.18 yields the point d when projected onto the (ρ_1, ρ_3) plane. As in the 2R$\underline{\text{P}}$R-PR robot, this shape of the reduced configuration space explains why it is possible to perform a nonsingular transition by encircling the points of the joint space with quadruple solutions to the forward kinematics.

Finally, it is interesting to visualize the trajectory described by the robot in the output space (i.e., the space of the x, y, and ϕ coordinates that define the position and orientation of the mobile platform), along with the singularity locus, to confirm that the trajectory followed by the robot is singularity-free. Figure 3.19 represents the singularity locus in the output space. The singularity locus, for the ranges of the (x, y, ϕ) variables used in Figure 3.19, is composed of a surface and a straight line segment l defined by: $\{y = 0, \phi = 0\}$. This segment l corresponds to the part of the curve γ_0^- that can be encircled in the joint space. The curve t_c is the trajectory described by the solution σ_1 when the joint coordinates perform the trajectory of Equation (3.49). Figure 3.19 shows that the trajectory t_c does not cross the singular surface. To show in the same figure that the segment l of the singularity locus is also avoided by the trajectory t_c, this trajectory has been projected onto the plane $x = -2$, obtaining the projected trajectory t_p. This projection clearly shows that the trajectory does not intersect the segment l, confirming that the transition is nonsingular.

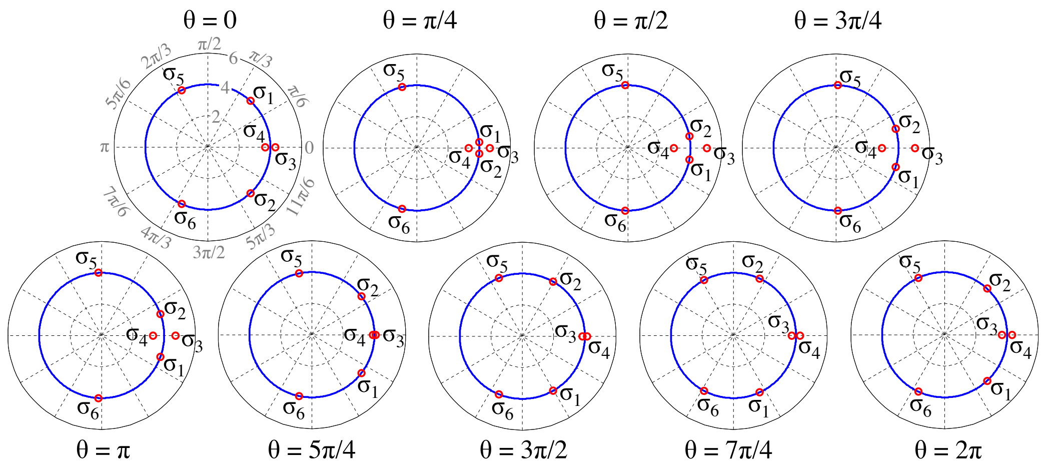
3.4.2.2 Triple zero of $P(\psi)$

Finally, Equation (3.10) will be solved for the geometry under study to check if $P(\psi)$ can have triple zeros in $[-1, 1]$. Using the geometric parameters indicated at the beginning of Section 3.4.2, the third equation of the system (3.10) becomes:

$$39000x_1 + 26700x_2 - 78925x_3 + 33264\psi = 40653 \tag{3.50}$$



(a)



(b)

Figure 3.17: (a) Slice of the joint space of Figure 3.15a at $\rho_2 = 1$. (b) Polar plots with the evolution of the six solutions σ_i $(i = 1, \ldots, 6)$ to the forward kinematics along the circle centered at d, shown in Figure 3.17a.

where $x_i = \rho_i^2$ $(i = 1, 2, 3)$. Solving x_2 from Equation (3.50) and inserting the solution into the first and second equations of the system (3.10) yields the following pair of equations:

$$\frac{308\psi+317}{792100}x_1^2 - \frac{169\psi+154}{190104}x_1x_3 + \frac{44352\psi+59977}{114062400}x_3^2 + \frac{255024\psi^2-82216\psi-189883}{910915000}x_1 + \frac{3060288\psi^2-2959633\psi+119954}{10930980000}x_3 - \frac{140152320\psi^3-731253600\psi^2+1065952516\psi-475317725}{4190209000000} = 0 \tag{3.51}$$

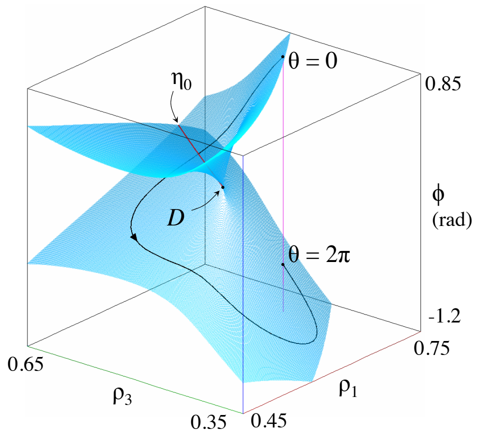


Figure 3.18: Trajectory of the solution σ_1 in the reduced configuration space of the 3-RPR robot, for the trajectory of Equation (3.49) and $\rho_2 = 1$

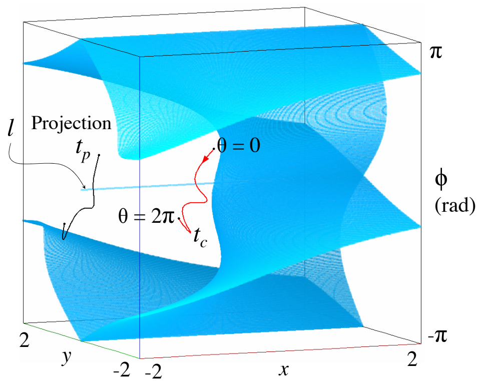


Figure 3.19: Trajectory of the solution σ_1 in the output space of the 3-RPR robot, for the trajectory of Equation (3.49) and $\rho_2 = 1$

$$\begin{aligned}\frac{40733}{198025}x_1^2 - \frac{89401}{190104}x_1x_3 + \frac{40733}{198025}x_3^2 + \frac{23(3060288\psi + 3682079)}{475260000}x_3 + \\ + \frac{23(31878\psi - 13049)}{4950625}x_1 + \frac{316274112\psi^2 - 585585000\psi + 267121871}{1980250000} = 0 \qquad (3.52)\end{aligned}$$

Equations (3.51) and (3.52) define respectively two conics C_1 and C_2 in the space (x_1, x_3). The coefficients of these conics depend on ψ, which must be in $[-1, 1]$ to obtain a triple zero of $P(\psi)$ that translates into a real angle ϕ. The next step is to calculate the intersection of C_1 and C_2 for $-1 \leq \psi \leq 1$.

To simplify the calculation, the coefficients of the conics can be used to classify them in terms of ψ (see [175], page 43). Studying the coefficients of C_2, it can be checked that it is a hyperbola $\forall \psi \in \mathbb{R}$. The analysis of the coefficients of C_1 results in more cases:

- If $\psi = -1$, C_1 is a double line that does not intersect C_2.
- If $-1 < \psi < 0$, C_1 is an imaginary ellipse.

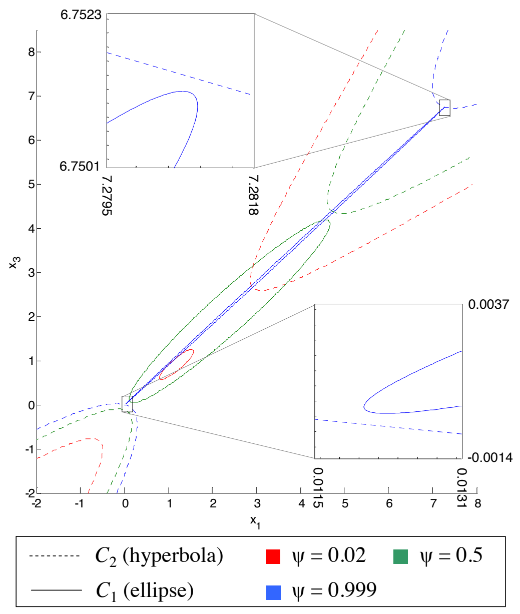


Figure 3.20: Conics C_1 (continuous line) and C_2 (dashed line) for different values of $0 < \psi < 1$. Each value of ψ is indicated with a different color. For $\psi = 0.999$, C_1 and C_2 almost intersect, as shown in the zoomed areas.

- If $\psi = 0$, C_1 is a point that does not belong to C_2.

For the previous cases, there is no real solution to Equations (3.51) and (3.52). For $0 < \psi < 1$, C_1 is a real ellipse whose size increases with ψ. C_1 and C_2 become closer as ψ approaches 1, but apparently they never intersect (see Figure 3.20).

Finally, for $\psi = 1$ the conic C_1 is another double line (i.e., a pair of coincident lines) that intersects C_2 at two points, which yields two solutions p^+ and p^- with the following joint coordinates:

$$p^{\pm} : \rho_1 = \frac{323}{230} \pm \frac{12\sqrt{154}}{115},\ \rho_2 = \frac{83}{46} \pm \frac{12\sqrt{154}}{115},\ \rho_3 = \frac{30}{23} \pm \frac{12\sqrt{154}}{115}$$

where $\rho_2 = \sqrt{x_2}$ is obtained substituting $\psi = 1$ and the corresponding values of x_1 and x_3 into Equation (3.50). At the point p^+, $\psi = 1$ is a triple zero of $P(\psi)$. A zero at $\psi = 1$ yields a single angle $\phi = 0$ (regarding angles differing by an integer multiple of 2π as the same angle). Equation (3.34) gives a single value for the x coordinate: $x = x_{p^+} = -323/230 - 12\sqrt{154}/115$. Finally, if the radicand in Equation (3.35) was positive, two different real values would be obtained for y. In that case, the forward kinematics would have two *triple* real solutions: $(\phi = 0, x = x_{p^+}, y = \sqrt{\rho_1^2 - x^2})$ and $(\phi = 0, x = x_{p^+}, y = -\sqrt{\rho_1^2 - x^2})$. However, the mentioned radicand is zero at p^+, which means that these two solutions become a single solution with multiplicity six. It can be checked that this also happens at p^-. The points p^+ and p^-, where the sextuple solutions occur, belong to parts of the curve γ_0^- that lie on the singularity surface that delimits the region of the joint space where the robot can be assembled. Hence, these parts of γ_0^- cannot be encircled.

3.5 Stability Analysis of Quadruple Solutions

In the previous sections, we thoroughly analyzed the solutions of the forward kinematic problem of two analytic parallel mechanisms: the 2R$\underline{\text{P}}$R-PR mechanism (Section 3.1.1) and the 3R$\underline{\text{P}}$R robot (Section 3.4.1). We demonstrated that both mechanisms are able to perform nonsingular transitions by enclosing quadruple solutions of the forward kinematic problem. Also, in Section 3.3 we checked that this phenomenon will not affect the climbing robot studied in this thesis, since the joint limits of the linear actuators will impede both singular and nonsingular transitions in the analytic 2R$\underline{\text{P}}$R-PR mechanisms used in the legs of this climbing robot.

However, although nonsingular transitions will not occur in the climbing robot studied in this thesis, it is very important to pose the following question: *how stable is the ability to perform nonsingular transitions of the mechanisms studied in Sections 3.2.1 and 3.4.2, under perturbations in the geometric design of these mechanisms?* The analytic parallel mechanisms studied in these sections have very special geometric designs. The analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b is a particular case of the general mechanism of Figure 3.1a, obtained when the mobile platform B-B$_1$-B$_2$ is flat and the passive slider OB is perpendicular to the base A$_2$-A$_1$. Similarly, the analytic 3R$\underline{\text{P}}$R robot of Figure 3.14b is a particular case of the general robot of Figure 3.14a, in which both the mobile and fixed platforms are flat. When a parallel mechanism has a special geometric design like in these two previous examples, it is said that its geometry is *non-generic*, and in that case, the singularity locus may exhibit higher-order singularities, like the isolated singularity λ_π of Figure 3.4.

The isolated point λ_π studied in previous sections is not the only higher-order singularity. Other well-known higher-order singularities that may appear when the geometry of a 2-DOF mechanism is non-generic are the *lips*, *beaks*, and *swallowtail* singularities [182]. These singularities are illustrated in Figures 3.21b, 3.21e, and 3.21h, assuming a 2-DOF parallel mechanism with two active joint coordinates ρ_1 and ρ_2. Higher-order singularities are unstable, since they only exist when the geometry of a mechanism is *exactly* non-generic (for example, the non-generic geometry of the analytic 3R$\underline{\text{P}}$R robot of Figure 3.14b requires both the mobile and flat platforms to be *exactly* flat).

However, due to finite precision or tolerances in the manufacturing of real parallel mechanisms and robots, in practice it will be impossible to build a real mechanism exactly with a non-generic geometry: there will always be small deviations from this non-generic geometry. Since higher-order singularities are unstable and only exist when the mechanism exactly has a non-generic geometry, a slight deviation from a non-generic geometry will transform these singularities as depicted in Figure 3.21 and explained next [182]. Depending on the direction of the perturbation...

- ... a *lips* singularity (Figure 3.21b), which is an isolated point, will either disappear (Figure 3.21c) or transform into a small bicuspid closed curve (Figure 3.21a).

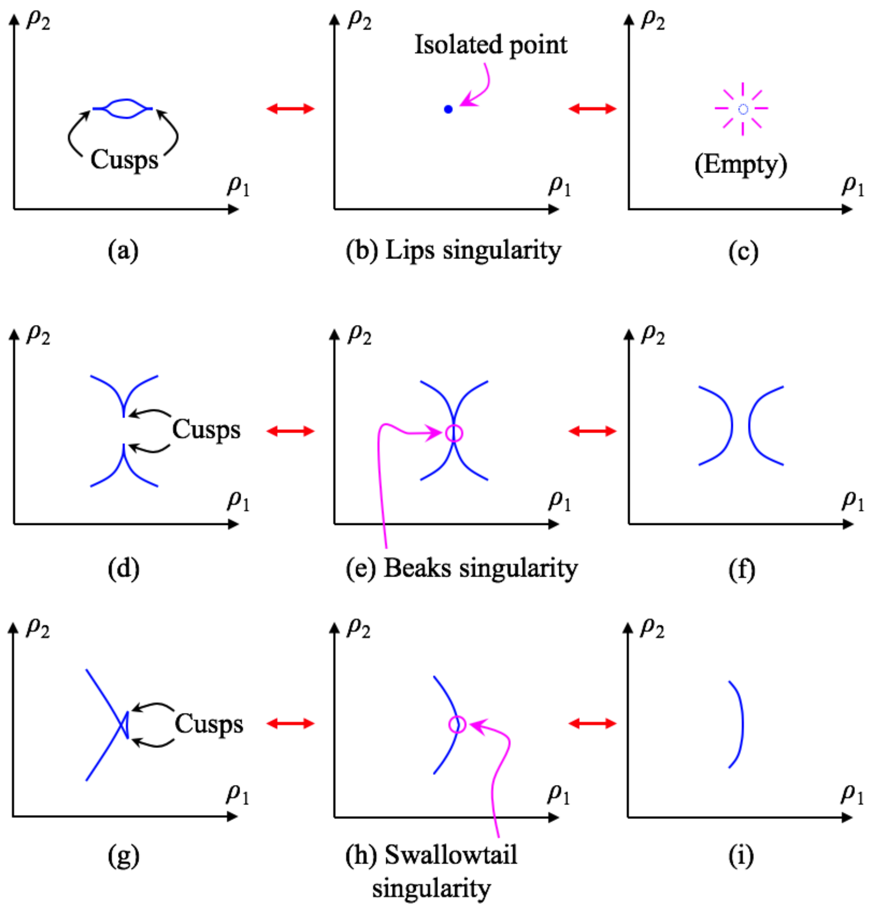


Figure 3.21: Higher-order unstable singularities and their perturbations.

- . . . a *beaks* singularity (Figure 3.21e) will transform either into a couple of smooth curves (Figure 3.21f) or a couple of curves with one cusp each (Figure 3.21d).
- . . . a *swallowtail* singularity (Figure 3.21h), will transform either into a smooth curve (Figure 3.21i) or a self-intersecting curve with two cusps (Figure 3.21g).

Accordingly, when perturbing one of these three higher-order singularities, two cusps are either created (cases of Figures 3.21a, 3.21d, 3.21g) or destroyed (cases of Figures 3.21c, 3.21f, 3.21i). Therefore, if a non-generic mechanism has a higher-order singularity, and its geometry is perturbed in the direction that destroys two cusps, the mechansim will lose the ability to perform nonsingular transitions (at least locally, near the destroyed cusps - the mechanism may still have additional cusps in other regions of its active joint space).

Like the lips singularity, the quadruple points λ_π (Figure 3.4) and d (Figure 3.17a), studied previously, are also higher-order isolated singularities. As we have just said, a lips singularity can either transform into a closed curve with two cusps, or be destroyed (in which case, the mechanism will lose the ability to perform nonsingular transitions). Considering this precedent, it is important to analyze how the isolated singularities studied in Sections 3.2.1 and 3.4.2 will transform under perturbations of the geometry, to check whether these singularities will behave similarly to lips singularities or not.

Next, we will analyze the stability of the isolated quadruple singularities of the analytic 3R$\underline{\text{P}}$R robot of Figure 3.14b, whose geometry is non-generic. This analysis will consist in perturbing numerically the geometry of the robot so that at least one of its platforms is no longer perfectly flat (and the robot is no longer analytic and non-generic), and observing how these isolated quadruple singularities transform under these perturbations.

3.5.1 Another Example of Nonsingular Transitions in Analytic 3R$\underline{\text{P}}$R Robots

In section 3.4.1, we demonstrated that the 3R$\underline{\text{P}}$R robot with flat and non-similar fixed and mobile platforms exhibits points in its three-dimensional active joint space at which its forward kinematic problem admits quadruple solutions. These points lie on polygonal curves denoted by $\gamma_0^{\pm}$ and $\gamma_\pi^{\pm}$ in Section 3.4.1. Some parts of these polygonal curves can be encircled by three-dimensional trajectories in which all three joint coordinates vary, which produces nonsingular transitions (see Section 3.4.2).

In subsection 3.4.2.1, we also analyzed this phenomenon when one of the active joint coordinates was kept constant. In that case, the robot enclosed the isolated quadruple singularity d obtained as the intersection of one of the aforementioned polygonal curves with a plane (this point was represented in Figure 3.17a).

In this section, we will again keep constant one of the active joint coordinates of the analytic 3R$\underline{\text{P}}$R robot of Figure 3.14b (in this case, the length of ρ_3 will be kept constant). Then, we will analyze how the isolated quadruple singularities of this analytic robot transform when perturbing its non-generic geometry. The reason for keeping constant one active joint coordinate of the 3R$\underline{\text{P}}$R robot is to have a 2-DOF mechanism, which allows us to analyze the effects of the perturbation in the same context as the higher-order singularities of Figure 3.21, which assume 2-DOF mechanisms.

Moreover, analyzing the 2-DOF mechanism obtained by keeping constant the length ρ_3 of the 3R$\underline{\text{P}}$R robot will also allow us to directly extend the results observed in this mechanism to the quadruple singularities of the (2-DOF) analytic 2R$\underline{\text{P}}$R-PR mechanism of Figure 3.1b. It should be mentioned that the transformations suffered by the quadruple singularities λ_0 and λ_π of this 2R$\underline{\text{P}}$R-PR mechanism under geometric perturbations were studied in [40], where these quadruple singularities were characterized in much greater detail than in the present chapter. In [40], these two quadruple singularities were analyzed based on the characteristic curves and uniqueness domains of the (ϕ, y) workspace of the 2R$\underline{\text{P}}$R-PR parallel mechanism, and were identified with the singularities of two complex mappings, namely: the complex square mapping and the "quarto" mapping.

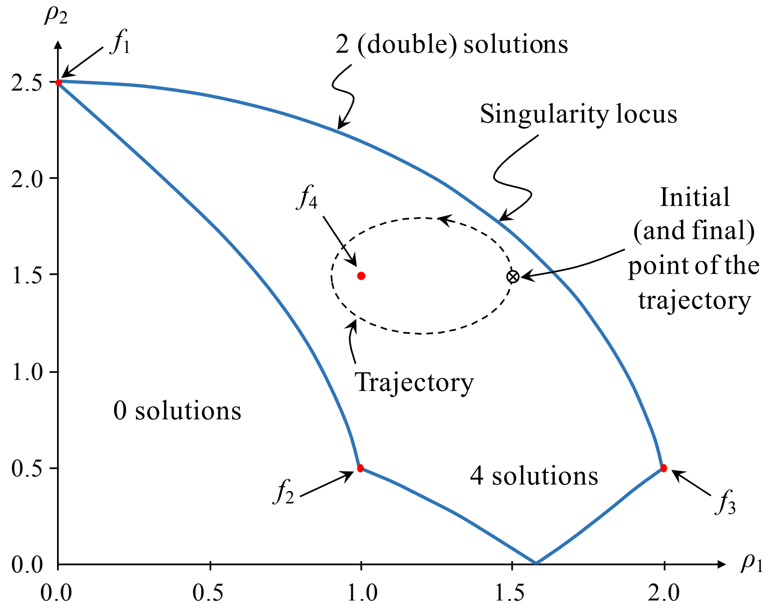


Figure 3.22: Singularity locus of an analytic 3R$\underline{\text{P}}$R robot with the following geometry: $c_2 = 1.5$, $c_3 = l_3 = l_1 = 0.5$ (also, the third input is kept constant at $\rho_3 = 1$). The figure indicates the number of different real solutions to the forward kinematic problem in each region of the (ρ_1, ρ_2) plane.

Returning to the 3R$\underline{\text{P}}$R robot, consider an analytic 3R$\underline{\text{P}}$R robot (i.e., a robot satisfying $d_3 = 0$ and $\beta = \pi$ rad) with the following geometric design parameters: $c_2 = 1.5$, $c_3 = l_3 = l_1 = 0.5$. As indicated above, assume also that only the inputs ρ_1 and ρ_2 are varied, keeping $\rho_3 = 1$. The corresponding singularity locus in the input plane (ρ_1, ρ_2) is shown in Figure 3.22.

Similarly to the example of Figure 3.17a, the singularity locus shown in Figure 3.22 is a closed curve that divides the (ρ_1, ρ_2) plane into two regions: the region outside the singularity curve, and the region inside it. In the outer region, the number of real solutions to the forward kinematic problem is zero: this means that the robot cannot be assembled for these combinations of ρ_1 and ρ_2. In the inner region, the forward kinematic problem of the robot has four different real solutions, i.e., the mobile platform adopts four different configurations for a given pair (ρ_1, ρ_2) enclosed by the singularity curve. For the pairs (ρ_1, ρ_2) that lie exactly on the singularity curve, the forward kinematic problem of the robot has two different double solutions (two different solutions with multiplicity two each), as in the vertex of an alpha-curve (point D in Figure 3.5b). Finally, the singularity locus also has four special points, labeled as f_1, f_2, f_3, and f_4 in Figure 3.22 (actually, the singular point f_4 is isolated, i.e., it does not lie on the singular curve, although it is a singularity too).

The points f_1, f_2, f_3, and f_4 are the intersections of the aforementioned polygonal curves ($\gamma_0^{\pm}$ and $\gamma_\pi^{\pm}$) with the plane $\rho_3 = 1$, in this example. At any of the points $\{f_1, f_2, f_3\}$, the forward kinematic problem of the robot has a single real solution with multiplicity four. At the isolated point f_4, the forward kinematic problem of this robot has three different real solutions: two of them are simple (they have multiplicity one) and the third one is quadruple (it has multiplicity four).

Next, we will illustrate another example of nonsingular transition by performing a feasible closed trajectory in the (ρ_1, ρ_2) plane that encloses one of the mentioned quadruple points. As Figure 3.22 shows, only the isolated point f_4 can be encircled by a closed trajectory, since any closed trajectory that encircles any of the other points $\{f_1, f_2, f_3\}$ crosses the singularity curve and invades the outer region, in which the robot has no real solutions to the forward kinematic problem (real robots cannot invade the outer region because the robot cannot be assembled in these regions). It is worth to mention that, if the mobile and fixed platforms become similar (i.e., if $c_3 l_2 = c_2 l_3$), then the isolated point f_4 shifts toward the singularity curve too (like the points $\{f_1, f_2, f_3\}$), which impedes the execution of closed trajectories enclosing f_4. This is why the non-similarity condition has been imposed so far in this chapter.

According to the previous paragraph, the following circular trajectory is executed, which encloses the point f_4:

$$\begin{cases} \rho_1 = 1.2 + 0.3 \cos\psi \\ \rho_2 = 1.5 + 0.3 \sin\psi \end{cases} , \quad 0 \leq \psi \leq 2\pi \tag{3.53}$$

This trajectory is shown in dashed line in Figure 3.22. The trajectories described by the six solutions of the forward kinematic problem in the complex plane of the angle ϕ when the inputs perform the trajectory of (3.53) are shown in Figure 3.23 in different colors[2]. Actually, the trajectories followed by the solutions $\{1, 2, 3, 4\}$ are completely contained in the real axis. However, since these four trajectories overlap, the trajectories described by the solutions 3 and 4 have been slightly displaced vertically to facilitate the visualization of the trajectories. The trajectories described by the solutions 5 and 6 are completely contained in the imaginary axis. Similarly, since these two trajectories overlap, they have been slightly displaced horizontally. Note that this behavior of the trajectories (four real trajectories and two imaginary trajectories) coincides with the fact that the trajectory described by the input variables ρ_1 and ρ_2 is completely contained in the region of the (ρ_1, ρ_2) plane enclosed by the singularities, in which the forward kinematic problem has four different real solutions (see Figure 3.22).

In Figure 3.23, the initial point of each trajectory is represented by a cross "$\times$", whereas the final point of each trajectory is represented by a circle "$\circ$" (this representation is reminiscent of the well-known *root locus* studied in Control Engineering). Looking at the initial and final points of each trajectory, we can see that the solutions 1 and 2 describe closed trajectories (the initial and final points coincide in each of these trajectories). On the contrary, the solutions $\{3, 4, 5, 6\}$ describe open trajectories since their initial and final points do not coincide. Of particular interest are the open trajectories described by the solutions 3 and 4, which are real: these trajectories correspond to nonsingular changes of the assembly mode of the robot. Indeed, if the robot

[2]Since the trajectories described by the complex solutions of angle ϕ do not wrap in this example, it is sufficient to use here the rectangular representation of Figure 3.6a, instead of the modified polar representation used in the previous examples of this chapter.

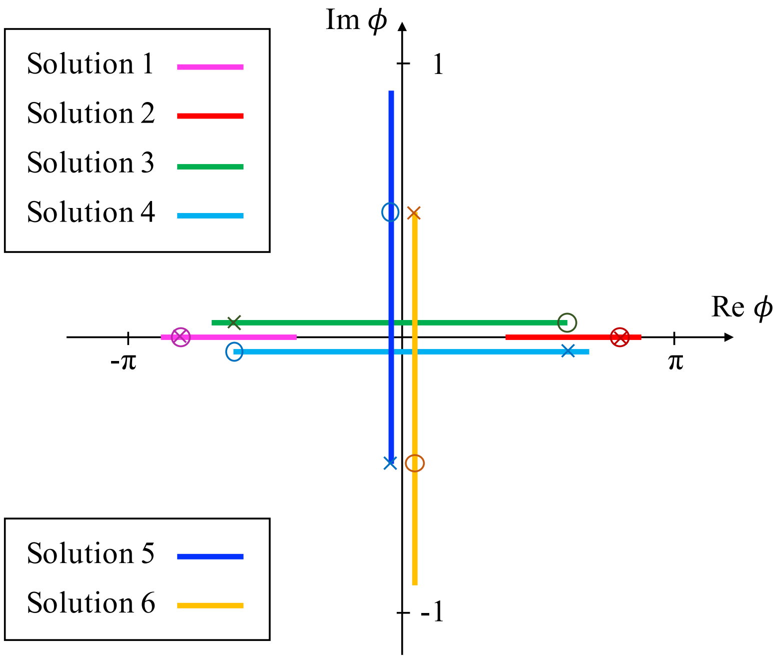


Figure 3.23: Trajectories described by the six solutions of the forward kinematic problem of the analytic 3R$\underline{\text{P}}$R robot when encircling the point f_4. The trajectories of the solutions $\{1, 2, 3, 4\}$ lie on the real axis, but since they overlap, the trajectories of the solutions 3 and 4 have been displaced vertically to facilitate the visualization. The trajectories of the solutions 5 and 6 are contained in the imaginary axis, but they have been shifted horizontally since they also overlap.

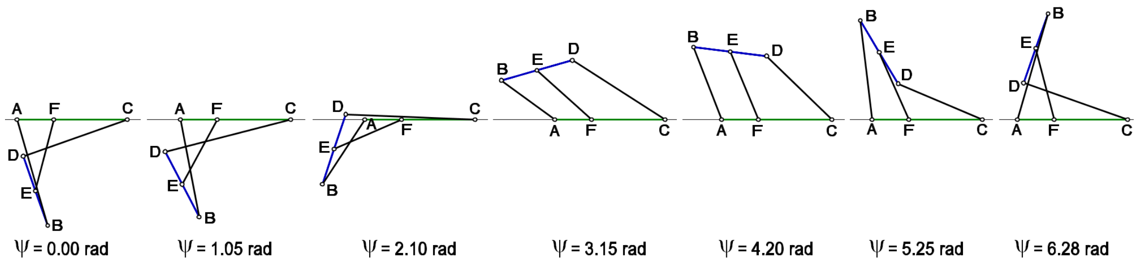


Figure 3.24: Evolution of the configuration of the robot along the solution 4 of Figure 3.23. The mobile platform BED begins ($\psi = 0$) and ends ($\psi = 6.28$ rad) the trajectory with different configurations: a nonsingular change of assembly mode occurs.

follows the trajectory described by the solution 3 (or 4), then the configuration of the mobile platform at the beginning of the trajectory will be different from the configuration at the end of the trajectory (see Figure 3.24), even though the input variables (ρ_1, ρ_2) have the same values at the beginning and at the end of such a trajectory. Since no singularities are crossed along this trajectory, it is a nonsingular (i.e., an easily controllable) transition.

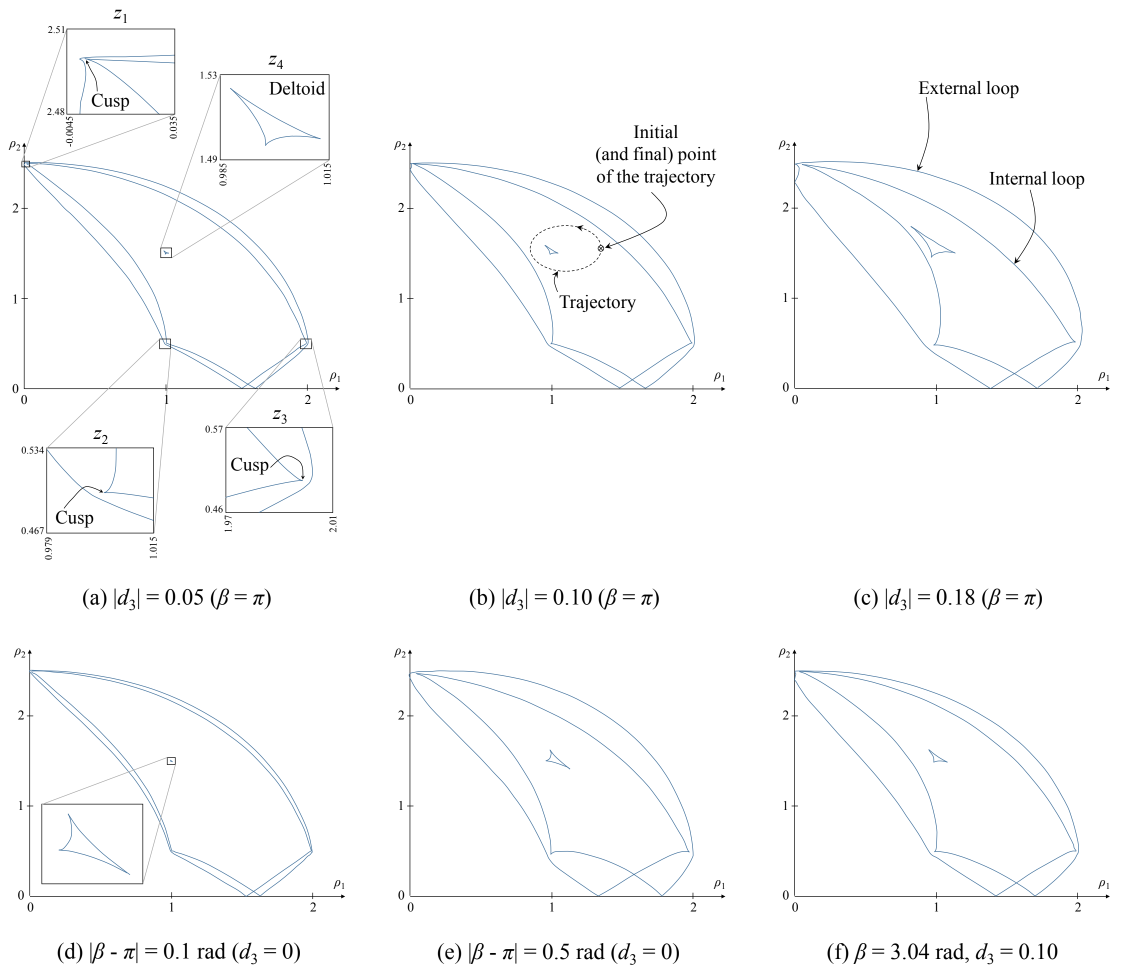
3.5.2 Perturbation of the Geometry of Non-generic 3R$\underline{\text{P}}$R Robots

As demonstrated in Section 3.4.1, the points $\{f_1, f_2, f_3, f_4\}$ (at which the forward kinematic problem of the analytic 3R$\underline{\text{P}}$R robot has quadruple solutions) always exist, independently of the values of the geometric design parameters $\{c_3, c_2, l_3, l_1\}$. However, this is true only if the robot remains analytic, i.e., if $d_3 = 0$ and $\beta = \pi$. In practice, due to the geometric tolerances and the finite accuracy in the manufacturing and mechanical assembly of the robot, it will be impossible to attain a perfect alignment between the revolute joints of the base or of the mobile platform, which means that d_3 and β will never be exactly equal to 0 and π, respectively.

Although the singular points f_i are stable with respect to the geometric parameters $\{c_3, c_2, l_3, l_1\}$ (in the sense that these points do not disappear when modifying these geometric parameters, only their position in the input plane is modified), these points may undergo important changes when the parameters d_3 or β deviate from 0 or π, respectively, since in that case the robot is no longer analytic. For example, consider what happens to an (isolated) lips singularity: as described at the beginning of the present section, perturbing a geometric parameter in a robot which has a lips singularity may lead to the transformation of the lips into a closed curve with two cusps (Figure 3.21a), which allows the robot to perform nonsingular transitions (by encircling any of these cusps). However, the aforementioned perturbation of a geometric parameter may also lead to the disappearance of the lips singularity, in which case the robot loses the ability to perform these nonsingular transitions (see Figure 3.21c). Considering this precedent, we will next study, using the example of Figure 3.22, what happens to the singular points f_i (and especially the isolated point f_4) when the geometry of the robot is perturbed from the analytic case, i.e., when d_3 and β deviate from 0 and π, respectively.

Keeping the four geometric parameters $\{c_3, c_2, l_3, l_1\}$ constant at the values of the previous example ($c_2 = 1.5$, $c_3 = l_3 = l_1 = 0.5$), d_3 and β will be perturbed from 0 and π next. Figures 3.25a-c show the effect of perturbing d_3 from zero (keeping $\beta = \pi$), whereas Figures 3.25d-e show the effect of perturbing β from π (keeping $d_3 = 0$). Figure 3.25f shows the effect of perturbing both variables simultaneously. As indicated in Figure 3.25, the sign of these perturbations does not affect the deformation of the singularity curves, i.e., only the absolute value of the perturbation matters (symmetric perturbations produce the same effect).

Comparing Figures 3.22 and 3.25, we can see that when the geometry of the robot is perturbed from the analytic case, the closed singularity curve of Figure 3.22 is split into two loops: an internal loop and an external loop (e.g., see Figure 3.25c). These two loops exactly overlap when the robot tends to the analytic geometry, i.e., when $d_3 = 0$ and $\beta = \pi$. In other words, the two loops become more different as the magnitude of the perturbation from the analytic case increases, i.e., when increasing $|d_3|$ or $|\beta - \pi|$. Regarding the three points $\{f_1, f_2, f_3\}$ of Figure 3.22, at which the forward kinematic problem of the analytic robot has a quadruple real solution, when the geometry of the robot is perturbed from the analytic geometry, these three points



(a) $|d_3| = 0.05$ ($\beta = \pi$)

(b) $|d_3| = 0.10$ ($\beta = \pi$)

(c) $|d_3| = 0.18$ ($\beta = \pi$)

(d) $|\beta - \pi| = 0.1$ rad ($d_3 = 0$)

(e) $|\beta - \pi| = 0.5$ rad ($d_3 = 0$)

(f) $\beta = 3.04$ rad, $d_3 = 0.10$

Figure 3.25: Deformation of the singularity locus as the geometry of the 3R$\underline{\text{P}}$R robot is slightly perturbed from the analytic geometry. The points $\{f_1, f_2, f_3\}$ transform into cusps. The point f_4 transforms into a deltoid with three cusps. The size of the deltoid increases with the magnitude of the perturbation from the analytic geometry. The remaining geometric parameters of the robot are: $c_2 = 1.5$, $c_3 = l_3 = l_1 = 0.5$.

transform into three cusps of the internal loop of the singularity locus. These three cusps are shown in detail in the zoomed areas z_1, z_2, and z_3 of Figure 3.25a.

Regarding the isolated point f_4 of Figure 3.22, at which the analytic robot has three different real solutions (two of them are simple, and the remaining one is quadruple), Figure 3.25 shows that this point transforms into a deltoid when the geometry of the robot deviates from the analytic geometry. This deltoid is a closed curve with three cusps at its vertices, and its size increases with $|d_3|$ or $|\beta - \pi|$. In other words, as d_3 and β tend to 0 and π, respectively, the deltoid becomes smaller, until it shrinks to the isolated point f_4 in the limit, when the robot becomes analytic (see the zoomed area z_4 in Figure 3.25a and the zoomed area in Figure 3.25d). A similar behavior was observed in [38] for the 3R$\underline{\text{P}}$R robot with congruent base and mobile platform, where it was shown that the singularity locus in the input plane has a deltoid that shrinks to a point as the length of one of the three actuated legs tends to infinity.

At the points enclosed by the deltoids, the forward kinematic problem of the perturbed 3R$\underline{\text{P}}$R robot (which is no longer analytic) has six different real solutions, instead of the quadruple solution that occurs at f_4. Thus, this demonstrates that the quadruple solutions occurring at f_4 are not stable, since they cease to exist when d_3 or β are not exactly equal to 0 or π, respectively. This will be the case in practice, since it is impossible to build a robot satisfying exactly $d_3 = 0$ and $\beta = \pi$, due to manufacturing tolerances.

Although the quadruple solutions are not stable with respect to perturbations in d_3 or β, let us study the stability of the behavior of the robot when encircling the deltoids. Consider a real robot manufactured so that $|d_3|$ and $|\beta - \pi|$ are small but not necessarily equal to zero, due to limited accuracy in the manufacturing (i.e., an "almost analytic" robot). According to the previous paragraphs, when such a robot performs a closed trajectory in the input plane that encircles the point f_4 (as in Figure 3.22), actually the robot will be encircling a small deltoid, instead of an isolated point. Will the robot undergo nonsingular transitions as in the analytic case, or will the robot lose the ability to perform such transitions due to the transformation of the point f_4 into a deltoid? To answer this, consider the following geometry, which corresponds to the singularity locus of Figure 3.25b: $c_2 = 1.5$, $c_3 = l_3 = l_1 = 0.5$, $d_3 = 0.1$, $\beta = \pi$ (note that this is an exaggerated example, since in this case d_3 is not negligible compared to the other geometric parameters of the robot). Next, we will study the evolution of the six solutions of the forward kinematic problem of this robot when the inputs describe the following closed trajectory in the (ρ_1, ρ_2) plane:

$$\begin{cases} \rho_1 = 1.1 + 0.25\cos\psi \\ \rho_2 = 1.55 + 0.25\sin\psi \end{cases} , \quad 0 \leq \psi \leq 2\pi \tag{3.54}$$

This closed trajectory is represented in Figure 3.25b. The evolution of the six solutions to the forward kinematic problem along this trajectory is represented in Figure 3.26, in the complex plane of the angle ϕ. As in the analytic case (see Figure 3.23), four solutions (solutions 1, 2, 3, and 4) describe trajectories that are contained in the real axis. Since these four trajectories overlap again, the trajectories of the solutions 3 and 4 have been vertically displaced from the real axis to facilitate their visualization in Figure 3.26. As Figure 3.26 shows, the remaining two solutions (solutions 5 and 6) describe complex (= non-real) trajectories, but unlike in the analytic case (see Figure 3.23), the trajectories described by these two solutions are no longer contained in the imaginary axis.

Comparing Figures 3.23 and 3.26, it can be observed that the behavior of the six solutions when encircling the deltoid is very similar to the behavior observed when the isolated point f_4 is encircled in the analytic case. First, there are two solutions that describe closed trajectories in the real axis (solutions 1 and 2). For these two solutions, the mobile platform of the robot begins and ends the trajectory with the same

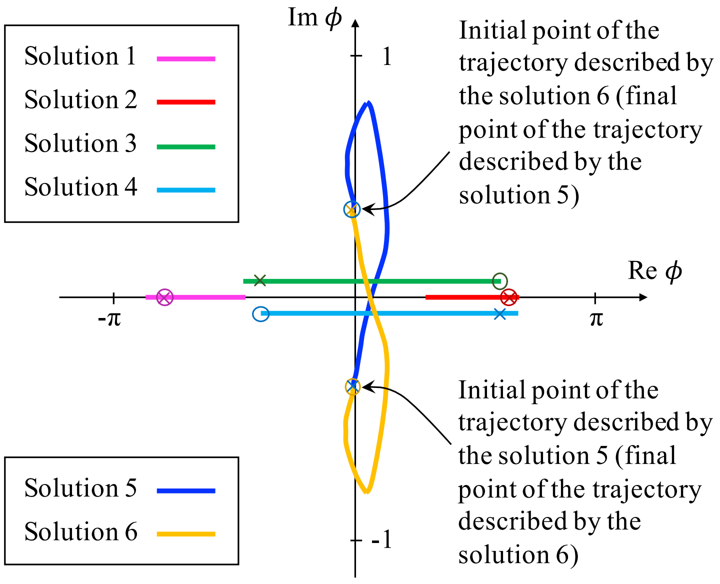


Figure 3.26: Trajectories described by the six solutions to the forward kinematic problem of an "almost analytic" 3R$\underline{\text{P}}$R robot when encircling the deltoid of Figure 3.25b. The trajectories of the solutions $\{1, 2, 3, 4\}$ are contained in the real axis, but since they overlap, the trajectories of the solutions 3 and 4 have been shifted vertically to facilitate the visualization.

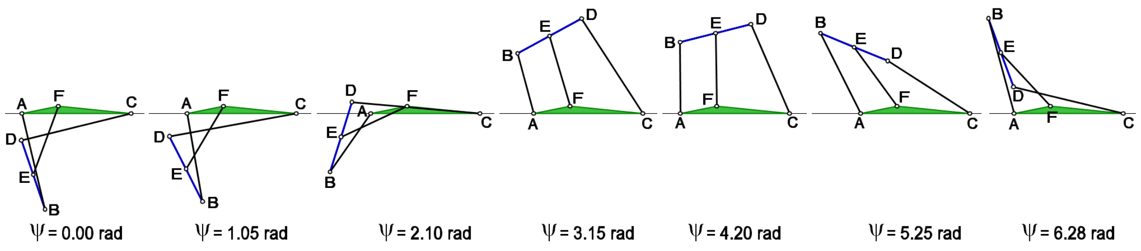


Figure 3.27: Evolution of the configuration of the robot along the solution 4 of Figure 3.26. The mobile platform BED begins ($\psi = 0$) and ends ($\psi = 6.28$ rad) the trajectory with different configurations: a nonsingular change of assembly mode occurs.

configuration, therefore these two solutions do not experience a change of assembly mode. The solutions 5 and 6 describe non-real open trajectories. Finally, the solutions 3 and 4 describe real open trajectories that correspond to nonsingular changes of assembly mode: the configuration of the mobile platform at the beginning of the trajectory described by the solution 3 (or 4) is different from the configuration at the end of the same trajectory (see Figure 3.27).

Therefore, although the singular isolated point f_4 (at which the analytic robot has a quadruple solution to its forward kinematic problem) is not stable since it transforms into a deltoid when the geometry of the robot is perturbed from the analytic case, we can conclude that the behavior of the robot when encircling the point f_4 is stable. This is because the solutions of the forward kinematic problem evolve similarly when encircling the point f_4 (in the analytic case) as when encircling the deltoid (in the non-analytic case, obtained perturbing the analytic case). In particular, and in contrast to what happens to lips singularities (see Figures 3.21a,b,c), we have shown that the ability of the robot to perform nonsingular transitions by encircling the point f_4 is conserved, even when this point transforms into a small deltoid.

3.6 Second-order Taylor Stability Analysis of Isolated Singularities

In the previous section, we have demonstrated that the isolated quadruple singularities of the 3R$\underline{\text{P}}$R analytic robot transform into small deltoids when at least one of its platforms is no longer flat. When encircling this deltoid, the solutions of the forward kinematic problem evolve similar to when encircling the isolated singularity in the analytic case (including the occurrence of nonsingular transitions). Unlike in isolated lips singularities, the experiments shown in Figure 3.25 seem to indicate that the isolated quadruple singularity f_4 cannot be destroyed by small geometric perturbations (i.e., it cannot be made to disappear, it only can be transformed into a deltoid). Therefore, it seems that the analytic 3R$\underline{\text{P}}$R robot will never lose the ability to perform nonsingular transitions due to small perturbations.

However, the analysis performed in the previous section was not exhaustive, because it only analyzed a finite number of perturbations: the examples of Figure 3.25 do not constitute a formal proof that the isolated point f_4 can never be destroyed. How do we know there is not a particular perturbation of d_3 or β that *does* destroy the singularity f_4? Maybe such a "destructive" perturbation exists, but we simply missed it when doing the experiments of Figure 3.25.

This section presents a more rigorous and formal method for determining how isolated singularities of parallel mechanisms transform when perturbing the geometry of the mechanism. This method is based on second-order Taylor expansions and is valid for isolated singularities (excluding some special exceptions discussed in Section 3.6.4), including lips singularities and isolated quadruple singularities, like the point f_4 of the 3R$\underline{\text{P}}$R robot or the point λ_π of the 2R$\underline{\text{P}}$R-PR mechanism. Using the method proposed in this section, it will be possible to predict how isolated singularities will transform under *any* perturbation of the geometric parameters of the mechanism (i.e., not only under a finite number of perturbations, as we did in Figure 3.25). In particular, it will allow us to demonstrate that, indeed, it is impossible to find perturbations of the parameters d_3 or β which can destroy the isolated singularity f_4 analyzed in the previous section. This means that this singularity will always transform into a deltoid when perturbing the geometry of the robot, i.e., it will never disappear (unlike lips singularities).

3.6.1 Formulation of the Method

Consider a parallel mechanism with 2 degrees of freedom, with inputs $\mathbf{x} = [x_1, x_2]^T$ and outputs $\mathbf{y} = [y_1, y_2]^T$. The inputs are two actuated joint coordinates, whereas the outputs are two parameters that define the position and/or orientation of the mobile platform of the mechanism. Let $\mathbf{g} = [g_1, \ldots, g_d]^T$ be the vector of all geometric design parameters of the mechanism. Inputs, outputs, and geometric parameters are related through a system of two simultaneous scalar equations that represent the kinematic/assembly constraints of the mechanism:

$$f_1(\mathbf{x}, \mathbf{y}, \mathbf{g}) = 0 \quad \text{AND} \quad f_2(\mathbf{x}, \mathbf{y}, \mathbf{g}) = 0 \tag{3.55}$$

where f_1 and f_2 are *constraint functions*. The forward kinematic problem consists in solving $\mathbf{y}$ from the system (3.55) for given $\mathbf{x}$ and $\mathbf{g}$. For example: in the analytic 2R$\underline{\text{P}}$R-PR parallel mechanism of Figure 3.1b, the inputs are $x_1 = l_1$ and $x_2 = l_2$, the outputs are $y_1 = \phi$ and $y_2 = y$, the geometric parameters are $\mathbf{g} = [a_1, a_2, b_1, b_2]^T$, and the scalar equations that relate all these variables are Equations (3.1) and (3.2).

As we have previously seen in Section 3.1.2, the parallel singularity locus is the set of configurations at which the output velocities $\dot{\mathbf{y}} = [\dot{y}_1, \dot{y}_2]^T$ are not uniquely determined by the feasible input velocities $\dot{\mathbf{x}} = [\dot{x}_1, \dot{x}_2]^T$. Assume that this singularity locus is defined in the output plane (y_1, y_2) by the following equation:

$$S(\mathbf{y}, \mathbf{g}) = 0 \tag{3.56}$$

where:

$$S(\mathbf{y}, \mathbf{g}) := \det \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{bmatrix} = \frac{\partial f_1}{\partial y_1}\frac{\partial f_2}{\partial y_2} - \frac{\partial f_1}{\partial y_2}\frac{\partial f_2}{\partial y_1} \tag{3.57}$$

For a given geometry $\mathbf{g}$ of the mechanism, Equation (3.56) defines a set of singularity curves in the (y_1, y_2) plane. The concrete shape of these curves depends on the geometry $\mathbf{g}$. Assume that, for a given non-generic geometry $\mathbf{g}_0$, the singularity curves exhibit an isolated point at $\mathbf{y}_0$. Next, S will be approximated by its second-order Taylor expansion about $(\mathbf{y}_0, \mathbf{g}_0)$:

$$\begin{aligned} S(\mathbf{y}, \mathbf{g}) \approx S(\mathbf{y}_0, \mathbf{g}_0) + \left[\frac{\partial S}{\partial \mathbf{y}}(\mathbf{y}_0, \mathbf{g}_0)\right] \Delta\mathbf{y} + \\ + \left[\frac{\partial S}{\partial \mathbf{g}}(\mathbf{y}_0, \mathbf{g}_0)\right] \Delta\mathbf{g} + [\Delta\mathbf{y}^T, \Delta\mathbf{g}^T] \, \frac{\mathbf{H}(\mathbf{y}_0, \mathbf{g}_0)}{2} \begin{bmatrix} \Delta\mathbf{y} \\ \Delta\mathbf{g} \end{bmatrix} \end{aligned} \tag{3.58}$$

where $\mathbf{H}$ is the (symmetric) Hessian matrix of S with respect to $\mathbf{y}$ and $\mathbf{g}$, $\Delta\mathbf{y} = \mathbf{y} - \mathbf{y}_0$ and $\Delta\mathbf{g} = \mathbf{g} - \mathbf{g}_0$. Note that $S(\mathbf{y}_0, \mathbf{g}_0) = 0$ because the point $\mathbf{y}_0$ belongs to the singularity curves corresponding to the geometry $\mathbf{g}_0$. Moreover, since $\mathbf{y}_0$ is an isolated point (thus, a critical or special point) of these curves, then:

$$\frac{\partial S}{\partial \mathbf{y}}(\mathbf{y}_0, \mathbf{g}_0) = [0, 0] \tag{3.59}$$

which justifies the need for a second-order expansion [otherwise, the following Equation (3.60) would not define a curve in the output plane]. Substituting (3.58) into Equation (3.56) yields the equation defining the singularity locus near the isolated singular point $\mathbf{y}_0$ and near $\mathbf{g}_0$:

$$\mathbf{S_g}\Delta\mathbf{g} + [\Delta\mathbf{y}^T, \Delta\mathbf{g}^T] \, \frac{\mathbf{H}(\mathbf{y}_0, \mathbf{g}_0)}{2} \begin{bmatrix} \Delta\mathbf{y} \\ \Delta\mathbf{g} \end{bmatrix} = 0 \tag{3.60}$$

where $\mathbf{S_g} = \frac{\partial S}{\partial \mathbf{g}}(\mathbf{y}_0, \mathbf{g}_0)$. Next, the Hessian $\mathbf{H}$ is partitioned as follows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{H}_{12}^T & \mathbf{H}_{22} \end{bmatrix} \tag{3.61}$$

where the sizes of $\mathbf{H}_{11}$, $\mathbf{H}_{12}$ and $\mathbf{H}_{22}$ are 2×2, $2 \times d$ and $d \times d$, respectively. Using this partition of $\mathbf{H}$, Equation (3.60) can be rewritten as follows:

$$[\Delta\mathbf{y}^T, 1] \underbrace{\begin{bmatrix} \mathbf{H}_{11}/2 & \mathbf{K} \\ \mathbf{K}^T & u \end{bmatrix}}_{\mathbf{C}} \begin{bmatrix} \Delta\mathbf{y} \\ 1 \end{bmatrix} = 0 \tag{3.62}$$

where:

$$\mathbf{K} = \frac{\mathbf{H}_{12}\Delta\mathbf{g}}{2} \quad \text{and} \quad u = \left(\Delta\mathbf{g}^T \frac{\mathbf{H}_{22}}{2} + \mathbf{S_g}\right)\Delta\mathbf{g} \tag{3.63}$$

Equation (3.62) defines a conic in the output plane (y_1, y_2). The type of conic defined depends on the coefficient matrix $\mathbf{C}$ [175]. Note that $\mathbf{C}$ depends on the perturbation $\Delta\mathbf{g}$ from the non-generic geometry $\mathbf{g}_0$. Thus, to study how the perturbations in the geometry of the mechanism affect the stability of the isolated singularity $\mathbf{y}_0$, we only need to study and classify the type of conic defined by $\mathbf{C}$ in terms of $\Delta\mathbf{g}$.

In the next sections, we will illustrate this method with different isolated singularities of parallel mechanisms.

3.6.2 Example 1: Lips Singularity

Before using the proposed method to analyze more formally the stability of the isolated quadruple singularity f_4 studied in Section 3.5.1, in this first example we will illustrate the application of the proposed Taylor-based method to analyze the stability of an isolated lips singularity of the 3R$\underline{\text{P}}$R robot. In this way, we will be able to compare later the results obtained by our method for lips singularities and quadruple singularities.

Before we begin, let us parameterize here the position of the mobile platform of the 3R$\underline{\text{P}}$R robot using parameters different from those used in Section 3.4.1. In Section 3.4.1, we parameterized the position of the mobile platform of the 3R$\underline{\text{P}}$R robot using the (x, y) coordinates of joint B (see Figure 3.14a). However, in this section it will be much more convenient to parameterize the position of the mobile platform using the polar coordinates (ρ_3, θ_3) of its joint E (see Figure 3.14a). This is because, for the reasons explained in Section 3.5, it is necessary to keep constant one joint coordinate in order to work with a 2-degrees-of-freedom parallel mechanism. Since we will keep constant the length ρ_3 (as we have done in Section 3.5), the mathematical analysis of this example will be simpler if we parameterize the position of the mobile platform using the polar coordinates (ρ_3, θ_3) (similarly, if we kept constant ρ_1 or ρ_2 instead of ρ_3, it would be more convenient to parameterize the position of the mobile platform using the polar coordinates of joints B or D, respectively). As in Section 3.4.1, we will still parameterize the orientation of the mobile platform by angle ϕ.

If we assume that ρ_3 is kept constant, then we get a 2-DOF parallel mechanism with inputs $\mathbf{x} = [\rho_1, \rho_2]^T$, outputs $\mathbf{y} = [\theta_3, \phi]^T$ and geometric parameters $\mathbf{g} = [c_2, c_3, d_3, l_1, l_3, \beta, \rho_3]^T$ (ρ_3 can be considered as a geometric parameter since it is kept constant in this example). Next, we will obtain the two scalar equations of the general formulation [system (3.55)] that relate $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{g}$. To this end, we write the following vector loop equations (based on Figure 3.14a):

$$\mathbf{AB} = \mathbf{AF} + \mathbf{FE} + \mathbf{EB} = \begin{bmatrix} c_3 + \rho_3 \cos\theta_3 - l_3 \cos\phi \\ d_3 + \rho_3 \sin\theta_3 - l_3 \sin\phi \end{bmatrix} \tag{3.64}$$

$$\mathbf{CD} = \mathbf{CF} + \mathbf{FE} + \mathbf{ED} = \begin{bmatrix} c_3 - c_2 + \rho_3 \cos\theta_3 + l_1 \cos(\phi + \pi - \beta) \\ d_3 + \rho_3 \sin\theta_3 + l_1 \sin(\phi + \pi - \beta) \end{bmatrix} \tag{3.65}$$

Next, we impose the condition that the lengths of the previous vectors $\mathbf{AB}$ and $\mathbf{CD}$ must be ρ_1 and ρ_2, respectively. This yields the following equations:

$$\left\| \begin{bmatrix} c_3 + \rho_3 \cos\theta_3 - l_3 \cos\phi \\ d_3 + \rho_3 \sin\theta_3 - l_3 \sin\phi \end{bmatrix} \right\|^2 - \rho_1^2 = 0 \tag{3.66}$$

$$\left\| \begin{bmatrix} c_3 - c_2 + \rho_3 \cos\theta_3 + l_1 \cos(\phi + \pi - \beta) \\ d_3 + \rho_3 \sin\theta_3 + l_1 \sin(\phi + \pi - \beta) \end{bmatrix} \right\|^2 - \rho_2^2 = 0 \tag{3.67}$$

The left-hand sides of Equations (3.66) and (3.67) are the constraint functions f_1 and f_2 in this example. If we had parameterized the position of the mobile platform using the (x, y) coordinates of joint B (as we did in Section 3.4.1) instead of the polar coordinates of joint E, then x and y would become dependent when keeping constant ρ_3, and we would need to manipulate Equations (3.29)-(3.31) in order to simultaneously eliminate one of these equations and x (or y). On the contrary, by parameterizing the position of the mobile platform using the polar coordinates (ρ_3, θ_3) of joint E (and considering that ρ_3 is constant), we obtain directly the two constraint equations (3.66) and (3.67) without having to eliminate any equation or variable. This convenient polar parameterization of the position of the mobile platform of the 3R$\underline{\text{P}}$R parallel robot was proposed in [206].

The singularity locus in the (θ_3, ϕ) plane of this mechanism is defined by the following equation:

$$S(\mathbf{y}, \mathbf{g}) = \frac{\partial f_1}{\partial \theta_3}\frac{\partial f_2}{\partial \phi} - \frac{\partial f_1}{\partial \phi}\frac{\partial f_2}{\partial \theta_3} = 0 \tag{3.68}$$

The concrete shape of the singularity locus defined by Equation (3.68) will depend on the value of the geometric parameters $\mathbf{g}$. Next, we will analyze the singularity locus for the following non-generic geometry: $\mathbf{g}_0 = [1.4, 2, -1.5, 1.06, 1.1, 5.65 \text{ rad}, 2.800304375]^T$. This geometry is non-generic since it corresponds with a singularity locus that exhibits the following isolated point: $\mathbf{y}_0 = [1.953791747, 1.571336043]^T$ rad (see Figure 3.28). This isolated point is a higher-order *lips* singularity (see Section 3.5), which is unstable because, when perturbing the geometry of the mechanism away from its non-generic design $\mathbf{g}_0$, it either transforms into a bicuspid closed curve in the input plane or is destroyed (depending on the direction of such perturbation). As discussed in Section

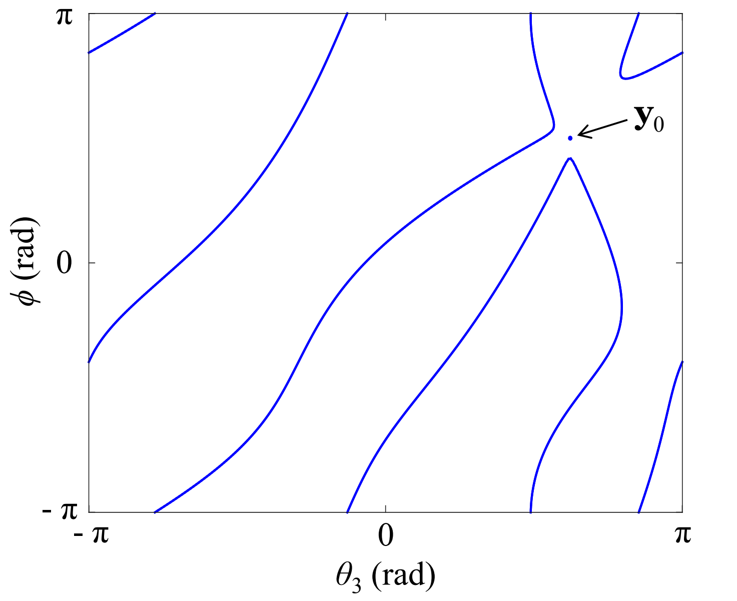


Figure 3.28: Representation of the singularity locus in the (θ_3, ϕ) plane of a 3R$\underline{\text{P}}$R parallel robot with non-generic geometry.

3.5, this destruction implies the loss of two cusps and, as a consequence, the local loss of the ability to perform nonsingular transitions. Next, we will apply the method described in subsection 3.6.1 in order to determine how the isolated singularity $\mathbf{y}_0$ transforms depending on how the geometry of the mechanism deviates from $\mathbf{g}_0$.

Next, assume that all geometric parameters suffer a small perturbation $\Delta\mathbf{g} = [\Delta c_2, \Delta c_3, \Delta d_3, \Delta l_1, \Delta l_3, \Delta\beta, \Delta\rho_3]^T$ which deviate them from the non-generic geometry $\mathbf{g}_0$ indicated in the previous paragraph. Substituting $\mathbf{y}_0$ and $\mathbf{g}_0$ into Equation (3.62), we obtain the equation of a conic curve which is an approximation of the perturbed singularity locus in the (θ_3, ϕ) plane, where:

$$\frac{\mathbf{H}_{11}}{2} = \begin{bmatrix} -11.8582 & 0.6271 \\ 0.6271 & -2.2934 \end{bmatrix} \tag{3.69}$$

$$\mathbf{K} = \begin{bmatrix} 1.9288 & -4.4937 \\ -2.6264 & 4.4935 \\ 14.9892 & -6.2409 \\ 8.8979 & 0.0256 \\ -22.3009 & 8.0256 \\ -3.0089 & 0.0440 \\ 14.3325 & -7.4679 \end{bmatrix}^T \Delta\mathbf{g} \tag{3.70}$$

$$\begin{aligned} u = {} & -2.5537\Delta c_2\Delta c_3 + 3.5688\Delta c_2\Delta d_3 \\ & + 7.9784\Delta c_2\Delta l_1 - 13.481\Delta c_2\Delta l_3 + 6.2140\Delta c_2\Delta\beta \\ & + 3.3911\Delta c_2\Delta\rho_3 - 2.4456\Delta c_2 + 2.5536\Delta c_3^2 \\ & - 5.4854\Delta c_3\Delta d_3 - 7.9347\Delta c_3\Delta l_1 + 16.382\Delta c_3\Delta l_3 \\ & - 6.2154\Delta c_3\Delta\beta - 6.1244\Delta c_3\Delta\rho_3 + 2.4422\Delta c_3 \\ & - 2.6245\Delta d_3^2 - 7.0749\Delta d_3\Delta l_1 + 6.6182\Delta d_3\Delta l_3 \\ & + 13.546\Delta d_3\Delta\beta - 3.9281\Delta d_3\Delta\rho_3 - 3.1068\Delta d_3 \\ & + 3.8633\Delta l_1\Delta l_3 - 0.013636\Delta l_1\Delta\beta - 3.5808\Delta l_1\Delta\rho_3 \\ & + 0.044585\Delta l_1 - 16.061\Delta l_3\Delta\beta + 1.4638\Delta l_3\Delta\rho_3 \\ & + 4.0521\Delta l_3 - 0.023632\Delta\beta^2 + 14.882\Delta\beta\Delta\rho_3 \\ & - 0.014455\Delta\beta - 1.3549\Delta\rho_3^2 - 3.7941\Delta\rho_3 \end{aligned} \tag{3.71}$$

The type of conic defined by Equation (3.62) depends on $\mathbf{H}_{11}$, $\mathbf{K}$, and u [175]. Firstly, since $\det(\mathbf{H}_{11}) > 0$, then the perturbed singularity locus is a (real or imaginary) ellipse. The type of ellipse defined by Equation (3.62) will depend on the sign of $\omega = c_{11}\det(\mathbf{C})$, where c_{11} is the first element of the first row of $\mathbf{C}$: if $\omega > 0$, then Equation (3.62) defines an imaginary ellipse, whereas this ellipse is real if $\omega < 0$. If $\omega = 0$, this ellipse degenerates into a single isolated point. The perturbation $\Delta\mathbf{g}$ of the geometric parameters will determine the sign of ω and, therefore, will determine the type of ellipse into which the isolated point $\mathbf{y}_0$ transforms when the geometry of the mechanism slightly deviates from $\mathbf{g}_0$.

3.6.2.1 Perturbation of a single geometric parameter

For simplicity, consider first that we only perturb parameter ρ_3, i.e., $\Delta\mathbf{g} = [0,\ 0,\ 0,\ 0,\ 0,\ 0,\ \Delta\rho_3]^T$. In that case:

$$\omega = (1206.22 - 11406.38\Delta\rho_3)\Delta\rho_3 \tag{3.72}$$

If we plot Equation (3.72) in Figure 3.29, we can identify three cases for sufficiently small perturbations $\Delta\rho_3$:

- If $\Delta\rho_3 < 0$, then $\omega < 0 \rightarrow$ the singularity locus is a real ellipse.
- If $\Delta\rho_3 > 0$, then $\omega > 0 \rightarrow$ the singularity locus is an imaginary ellipse.
- If $\Delta\rho_3 = 0$, then $\omega = 0 \rightarrow$ the singularity locus is a real ellipse that has degenerated into point $\mathbf{y}_0$.

Therefore, if we depart from the non-generic value of ρ_3 and slightly decrease it, the isolated singularity $\mathbf{y}_0$ will transform into a small ellipse E_r in the (θ_3, ϕ) plane. If we depart from $\Delta\rho_3 < 0$ and we continuously increase ρ_3 so that $\Delta\rho_3$ tends to zero from the left, the size of E_r continuously decreases, until this small real ellipse degenerates again into point $\mathbf{y}_0$. If the perturbation $\Delta\rho_3$ is further increased and becomes positive,

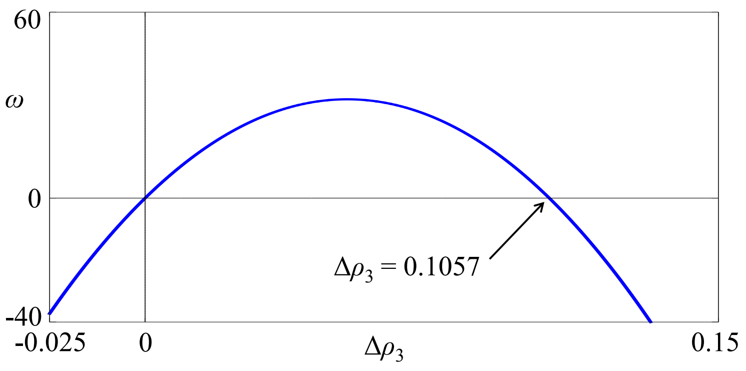


Figure 3.29: Variation of ω with $\Delta\rho_3$.

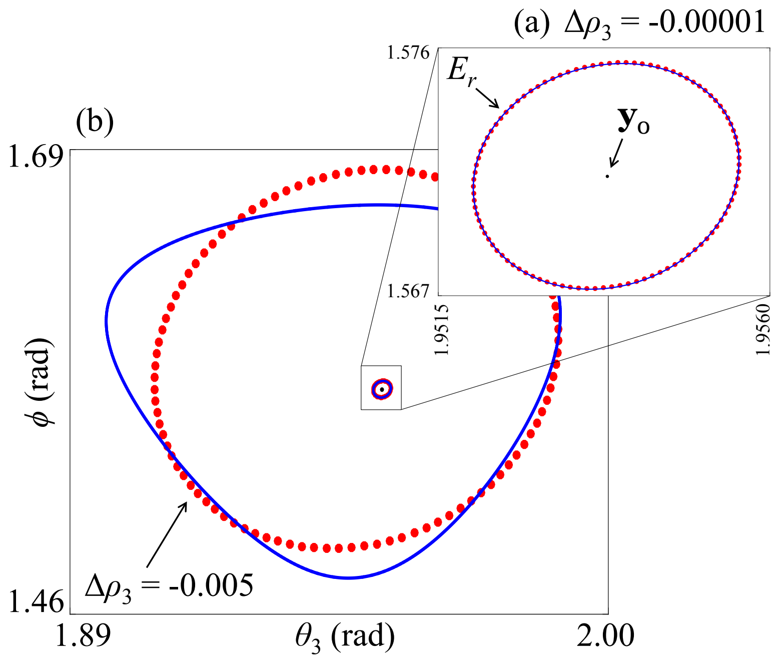


Figure 3.30: When ρ_3 decreases slightly, the isolated point $\mathbf{y}_0$ transforms into a closed curve which can be approximated by an ellipse.

then point $\mathbf{y}_0$ transforms into an imaginary ellipse, i.e., $\mathbf{y}_0$ disappears from plane (θ_3, ϕ).

Figure 3.30a illustrates the transformation of $\mathbf{y}_0$ into an approximately elliptic closed curve E_r for the perturbation $\Delta\rho_3 = -0.00001$: the ellipse defined by Equation (3.62) is represented in red dotted line, whereas the exact singularity locus [i.e., the curve defined by Equation (3.68)] is represented in continuous blue line. Note that Equation (3.62) is an accurate approximation of the exact singularity locus for sufficiently small perturbations, whereas this approximation is not good for large perturbations (for example, see Figure 3.30b, where $\Delta\rho_3 = -0.005$).

By solving the inverse kinematics of this mechanism [i.e., solving ρ_1 and ρ_2

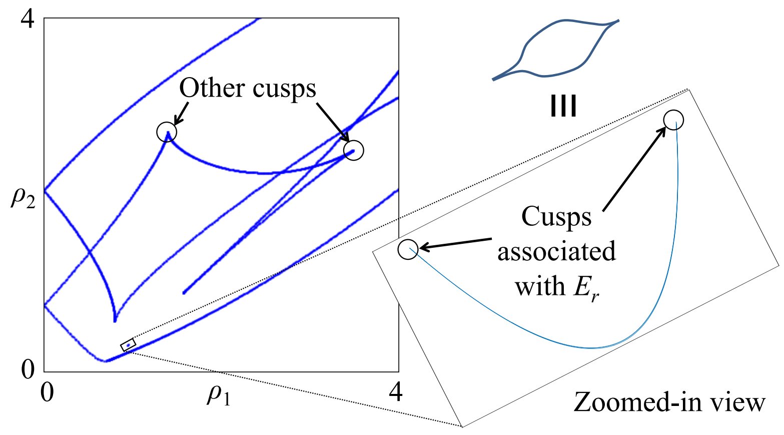


Figure 3.31: Singularity locus of a perturbed 3R$\underline{P}$R robot with $\Delta\rho_3 = -0.00001$. The figure shows a zoomed view of the closed bicuspid curve which is the image of real ellipse E_r of Figure 3.30a in (ρ_1, ρ_2) plane.

from Equations (3.66) and (3.67) for given θ_3 and ϕ], it is possible to transform the real ellipse E_r to the input plane (ρ_1, ρ_2). The image of ellipse E_r in the input plane is a small closed curve with two cusps (see Figure 3.31). As described in Section 3.2, these cusps allow the mechanism to reconfigure itself between different assembly modes without crossing singularities. Therefore, the destruction of the real ellipse E_r (when $\omega > 0$) implies the destruction of these cusps, and the mechanism loses this ability (reconfiguring without crossing singularities) in the region of the (ρ_1, ρ_2) plane near the closed bicuspid curve of Figure 3.31. (However, as it can be observed in this figure, the singularity locus of this mechanism exhibits other additional cusps which are not destroyed along with E_r; these additional cusps still enable nonsingular transitions after E_r vanishes.)

Note that, according to Figure 3.29, ω becomes again negative for $\Delta\rho_3 > 0.1057$, which implies that the real ellipse E_r defined by Equation (3.62) reappears again for $\Delta\rho_3 > 0.1057$. This may erroneously suggest that the exact singularity locus [defined by Equation (3.68)] should also exhibit a small (approximately elliptic) closed curve in plane (θ_3, ϕ) for $\Delta\rho_3 > 0.1057$, due to the reappearance of real ellipse E_r. However, this is not true since perturbation $\Delta\rho_3 = 0.1057$ is too large for Equation (3.62) to be a valid approximation of the exact singularity locus. Therefore, the analysis of the sign of ω in Equation (3.72) is only valid for sufficiently small values of $|\Delta\rho_3|$.

3.6.2.2 Perturbation of two geometric parameters

Next, assume that both ρ_3 and c_2 are perturbed, i.e.: $\Delta\mathbf{g} = [\Delta c_2, 0, 0, 0, 0, 0, \Delta\rho_3]^T$. In that case, the expression of ω is:

$$\omega = -2811.780\Delta c_2^2 - 10847.38\Delta c_2\Delta\rho_3 + 777.4378\Delta c_2 - 11406.38\Delta\rho_3^2 + 1206.22\Delta\rho_3 \quad (3.73)$$

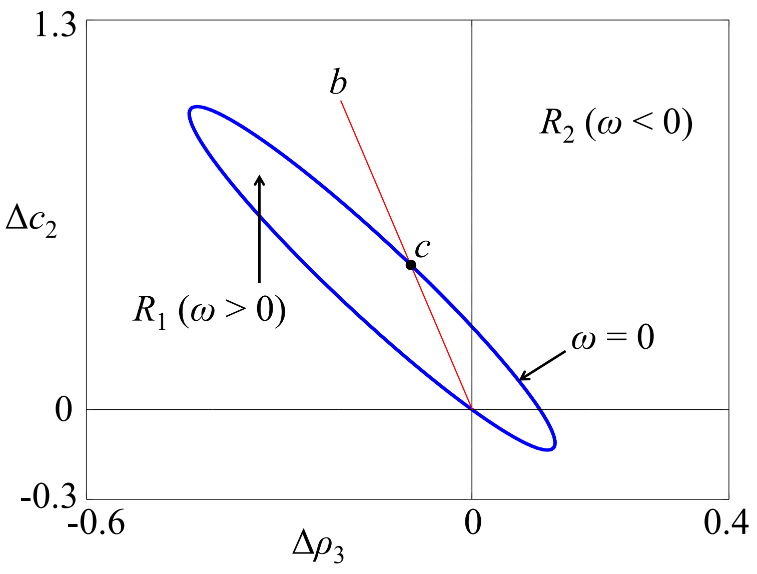


Figure 3.32: Variation of the sign of ω in terms of perturbations $\Delta\rho_3$ and Δc_2.

Figure 3.32 shows the variation of the sign of ω in terms of the perturbations $\Delta\rho_3$ and Δc_2. The condition $\omega = 0$ defines an ellipse that divides the $(\Delta\rho_3, \Delta c_2)$ plane into two regions: region R_1 interior to this ellipse, and region R_2 exterior to this ellipse. Since $\omega < 0$ in region R_2, Equation (3.62) defines a real ellipse for perturbations falling in this exterior region. I.e., for $(\Delta\rho_3, \Delta c_2) \in R_2$, the isolated point $\mathbf{y}_0$ deforms into a closed curve with an approximately elliptic shape, which in turn transforms into a bicuspid curve when mapped to the (ρ_1, ρ_2) plane.

For perturbations belonging to region R_1 we have $\omega > 0$, which means that Equation (3.62) defines an imaginary ellipse. This means that point $\mathbf{y}_0$ disappears, and the robot loses the ability to reconfigure between different solutions of the forward kinematic problem without crossing singularities. This ability is lost only locally because, as shown in Figure 3.31, the singularity locus of this robot exhibits other cusps that are not destroyed along with $\mathbf{y}_0$.

Finally, it is important to emphasize again that the behavior of the exact singularity locus [defined by Equation (3.68)] under large perturbations cannot be predicted by analyzing the transformations suffered by the ellipse defined by Equation (3.62). For example, if we depart from the non-generic geometry $\mathbf{g}_0$ (i.e., from the origin $\Delta\rho_3 = \Delta c_2 = 0$) and we perturb these two geometric parameters along segment b (see Figure 3.32), when crossing point c (i.e., when passing from region R_1 to region R_2) we will not observe the appearance of any (approximately elliptic) closed curve in the exact singularity locus, despite the fact that the ellipse defined by Equation (3.62) switches from imaginary to real. This is because crossing point c requires perturbations so large that render the quadratic approximation of Equation (3.62) invalid.

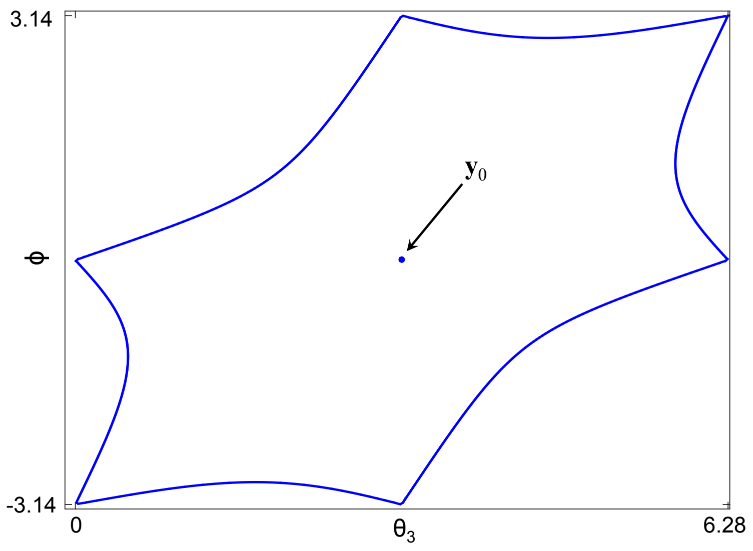


Figure 3.33: Singularity locus of a 3R$\underline{\text{P}}$R robot with flat non-similar platforms.

3.6.3 Example 2: Isolated Quadruple Singularity

After using the proposed method to analyze the transformations suffered by a lips singularity under perturbations of the geometry of the 3R$\underline{\text{P}}$R robot, in this second example we will apply this method to analyze more formally the stability of the isolated quadruple singularity f_4 that was analyzed in Section 3.5. As in Section 3.5.1, we will assume that the robot has the non-generic geometry $\mathbf{g}_0 = [1.5, 0.5, 0, 0.5, 0.5, \pi \text{ rad}, 1]^T$ (i.e., both platforms are exactly flat and non-similar, and the length ρ_3 is kept constant and set to $\rho_3 = 1$). The singularity locus in the (θ_3, ϕ) plane corresponding to this non-generic geometry is represented in Figure 3.33. This singularity locus exhibits an isolated point at $\mathbf{y}_0 = [\pi, 0]$ rad. Note that the image of $\mathbf{y}_0$ in the plane of inputs (ρ_1, ρ_2) is the isolated point f_4 analyzed in Section 3.5.1.

Next, we will apply the proposed Taylor-based method to analyze the stability of this isolated singularity. Consider that all geometric parameters suffer a small deviation from $\mathbf{g}_0$, i.e.: $\Delta\mathbf{g} = [\Delta c_2, \Delta c_3, \Delta d_3, \Delta l_1, \Delta l_3, \Delta\beta, \Delta\rho_3]^T$. Substituting $\mathbf{y}_0$ and $\mathbf{g}_0$ into Equation (3.62), which approximates the singularity locus in the output plane near $\mathbf{y}_0$, yields:

$$\frac{\mathbf{H}_{11}}{2} = \begin{bmatrix} 1 & -0.25 \\ -0.25 & 1.5 \end{bmatrix} \tag{3.74}$$

$$\mathbf{K} = [1.5\Delta d_3 - 0.5\Delta\beta, -2.5\Delta d_3 - 0.75\Delta\beta]^T \tag{3.75}$$

$$u = 5\Delta d_3 \Delta\beta - 4\Delta d_3^2 \tag{3.76}$$

Although all the geometric parameters are perturbed, according to Equations (3.75) and (3.76), the transformation of $\mathbf{y}_0$ depends only on the perturbations of d_3 and β, which are precisely the only two geometric parameters that determine whether $\mathbf{g}_0$ is a generic geometry or not (since these two parameters determine if the fixed and mobile

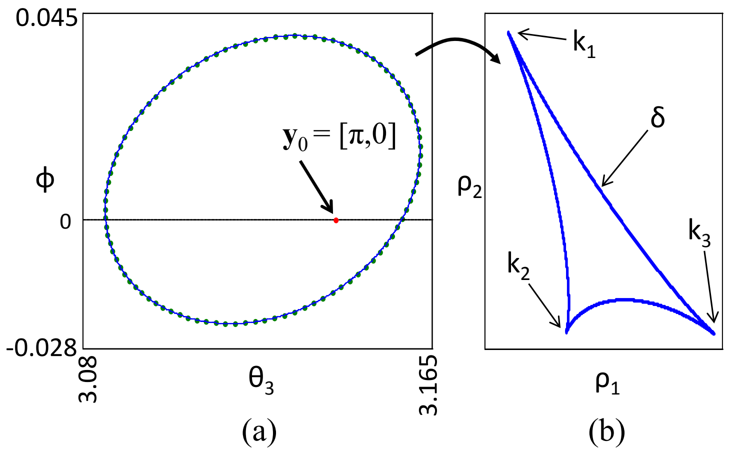


Figure 3.34: (a) (Approximately elliptic) singularity locus near $\mathbf{y}_0$ when the non-generic geometry of a 3R$\underline{\text{P}}$R robot with flat platforms is slightly perturbed ($\Delta d_3 = 0.01$, $\Delta\beta = -0.01$). (b) The image of this ellipse in the input plane is a deltoid δ, with cusps: $k_1 \approx (0.9995, 1.5014)$, $k_2 \approx (0.9999, 1.4999)$ and $k_3 \approx (1.0009, 1.4999)$.

platforms are perfectly flat). In contrast to this, in the previous example we observed that the transformation of the isolated lips singularity depended on the perturbations of all geometric parameters [see Equations (3.70) and (3.71)].

Since $\det(\mathbf{H}_{11}) > 0$, Equation (3.62) defines a real or imaginary ellipse, depending on the sign of $\omega = c_{11}\det(\mathbf{C})$:

$$\omega = -1.125(12\Delta d_3^2 - 5\Delta d_3\Delta\beta + \Delta\beta^2) \tag{3.77}$$

ω in Equation (3.77) is a negative definite quadratic form, i.e., $\omega < 0\ \forall(\Delta d_3, \Delta\beta) \neq (0,0)$. Therefore, if any of the two geometric parameters $\{d_3, \beta\}$ deviates from its non-generic value, then Equation (3.62) defines a real ellipse in the output plane, independently of the direction of these perturbations. This means that the isolated point $\mathbf{y}_0$ of the exact singularity locus *always* deforms into a small loop that can be approximated by an ellipse if the perturbations are sufficiently small. If this ellipse is mapped to the input plane (ρ_1, ρ_2), then it transforms into a deltoid δ (see the example of Figure 3.34), which is the small deltoid that was observed in Figure 3.25.

In this way, the proposed method based on second-order Taylor expansions completes the analysis started in Section 3.5.2, and demonstrates more formally that, unlike a lips singularity, the deltoid δ cannot be destroyed by any combination of perturbations from the non-generic geometry $\mathbf{g}_0$: in the analyzed 3R$\underline{\text{P}}$R robot, these perturbations *always* transform the isolated point $\mathbf{y}_0$ into a real ellipse, and the image of this real ellipse in the input plane is the deltoid δ.

3.6.4 Example 3: Exceptions of the Method

It is important to emphasize that the proposed Taylor-based method is only valid if the perturbed singularity locus in the output plane (y_1, y_2) near the isolated singularity

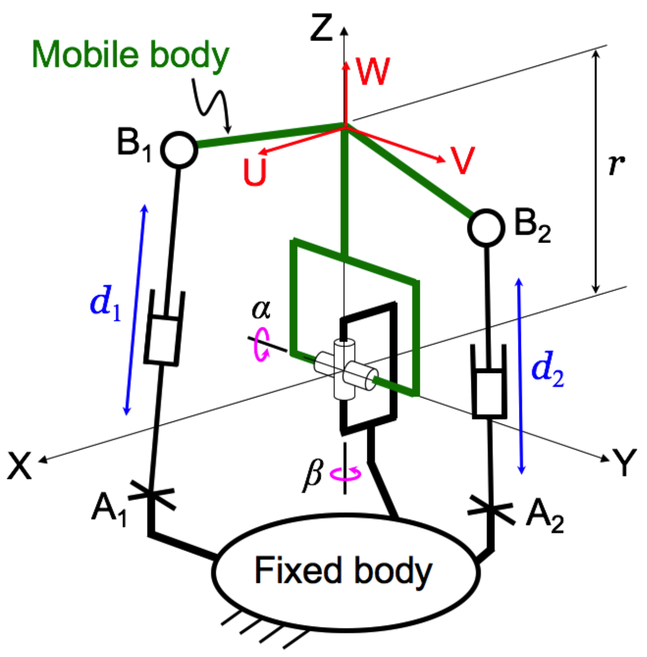


Figure 3.35: A 2U$\underline{\text{P}}$S-U parallel robot.

$\mathbf{y}_0$ can be approximated by a small ellipse. In that case, the proposed method can be used for studying how the corresponding isolated singularity $\mathbf{x}_0$ in the input plane (x_1, x_2) (which is the preimage of $\mathbf{y}_0$ under the forward kinematics) transforms into a small multi-cusped closed curve under perturbations, as previous examples have demonstrated.

However, it may occur that a parallel robot has an isolated singularity $\mathbf{x}_0$ in the input plane (which transforms into a multi-cusped closed curve under perturbations) which is not the preimage of an isolated singularity $\mathbf{y}_0$ in the output plane (and, therefore, it does not transform into a small ellipse under perturbations), being instead the preimage of a curve. In that case, the proposed Taylor-based method cannot be used.

For example, consider a 2U$\underline{\text{P}}$S-U parallel robot as the one shown in Figure 3.35. This robot has mobile and fixed platforms connected through a universal passive joint, as well as two U$\underline{\text{P}}$S legs with controllable lengths (d_1 and d_2). The inputs of this robot are these lengths (d_1, d_2), whereas its outputs are angles (α, β) associated to the passive universal joint that directly connects both platforms, as depicted in Figure 3.35.

When both universal joints A$_1$ and A$_2$ of the U$\underline{\text{P}}$S legs of this robot belong to axis X, then the singularity locus in the input plane (d_1, d_2) exhibits two isolated singularities, which are indicated in Figure 3.36a. When the design of the robot is slightly perturbed so that A$_1$ and A$_2$ do not belong to axis X simultaneously, then these two isolated singularities transform into small "diamonds" (see Figure 3.36b), which are closed curves that have four cusps each [38]. According to some experiments performed

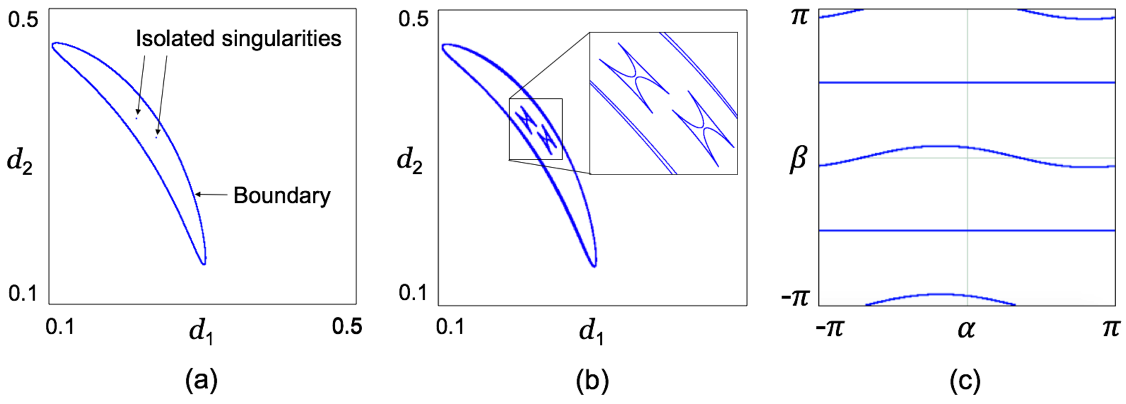


Figure 3.36: Special singularities of a non-generic 2U$\underline{\text{P}}$S-U robot. The details of this example (e.g., numerical values of the geometric parameters of the robot) are omitted here, but can be found in [130].

with the simulator presented in [130], this perturbation seems stable, in the sense that perturbing A_1 or A_2 in any direction (so that these two joints do not simultaneously lie on axis X) always transforms these two isolated singularities into small diamonds, never destroying them.

As we did in previous examples, we might try to apply the proposed Taylor-based method to rigorously study this apparent stability observed experimentally, for any possible perturbation. However, this will not be possible in this example, because these isolated singularities are not preimages of isolated singularities of the output plane (α, β). Figure 3.36c represents the parallel singularities of this robot in plane (α, β), which shows four disjoint curves (two non-straight and two horizontal straight curves). The preimage of each of the non-straight curves of Figure 3.36c is the complete boundary indicated in Figure 3.36a, which encloses both isolated singularities. The preimage of each of the horizontal straight singular curves of Figure 3.36c is one of the isolated singularities of Figure 3.36a.

This is a well-known special finite-motion singularity also observed in other robots [27], in which the inputs are locked but the mobile platform is free to perform finite displacements (in this example, these finite displacements correspond to complete revolutions of the mobile platform about axis X [130]). In the next chapter, we will return to this special singularity.

Note that, since the images of these "point diamonds" (i.e., diamonds shrunk to points) in the output plane (α, β) are not points but horizontal straight lines, the Taylor-based method proposed in this section cannot be used for studying their stability.

3.7 Conclusions

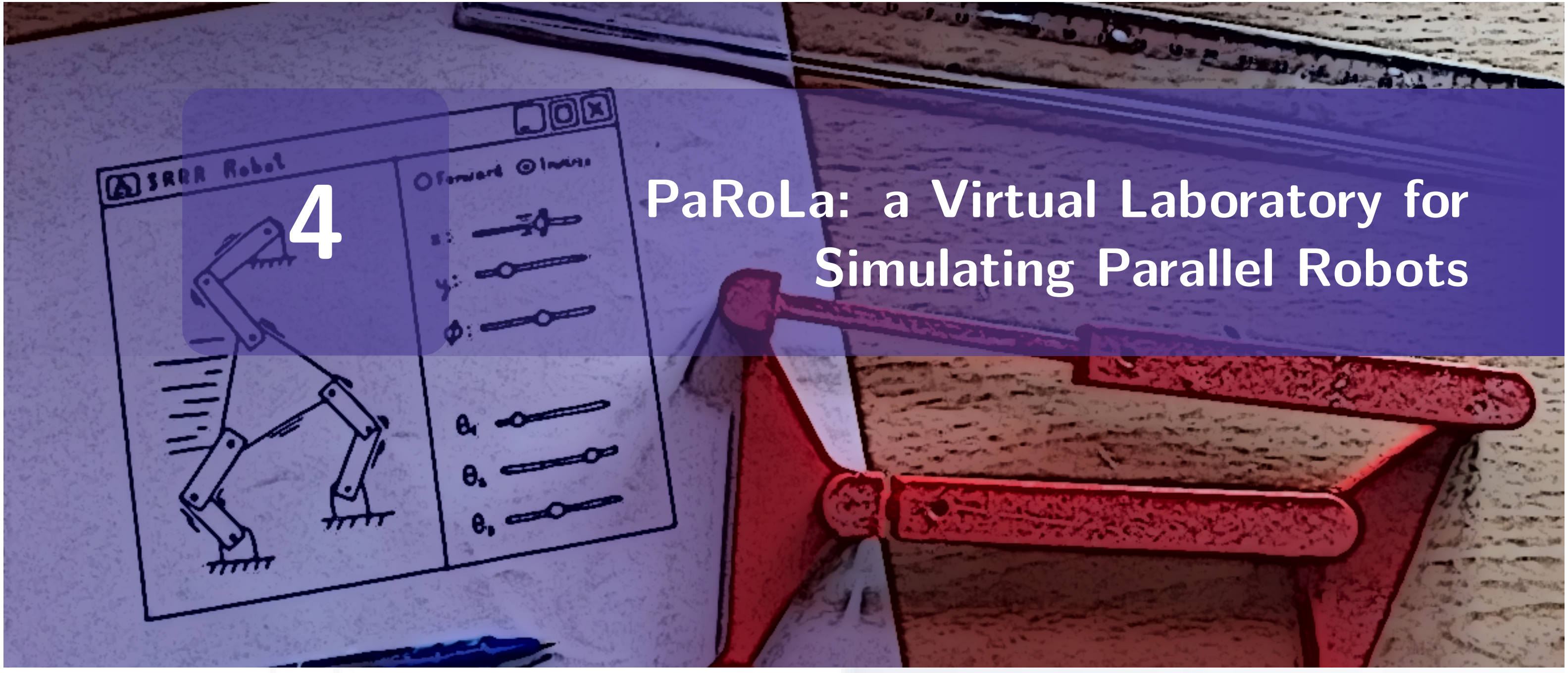
In this chapter, we have thoroughly analyzed the kinematics of two classes of analytic parallel mechanisms of types 2R$\underline{\text{P}}$R-PR and 3R$\underline{\text{P}}$R, and we have demonstrated that they are able to perform nonsingular transitions by describing closed trajectories that enclose isolated singularities at which the forward kinematic problem of these mechanisms admits quadruple solutions (Sections 3.1, 3.2, and 3.4). This may seem counter-intuitive at first since the singularity loci of these mechanisms apparently lack cusps and α-curves (the only two generic singularities that are known to enable nonsingular transitions), although it was later demonstrated in [40] that there are indeed "hidden cusps" in this case: these isolated quadruple singularities actually are small deltoids (i.e., closed curves with three cusps) degenerated into points. Thus, encircling the isolated quadruple singularity actually means encircling three "hidden" coinciding cusps. This deltoid emerges when the geometry of the mechanism is slightly perturbed away from the special design that gives rise to isolated quadruple singularities, and we have demonstrated that encircling the deltoid has practically the same effect as encircling the isolated quadruple singularities, regarding the occurrence of nonsingular transitions (Section 3.5.2). Finally, in Section 3.6 we have presented a method based on second-order Taylor expansions to predict how isolated singularities of parallel mechanisms will transform under any perturbation in their geometric design, and we have illustrated this method with two types of isolated singularities: lips and quadruple singularities.

3.8 Publications Related to this Chapter

The main results presented in this chapter are related to the following publications:

- A. Peidró, J.M. Marín, A. Gil, and O. Reinoso. Performing nonsingular transitions between assembly modes in analytic parallel manipulators by enclosing quadruple solutions. *ASME Journal of Mechanical Design*, 137(12):122302, 2015 [140] **(SCI-JCR Impact Factor: 1.444, Q2)**.
  - This paper presents the forward kinematic analysis of analytic 2RPR-PR and 3RPR-PR robots presented in Sections 3.1, 3.2, and 3.4. In this paper, the special quadruple singularities of these two analytic parallel robots were identified, and it was demonstrated that encircling them in the actuated joint space produces nonsingular transitions.
- A. Peidró, A. Gil, J.M. Marín, L. Payá, and Ó. Reinoso. On the stability of the quadruple solutions of the forward kinematic problem in analytic parallel robots. *Journal of Intelligent & Robotic Systems*, 86(3):381–396, 2017 [135] **(SCI-JCR Impact Factor: 1.583, Q3)**.
  - This paper presents the stability analysis of quadruple singularities presented in Section 3.5 for the 3R$\underline{P}$R robot, in which it was found that nonsingular transitions are still produced when encircling a small deltoid instead of an isolated quadruple singularity f_4.
- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Second-order Taylor stability analysis of isolated kinematic singularities of closed-chain mechanisms. In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics - Volume 2*, pages 351–358. INSTICC, SciTePress, 2017 [148].

– This paper presents the Taylor-based method presented in Section 3.6 for predicting how isolated singularities of parallel mechanisms transform under perturbations of their design.



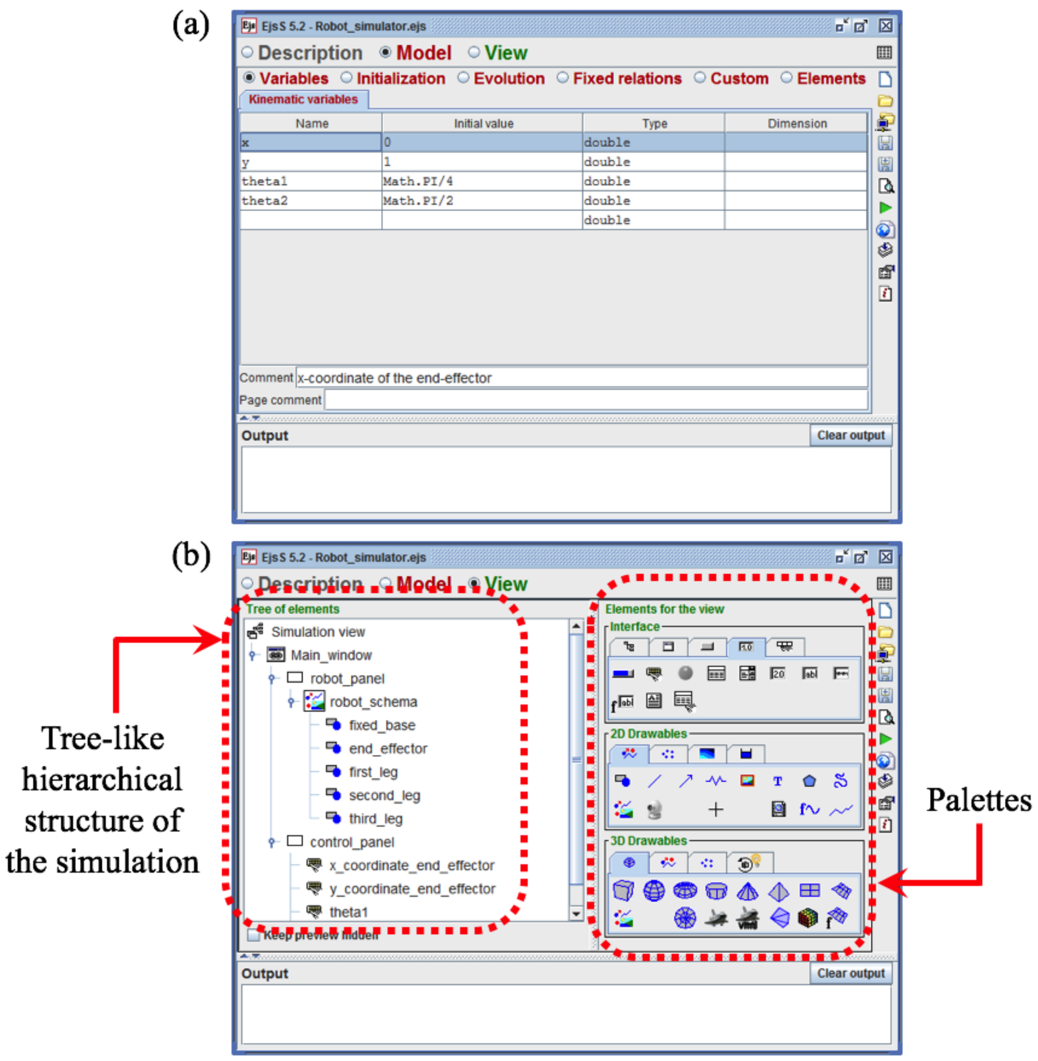
4 PaRoLa: a Virtual Laboratory for Simulating Parallel Robots

In parallel to the development of this thesis, a virtual and remote laboratory called PaRoLa (Parallel Robotics Laboratory) has been developed. This laboratory consists of a collection of simulators and graphical interfaces that allow the user to experiment with several simulated or real parallel robots (remotely), with the purpose of understanding and learning about different kinematic and dynamic notions of these robots. The original objective of developing this virtual and remote laboratory was to build simulation tools for supporting many of the analyses performed during the present thesis: from the kinematic analyses presented in the previous chapter to the analyses that will be presented in later chapters. After developing these tools, we decided to freely publish them on the Internet, so that any user interested in parallel robots can use them either for learning or research purposes.

This chapter is organized as follows. Section 4.1 briefly introduces Easy Java Simulations, which is the authoring tool that has been used in this thesis to develop PaRoLa. Section 4.2 introduces the virtual and remote laboratory PaRoLa. Then, Section 4.3 describes the virtual and remote robots available in this virtual laboratory. Finally, Section 4.4 describes the functionalities of this virtual laboratory, hinting some of their possible applications through several examples.

4.1 Easy Java Simulations

The simulators described in this chapter were developed by means of the authoring tool Easy Java Simulations [54]. Easy Java Simulations (EJS) is an authoring tool for developing graphical interactive simulations in Java, mainly with educational purposes. The main advantage of Easy Java Simulations is evident from its name: *easy* means that EJS greatly simplifies the development of graphical simulators, reducing the programming effort of the author of the simulations. EJS abstracts the process of programming graphical user interfaces, such that the author can focus on programming the behavior of the simulated system (in our case, the kinematics and dynamics of parallel robots) instead of "getting lost" with low-level implementation details of the graphical interface. EJS accomplishes this by means of a very intuitive graphical user interface divided mainly into two parts: *view* and *model* (see Figure 4.1).


(a)
EjsS 5.2 - Robot_simulator.ejs
Description [✓ Model] View
[✓ Variables] Initialization Evolution Fixed relations Custom Elements
Kinematic variables

Name	Initial value	Type	Dimension
x	0	double	
y	1	double	
theta1	Math.PI/4	double	
theta2	Math.PI/2	double	
		double	

Comment: x-coordinate of the end-effector

Figure 4.1: (a) Model and (b) view parts of Easy Java Simulations.

The *model* part contains most of the "logic" that models the behavior of the simulation. The model part comprises the following sections or tabs (see Figure 4.1a):

- **Variables tab:** consists of several tables where the user defines the global variables of the simulation. These variables will be accessible and manipulable from any other part of the simulation. For each global variable, the user specifies its name, initial value, type of variable (which can be any Java primitive type such as int, double, or Boolean, but it can be also a predefined or user-defined Java class), and its size (for variables which are multidimensional arrays).

- **Initialization tab:** contains the code that is run when initiating or resetting the simulation.
- **Evolution tab:** contains code which is run cyclically, as long as the simulation is in "running" state. This section can contain differential equations which model the dynamics of a system (EJS automatically integrates these equations numerically, using Runge-Kutta or other methods). Also, this section can contain plain text Java code pages that are sequentially executed. If the simulator is in "pause" state, the code pages under this tab are ignored. However, this does not mean that the simulator is doing nothing: even when the simulator is paused, it responds to user interaction.
- **Fixed relations tab:** contains code which is *continuously* executed if the state of the simulation is "running" instead of "paused". If the simulation is paused, all code under this tab is executed whenever the user interacts with the simulation (e.g., when clicking a graphical window of the simulation, or when rotating a camera or pressing a button, etc.). Since it is not possible to accurately control when the code under this tab is executed, it is advisable to move all this code to user-defined functions under the "Custom" tab, calling these functions whenever these are necessary (e.g., whenever the user presses a button).
- **Custom tab:** under this tab, the user can define general-purpose custom functions to model the behavior of the simulation. For example: in the context of parallel robots, here we could define a function which would be summoned when dragging the end-effector of the robot, and this function would solve the inverse kinematic problem. All functions defined under this tab are globally accessible from any other part of the code of the simulation.
- **Elements tab:** this is a library of external or accessory elements to enrich or augment the simulation. For example, under this tab we find blocks for interfacing our simulation with Arduino. Actually, this tab has not been necessary during the development of this thesis.

As for the *view* part, it contains two parts (see Figure 4.1b): a white panel on the left and three palettes on the right. These palettes contain several elements which can be used to build the graphical user interface of the simulation, such as windows, panels, buttons, 2D- or 3D-plots, geometric shapes, etc. The user can drag and drop these elements from the palettes into the white panel on the left, in order to build the interface hierarchically, following a tree-like structure. For example (see Figure 4.1b): first, one would define a window, which would be divided into several panels. Then, each panel would contain a different thing: one of them would contain a 2D plot to visualize the robot, other panel would contain buttons and sliders for modifying the joint coordinates of the robot, etc. By means of these palettes and the drag-and-drop philosophy, EJS greatly facilitates the task of building the graphical user interface of the simulation.

After defining hierarchically the graphical user interface of our simulation, EJS allows us to configure the elements of our interface, defining their position/orientation, their size and color, their interaction capabilities (e.g., the code that should be executed when clicking on some part of the robot), etc. Moreover, all elements available on the *view* part of the simulation are completely linked to the *model* part. This means that, for example, we can associate the position of a geometric shape (of the view part) with a variable defined in the model part. Or it is possible to call functions defined under the "Custom" tab of the model part when interacting with some element of the view part. This interconnection between both parts of the simulation (model and view) allows the user to build fairly sophisticated interactive simulations with relatively little programming effort, which is the main advantage of Easy Java Simulations.

This chapter will not explain further details on how to implement simulations in EJS, for two reasons. Firstly, there already exist some tutorials of EJS [53]. Secondly, we consider that EJS is so intuitive that the user can autonomously master it after using it for just a few hours or days: the learning curve of EJS is really smooth.

As explained above, EJS comprises two parts (model and view) which are intimately intertwined. This means that, when building a simulation, the graphical user interface with which the user interacts can be entirely developed using the elements of the *view* part, whereas all variables and functions necessary to "animate" the graphical elements of the interface can be entirely defined using the different tabs under the *model* part. However, from our experience, EJS seems to "slow down" when the model part is overloaded (e.g., when we write too much code under the "Custom" tab), which may end up hindering the development of the simulation.

To avoid this, during the development of PaRoLa, we have been progressively migrating most of the code from the model part of our simulations to external Java libraries. In this way, EJS can be used to define the graphical/interactive part of the simulation (which is the part at which EJS is really powerful), summoning most of the functions and code from linked Java libraries which contain custom-made functions for simulating the kinematics or dynamics of parallel robots. This alleviates a great deal of workload from EJS, avoiding undesirable lags or malfunctioning of this tool.

Note that this solution (migrating most of the code to external Java libraries) also favors a more efficient and modular way of programming simulators, since many of the parallel robots implemented in PaRoLa share many problems and calculations that can be solved by a single routine available in an external Java library (instead of implementing repeatedly the same calculations once and again for different robots, which is not efficient). A clear example of this can be found in the inverse kinematics of the 5R and 3$\underline{R}$RR robots (these robots will be introduced later, in Section 4.3.1): the inverse kinematic problem for the legs of both these robots is identical. Thus, instead of programming several times the resolution of these problems (once per each leg of these robots), it is more efficient to program a general solver routine in an external library and then summon this solver from each individual robot.

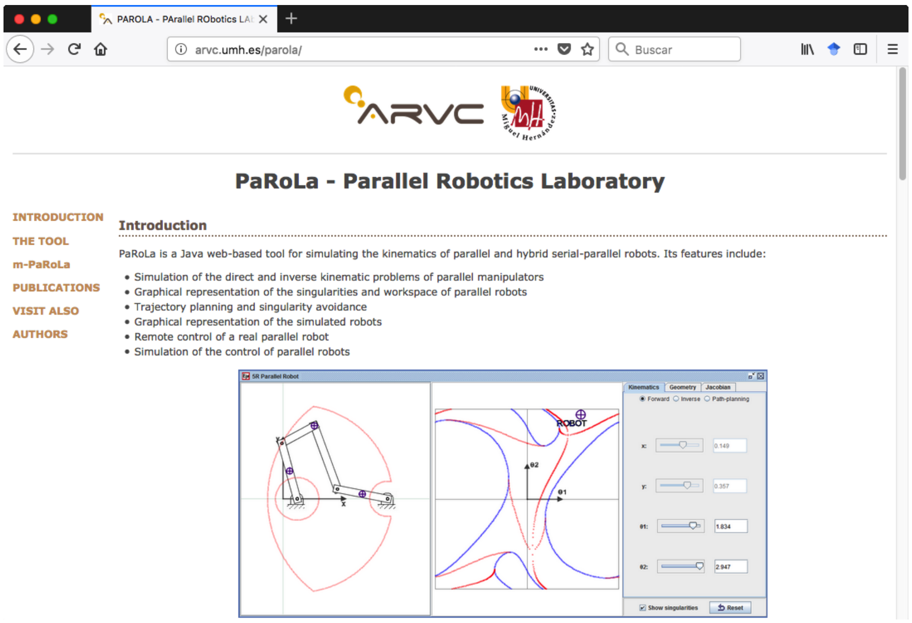
Finally, everything said above is only valid for the Java *flavour* of EJS. Originally, EJS was aimed at developing simulations in Java, since these simulations were very common in the Internet in the past, in the form of educational applets embedded in webpages. However, during the recent years, most popular web browsers have been progressively abandoning their support to Java applets, due to security issues. Therefore, nowadays it is impossible to execute Java applets in webpages: it is necessary to download these simulators (in the form of `.jar` packages) into a computer with Java installed and run them. To solve this, since its version 5.0, Easy Java Simulations (now called Easy Java/Javascript Simulations) can be used to develop simulations both in Java and Javascript programming languages. The advantage of simulators based on Javascript is that they can be run on any current web browser, both on desktop computers and on mobile devices (such as tablets and smartphones). Most of the simulation tools presented in this chapter have been developed in Java, although recently we have also started to migrate these simulations to Javascript, as explained in next section.

4.2 PaRoLa

By using EJS, a virtual and remote laboratory called PaRoLa has been developed in this thesis for studying parallel robots (as well as other serial and hybrid robots, like the HyReCRo robot studied in this thesis), with educational and research purposes. PaRoLa can be accessed at `http://arvc.umh.es/parola`, which lands on the website shown in Figure 4.2. In this website, the user can download the `.jar` files of the available robots, which can be run on any desktop computer with Java installed (it works on Mac, Windows, and Linux). In the next sections, we will describe the robots currently available in PaRoLa, as well as the functionalities and applications offered by this tool.

As explained above, considering that Java applets are no longer supported by web browsers, recently we have started to migrate PaRoLa to a Javascript-based version, by means of the latest versions of EJS (version 5.0 and above). The objective will be to "translate" the Java simulations currently available in PaRoLa into Javascript versions. However, this translation is not straightforward, since it requires practically remaking the complete simulation from scratch. In [153], we presented m-PaRoLa, which is the beginning of a Javascript version of PaRoLa. The name "m-PaRoLa" is due to the fact that, since the simulators are based on Javascript, they can be run on any web browser, including those present in mobile devices such as smartphones or tablets. This allows for the use of these simulations in mobile learning (m-learning) methodologies. Currently, m-PaRoLa only allows the user to simulate the forward/inverse kinematic problems of 5R and 3$\underline{R}$RR robots, as well as visualize their singularities and workspace.

To avoid having to remake all simulators from scratch, and with the purpose of recycling most of the code written for the Java simulators of PaRoLa, in the future


PAROLA - PArallel RObotics LA

arvc.umh.es/parola/

Buscar

PaRoLa - Parallel Robotics Laboratory

INTRODUCTION
THE TOOL
m-PaRoLa
PUBLICATIONS
VISIT ALSO
AUTHORS

Introduction

PaRoLa is a Java web-based tool for simulating the kinematics of parallel and hybrid serial-parallel robots. Its features include:

- Simulation of the direct and inverse kinematic problems of parallel manipulators
- Graphical representation of the singularities and workspace of parallel robots
- Trajectory planning and singularity avoidance
- Graphical representation of the simulated robots
- Remote control of a real parallel robot
- Simulation of the control of parallel robots

Figure 4.2: Website of the virtual laboratory PaRoLa.

we will also explore the solution proposed by Saenz et al. [164], who propose using a client-server architecture in which the client is the Javascript simulator of the robot, whereas the server contains the Java code already developed for the Java version of PaRoLa.

4.3 Robots Available in PaRoLa

PaRoLa mainly includes parallel robots, although it also includes hybrid robots (such as the HyReCRo robot) and other climbing robots (like the 3DCLIMBER robot [178]). Moreover, it comprises both simulated robots and real remote robots, which can be remotely controlled over the Internet (these remote robots are installed in the laboratories of the Automation, Robotics and Computer Vision research group of the Miguel Hernández University in Elche, Spain). Despite this, in this chapter we will especially focus on the simulators of the parallel robots, since they are more important for previous and later chapters of this thesis.

4.3.1 Parallel Robots Implemented

Currently, the following parallel robots can be simulated in PaRoLa. These robots, shown in Figure 4.3, are sorted in increasing number of degrees of freedom (DOF):

- **Four-bar mechanism** (Figure 4.3a): strictly speaking, this is not a parallel robot, but it is the simplest closed-chain mechanism possible. The main interest of studying and incorporating this robot to PaRoLa is twofold:
  - On the one hand, the position problem of this mechanism (which consists in solving all three angles θ_1, θ_2, and θ_3 of Figure 4.3a given one of these

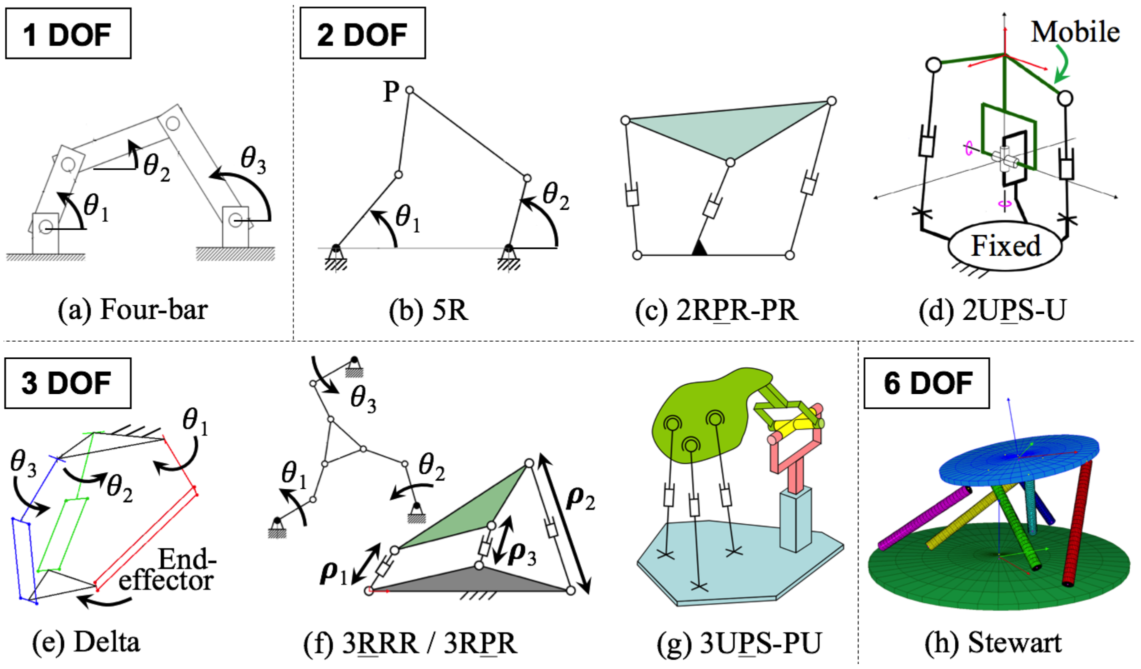


Figure 4.3: Simulated parallel robots implemented in PaRoLa.

angles) appears in the forward and inverse kinematic analyses of many other parallel robots which are more complicated.

– On the other hand, since this mechanism only has three variables (angles θ_1, θ_2, and θ_3), we can represent its complete configuration space in a three-dimensional space (the configuration space is the set of triplets $[\theta_1, \theta_2, \theta_3]$ that result in feasible assemblies of the mechanism). Robots with two or more degrees of freedom (like the rest of the robots implemented in PaRoLa) have more than three kinematic variables, so it is not possible to represent all their kinematic variables in three dimensions (for example: a 5R robot has two input variables and two output variables, which would require a four-dimensional space to represent their assemblable 4-tuples). Therefore, for other robots it is necessary to project these configuration spaces into three-dimensional spaces, obtaining *reduced configuration spaces* like those studied in Figures 3.10, 3.13 and 3.18 of the previous chapter.

- **5R robot** (Figure 4.3b): this is a five-bar mechanism with two degrees of freedom, in which angles θ_1 and θ_2 are controlled with the purpose of controlling the planar position of point P, which is the end-effector of this robot.

- **2RPR-PR robot** (Figure 4.3c): these are the two-DOF parallel mechanisms used in the legs of the HyReCRo robot, and they have been comprehensively analyzed in the previous chapter.

- **2UPS-U mechanism** (Figure 4.3d): this is a two-DOF mechanism composed of two bodies (one fixed and another mobile) interconnected through a passive universal joint (which constrains the relative motion between these two bodies) and two legs of type U$\underline{\text{P}}$S (which do not constrain their relative motion). The objective of this mechanism is to control the relative orientation between these two bodies by regulating the length of the U$\underline{\text{P}}$S legs. This robot appeared in Section 3.6.4 of the previous chapter, which analyzed an exception to the Taylor-based method presented in Section 3.6.

- **Delta robot** (Figure 4.3e): this is the well-known parallel robot used for pick-and-place tasks in industry. This robot can be regarded as a three-dimensional version of the 5R robot. In the Delta robot, we control the spatial position of the end-effector by means of angles θ_1, θ_2, and θ_3.

- **3R$\underline{\text{P}}$R and 3$\underline{\text{R}}$RR robots** (Figure 4.3f): these are three-DOF planar parallel robots in which three identical legs are used for positioning and orienting a triangular mobile body in the plane. In the 3R$\underline{\text{P}}$R robot, these legs are of type R$\underline{\text{P}}$R and have variable length (its inputs are the lengths ρ_1, ρ_2, and ρ_3). In the 3$\underline{\text{R}}$RR robot, these legs are of type $\underline{\text{R}}$RR and the inputs are the angles $\{\theta_1$, θ_2, $\theta_3\}$, which are the orientations of the links connected to the fixed ground. The 3R$\underline{\text{P}}$R robot has been comprehensively studied in the previous chapter, with the purpose of analyzing its special quadruple singularities. As for the 3$\underline{\text{R}}$RR robot, this robot has been studied in this thesis mainly with the purpose of analyzing the interior barriers of its workspace, in chapter 7.

- **3U$\underline{\text{P}}$S-PU robot** (Figure 4.3g): this is a 3-DOF parallel robot which is part of the well-known serial-parallel robot Tricept used for machining [124]. Moreover, it can be regarded as a three-dimensional generalization of the 2R$\underline{\text{P}}$R-PR parallel module studied in the previous chapter, and therefore, it exhibits eightfold special singularities which are analogous to the singularities λ_0 and λ_π studied in the previous chapter [141].

- **Stewart platform, or 6U$\underline{\text{P}}$S robot** (Figure 4.3h): this is a 6-DOF parallel robot in which a mobile platform is connected to a fixed platform through six U$\underline{\text{P}}$S legs, whose lengths are controlled by means of linear actuators. By regulating the lengths of these six legs, it is possible to control the position and orientation of the mobile platform in three-dimensional space. In this thesis, this robot has been mainly used for analyzing the interior barriers of its workspace, in chapter 7.

Note that this collection of parallel robots is not arbitrary. It includes three important families of parallel robots used in industry: the Delta robot of Figure 4.3e (and its "planar version", which is the 5R robot of Figure 4.3b), the Tricept robot of Figure 4.3g (and its planar version, which is the 2R$\underline{\text{P}}$R-PR robot of Figure 4.3c), and the Stewart platform of Figure 4.3h (and its planar version, which is the 3R$\underline{\text{P}}$R robot of Figure 4.3f).

4.3.2 Other Robots Available in PaRoLa

In addition to the simulators of parallel robots described in the previous subsection, PaRoLa also houses simulators of climbing robots and remote prototypes of parallel robots:

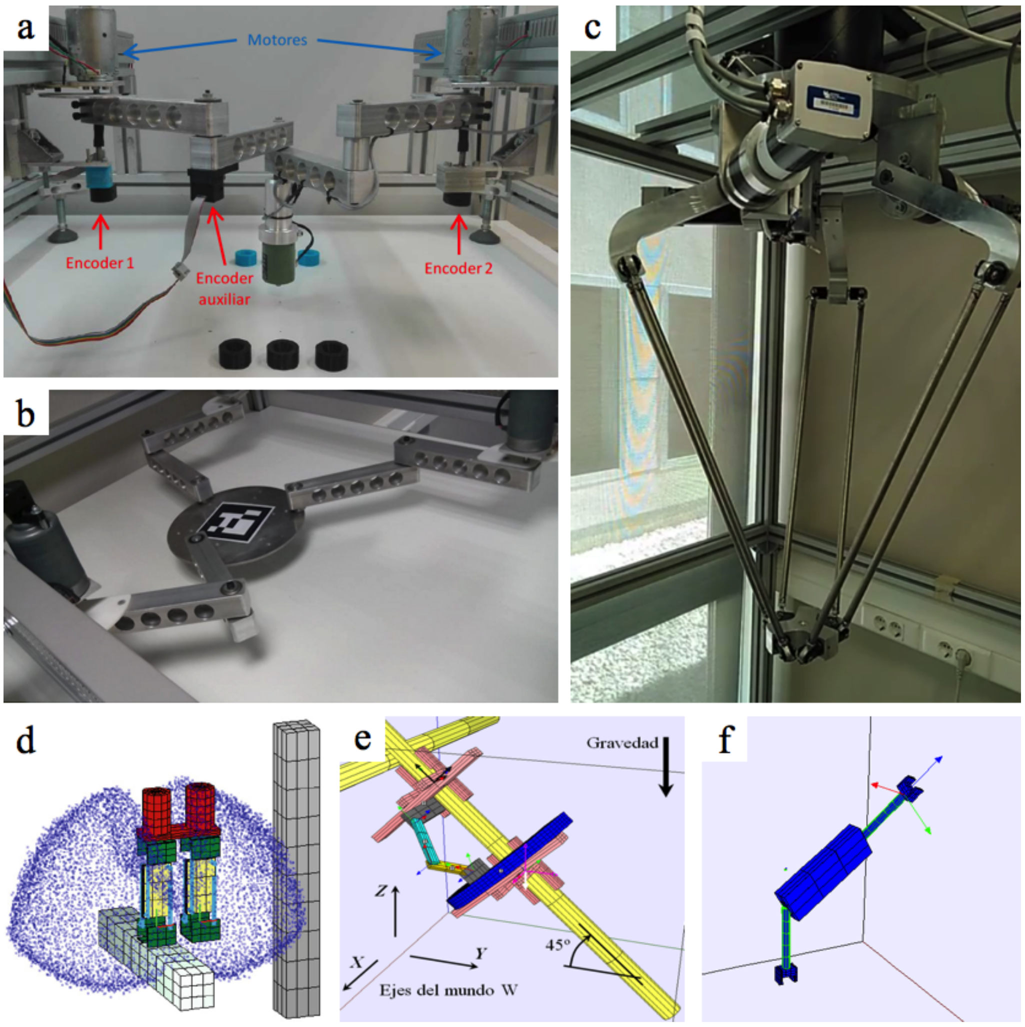


Figure 4.4: Other remote and simulated robots implemented in PaRoLa.

- **Remote 5R robot** (Figure 4.4a): PaRoLa allows the user to remotely control a real five-bar parallel robot. The user can visualize the motion of the robot in streaming through a webcam, while experimenting with trajectory planning and the Proportional-Integral-Derivative control of this robot.

- **Remote 3RRR robot** (Figure 4.4b): the objective of this prototype will be to study nonsingular transitions (by enclosing cusps) and singular transitions (by crossing singularities) with a real parallel robot, neglecting the effects of gravity (since the robot will move in a horizontal plane).

- **Remote Delta robot** (Figure 4.4c): this prototype will be used to study singular transitions (which are the only possible transitions between assembly modes in the Delta robot) considering the effects of gravity.
- **Simulator of the HyReCRo robot** (Figure 4.4d): during this thesis, a simulator of the HyReCRo robot has been developed with the purpose of studying the kinematics and workspace of this climbing robot. This simulator was developed as a support tool to aid in the analysis and study of this robot. Also, this tool can act as a graphical user interface for controlling the prototype of the HyReCRo robot, which will be described in chapter 8. The simulator of the HyReCRo robot will not be analyzed in this chapter, but it will be described in later chapters of this thesis, in which the kinematics and workspace of the HyReCRo robot will be analyzed (it is necessary to explain the kinematic problems of the HyReCRo robot before introducing the tool developed for simulating these problems).
- **Simulator of the 3DCLIMBER robot** (Figure 4.4e): PaRoLa also includes a simulator of the 3DCLIMBER climbing robot [178]. This simulator was developed with the purpose of facilitating the analysis of adhesion forces and torques required in the grippers of climbing robots [138].
- **Simulator of the ROMA robot** (Figure 4.4f): finally, PaRoLa also houses a simulator of the ROMA climbing robot [14]. The objective of developing this simulator was to compare this climbing robot with other similar climbing robots, in terms of workspace.

4.4 Functionalities of PaRoLa

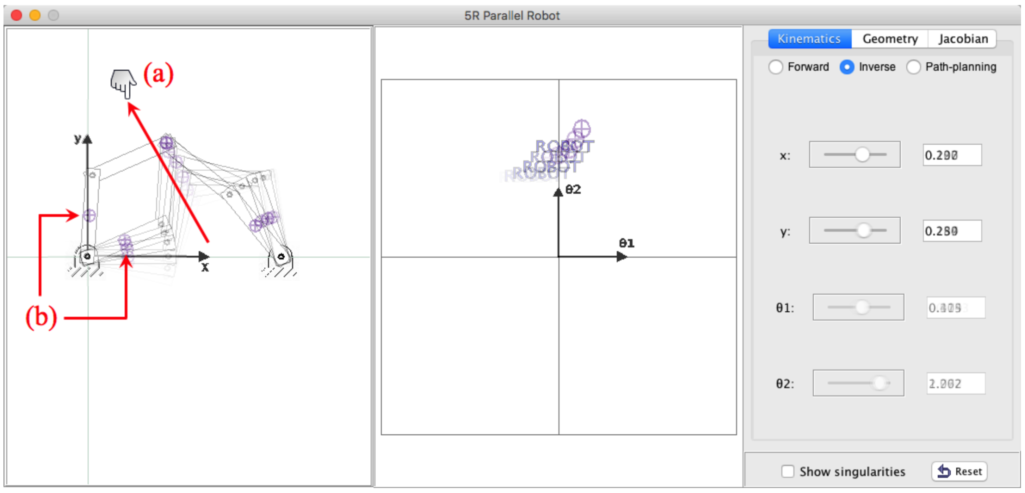
This section describes the main functionalities of the virtual laboratory PaRoLa. Although this virtual laboratory also includes remote robots, as well as climbing robots, in the remaining of this chapter we will focus only on the simulators of parallel robots. The main functionalities of PaRoLa are described in next subsections, illustrating these functionalities with examples.

4.4.1 Simulation of the Inverse Kinematics

The inverse kinematics consists in specifying the desired position and/or orientation for the end-effector of a parallel robot, and computing the values of the actuated joint coordinates necessary for attaining the desired position/orientation. The simulators of PaRoLa allow the user to specify the desired pose (position and orientation) for any parallel robot, obtaining as a result the required values of the actuated joint coordinates. The desired pose can be specified numerically or by means of sliders, or by directly clicking and dragging the end-effector of the robot in the simulator (Figure 4.5a).

In general, the inverse kinematic problem of parallel robots has several possible solutions (known as working modes): different values of the actuated joint coordinates that place the end-effector at the desired position and with the desired orientation.

In this aspect, PaRoLa allows the user to compare the different possible solutions for a desired pose of the end-effector, comparing the posture of the robot for different solutions (Figure 4.5b). To switch between different solutions of the inverse kinematics, the user must click the small circle placed on the center of the actuated links of the robots.



x: 0.290
y: 0.280
θ1: 0.809
θ2: 2.002

Figure 4.5: Simulating the inverse kinematics in PaRoLa. (a) Dragging the end-effector. (b) Switching between different solutions of the inverse kinematic problem.

4.4.2 Simulation of the Forward Kinematics

The forward kinematic problem consists in determining the position and orientation of the end-effector of a parallel robot for given values of its actuated joint coordinates. PaRoLa also allows the user to "move" the robot under "forward kinematics mode", where the user moves and drags the actuated joints of the robot, and the robot moves as a consequence (see Figure 4.6a). It is also possible to simulate the forward kinematics of the robots by dragging their actuated joint coordinates in the actuated joint space (e.g., see Figure 4.10d for the Delta Robot), or by introducing the values of the actuated joint coordinates by means of sliders and numeric boxes.

Also, recall from the previous chapter that the forward kinematic problem has several solutions, which are called assembly modes. All these assembly modes can be visualized for each parallel robot in PaRoLa (see Figure 4.6b). The different solutions can be visualized using two possible representations, as explained in the next subsection.

4.4.3 Multiple Visualizations of the Solutions of the Forward Kinematics

As illustrated in Figure 4.6, PaRoLa allows the user to visualize the complete posture of the robot for each of the solutions of the forward kinematics. PaRoLa uses two different representations to represent the solutions of the forward kinematics: the **complex**

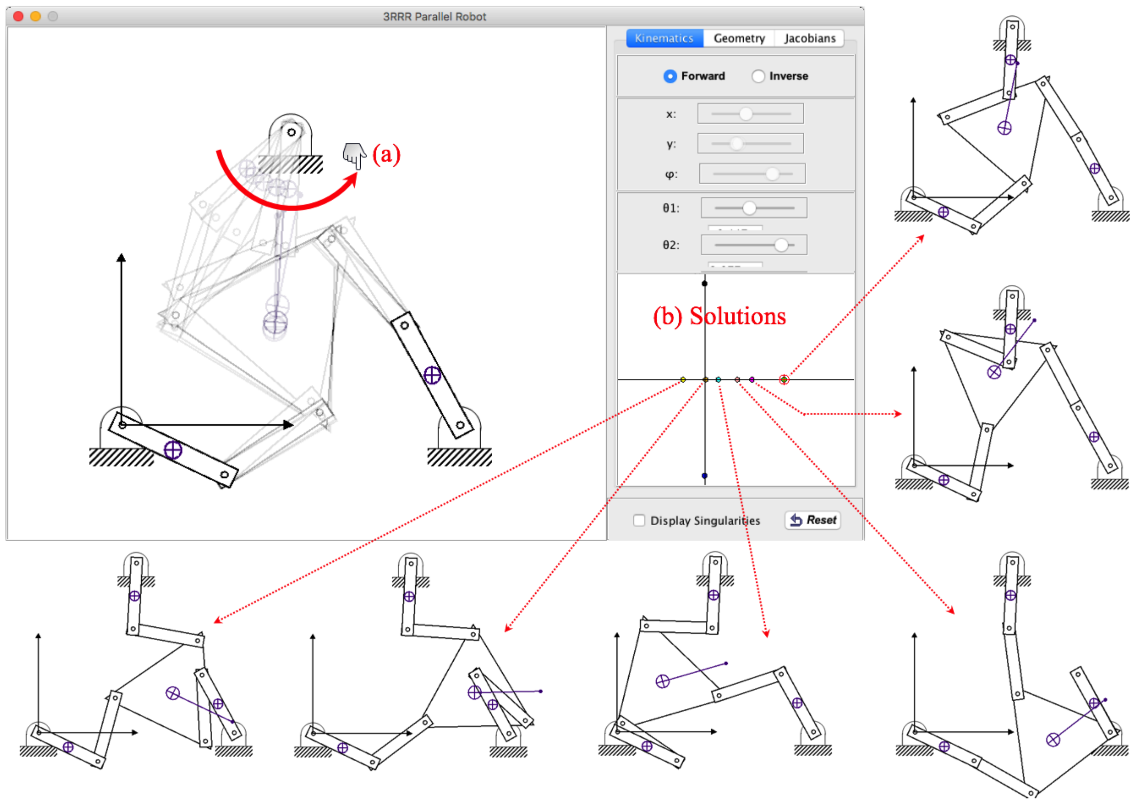


Figure 4.6: Simulating the forward kinematics in PaRoLa. (a) Dragging the actuated joints. (b) Switching between different solutions of the forward kinematic problem.

plane and the **real plane**. These representations are especially useful for analyzing how the different solutions of the forward kinematics evolve when varying the actuated joint coordinates, and this can be used for analyzing nonsingular transitions, as it has been done in the previous chapter. The two representations used in PaRoLa are explained next.

4.4.3.1 Visualizing solutions in the complex plane

All solutions are represented as colored points in the complex domain. For example: Figure 4.6b represents the solutions of the forward kinematics of the 3$\underline{\text{R}}$RR robot projected to complex plane (Re(ϕ), Im(ϕ)), where ϕ is the orientation angle of the triangular end-effector. When varying the actuated joint coordinates of the robot, these solutions describe trajectories in this complex plane. As explained in the previous chapter, "complex angles" can also be represented using modified polar coordinates or cylindrical coordinates (Figure 3.6). As explained also in the previous chapter, these two representations (cylindrical and modified-polar) are better than the rectangular representation for representing "complex angles", since they avoid the discontinuity due to the wrapping of the real part of the angle. In PaRoLa, the user can click on each solution in the complex domain, and then the simulator represents the posture of the clicked solution. Obviously, clicking on solutions with non-zero imaginary part will result in a wrong representation of the posture of the robot, since imaginary assembly modes have no physical meaning.

4.4.3.2 Visualizing solutions in the real plane

The second representation used in PaRoLa is valid only for robots with two degrees of freedom. In 2-DOF parallel robots, we have two inputs (two actuated joint coordinates) and two outputs (two variables that parameterize the position and/or orientation of the mobile platform). These inputs and outputs are related through two scalar equations, which encode the kinematic restrictions imposed by the robot. These two scalar equations can be regarded as defining two curves in the plane of outputs. Therefore, the real solutions of the forward kinematics are obtained as the intersections between these two planar curves.

Let us explain this representation with a concrete example. For example, for the 2R$\underline{\text{P}}$R-PR parallel robot analyzed in the previous chapter, the inputs are the lengths $\{l_1, l_2\}$, the outputs are $\{\phi, y\}$, and the two scalar equations are {Equation (3.1), Equation (3.2)}, which are repeated next:

$$(b_1 \cos\phi - a_1)^2 + (y + b_1 \sin\phi)^2 = l_1^2 \tag{4.1}$$

$$(-b_2 \cos\phi - a_2)^2 + (y - b_2 \sin\phi)^2 = l_2^2 \tag{4.2}$$

For given values of the geometric design parameters (a_1, a_2, b_1, and b_2), and for given values of the inputs (lenghts l_1 and l_2), the previous equations define two curves in plane (ϕ, y), as shown in Figure 4.7. The intersection points of these curves yield the real solutions of the forward kinematics (in this case, there are four different real solutions, which are denoted as in the previous chapter: σ_1, σ_2, σ_3, and σ_4). If we continuously modify the values of the inputs (l_1, l_2), these two curves are continuously deformed, and their intersection points vary as a consequence (the solutions of the forward kinematics move, as shown in Figures 4.7 and 4.8).

Like the representation in the complex plane, this representation of solutions as intersection points of curves in the real plane can be used for studying nonsingular transitions and special singularities. For example:

- **Analyzing nonsingular transitions:** Figure 4.7 represents the evolution of curves (4.1) and (4.2) when the actuated joint coordinates l_1 and l_2 describe the circular trajectory of Figure 3.4. As analyzed in the previous chapter, this circular trajectory encloses a special singularity λ_π of the 2R$\underline{\text{P}}$R-PR robot, which produces nonsingular transitions between solutions of the forward kinematics. This can be observed in Figure 4.7: solutions σ_3 and σ_4 swap their positions in the output plane after completing one turn around λ_π.
- **Analyzing special singularities**: Figure 4.8 represents the evolution of curves (4.1) and (4.2) when the actuated joint coordinates l_1 and l_2 approach the special singularity λ_π of the 2R$\underline{\text{P}}$R-PR robot through the vertical trajectory t_π shown in Figure 3.4. We observe that, when this special singularity is approached,

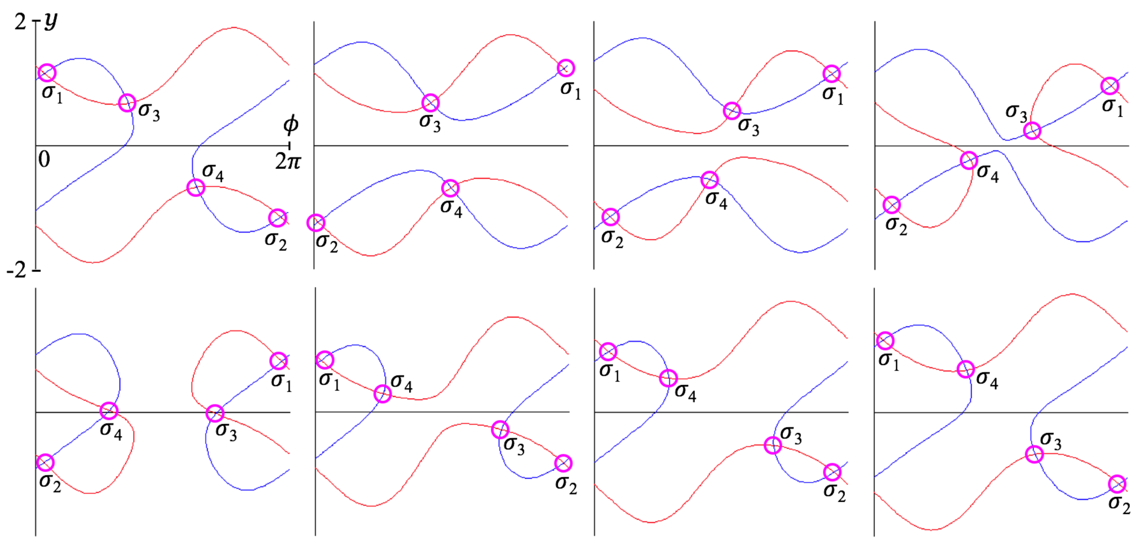


Figure 4.7: Evolution of the kinematic constraints defined by Equations (4.1) (red curve) and (4.2) (blue curve), when describing the circular trajectory shown in Figure 3.4. Solutions of the forward kinematics, which are the intersection points between the red and blue curves, are enclosed by magenta circles. Solutions σ_3 and σ_4 swap their positions along this trajectory, resulting in a nonsingular transition (see Section 3.2.1).

these curves have a special intersection: each curve self-intersects at ($\phi = \pi$, $y = 0$), where each curve also meets the other curve (see Figure 4.8). Thus, this special intersection corresponds to a higher-multiplicity solution of the forward kinematics.

What is the point in representing the solutions of the forward kinematics as the intersection of two planar curves? There are two advantages in graphically solving the forward kinematics. Firstly, plotting these curves is very easy and straightforward, using any mathematical package or software able to plot implicit curves (e.g.: "ezplot" function in Matlab), whereas numerically solving the forward kinematics can be quite tedious for many 2-DOF robots. For example: consider a general 3$\underline{\text{R}}$RR robot in which one of the input angles has been locked, so that the robot becomes a 2-DOF mechanism. Solving the forward kinematics of this robot by elimination leads to a sixth-degree polynomial whose coefficients are so lengthy that this approach may become impractical. On the contrary, plotting the curves whose intersections are the solutions of the forward kinematics is straightforward.

Nevertheless, the true reason for considering this representation of the solutions of the forward kinematics is that it allows one to analyze degenerate singular cases for which the first representation (solutions projected to the complex plane) fails. Typically, for parallel robots the forward kinematic problem has a zero-dimensional solution set, i.e., the solutions are a finite number of isolated discrete points in the complex domain. In that case, the representation based on projecting the solutions to the complex plane is valid. However, in some special cases, there exist also positive-dimensional solution sets (e.g., curves) in addition to these isolated solutions. In those cases, representing

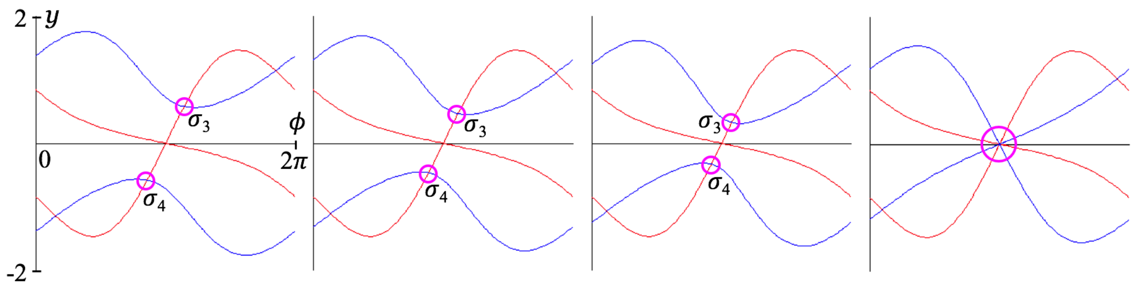


Figure 4.8: Evolution of the kinematic constraints defined by Equations (4.1) (red curve) and (4.2) (blue curve), when approaching singularity λ_π shown in Figure 3.4.

the solutions as a set of points in the complex domain obviously misses information. This will be better illustrated through the following example, which was presented in [130].

Consider again the 2U$\underline{\text{P}}$S-U robot analyzed in Section 3.6.4 of the previous chapter. In this robot, the inputs are the lengths d_1 and d_2 of the U$\underline{\text{P}}$S legs, whereas the outputs are the angles α and β that parameterize the relative orientation between the mobile and fixed platforms. As studied in Section 3.6.4, when both universal joints A_1 and A_2 of the U$\underline{\text{P}}$S legs tend to the X axis (Figure 4.9a), two diamond-like closed curves of the singularity locus shrink to points in plane $(d_1,\ d_2)$ (Figures 4.9b-d). Considering the analyses presented in the previous chatper, in which we analyzed the evolution of the assembly modes when approaching deltoids degenerated into points, one may wish to investigate how the solutions of the forward kinematics evolve when the joint coordinates approach the degenerate point diamonds of Figure 4.9d, e.g., through the vertical trajectory γ depicted in Figure 4.9d.

If one visualizes the evolution of the solutions in the complex plane of variable β (Figure 4.9e), then one observes that, apparently, four solutions seem to converge when approaching the point-diamond singularity (actually, there seem to be six converging solutions, since each of the U-shaped trajectories shown in Figure 4.9e is double, i.e., it is described by two coinciding complex solutions). However, if one observes the evolution of the solutions in the complex plane of α (not shown here), these solutions seem to move erratically, not converging at all. This apparently strange behavior can be understood if we visualize the solutions of the forward kinematics as the intersection of two curves in real plane $(\alpha,\ \beta)$, as Figures 4.9f-h show (these curves represent the kinematic restrictions of the 2U$\underline{\text{P}}$S-U robot [130]).

Figures 4.9f-h show the evolution of these curves as the joint coordinates approach the studied isolated singularity through vertical trajectory γ . Since this trajectory is vertical, d_1 is kept constant and the red curve, which only depends on d_1, does not change along the trajectory. This red curve consists of two intersecting portions: a horizontal segment and an (approximately) sinusoid portion (Figure 4.9f).

On the other hand, the blue curves, which only depend on d_2, are two loops. During the execution of trajectory γ, the red and blue curves intersect at four different points (and all four intersections are simple). However, as Figures 4.9f-h show, the blue curves deform and adopt a similar shape to the red curves as the isolated singularity is approached in plane (d_1, d_2). In the limit, when the isolated singularity is approached, the blue curves consist also of two intersecting portions, like the red curves: a horizontal segment and another (approximately) sinusoid portion. And this is where the previous apparently strange behavior of the complex solutions can be understood, as explained next.

The sinusoid portions of the red and blue curves of Figure 4.9h intersect at two points (and these intersections are simple). These are the simple isolated solutions properly represented in Figure 4.9e. However, the horizontal portions of these curves are coincident, i.e., these portions have infinitely many different intersection points. This means that, in addition to the two mentioned simple solutions, this degenerate mechanism admits infinitely many solutions with $\beta = -\pi/2$, with angle α being free: the mechanism can freely rotate about axis X, without control [130]. This is a special self-motion parallel singularity, in which the linear actuators are locked (lengths d_1 and d_2 are constant) but the mechanism admits finite uncontrolled motions.

This example demonstrates the usefulness and importance of visualizing the solutions of the forward kinematic problem as the intersections of planar curves in the output plane. Although visualizing the solutions in the complex domain is useful to understand how different real and complex solutions coalesce at singularities (as we have seen in the previous chapter), this representation is only valid when all solutions to the forward kinematic problem are isolated. In degenerate cases like the one studied here, it is necessary to visualize the forward kinematic problem from a more global perspective, such as the intersection of the planar curves that represent the kinematic restrictions of the mechanism. In these cases, representations of discrete solutions miss important information and solutions, as we have seen in Figure 4.9e.

4.4.4 Visualization of the Workspace and Singularities

PaRoLa represents the workspace and parallel singularities of parallel robots in the Cartesian and joint spaces, respectively. For robots in which the degrees of freedom of the end-effector involve both position and orientation coordinates, e.g. like the 3$\underline{\text{R}}$RR robot, PaRoLa allows the user to visualize two types of workspaces:

- Constant orientation workspace, in which the orientation of the end-effector remains constant (Figure 4.10a)
- Reachable workspace, which are the positions reachable with at least one orientation (Figure 4.10b).

As for the representation of parallel singularities in the actuated joint space: if the robot has more than two degrees of freedom, e.g. like the Delta robot, PaRoLa represents planar slices of the singularity locus in the coordinate planes of the joint space, considering that all joint coordinates but two remain fixed (Figure 4.10c). This is

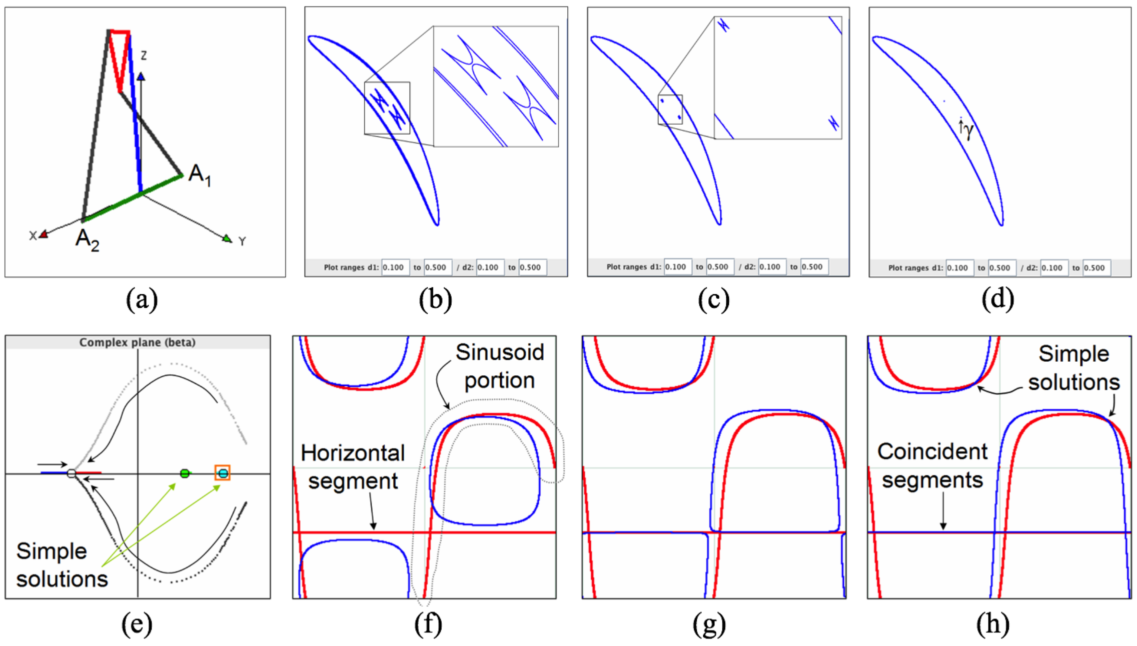


Figure 4.9: (a): a non-generic 2U$\underline{\text{P}}$S-U parallel robot with aligned universal joints. (b-d): two diamonds degenerate into points for this non-generic robot. (e): evolution of the solutions in the complex plane when approaching one of the point diamonds. (f-h): evolution of the solutions in the real plane when approaching one of the point diamonds.

because the surfaces of parallel singularities in the actuated joint space usually have complex shapes which are difficult to visualize in three dimensions, whereas the singularity curves resulting from planar slices of these singularity surfaces are much easier to visualize.

4.4.5 Modification of the Geometry of Parallel Robots

This is one of the most powerful features of PaRoLa: with this tool, the user can modify the geometric parameters of parallel robots, i.e., one can modify the linear and angular dimensions that define the design of a robot. This is especially useful for studying how the design of the robot affects its kinematic capabilities. By modifying the design parameters in PaRoLa, we can study how the workspace of a parallel robot transforms as a consequence. For example, Figure 4.11a shows how the workspace of a 5R robot transforms when its proximal links are made shorter. Initially, as Figure 4.11a shows, all four links have the same length, and the resulting workspace is lens-shaped and contains parallel singularities (blue closed curve wrapped along the horizontal axis in Figure 4.11a-right). As the proximal links are made shorter in PaRoLa, this lens-shaped workspace becomes smaller and smaller at the same time that two circular voids appear around the fixed supports of the robot. Then, when the proximal links are made sufficiently short, the workspace splits into two disconnected components, as shown in Figure 4.11b. As this Figure 4.11b also shows, this design is singularity-free, but the robot will be restricted to work only in one of the two components of the workspace, without being able to move to the other component. In this aspect, PaRoLa is useful

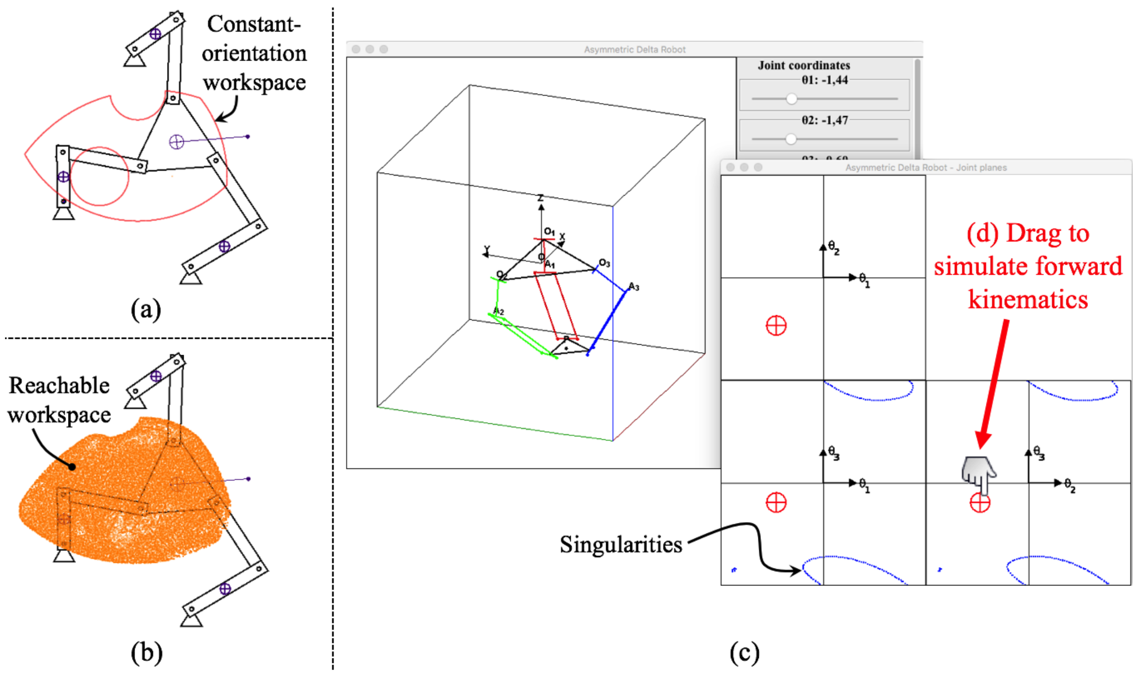

Constant-orientation workspace
(a)
Reachable workspace
(b)
Asymmetric Delta Robot
Joint coordinates
θ1: -1,44
θ2: -1,47
Asymmetric Delta Robot - Joint planes
(d) Drag to simulate forward kinematics
Singularities
(c)

Figure 4.10: (a) Representation of the constant-orientation workspace in PaRoLa. (b) Reachable workspace. (c) Planar slices of the singularity locus in the joint space. (d) The forward kinematics can also be simulated by dragging the actuated joint coordinates in the actuated joint space.

for analyzing how different designs yield larger or smaller workspaces, with more or less singularities (or without singularities at all).

This functionality is also useful for analyzing how parallel singularities transform under changes in the design of the robot. For example: Figure 4.11c shows a 3R$\underline{P}$R robot, whereas Figures 4.11d-g illustrate how the singularity locus of this robot is deformed when joint F of the robot is shifted to the left using PaRoLa. Initially (Figure 4.11d), the singularity locus contains several cusps, although we will focus this analysis on the two cusps indicated by red arrows in Figure 4.11d. When joint F of the robot is shifted to the left, these two cusps come closer to each other (Figure 4.11e) until they cross a swallowtail singularity (indicated in Figure 4.11f by a red arrow) and vanish. Thus, the robot would lose the ability to perform nonsingular transitions in the vicinity of this swallowtail singularity due to the vanishing of two cusps. In this aspect, PaRoLa can be used to analyze how parallel robots gain or lose the ability to perform nonsingular transitions by studying how cusps are created or vanish when changing the design of the robot.

4.4.6 Path Planning

PaRoLa allows the user to compare different trajectory planners, to understand the importance of singularities in path planning. The user can select initial and final points of the workspace, and the simulator plans a trajectory between these points taking into account different criteria and restrictions. For example: Figure 4.12 compares

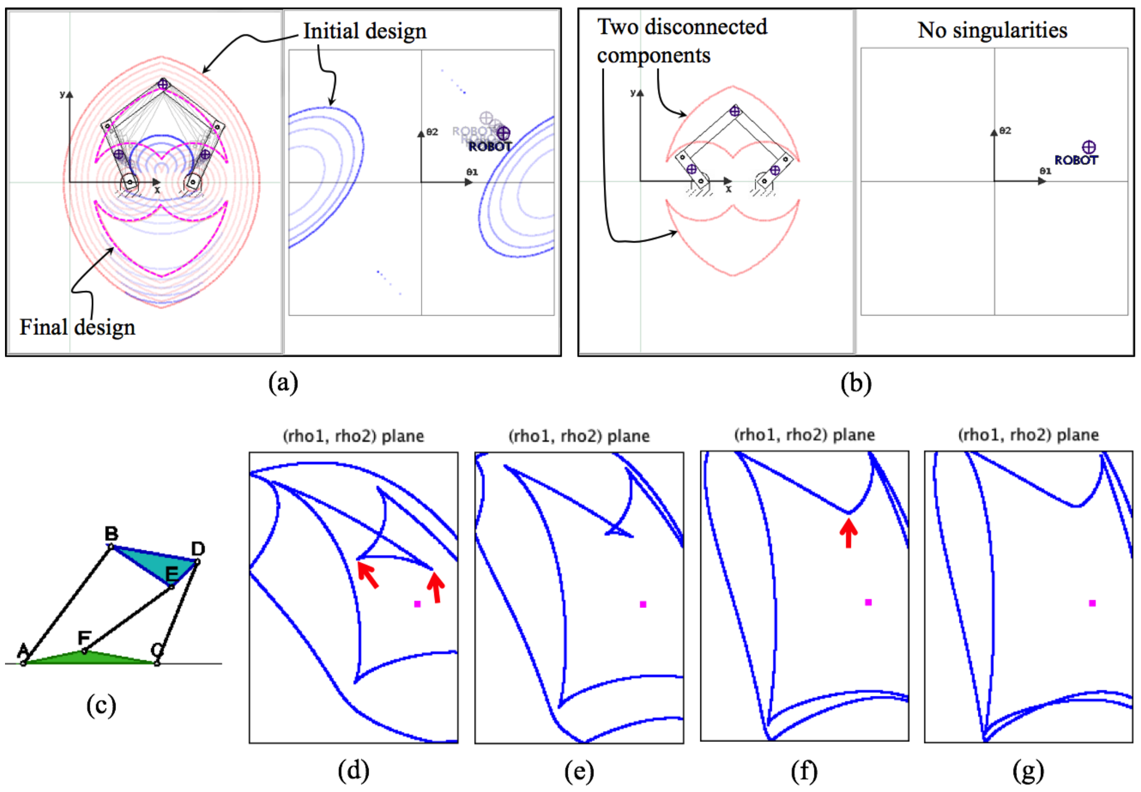


Figure 4.11: PaRoLa is useful for analyzing how the workspace and singularities transform under changes in the design of parallel robots.

two trajectories between two points (Start and Goal points) of the workspace of the 5R robot. In the first trajectory (Figure 4.12a-g), the robot is "broken" during a part of the trajectory (Figures 4.12c-d), which means that this trajectory is not feasible in practice. In the second trajectory (Figures 4.12h-n), the trajectory is satisfactorily executed. In the first case, a "naive" algorithm planned a straight line trajectory between the preimages of the start and goal points in the joint space (green trajectory shown in Figure 4.13-left), and this straight trajectory crosses the forbidden area of the joint space in which all solutions of the forward kinematics are non-real: for all joint angles in this forbidden region, the robot cannot be assembled.

On the contrary, the trajectory of Figure 4.12h-n was planned by an "intelligent" algorithm that searches the active joint space through the A* algorithm, and builds a joint-space trajectory connecting the preimages of the start and goal points without invading the forbidden area at which the robot cannot be assembled (green trajectory in Figure 4.13-right).

This functionality is useful for understanding the problem of planning trajectories in parallel robots, a problem which can become complex due to parallel singularities and the non-uniqueness of the solutions of the forward and inverse kinematic problems (i.e., both the inverse and forward kinematic problems generally yield several possible solutions for parallel robots).

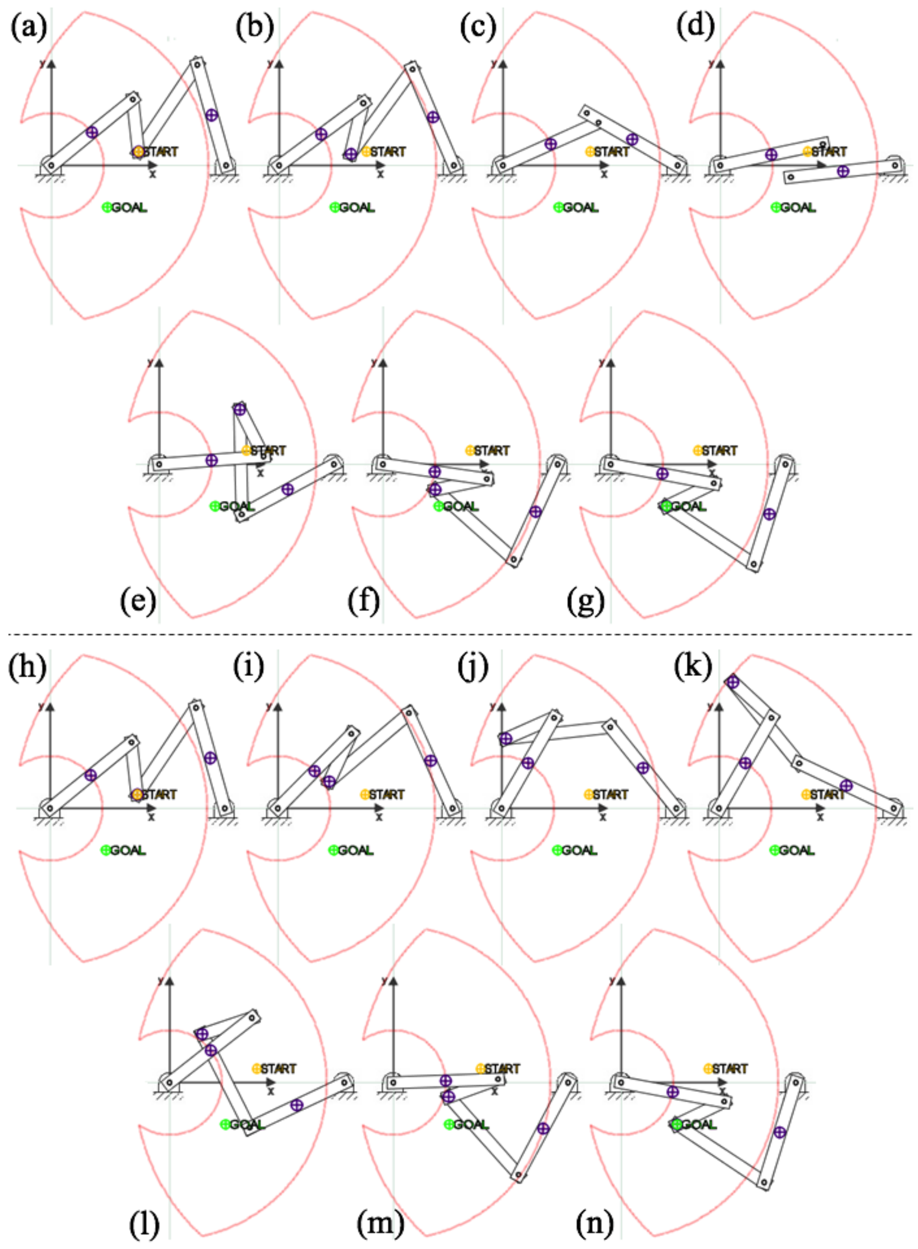


Figure 4.12: Trajectories of a 5R robot simulated in PaRoLa. (a-g): an unfeasible trajectory. (h-n): a feasible trajectory.

4.4.7 Dynamics and Control Simulation

To conclude, PaRoLa allows the user to simulate the closed-loop control of parallel robots, with the main purpose of analyzing singular transitions between different assembly modes (i.e., transitions by crossing singularities). These transitions cannot be properly reproduced by a purely kinematic simulation, i.e., the dynamics must be implemented too. This is because two or more different assembly modes coalesce at parallel singularities, and it is not possible to decide which assembly mode will be adopted by the robot when leaving a singularity if we simulate its motion based solely on the

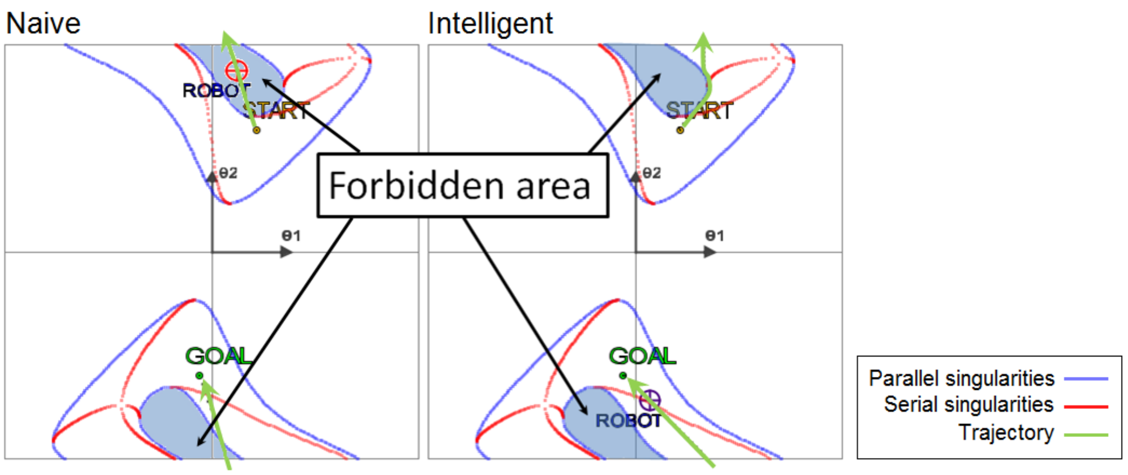


Figure 4.13: Joint space of the 5R robot, in which trajectories are planned. Left: the "naive" planner does not take into account singularities, and plans straight trajectories that may contain unfeasible values of the joint coordinates. Right: the "intelligent planner" avoids unfeasible configurations of the joint coordinates, finding a feasible joint trajectory.

forward kinematics: the robot should carry momentum when simulating a singularity crossing to decide which assembly mode will be adopted after this crossing, and this requires performing a dynamic simulation.

To facilitate the simulation of singularity crossings, PaRoLa allows the user to simulate the PID control of parallel robots [143]. Each actuated joint coordinate of the robot is controlled by an independent PID controller, in which the user can manually tune the Proportional, Integral and Derivative gains. Although this is not the best controller available for parallel robots, it suffices for the purposes of this functionality, since the only objective of these controllers is to drive the robot to the joint space points specified by the user while simulating the dynamics of the robot. In other words: thanks to these PID controllers, PaRoLa allows the user to simulate some sort of "forward kinematics" that takes into account the dynamics of the robot, in which the user specifies step references to be followed by the actuated joint coordinates, instead of specifying directly the actual values for these joint coordinates (as one does while simulating the forward kinematics, as described in Section 4.4.2). Thus, the mission of the PID controllers is to try to move the actual joint coordinates to these references specified by the user.

By using this functionality of PaRoLa, singular transitions between different assembly modes can be simulated as explained through the following example.

Consider a 5R robot which is initially assembled as shown in Figure 4.14a. With this initial assembly mode, the points of the region R at the top part of its workspace cannot be attained. To reach these points, the robot must cross a singularity and change its configuration to adopt the assembly mode indicated in dashed line in Figure 4.14a, for which region R can be accessed. This requires performing a trajectory in the joint space which is reflected at the singularity locus [195], in order to produce a change of assembly mode. Next, we will use the simulator to analyze different simple strategies that may be used to change the assembly mode by crossing a singularity in the 5R robot.

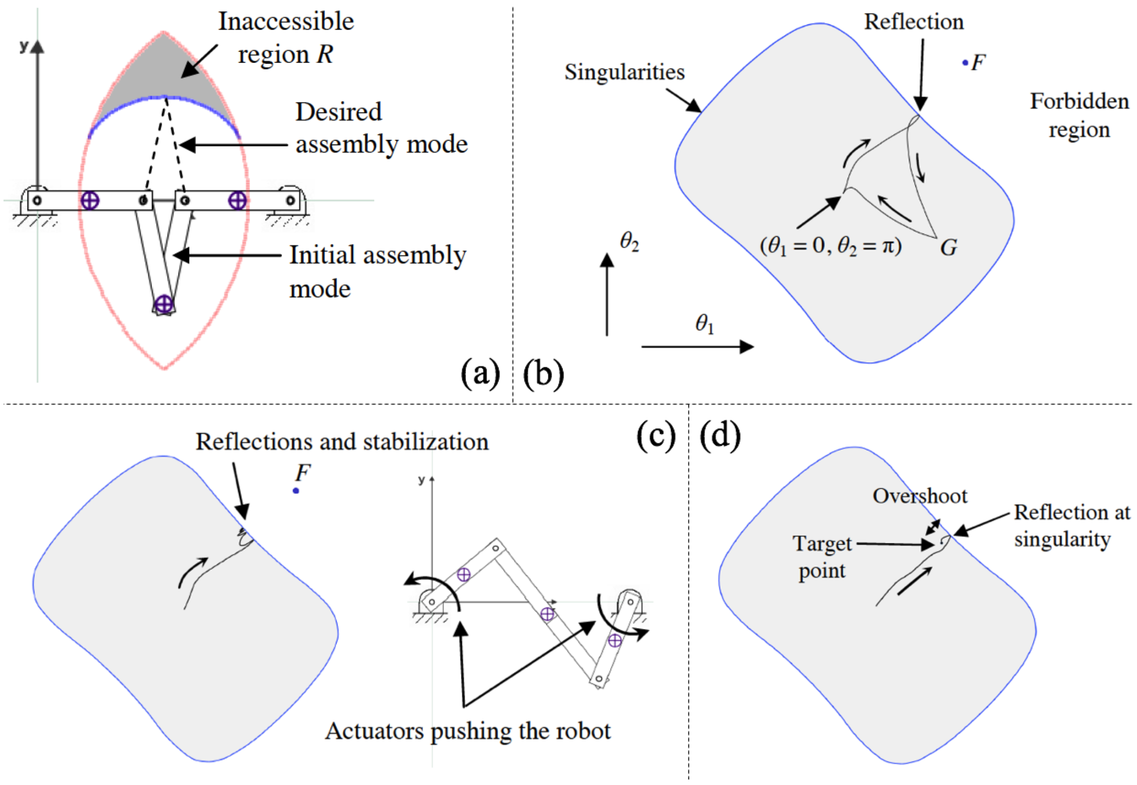


Figure 4.14: Simulating singular transitions in PaRoLa.

First, it can be checked that it is very easy to perform this change manually in the simulator, following the steps described next. Beginning at the initial configuration ($\theta_1 = 0$, $\theta_2 = \pi$), a point outside the region of the joint plane enclosed by the singularities (point F of Figure 4.14b) is set as reference for the joint coordinates of the robot. As in Figure 4.13, the points outside the region enclosed by singularities correspond to forbidden configurations because the robot cannot be assembled at them. In this way, the controller will try to drive the robot toward a forbidden configuration. When the trajectory of the robot arrives at the singularities, it will suffer a reflection that corresponds to a change of assembly mode. After detecting visually the first reflection (change of assembly mode), the user must set a point inside the region enclosed by singularities (e.g. the point G of Figure 4.14b) as a new reference for the controllers. Finally the initial point ($\theta_1 = 0$, $\theta_2 = \pi$) must be introduced as the new reference to reach the dashed configuration of Figure 4.14a, leaving the robot at a configuration where it can access region R without crossing more singularities.

If the user does not command a point G inside the region enclosed by singularities after the first reflection at the singularities has occurred, then the controller will continue trying to bring the robot to the forbidden point F, and this will produce more impacts and reflections at the singularities, as shown in Figure 4.14c (and each reflection will produce a change of assembly mode of the robot). Depending on the used controller, these reflections will decrease in amplitude until the controller stabilizes the robot at a singularity, as shown in Figure 4.14c. In this stabilized situation, the actuators are pushing the robot and trying to drive it to the forbidden region outside the singularities, a dangerous situation that may end up overheating and damaging the DC motors in a real robot.

The previous strategy for performing a singular change of assembly mode is manual: the user detects that the robot has changed the configuration after the reflection at the singularity occurs, and then she/he commands another point inside the region enclosed by singularities to avoid more reflections that produce more changes of assembly mode. This detection may be automated by introducing a sensor that detects the first singularity crossing (for example, detecting that the angle between the distal links crosses π rad, which is the singularity condition for this robot).

Alternatively, this may also be achieved by commanding as a control reference a point inside the region enclosed by singularities and close to them, instead of commanding a point outside this region. Then, the ratio proportional gain/derivative gain of the controller can be increased so that the trajectory presents some overshoot that exceeds the target point, touching the singularities and being reflected at them, producing a change of assembly mode (see Figure 4.14d). However, this solution is not robust or predictable (especially in a real robot), since sometimes the response may lack the necessary overshoot (not producing the change of assembly mode), or the trajectory may suffer successive oscillations and reflections that may produce several undesired changes of assembly mode, as in Figure 4.14c.

Summing up, PaRoLa can be used to get a glimpse of the complexity of performing singular transitions in real parallel robots. Actually, in practice, instead of using the simple strategies described above, one should use Computed Torque control laws guaranteeing the non-degeneracy of the dynamic model of the robot [127] for robustly and effectively crossing parallel singularities in real parallel robots.

4.5 Conclusions

In this chapter, we have presented PaRoLa, a virtual laboratory developed to aid in the kinematic analysis of parallel robots. This virtual laboratory consists of a collection of Java applets which allow the user to simulate the kinematics and dynamics of several parallel robots, which include well-known industrial robots such as the Stewart platform and the Delta and Tricept robots, as well as other robots which can be regarded as their planar versions. These simulators were developed using Easy Java Simulations, they are graphical and very intuitive, and require no installation. The functionalities offered by these simulators are: simulation of the inverse and forward kinematic problems of parallel robots, visualization of all the solutions in the complex and real domains, visualization of the workspace and singularities of these robots, possibility of changing the design of the robot in order to study how the workspace and singularities deform as a consequence, simulation of path planning, and simulation of the control of parallel robots to study singular transitions. In the present thesis, the developed simulations have been especially useful for analyzing the kinematics and singularities of the 2R$\underline{\text{P}}$R-PR parallel mechanisms that make up the legs of the HyReCRo robot. A simulator of the complete HyReCRo robot is also included in the developed virtual laboratory, but this simulator will be described in the next chapters.

4.6 Publications Related to this Chapter

The main results presented in this chapter are related to the following publications:

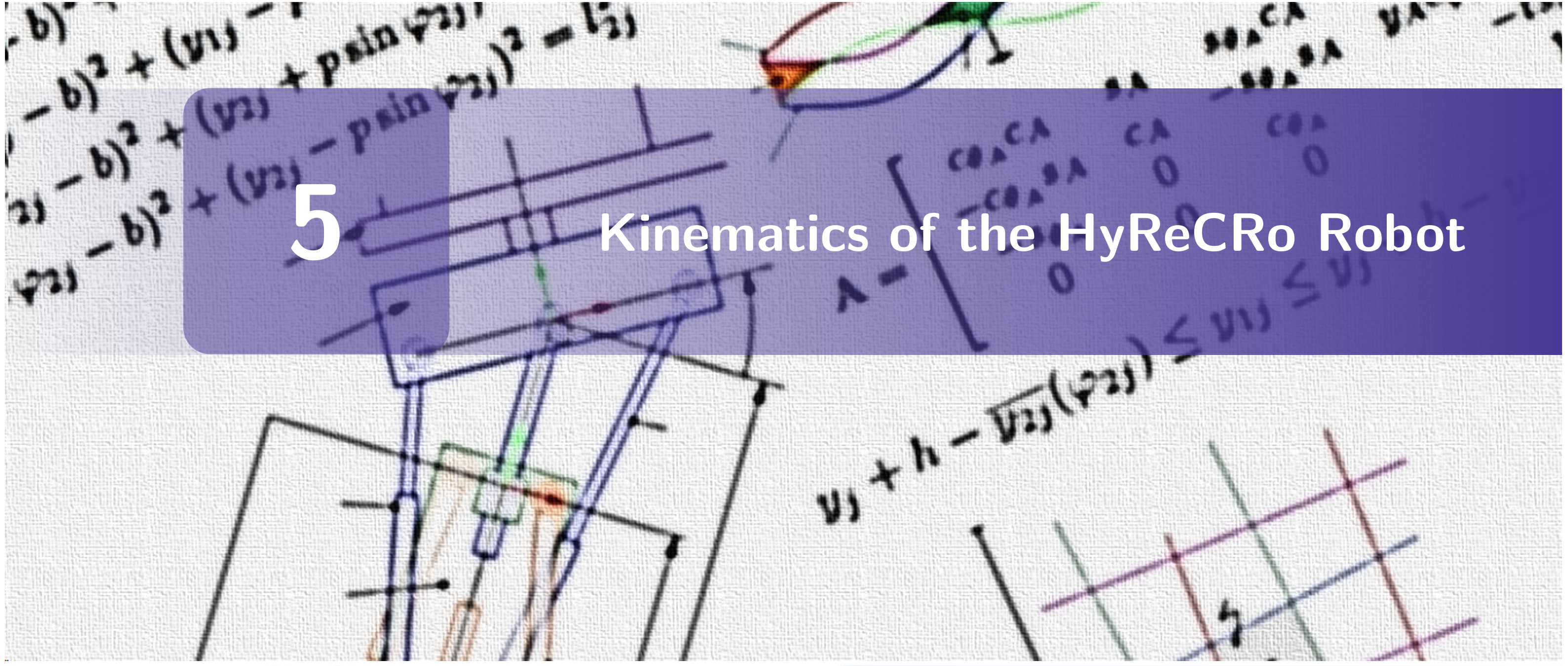
- A. Peidró, A. Gil, J.M. Marín, and Ó. Reinoso. A web-based tool to analyze the kinematics and singularities of parallel robots. *Journal of Intelligent & Robotic Systems*, 81(1):145–163, 2016 [137] **(SCI-JCR Impact Factor: 1.512, Q3)**.
  - This paper presents the virtual laboratory PaRoLa, with several examples illustrating how to use this tool to simulate the forward and inverse kinematics of 5R, 3$\underline{\text{R}}$RR, and Delta parallel robots, as well as visualize their singularities and workspace.
- A. Peidró, C. Tendero, J.M. Marín, A. Gil, L. Payá, and Ó. Reinoso. m-PaRoLa: a mobile virtual laboratory for studying the kinematics of five-bar and 3RRR planar parallel robots. *IFAC-PapersOnLine*, 51(4):178 – 183, 2018. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID 2018) [153].
  - This paper presents the first version of m-PaRoLa, the Javascript version of PaRoLa, which can be run on web browsers of desktop computers and mobile devices.
- A. Peidró, O. Reinoso, A. Gil, J.M. Marín, and L. Payá. A virtual laboratory to simulate the control of parallel robots. *IFAC-PapersOnLine*, 48(29):19 – 24, 2015 [143].
  - This paper presents the functionality to simulate the control of parallel robots in PaRoLa, for the 5R and 3$\underline{\text{R}}$RR robots, with the purpose of simulating singular transitions as explained in Section 4.4.7.
- A. Peidró, Ó. Reinoso, A. Gil J.M. Marín, and L. Payá. A simulation tool to study the kinematics and control of 2RPR-PR parallel robots. *IFAC-PapersOnLine*, 49(6):268–273, 2016. 11th IFAC Symposium on Advances in Control Education (ACE 2016) [149].
  - This paper presents a tool for simulating 2R$\underline{\text{P}}$R-PR parallel robots, which is included in PaRoLa.
- A. Peidró, Ó. Reinoso, J.M. Marín, A. Gil, L. Payá, and Y. Berenguer. A simulation tool for visualizing the assembly modes and singularity locus of 3RPR planar parallel robots. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno Lau,

and Carlos Cardeira, editors, *ROBOT 2017: Third Iberian Robotics Conference: Volume 1*, pages 516–528. Springer International Publishing, 2018 [150].

  – This paper presents a tool for simulating 3R$\underline{P}$R parallel robots, which is included in PaRoLa.

- A. Peidró, A. Belando, D. Valiente, O. Reinoso, and L. Payá. A multi-perspective simulator for visualizing and analyzing the kinematics and singularities of 2UPS/U parallel mechanisms. In *INTED2018 Proceedings*, 12th International Technology, Education and Development Conference, pages 3785–3793. IATED, 2018 [130].

  – This paper presents a tool for simulating 2U$\underline{P}$S-U parallel robots, which is included in PaRoLa.



5 Kinematics of the HyReCRo Robot

This chapter presents the kinematic analysis of the HyReCRo climbing robot. First, section 5.1 describes in detail the hybrid serial-parallel architecture of the HyReCRo robot. Next, section 5.2 presents the solution of its forward kinematic problem, which consists in computing the relative position and orientation between the feet of the robot for given values of its ten active joint coordinates. Following, section 5.3 analyzes the inverse kinematics and workspace for a subset of planar and symmetric postures of this robot, which are useful for planning basic movements that are necessary for exploring three-dimensional structures. Section 5.4 solves the general inverse kinematic problem, which consists in determining the necessary values of the ten active joint coordinates of this robot in order to reach a desired general three-dimensional relative position and orientation between its feet. Finally, Section 5.5 presents an interactive simulation tool for simulating the forward and inverse kinematic problems of the HyReCRo robot, and for studying graphically the solutions of these problems.

5.1 Description of the HyReCRo Robot

Figure 5.1a shows a 3D CAD model of the HyReCRo biped climbing robot. The robot has two identical legs (A and B) connected to the hip through revolute joints driven by motors (angles θ_A and θ_B). Each leg has three links: a core link and two platforms. The lower platform is the foot of the leg and carries the magnetic gripper that fixes the robot to the structure (the grippers are not considered in the kinematic analysis presented in this chapter: they will be analyzed and designed in Chapter 8). The upper platform is connected to the hip through the aforementioned revolute joint. Each platform is connected to the core link by means of two prismatic (or linear) actuators in parallel and a passive slider.

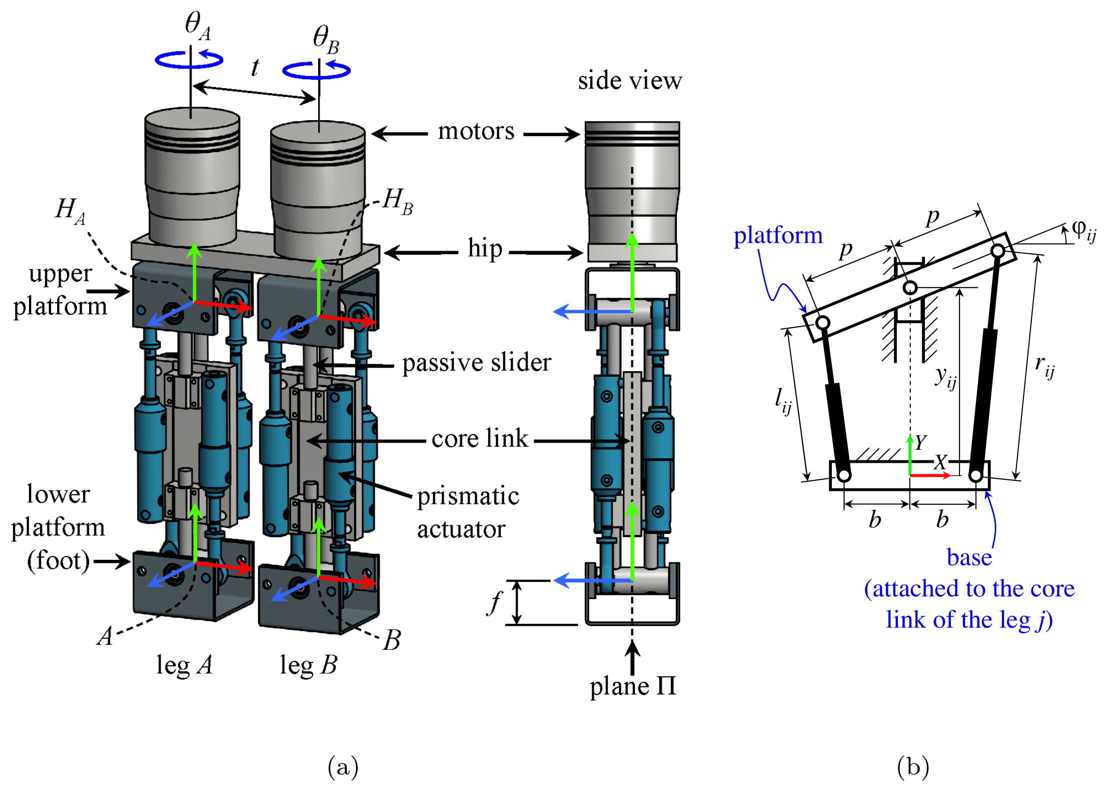


Figure 5.1: (a) 3D model of the climbing robot. (b) A symmetric 2R$\underline{\text{P}}$R-PR parallel mechanism used in the legs of the HyReCRo robot.

The mechanism composed of the core link, one platform, and the two prismatic actuators that connect these two elements, is a 2R$\underline{\text{P}}$R-PR parallel mechanism like those studied in chapter 3, but with symmetric design (i.e., with $a_1 = -a_2 = b$ and $b_1 = b_2 = p$). These symmetric parallel mechanisms, which will be referred to as "parallel modules" in this chapter, are represented schematically in Figure 5.1b. Hence, each leg is the serial combination of parallel module 1 (which is connected to the foot) and 2 (which is connected to the hip). The prismatic actuators of each parallel module lie in opposite sides of plane Π_j, which is one of the planes of symmetry of the core link of leg j (see the side view in Figure 5.1a). This is indicated with dashed lines in Figure 5.2.

Figure 5.1a also shows some reference frames attached to different parts of the robot (the origins of all these frames belong to plane Π_j). In this chapter, the X, Y, and Z axes of reference frames will be represented in red, green, and blue, respectively. Frames H_A and H_B are fixed to the hip of the robot, whereas frames A and B are respectively attached to the feet of legs A and B.

This biped climbing robot has ten degrees of freedom: two rotation angles θ_A and θ_B in the hip, and four linear actuators in each leg. Since ten degrees of freedom are used for positioning and orienting one foot of this robot with respect to the other (and this relative position/orientation has six degrees of freedom), the HyReCRo robot is kinematically redundant. This kinematic redundancy means that its inverse kinematic problem will admit infinitely many possible solutions, as detailed in sections 5.3 and 5.4. Actually, it has been shown that a climbing robot needs only 4 DOF to explore 3D structures [181, 178]. Thus, the HyReCRo robot has 6 redundant DOF to perform this task. It is important to remark that this redundancy was not a design criterion to be fulfilled when designing the robot, but rather a consequence of the original proposal for this robot, in which the robot would attain all postures necessary for performing plane transitions using purely binary actuation.

Kinematic redundancy offers some advantages for climbing 3D structures. A higher number of DOF allows a robot to travel from one point of the structure to another point using fewer and simpler movements than another robot with fewer DOF [62]. Moreover, the kinematic redundancy can be exploited to execute other secondary tasks in addition to the primary task of exploring the structure. For example, the redundancy can be used to avoid the singularities that difficult the movements of climbing robots [165], or to avoid joint limits and obstacles present in the workspace of the robot (e.g., other beams of the explored structure). The redundancy can also be used to minimize the torques in the actuators, or the energy consumption of the robot, which is useful to increase the autonomy of climbing robots powered by batteries [14]. Also, a kinematically redundant design allows for the use of closed-loop techniques to perform the kinematic callibration of the robot [63], which avoid the measurement of the relative pose between the feet of the robot. Robot calibration is necessary to increase the precision of climbing robots and guarantee that they can properly grasp the structure [179]. Other advantages of kinematic redundancy can be found in [32].

The geometric design of the HyReCRo robot is defined by the following six parameters: $\{b, p\}$ (which are dimensions of the 2R$\underline{\text{P}}$R-PR parallel mechanisms that compose the legs of the robot, as indicated in Figure 5.1b), $\{\rho_0, \Delta\rho\}$ (which are the joint limits $[\rho_0, \rho_0 + \Delta\rho]$ to which the linear actuators of the robot are subject, as defined in Section 3.3), distance t (which is the distance between the parallel axes of the rotations of the hip, as shown in Figure 5.1a), and h (which is the length of the core links of the legs, as indicated in Figure 5.2). All these six parameters, which influence the shape of the workspace of the robot, will be introduced in more detail during the next sections of this chapter.

In next sections, we will analyze and solve the forward and inverse kinematic problems of the HyReCRo climbing robot, as well as present a simulation tool developed for graphically studying these problems.

5.2 Forward Kinematic Problem (FKP)

In this section, the forward kinematic problem (FKP) of the HyReCRo robot is solved. The problem considered here consists in calculating the position and orientation of one foot with respect to the other foot when all ten joint coordinates are known: angles θ_A and θ_B and lengths (l_{ij}, r_{ij}) of the linear actuators of the parallel modules

($i \in \{1,2\}$, $j \in \{A,B\}$). First, the forward kinematics of the symmetric 2R$\underline{\text{P}}$R-PR parallel mechanisms of the legs is analyzed.

5.2.1 FKP of the Parallel Modules

Figure 5.1b shows the i-th parallel module of leg j ($i \in \{1,2\}$, $j \in \{A,B\}$), which is a 2R$\underline{\text{P}}$R-PR parallel mechanism like those studied in chapter 3. The main novelty here with respect to the analysis of chapter 3 is the fact that now these parallel mechanisms are symmetric, i.e., with: $a_1 = -a_2 = b$ and $b_1 = b_2 = p$. The forward kinematic analysis of section 3.1.1 is completely valid for the symmetric 2R$\underline{\text{P}}$R-PR parallel mechanisms analyzed in this chapter. However, next it will be convenient to solve again the forward kinematic problem of the 2R$\underline{\text{P}}$R-PR mechanisms for the particular case of symmetric mechanisms, since we will obtain simpler equations that will allow us to identify a subset of planar symmetric postures of the robot, which are useful for performing several movements on a structure by solving a simplified version of the general inverse kinematic problem.

Recall from section 3.1.1 that the forward kinematic problem of the i-th 2R$\underline{\text{P}}$R-PR parallel mechanism of leg j consists in calculating the position y_{ij} and the orientation φ_{ij} of the mobile platform for given lengths $\{l_{ij}, r_{ij}\}$ of the linear actuators. According to Figure 5.1b, (l_{ij}, r_{ij}) and (y_{ij}, φ_{ij}) are related through the following two equations:

$$(p\cos\varphi_{ij} - b)^2 + (y_{ij} + p\sin\varphi_{ij})^2 = r_{ij}^2 \tag{5.1}$$

$$(p\cos\varphi_{ij} - b)^2 + (y_{ij} - p\sin\varphi_{ij})^2 = l_{ij}^2 \tag{5.2}$$

These equations can be combined to obtain an equivalent system. Adding together Equations (5.1) and (5.2) yields Equation (5.3), whereas subtracting Equation (5.2) from Equation (5.1) results in Equation (5.4):

$$4bp\cos\varphi_{ij} = 2y_{ij}^2 + 2b^2 + 2p^2 - l_{ij}^2 - r_{ij}^2 \tag{5.3}$$

$$4y_{ij}p\sin\varphi_{ij} = r_{ij}^2 - l_{ij}^2 \tag{5.4}$$

Solving $\cos\varphi_{ij}$ from Equation (5.3) gives:

$$\cos\varphi_{ij} = \frac{2y_{ij}^2 + 2b^2 + 2p^2 - l_{ij}^2 - r_{ij}^2}{4bp} \tag{5.5}$$

Squaring Equation (5.4):

$$16y_{ij}^2p^2(1 - \cos^2\varphi_{ij}) = (r_{ij}^2 - l_{ij}^2)^2 \tag{5.6}$$

Finally, substituting Equation (5.5) into Equation (5.6) yields a cubic equation in $\Upsilon_{ij} = y_{ij}^2$:

$$\Upsilon_{ij}^3 + k_2^{ij}\Upsilon_{ij}^2 + k_1^{ij}\Upsilon_{ij} + k_0^{ij} = 0 \tag{5.7}$$

where:

$$k_2^{ij} = 2b^2 + 2p^2 - l_{ij}^2 - r_{ij}^2 \tag{5.8}$$

$$k_1^{ij} = \left[(b+p)^2 - \frac{l_{ij}^2 + r_{ij}^2}{2}\right]\left[(b-p)^2 - \frac{l_{ij}^2 + r_{ij}^2}{2}\right] \tag{5.9}$$

$$k_0^{ij} = b^2(l_{ij} + r_{ij})^2(l_{ij} - r_{ij})^2/4 \tag{5.10}$$

Equation (5.7) always has three roots, two of which may be complex. For a given strictly positive root Υ_{ij} of Equation (5.7), two solutions are obtained for $y_{ij} = \pm\sqrt{\Upsilon_{ij}}$. For each of these two values of y_{ij}, $\cos\varphi_{ij}$ is calculated from Equation (5.5), whereas $\sin\varphi_{ij}$ is obtained from Equation (5.4):

$$\sin\varphi_{ij} = \frac{r_{ij}^2 - l_{ij}^2}{4y_{ij}p} \tag{5.11}$$

Once $\cos\varphi_{ij}$ and $\sin\varphi_{ij}$ are known, φ_{ij} is unequivocally determined in $(-\pi, \pi]$. If $\Upsilon_{ij} = 0$, then $y_{ij} = 0$ and $\cos\varphi_{ij}$ is calculated using Equation (5.5). However, $\sin\varphi_{ij}$ cannot be calculated from Equation (5.11) since $y_{ij} = 0$. Instead, $\sin\varphi_{ij}$ is calculated as follows:

$$\sin\varphi_{ij} = \pm\sqrt{1 - \cos^2\varphi_{ij}} \tag{5.12}$$

obtaining again two solutions. Recall from section 3.1.2 that the forward kinematic problem of the 2R$\underline{\text{P}}$R-PR parallel mechanism has four different real solutions: for a given pair (l_{ij}, r_{ij}), the previous equations yield four different real pairs (y_{ij}, φ_{ij}). This is because Equation (5.7) can only have two non-negative roots [87] and, according to the previous equations, each of these non-negative roots results in two different pairs (y_{ij}, φ_{ij}).

Note that swapping the values of r_{ij} and l_{ij} neither affects Equation (5.7) nor Equation (5.5), but this changes the sign of $\sin\varphi_{ij}$ in Equation (5.11). Hence, swapping r_{ij} and l_{ij} changes the sign of φ_{ij}, leaving y_{ij} unchanged. This can also be deduced from Figure 5.1b, where swapping r_{ij} and l_{ij} is equivalent to rotating this figure π rad about the vertical Y axis. This fact will be exploited in Section 5.3 to analyze a simplified (yet useful) case of the inverse kinematics of the complete HyReCRo climbing robot.

5.2.2 FKP of the Complete Robot

The forward kinematics of the complete robot consists in calculating the position and orientation of one foot with respect to the other foot when all ten joint coordinates are known. This problem will be solved using Homogeneous Transformation Matrices (HTMs). An HTM has the following form [12]:

$$\mathbf{T}_{m/n} = \begin{bmatrix} \mathbf{R}_{m/n} & \mathbf{t}_{m/n} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{5.13}$$

where $\mathbf{0}_{1\times3} = [0, 0, 0]$. The matrix $\mathbf{T}_{m/n}$ encodes the position and orientation of a frame m with respect to another frame n. Indeed, $\mathbf{R}_{m/n} \in \mathbb{R}^{3\times3}$ is a rotation matrix

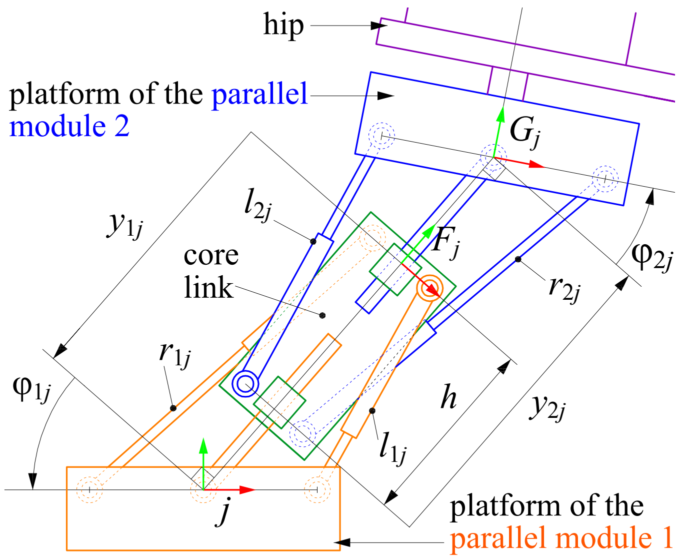


Figure 5.2: Kinematics of a generic leg $j \in \{A, B\}$.

whose columns are the vectors of frame m expressed in the basis formed by the vectors of frame n, whereas $\mathbf{t}_{m/n} \in \mathbb{R}^{3\times1}$ is the position of the origin of frame m in coordinates of frame n.

The forward kinematics of one leg can be easily solved using HTMs. Figure 5.2 represents a generic leg $j \in \{A, B\}$. Each leg has two parallel modules whose bases are attached to the core link. The platform of parallel module 1 is the foot of the leg, whereas the platform of parallel module 2 is connected to the hip of the robot by means of a revolute joint. Variables $(y_{1j}, \varphi_{1j}, y_{2j}, \varphi_{2j})$ are obtained from $(l_{1j}, r_{1j}, l_{2j}, r_{2j})$ as explained in Section 5.2.1 (i.e., solving the forward kinematics of the symmetric 2R$\underline{\text{P}}$R-PR parallel mechanisms). All reference frames of Figure 5.2 are contained in plane Π_j, which is one of the planes of symmetry of the core link of leg j (see Figure 5.1a). The transformation between frame j (fixed to the foot) and frame F_j (fixed to the core link) is:

$$\mathbf{T}_{F_j/j} = \begin{bmatrix} \cos\varphi_{1j} & \sin\varphi_{1j} & 0 & y_{1j}\sin\varphi_{1j} \\ -\sin\varphi_{1j} & \cos\varphi_{1j} & 0 & y_{1j}\cos\varphi_{1j} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.14}$$

Similarly, the transformation between frame G_j (attached to the platform of parallel module 2) and frame F_j is:

$$\mathbf{T}_{G_j/F_j} = \begin{bmatrix} \cos\varphi_{2j} & -\sin\varphi_{2j} & 0 & 0 \\ \sin\varphi_{2j} & \cos\varphi_{2j} & 0 & y_{2j} - h \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.15}$$

where h is a geometric constant ($=$ the length of the core link, as indicated in Figure 5.2). Finally, a rotation θ_j about the Y axis of frame G_j transforms it into frame H_j, which is attached to the hip:

$$\mathbf{T}_{H_j/G_j} = \begin{bmatrix} \cos\theta_j & 0 & \sin\theta_j & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_j & 0 & \cos\theta_j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.16}$$

The position and orientation of frame H_j with respect to frame j is obtained as follows:

$$\mathbf{T}_{H_j/j} = \mathbf{T}_{F_j/j}\mathbf{T}_{G_j/F_j}\mathbf{T}_{H_j/G_j} \tag{5.17}$$

which completes the solution of the FKP of any generic leg j. Once the forward kinematics of each leg is solved, it is straightforward to calculate the position and orientation of the foot of one leg $k \in \{A, B\} \setminus \{j\}$ with respect to the foot of the other leg j:

$$\mathbf{T}_{k/j} = \mathbf{T}_{H_j/j}\mathbf{T}_{H_k/H_j}\mathbf{T}_{k/H_k} \tag{5.18}$$

where $\mathbf{T}_{k/H_k} = \left(\mathbf{T}_{H_k/k}\right)^{-1}$ and $\mathbf{T}_{H_k/H_j}$ is the HTM that encodes the position and orientation of frame H_k with respect to frame H_j:

$$\mathbf{T}_{H_k/H_j} = \begin{bmatrix} \mathbf{I} & \mathbf{t}_{H_k/H_j} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{5.19}$$

which is constant because both frames are attached to the same rigid body (the hip). $\mathbf{I}$ is the 3×3 identity matrix. Moreover, according to Figure 5.1a: $\mathbf{t}_{H_B/H_A} = [t, 0, 0]^T = -\mathbf{t}_{H_A/H_B}$, where t is the distance between the parallel axes of the revolute actuators of the hip.

Note that, in theory, there are $4^4 = 256$ different solutions to the FKP of the complete robot. This is because the kinematic chain between the feet has four parallel modules connected in series and the FKP of each module has four real solutions at most. However, as discussed in section 3.3, only one of the four real solutions of each 2R$\underline{\text{P}}$R-PR parallel mechanism will be physically possible in practice, since the other three solutions imply some kind of collision between different parts of the robot (the only valid solution for each 2R$\underline{\text{P}}$R-PR parallel mechanism was denoted by H+ in section 3.3). Thus, if we consider that each parallel mechanism can only have one valid solution to its forward kinematic problem, then the forward kinematics of the complete HyReCRo robot (i.e., the robot composed of four 2R$\underline{\text{P}}$R-PR parallel mechanisms connected in series) will only have one valid solution, instead of 256.

5.3 Planar Symmetric Postures

In the previous section, we have solved the forward kinematic problem of the complete HyReCRo robot. This problem is relatively easy: given all ten joint coordinates θ_j, l_{ij}, r_{ij} ($i = 1, 2$, $j = A, B$), one only needs to solve the forward kinematics of all four symmetric 2R$\underline{\text{P}}$R-PR parallel mechanisms, and then concatenate in series the solutions obtained for these mechanisms, obtaining matrix $\mathbf{T}_{k/j}$, which represents the position and orientation of one foot k of the robot with respect to the other foot j [Equation (5.18)].

Conversely, the Inverse Kinematic Problem (IKP) can be formulated in the following way: given matrix $\mathbf{T}_{k/j}$, which codifies the desired relative position and orientation between the feet, one must compute all ten joint coordinates θ_j, l_{ij}, r_{ij} necessary to attain the desired pose $\mathbf{T}_{k/j}$. Solving the inverse kinematic problem is very useful and necessary for planning the movements of the robot, since it allows us to know the necessary values of all joint coordinates to place one foot of the robot at the position and orientation necessary for exploring a three dimensional structure. However, due to the kinematic redundancy of the HyReCRo robot (ten joint coordinates are used to place and orient a foot in space, which is a rigid body with only six degrees of freedom), the general inverse kinematic analysis of the HyReCRo robot is not easy and will be presented in section 5.4.

Instead of solving the general IKP of the HyReCRo robot, in this section we will focus on analyzing and solving the inverse kinematics and workspace of a subset of planar and symmetric postures of this robot, which are easier to analyze (from the perspective of the inverse kinematics) than general arbitrarily complex three-dimensional postures. In this section, we will refer to these planar and symmetric postures by the acronym "PSIK", which stands for "Planar Symmetric Inverse Kinematics". As we will see later, PSIK postures, in spite of their simplicity, are very useful since the basic movements necessary for exploring three dimensional structures (namely: longitudinal advancement along a beam, convex/exterior transition to change between adjacent faces of the same beam, and concave/interior transition to change between different adjacent beams) can be executed by means of planar and symmetric postures. Therefore, solving the inverse kinematics for PSIK postures can be very useful for easily planning planar and symmetric movements that allow the robot to explore three-dimensional structures. Moreover, as we will see later in this section, when analyzing the range of attainable planar and symmetric postures (i.e., the PSIK workspace), we will obtain very useful information concerning the optimal design of the robot, i.e., we will discover how the reach of the robot will be affected by its geometric design parameters.

5.3.1 Planar Symmetric Inverse Kinematic (PSIK) Problem

The planar and symmetric postures considered in this section are represented by Figure 5.3, where we assume that foot j is fixed to the structure and foot k is mobile ($j, k \in \{A, B\}, j \neq k$). It is assumed that the Z axes of the frames attached to both feet are parallel and point in the same direction. Moreover, the origin of the frame attached to foot k is contained in the XY plane of the frame attached to foot j. In this situation, any variation in the length of the prismatic actuators of the parallel modules only produces planar motions of frame k in the XY plane of frame j. In that case, the position and orientation of frame k relative to frame j can be calculated as follows:

$$\mathbf{T}_{k/j} = \mathbf{T}_{G_j/j} \begin{bmatrix} \mathbf{I} & [t, 0, 0]^T \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \left(\mathbf{T}_{G_k/k}\right)^{-1} \tag{5.20}$$

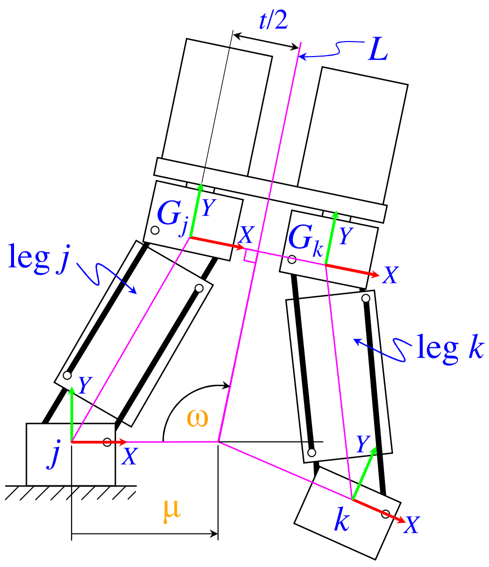


Figure 5.3: The Planar Symmetric Inverse Kinematic (PSIK) problem.

where $\mathbf{T}_{G_j/j} = \mathbf{T}_{F_j/j}\mathbf{T}_{G_j/F_j}$. Moreover, it is assumed that the joint coordinates of the parallel modules of the two legs are related as follows:

$$l_{ik} = r_{ij}, \quad r_{ik} = l_{ij} \quad (i = 1, 2) \tag{5.21}$$

This means that the joint coordinates of parallel module i of legs k and j are swapped. According to Section 5.2.1, this translates into:

$$y_{ik} = y_{ij}, \quad \varphi_{ik} = -\varphi_{ij} \quad (i = 1, 2) \tag{5.22}$$

It can be graphically checked that Equation (5.22) implies that legs k and j are symmetric with respect to line L, which is the axis of symmetry of the hip of the robot. Substituting Equation (5.22) into Equation (5.20), matrix $\mathbf{T}_{k/j}$ can be written in terms of only the variables of leg j and has the following expression:

$$\mathbf{T}_{k/j} = \begin{bmatrix} -\cos(2\omega) & -\sin(2\omega) & 0 & \mu\,(1-\cos(2\omega)) \\ \sin(2\omega) & -\cos(2\omega) & 0 & \mu\cdot\sin(2\omega) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.23}$$

where:

$$\mu = [t - 2(h - y_{1j} - y_{2j})\sin\varphi_{2j}] \,/\, [2\cos(\varphi_{1j} - \varphi_{2j})] \tag{5.24}$$
$$\omega = \varphi_{1j} - \varphi_{2j} + \pi/2 \tag{5.25}$$

Thus, under the condition of planar and symmetric motion, the position and orientation of foot k relative to foot j can be completely defined by only two parameters (μ, ω), which are indicated in Figure 5.3. We define the Planar Symmetric Inverse Kinematic (PSIK) problem, which consists in calculating the joint coordinates $(l_{1j}, r_{1j}, l_{2j}, r_{2j})$ needed to achieve a desired position and orientation (μ, ω). Since the joint coordinates do not appear explicitly in Equations (5.24)-(5.25), the kinematic equations of the parallel modules of leg j must be included:

$$(p\cos\varphi_{1j} - b)^2 + (y_{1j} + p\sin\varphi_{1j})^2 = r_{1j}^2 \tag{5.26}$$

$$(p\cos\varphi_{1j} - b)^2 + (y_{1j} - p\sin\varphi_{1j})^2 = l_{1j}^2 \tag{5.27}$$

$$(p\cos\varphi_{2j} - b)^2 + (y_{2j} + p\sin\varphi_{2j})^2 = r_{2j}^2 \tag{5.28}$$

$$(p\cos\varphi_{2j} - b)^2 + (y_{2j} - p\sin\varphi_{2j})^2 = l_{2j}^2 \tag{5.29}$$

Hence, the PSIK problem requires calculating $(l_{1j}, r_{1j}, l_{2j}, r_{2j}, y_{1j}, \varphi_{1j}, y_{2j}, \varphi_{2j})$ from Equations (5.24)-(5.29). Like the general inverse kinematic problem, the PSIK problem is underconstrained since eight unknowns must be obtained from six equations (the solution sets of this problem are two-dimensional). However, the PSIK problem involves less variables and simpler equations than the general inverse kinematic problem, so it is easier to solve. Next, we will illustrate through a representative and useful example how to solve the PSIK problem, assuming that the lengths of the prismatic actuators of the parallel modules have upper and lower limits: l_{ij} and r_{ij} must be in $[\rho_0, \rho_0 + \Delta\rho]$, where $\rho_0 > 0$ is the minimum length of the actuators and $\Delta\rho > 0$ is their stroke.

5.3.1.1 Example: performing a concave transition

Consider a HyReCRo robot with the following parameters: $b = p = 4$, $\rho_0 = 19$, $\Delta\rho = 6$, $t = 15.6$, and $h = 16$ (all values in cm). The robot, whose foot B is fixed, has to perform a concave transition between two perpendicular beams $\{b_1, b_2\}$ of a structure, as shown in Figure 5.4a. Assume that we wish to perform this concave transition using a PSIK posture. When using planar and symmetric postures, it is evident that orthogonal concave transitions are defined by $\omega = \pi/4$ (see Figure 5.4a), whereas length μ depends on the distance between the fixed foot and the vertical beam to which we wish to attach the free foot. Assume that $\mu = 27.4$ cm in this example. Next, we will show how to solve the PSIK problem in order to achieve the desired concave transition in this example.

First, we substitute all numerical values given in the previous paragraph into Equations (5.24)-(5.29) particularized for $j = B$ (since B is the fixed foot), obtaining the following system of six equations in eight unknowns (l_{1B}, r_{1B}, l_{2B}, r_{2B}, y_{1B}, φ_{1B}, y_{2B}, φ_{2B}):

$$\frac{15.6 - 2(16 - y_{1B} - y_{2B})\sin\varphi_{2B}}{2\cos(\varphi_{1B} - \varphi_{2B})} = 27.4 \tag{5.30}$$

$$\varphi_{2B} - \pi/4 = \varphi_{1B} \tag{5.31}$$

$$(4\cos\varphi_{1B} - 4)^2 + (y_{1B} + 4\sin\varphi_{1B})^2 = r_{1B}^2 \tag{5.32}$$

$$(4\cos\varphi_{1B} - 4)^2 + (y_{1B} - 4\sin\varphi_{1B})^2 = l_{1B}^2 \tag{5.33}$$

$$(4\cos\varphi_{2B} - 4)^2 + (y_{2B} + 4\sin\varphi_{2B})^2 = r_{2B}^2 \tag{5.34}$$

$$(4\cos\varphi_{2B} - 4)^2 + (y_{2B} - 4\sin\varphi_{2B})^2 = l_{2B}^2 \tag{5.35}$$

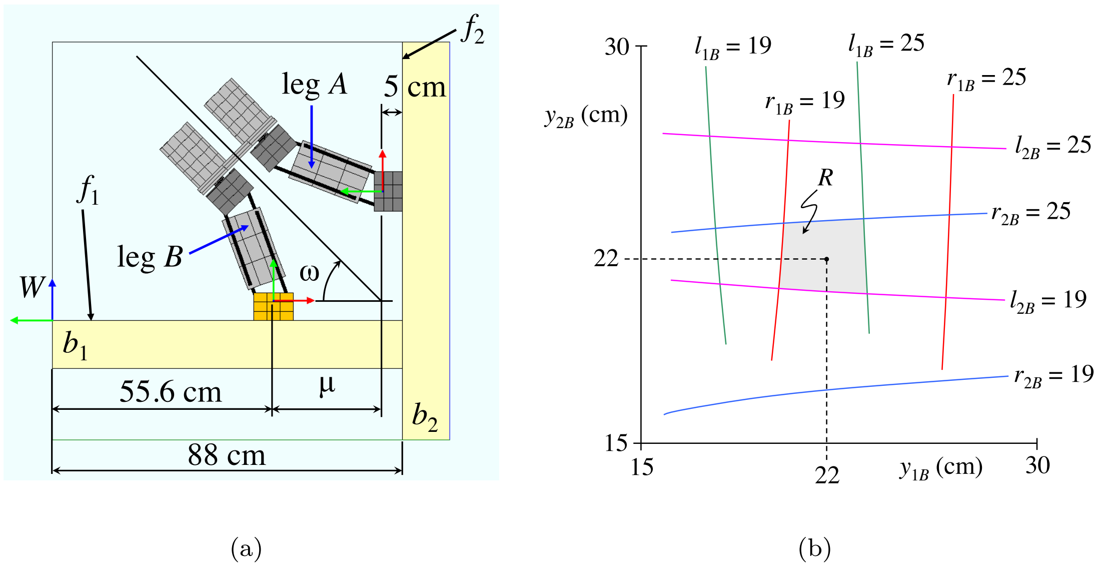


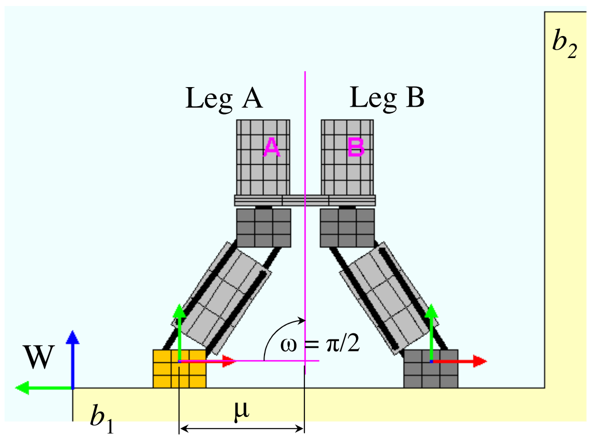
Figure 5.4: (a) Executing a concave transition using a planar and symmetric posture. (b) Two-dimensional solution set R of the PSIK problem.

Since we have two more unknowns than the number of equations, the system is underdetermined and, in principle, we should expect a two-dimensional set of infinitely many possible solutions. The objective will be to manipulate the previous equations to obtain this two-dimensional solution set. First, Equation (5.31) is used to eliminate φ_{1B} from Equation (5.30). Then, φ_{2B} is solved from the resulting equation:

$$\varphi_{2B} = \sin^{-1}\left(\frac{13.7\sqrt{2} - 7.8}{y_{1B} + y_{2B} - 16}\right) \tag{5.36}$$

This solution can be substituted into Equations (5.31)-(5.35) to express joint coordinates $\{l_{1B}, r_{1B}, l_{2B}, r_{2B}\}$ in terms of $\{y_{1B}, y_{2B}\}$. In that case, we can plot the region R of plane (y_{1B}, y_{2B}) for which joint coordinates $\{l_{1B}, r_{1B}, l_{2B}, r_{2B}\}$ satisfy the joint limits (i.e., $l_{iB}, r_{iB} \in [19, 25]$ for $i = 1, 2$). Figure 5.4b represents this (shaded) region R of pairs (y_{1B}, y_{2B}) that satisfy the joint limits, so any point inside this region is a valid solution to the PSIK problem. For example, picking the solution $y_{1B} = y_{2B} = 22$ cm yields the following lengths for the linear actuators: $r_{1B} \approx 20.595$, $l_{1B} \approx 23.407$, $r_{2B} \approx 23.656$, and $l_{2B} \approx 20.349$ (all values in cm). This solution precisely corresponds with the posture shown in Figure 5.4a.

Note that the procedure followed in this example is general and valid for any other values of (ω, μ) or for HyReCRo robots with other geometric parameters. For example, this procedure was also followed in [133] to solve the PSIK problem for executing a convex transition between different faces of the same beam.



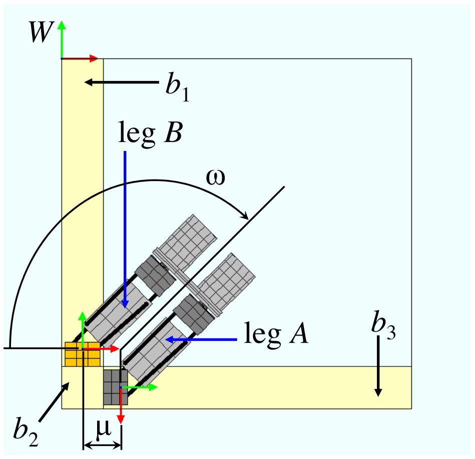


Figure 5.5: PSIK postures for (a) longitudinally advancing along a beam, and (b) performing convex transitions between adjacent faces of the same beam.

5.3.2 Workspace for PSIK Postures: Sensitivity Analysis

As we have just seen in the example of previous section 5.3.1.1, we can solve the PSIK problem to determine planar and symmetric postures that allow the robot to perform concave transitions ($\omega = \pi/4$) between different orthogonal beams of a structure. In general, PSIK postures are also useful for other basic movements and, in particular, they allow us to solve postures necessary for advancing longitudinally along a beam ($\omega = \pi/2$, see Figure 5.5a) or for performing convex transitions between different faces of the same beam ($\omega = 3\pi/4$, see Figure 5.5b). By combining these three basic movements (longitudinal movement, concave transition, and convex transition) with the rotations of the hip, it is possible to completely explore three-dimensional structures using PSIK postures.

In this section, we will study the PSIK workspace, i.e., the set of pairs (ω, μ) reachable by the robot when it adopts planar and symmetric postures. Studying such a workspace will allow us to find the reach μ for each of the three basic movements: longitudinal movement ($\omega = \pi/2$), concave transitions ($\omega = \pi/4$), and convex transitions ($\omega = 3\pi/4$). Also, we will be able to determine how the reach μ varies when varying each of the six geometric design parameters of the robot $(b, p, t, h, \rho_0, \Delta\rho)$. In this way, we will be able to determine which design parameters have a greater influence on the reach of the robot, and this information will be very useful for optimally designing and building a prototype of the HyReCRo robot in later chapters of this thesis.

Figures 5.6a-f show the PSIK workspaces for different choices of the geometric design parameters of the robot, obtained using a Monte Carlo algorithm [6]. Each subfigure of Figure 5.6 shows different PSIK workspaces that are obtained when the

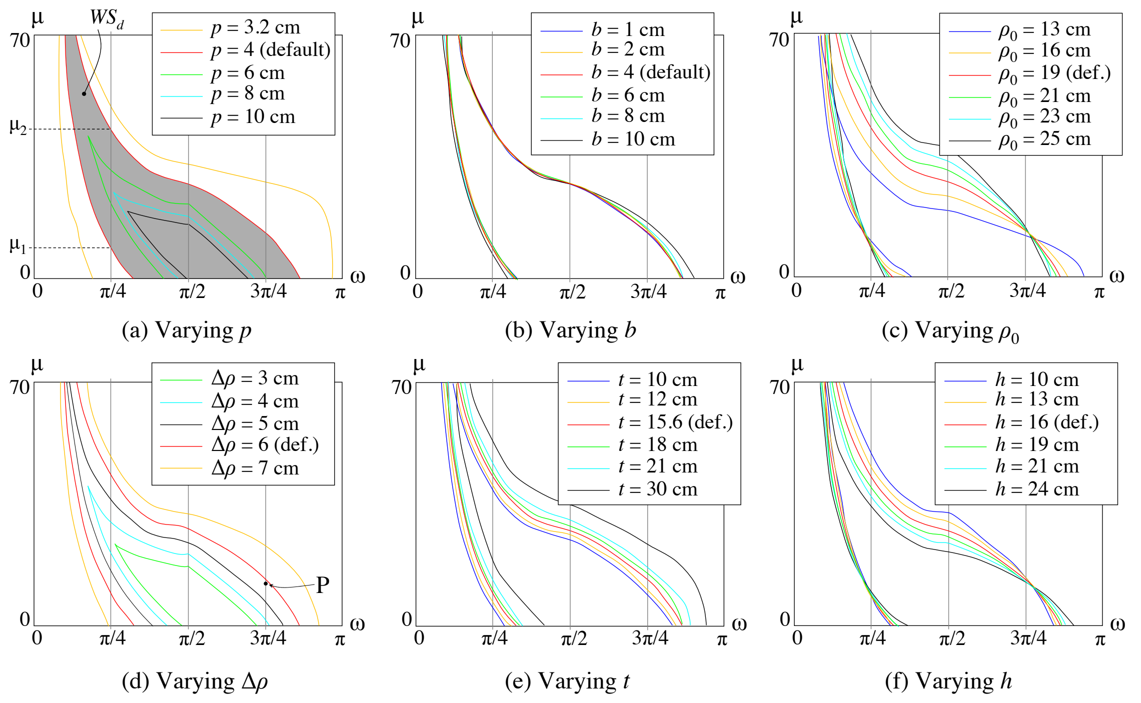


Figure 5.6: PSIK workspaces for different values of the geometric design parameters of the robot. In the axes, μ is in cm and ω is in rad.

corresponding geometric parameter of the robot is varied, keeping the remaining parameters at the default values used in the example of section 5.3.1.1, which are: $b = p = 4$, $t = 15.6$, $h = 16$, $\Delta\rho = 6$, and $\rho_0 = 19$ (all in cm). Curves having the same color in each subfigure of Figure 5.6 enclose the PSIK workspace corresponding to each geometry, i.e., these curves are the boundaries of the workspaces. This is illustrated in Figure 5.6a, where the PSIK workspace for the default geometry, denoted by WS_d and enclosed by red curves, has been shaded. It is evident from Figure 5.6 that the shape of the PSIK workspace is most sensitive to parameters p and $\Delta\rho$, whereas the shape changes little with b. As we will show in the next chapter, this observation about sensitivities of different parameters is not only valid for PSIK postures, but also for general three-dimensional postures.

Figures 5.6a-f also show three vertical lines at $\omega = \pi/4$, $\omega = \pi/2$, and $\omega = 3\pi/4$. As explained earlier, lines $\omega = \pi/4$ and $\omega = 3\pi/4$ represent concave (Figure 5.4b) and convex (Figure 5.5b) transitions, respectively. Line $\omega = \pi/2$ represents a posture that can be used to move the robot along a beam like an inchworm, extending and retracting the legs as shown in Figure 5.7. Alternatively, the robot may move along a beam following the gait shown in Figure 5.8, which uses the rotations of the hip (θ_A and θ_B). The segment of these three vertical lines that lies within the PSIK workspace of a given geometry is the set of lengths μ that can be achieved for the corresponding value of ω. For example, for the default workspace WS_d in Figure 5.6a and $\omega = \pi/4$, μ must be between μ_1 and μ_2. Next, we will analyze Figure 5.6 to discuss if the robot can perform convex and concave transitions using planar and symmetric postures in three scenarios.

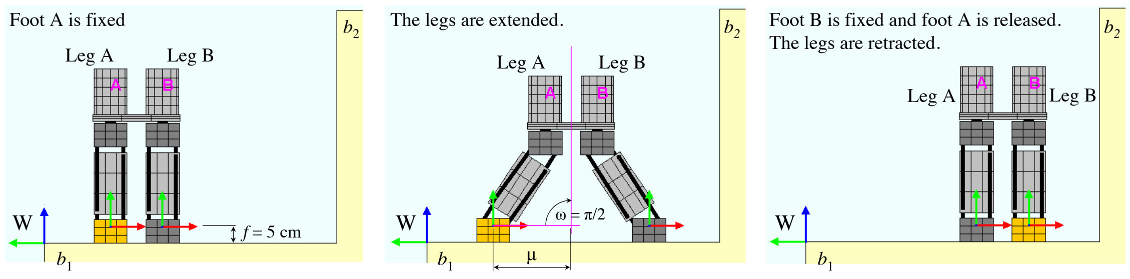


Figure 5.7: An inchworm-like gait to move along a beam.

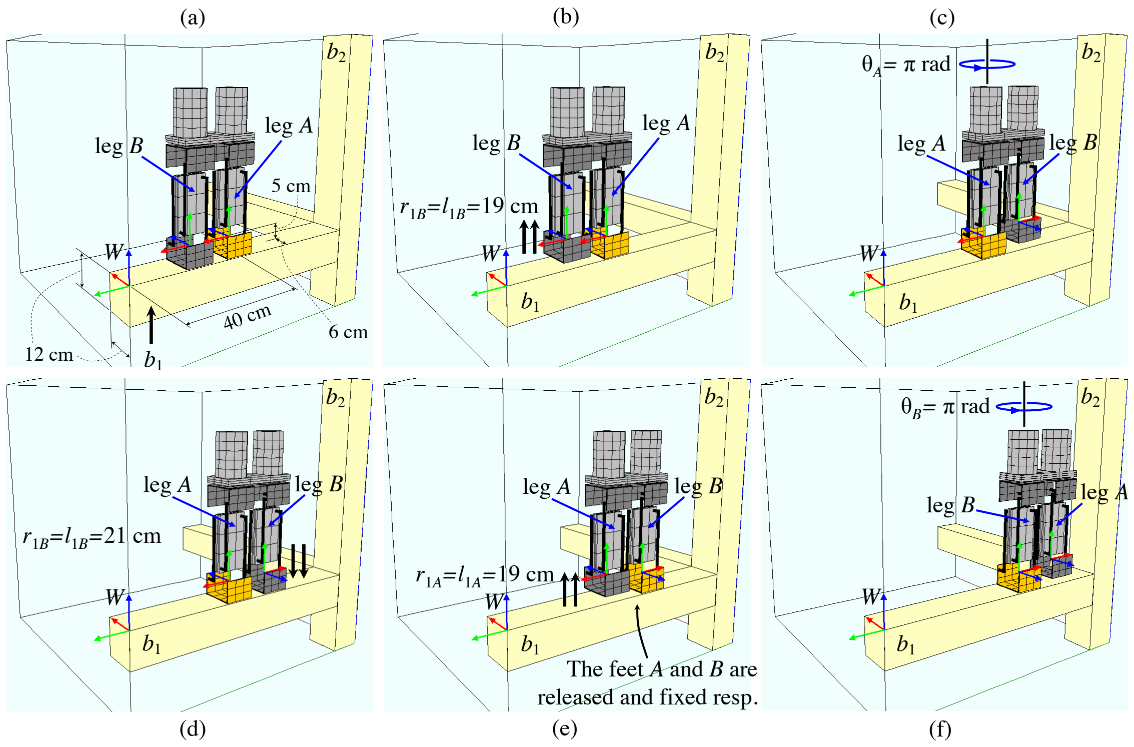


Figure 5.8: Advancing longitudinally along a beam by using the rotations θ_A and θ_B of the hip.

5.3.2.1 Scenario 1: the geometry of the robot is modified.

Can the robot perform convex and concave transitions if its geometry is different from the default one? As Figure 5.6 shows, vertical lines $\omega = \pi/4$ and $\omega = 3\pi/4$ (which represent concave and convex transitions, respectively) intersect the PSIK workspace for most of the geometries, which means that both transitions are possible. However, for some designs obtained by increasing p or decreasing $\Delta\rho$, the workspace is too small and does not intersect these vertical lines. Thus, these designs should be avoided.

5.3.2.2 Scenario 2: the initial position of the robot is modified.

Even if the PSIK workspace intersects vertical line $\omega = \pi/4$, the robot may be unable to perform a concave transition between two beams (e.g. to climb to beam b_2 in the example of Figure 5.4a) if the distance between the robot and the beam to be climbed is not adequate. This is because, according to Figure 5.6, for most of the geometries the feasible concave transitions (those obtained intersecting the PSIK workspace and line $\omega = \pi/4$) require μ to be between a minimum value μ_1 and a maximum value μ_2 (e.g. see the default PSIK workspace WS_d in Figure 5.6a). As a result, if the robot is too close ($\mu < \mu_1$) or too far ($\mu > \mu_2$) from beam b_2, the concave transition will be impossible (in that case, region R of solutions depicted in Figure 5.4b will be empty).

This can be a problem if the distance between the robot and the beam to be climbed cannot be adjusted continuously, but only by means of discrete increments, like when using purely binary actuators or the gait shown in Figure 5.8. With this gait, which uses the rotations $\{\theta_A, \theta_B\}$ of the hip, the robot can only travel a distance equal to an integer multiple of the distance t between the rotation axes of the hip. Thus, depending on the initial position of the robot, approaching beam b_2 using this gait may eventually place the robot at a point too close or too far from the beam, without the possibility to finely adjust its position to make possible the execution of a concave transition.

Nevertheless, this can be avoided if the inchworm-like gait of Figure 5.7 is used to approach beam b_2. With $\omega = \pi/2$, the robot travels a distance equal to 2μ, where μ can take any value from line $\omega = \pi/2$ lying inside the PSIK workspace for the considered geometry. Thus, the use of this gait permits the robot to continuously adjust its position along beam b_1 to place itself at a proper distance from beam b_2 to climb it performing a concave transition, as in Figure 5.4a.

5.3.2.3 Scenario 3: the width of the beams is modified.

Even if the PSIK workspace intersects vertical line $\omega = 3\pi/4$, the robot may be unable to perform a convex transition between different faces of a beam if the distance μ necessary to execute this transition (shown in Figure 5.5b) is too large, since the point representing this transition in the (ω, μ) plane may lie outside the PSIK workspace of the considered geometry. For example, the convex transition of Figure 5.5b, which is represented by point P in Figure 5.6d, can be performed using the default geometry because P lies inside the default PSIK workspace, but if the stroke of the linear actuators is decreased to $\Delta\rho = 5$ cm, P will lie outside the workspace and this transition will be impossible.

As Figure 5.5b shows, the distance μ necessary to perform a convex transition equals the sum of the size f of the foot (indicated both in Figure 5.1a and Figure 5.7-left) and 0.5 times the width of the cross section of the beam. This assumes that the fixed foot of the robot is placed at the middle of the cross section of the beam, which will be the most typical case. Thus, if f is too large, and/or the beam is sufficiently wide, the robot may be unable to perform a transition between different faces of a beam.

In case the beams of a given structure are too wide to allow the robot to execute convex transitions between different faces of the beams, the design of the robot should be modified. One possibility is to reduce the size f of the feet, which will depend on the type of adhesion device used for climbing (these devices will be discussed and designed for the HyReCRo robot in Chapter 8). Another possibility is to modify some of the design parameters of Figure 5.6 to increase the maximum value of μ that can be attained with $\omega = 3\pi/4$. Note that this maximum value changes little with parameters $\{b, \rho_0, h\}$, whereas it increases when increasing $\{t, \Delta\rho\}$ or decreasing p.

To sum up, the analysis presented in this section reveals that by choosing an appropriate design for the robot, the planar and symmetric postures of the PSIK problem provide a relatively high flexibility to explore 3D structures using the three basic movements defined in this section. This is partly thanks to the possibility of precisely adjusting the position of the robot along a beam by means of the inchworm-like gait of Figure 5.7. This result suggests that the architecture of the HyReCRo robot may be modified to design a simpler robot based on the PSIK problem, with an actuation scheme in which a single actuator may simultaneously drive various joints, producing symmetric movements. This symmetry in the movements was previously identified by Balaguer et al. [13] as one of the key criteria for designing climbing robots, since this symmetry reduces the number of actuators and hence the overall weight of the robot. However, these simplifications/variations of the HyReCRo robot will be studied in the future, and are beyond the scope of this thesis.

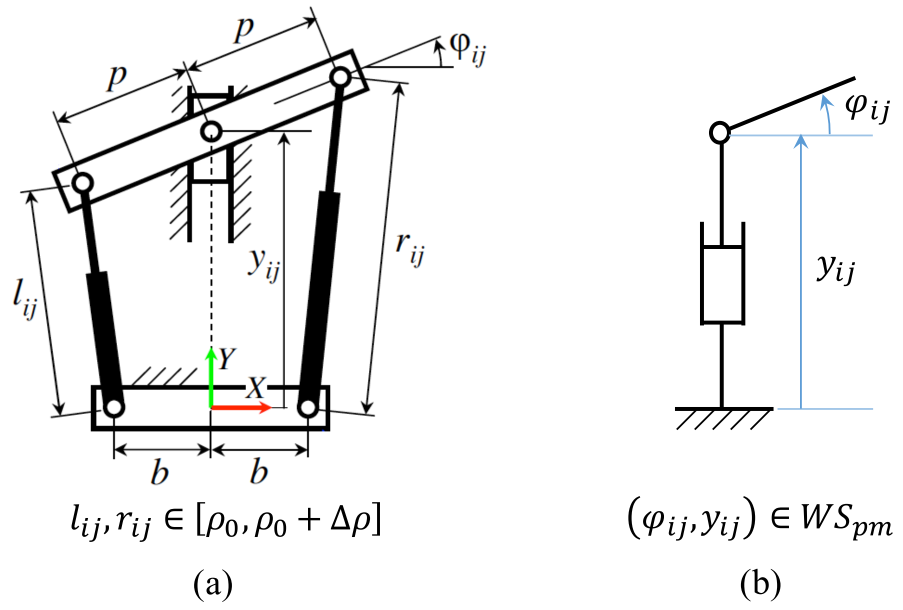
5.4 General Inverse Kinematic Problem

As we have seen in the previous section, solving the PSIK problem is very useful for determining planar and symmetric postures of the HyReCRo robot, which can be used for performing the basic movements necessary for exploring three-dimensional structures. However, in practice we will need to solve the inverse kinematics for planar and not necessarily symmetric postures. Furthermore, in general we will need to solve the inverse kinematic problem to achieve a desired arbitrary three-dimensional relative posture between the feet of the robot. Thus, although useful, the PSIK problem has a limited scope, and it is necessary to solve the general inverse kinematic problem of the HyReCRo robot, which consists in determining all ten active joint coordinates to achieve a desired 3D relative pose (position and orientation) between the feet.

In this section, we will solve the general inverse kinematic problem of the HyReCRo robot, without limiting our analysis to planar or symmetric postures. To this end, it will be convenient to begin by studying the workspace of the symmetric 2R$\underline{\text{P}}$R-PR parallel modules that compose the legs of the HyReCro robot.

5.4.1 Workspace of Symmetric 2R$\underline{\text{P}}$R-PR Parallel Mechanisms

This section analyzes the inverse kinematics and workspace of the symmetric 2R$\underline{\text{P}}$R-PR parallel mechanisms that compose the legs of the climbing robot. Figure 5.9a (repeated from Figure 5.1b) shows the i-th parallel mechanism of leg j ($i \in \{1, 2\}$, $j \in \{A, B\}$). The forward kinematics of this mechanism, which was analyzed in subsection 5.2.1, consists in determining the position y_{ij} and orientation φ_{ij} of the mobile platform in terms of the lengths $\{l_{ij}, r_{ij}\}$ of the two linear actuators. Conversely, the inverse



p
p
φij
lij
yij
rij
Y
X
b
b
$l_{ij}, r_{ij} \in [\rho_0, \rho_0 + \Delta\rho]$
(a)
φij
yij
$(\varphi_{ij}, y_{ij}) \in WS_{pm}$
(b)

Figure 5.9: (a) Symmetric 2R$\underline{\text{P}}$R-PR parallel modules and (b) their equivalent serial mechanisms.

kinematic problem consists in calculating the values of joint coordinates l_{ij} and r_{ij} that yield a desired position y_{ij} and orientation φ_{ij}. The solution to the inverse problem can be easily obtained from Equations (5.1)-(5.2):

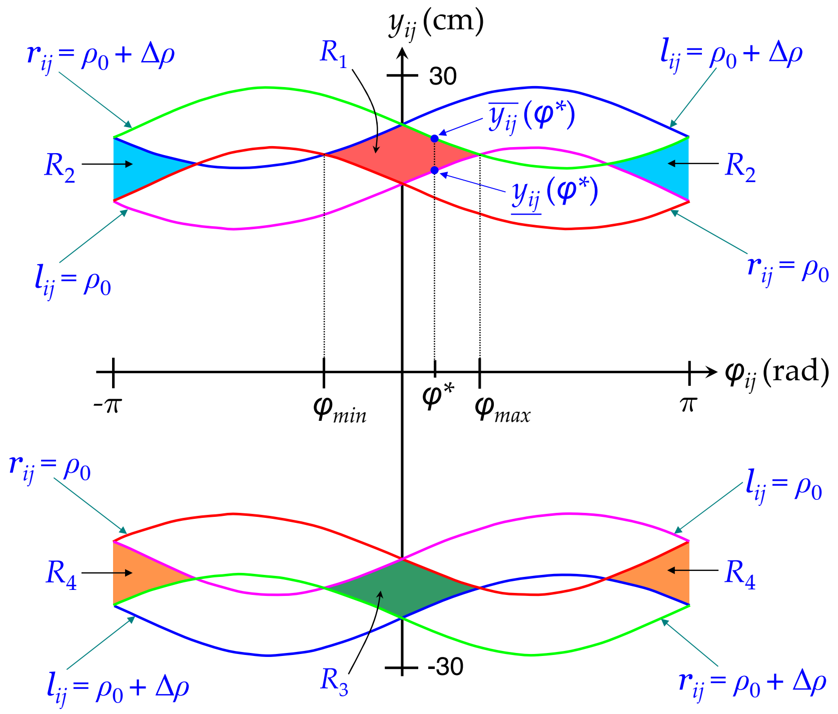
$$l_{ij} = \sqrt{(p\cos\varphi_{ij} - b)^2 + (y_{ij} - p\sin\varphi_{ij})^2} \tag{5.37}$$

$$r_{ij} = \sqrt{(p\cos\varphi_{ij} - b)^2 + (y_{ij} + p\sin\varphi_{ij})^2} \tag{5.38}$$

As stated earlier, in practice, joint coordinates have limits: both l_{ij} and r_{ij} must be in $[\rho_0, \rho_0 + \Delta\rho]$, where $\rho_0 > 0$ is the minimum length of the linear actuators and $\Delta\rho > 0$ is their stroke. Thus, the workspace of 2R$\underline{\text{P}}$R-PR parallel mechanisms can be defined as the set of pairs (φ_{ij}, y_{ij}) for which the right-hand side of both Equations (5.37) and (5.38) is between ρ_0 and $\rho_0 + \Delta\rho$.

For example, Figure 5.10 shows the workspace of a symmetric 2R$\underline{\text{P}}$R-PR parallel mechanism for the following parameters: $b = p = 4$ cm, $\rho_0 = 19$ cm, and $\Delta\rho = 6$ cm, which are the same values used in the example of subsection 5.3.1.1. For this geometry, the workspace is divided into four diamond-shaped regions R_1, R_2, R_3, and R_4, which are enclosed by the curves in which the joint coordinates equal either ρ_0 or $\rho_0 + \Delta\rho$.

Any point of these four regions satisfies the joint limits. However, not all these regions contain feasible configurations. It turns out that each one of these regions corresponds to one of the four assembly modes identified in section 3.3 for symmetric 2R$\underline{\text{P}}$R-PR mechanisms. In that section, assembly modes were identified by the following labels: $\{H+, X+, H-, X-\}$, which indicate the relative position of the linear



y_{ij} (cm)
30
-30
$r_{ij} = \rho_0 + \Delta\rho$
R_1
$l_{ij} = \rho_0 + \Delta\rho$
$\overline{y_{ij}}(\varphi^*)$
R_2
R_2
$\underline{y_{ij}}(\varphi^*)$
$l_{ij} = \rho_0$
$r_{ij} = \rho_0$
φ_{ij} (rad)
-π
φ_{min}
φ^*
φ_{max}
π
$r_{ij} = \rho_0$
$l_{ij} = \rho_0$
R_4
R_4
$l_{ij} = \rho_0 + \Delta\rho$
R_3
$r_{ij} = \rho_0 + \Delta\rho$

Figure 5.10: Workspace of symmetric 2R<u>P</u>R-PR parallel modules.

actuators ("H" means actuators almost parallel, "X" means actuators crossing) and the sign of y_{ij} ("+" means $y_{ij} > 0$, "−" means $y_{ij} < 0$). These four assembly modes are associated with regions R_1, R_2, R_3, and R_4 of Figure 5.10, respectively. As explained in section 3.3, assembly modes $\{X+, H-, X-\}$ are not feasible in practice due to mechanical interferences between different parts of the robot. For this reason, workspace regions $\{R_2, R_3, R_4\}$ will be discarded and, from now on, we will consider that the workspace WS_{pm} of symmetric 2R<u>P</u>R-PR parallel modules consists only of region R_1 of Figure 5.10, which can be described as follows:

$$WS_{pm} = \left\{(\varphi_{ij}, y_{ij}) : \varphi_{min} \le \varphi_{ij} \le \varphi_{max}, \underline{y_{ij}}(\varphi_{ij}) \le y_{ij} \le \overline{y_{ij}}(\varphi_{ij})\right\} \qquad (5.39)$$

where $\underline{y_{ij}}(\varphi^*)$ and $\overline{y_{ij}}(\varphi^*)$ are the lower and upper bounds of the variable y_{ij} for $\varphi_{ij} = \varphi^*$, respectively (see Figure 5.10). In the following, it will be assumed that the valid combinations of variables φ_{ij} and y_{ij} of the parallel mechanisms lie in regions defined by Equation (5.39). Note that, although Equation (5.39) has been introduced using the example region R_1 of Figure 5.10, it defines a more general region in $\mathbb{R}^2$. In particular, Equation (5.39) will still be a valid definition of the feasible workspace of the parallel mechanisms if the geometric parameters are perturbed from the values used in the previous example, provided that these perturbations are sufficiently small.

What is the point in analyzing the workspace of the 2R<u>P</u>R-PR parallel modules for solving the inverse kinematic problem of the complete HyReCRo robot? The answer is: to simplify. Note that, after studying the workspace of these parallel modules, we can omit the parallel architecture of these modules and study them as if they were serial mechanisms with PR architecture (see Figure 5.9b), with the restriction that $(\varphi_{ij}, y_{ij}) \in WS_{pm}$. In this way, we will be able to solve the inverse kinematic problem of the complete HyReCRo robot without having to involve lengths (l_{ij}, r_{ij}) (and their joint limits) in the calculations, since this information is implicitly contained in region WS_{pm} defined in Equation (5.39). This will allow us to analyze and solve the inverse kinematics of the complete HyReCRo robot (which has a serial-parallel architecture) as if it was a serial robot.

5.4.2 Intermediate Joint Coordinates

The inverse kinematics of the complete HyReCRo robot can be defined as the problem of finding the values of all ten actuated joint coordinates $\{l_{1A}$, r_{1A}, l_{2A}, r_{2A}, l_{1B}, r_{1B}, l_{2B}, r_{2B}, θ_A, $\theta_B\}$ which are necessary to achieve a desired relative position and orientation between the feet of the robot.

However, as argued in the previous subsection, instead of directly calculating the lengths of the linear actuators of the parallel modules (variables l_{ij} and r_{ij}), it will be more convenient to regard the HyReCRo robot as an equivalent serial robot and to consider the following variables as the unknowns of the problem, which will be called *intermediate joint coordinates*: $\{\varphi_{1A}$, φ_{2A}, φ_{1B}, φ_{2B}, y_{1A}, y_A, y_{1B}, y_B, θ_A, $\theta_B\}$. As discussed in the previous subsection, most of these unknowns are positions (y_{ij}) and orientations (φ_{ij}) of the mobile platforms of the parallel mechanisms. However, two new variables have appeared among the intermediate joint coordinates: y_A and y_B. These new variables are defined as follows:

$$y_j = y_{1j} + y_{2j} - h, \quad j \in \{A, B\} \tag{5.40}$$

Variable y_j can be seen as the length of leg j (i.e., the distance between the origins of frames j and G_j, see Figure 5.2). Note that, for the purpose of solving the inverse kinematics, solving intermediate joint coordinates is equivalent to solving the actuated joint coordinates. Indeed, if we can calculate all intermediate joint coordinates, we can use Equation (5.40) to compute y_{2j}. In this way, positions and orientations $\{y_{ij}, \varphi_{ij}\}$ are known for all parallel modules. Then, using the solutions of the inverse kinematics of parallel modules [Equations (5.37) and (5.38)], we can compute $\{l_{ij}, r_{ij}\}$ from $\{y_{ij}, \varphi_{ij}\}$, obtaining a unique solution. This is because a set of intermediate joint coordinates yields a unique set of actuated joint coordinates, since the inverse kinematic problem of each parallel module has a single solution given by Equations (5.37) and (5.38) (as opposed to their forward kinematic problem, which has four solutions as explained in Section 5.2.1).

5.4.3 Solving the Inverse Kinematics

As stated in the previous subsection, the inverse kinematics of the HyReCRo robot can be defined as the problem consisting in computing the intermediate joint coordinates that yield a desired relative position and orientation between the feet of the robot. The desired relative position and orientation between the feet of the robot can be defined by the homogeneous transformation matrix $\mathbf{T}_{B/A}$, which encodes the position and orientation of foot B relative to foot A. Assume that the input of the inverse kinematic problem is the desired matrix $\mathbf{T}_{B/A}$, with the following form:

$$\mathbf{T}_{B/A} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & p_x \\ R_{21} & R_{22} & R_{23} & p_y \\ R_{31} & R_{32} & R_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.41}$$

where R_{mn} are the entries of the rotation matrix encoding the orientation of foot B relative to foot A, whereas $[p_x, p_y, p_z]$ are the position coordinates of foot B relative to foot A. Note that, if the input of the inverse kinematics was $\mathbf{T}_{A/B}$ (i.e., the position and orientation of foot A relative to foot B) instead of $\mathbf{T}_{B/A}$, one only needs to invert $\mathbf{T}_{A/B}$ to obtain $\mathbf{T}_{B/A} = (\mathbf{T}_{A/B})^{-1}$, and proceed normally with the calculations as explained next.

If $\mathbf{T}_{B/A}$ is given, the inverse kinematics boils down to solving the intermediate joint coordinates from Equation (5.18), which is repeated next:

$$\mathbf{T}_{B/A} = \mathbf{T}_{H_A/A}\mathbf{T}_{H_B/H_A}(\mathbf{T}_{H_B/B})^{-1} \tag{5.42}$$

The left-hand side of Equation (5.42), which is known when solving the inverse kinematics, has the expression given in Equation (5.41). In regard to the right-hand side of Equation (5.42), the expression of matrix $\mathbf{T}_{H_B/H_A}$ was given in Equation (5.19), whereas the expression of $\mathbf{T}_{H_j/j}$ ($j = A, B$) can be obtained by multiplying the three matrices on the right-hand side of Equation (5.17), which yields:

$$\mathbf{T}_{H_j/j} = \begin{bmatrix} c_{\theta_j}c_j & s_j & s_{\theta_j}c_j & y_j s_{\varphi_{1j}} \\ -c_{\theta_j}s_j & c_j & -s_{\theta_j}s_j & y_j c_{\varphi_{1j}} \\ -s_{\theta_j} & 0 & c_{\theta_j} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.43}$$

where y_j was defined in Equation (5.40) and the following variables have been defined to abbreviate the trigonometric terms: $s_{\theta_j} = \sin\theta_j$, $c_{\theta_j} = \cos\theta_j$, $s_{\varphi_{1j}} = \sin\varphi_{1j}$, $c_{\varphi_{1j}} = \cos\varphi_{1j}$, $s_j = \sin(\varphi_{1j} - \varphi_{2j})$, $c_j = \cos(\varphi_{1j} - \varphi_{2j})$.

According to Equation (5.43), the right-hand side of Equation (5.42) contains the unknown intermediate joint coordinates that must be solved. Trying to solve Equation (5.42) directly can be difficult because all unknowns are coupled on its right-hand side, and the resulting equations are quite large and difficult to handle. However, if Equation (5.42) is postmultiplied by $\mathbf{T}_{H_B/B}$, it becomes:

$$\underbrace{\mathbf{T}_{B/A}\mathbf{T}_{H_B/B}}_{\mathbf{B}} = \underbrace{\mathbf{T}_{H_A/A}\mathbf{T}_{H_B/H_A}}_{\mathbf{A}} \tag{5.44}$$

In this way, the problem becomes easier since all unknowns associated with each leg have been decoupled and appear on different sides of the equation. Indeed, the matrix on the left-hand side of Equation (5.44) only depends on the variables of leg B $\{y_B, \varphi_{1B}, \varphi_{2B}, \theta_B\}$, whereas the right-hand side only depends on the variables of leg

$A\ \{y_A, \varphi_{1A}, \varphi_{2A}, \theta_A\}$. In what follows, the left-hand side of Equation (5.44) will be denoted by $\mathbf{B}$, whereas the right-hand side will be denoted by $\mathbf{A}$. The expressions of these matrices, which involve much simpler terms than the right-hand side of Equation (5.42), can be found in Equations (5.45) and (5.46):

$$\mathbf{A} = \begin{bmatrix} c_{\theta_A} c_A & s_A & s_{\theta_A} c_A & y_A s_{\varphi_{1A}} + t c_{\theta_A} c_A \\ -c_{\theta_A} s_A & c_A & -s_{\theta_A} s_A & y_A c_{\varphi_{1A}} - t c_{\theta_A} s_A \\ -s_{\theta_A} & 0 & c_{\theta_A} & -t s_{\theta_A} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.45}$$

$$\mathbf{B} = \left[\begin{array}{cc} c_{\theta_B}(c_B R_{11} - s_B R_{12}) - R_{13} s_{\theta_B} & s_B R_{11} + c_B R_{12} \\ c_{\theta_B}(c_B R_{21} - s_B R_{22}) - R_{23} s_{\theta_B} & s_B R_{21} + c_B R_{22} \\ c_{\theta_B}(c_B R_{31} - s_B R_{32}) - R_{33} s_{\theta_B} & s_B R_{31} + c_B R_{32} \\ 0 & 0 \end{array} \right. \cdots$$

$$\cdots \left. \begin{array}{cc} s_{\theta_B}(c_B R_{11} - s_B R_{12}) + R_{13} c_{\theta_B} & R_{11} y_B s_{\varphi_{1B}} + R_{12} y_B c_{\varphi_{1B}} + p_x \\ s_{\theta_B}(c_B R_{21} - s_B R_{22}) + R_{23} c_{\theta_B} & R_{21} y_B s_{\varphi_{1B}} + R_{22} y_B c_{\varphi_{1B}} + p_y \\ s_{\theta_B}(c_B R_{31} - s_B R_{32}) + R_{33} c_{\theta_B} & R_{31} y_B s_{\varphi_{1B}} + R_{32} y_B c_{\varphi_{1B}} + p_z \\ 0 & 1 \end{array} \right] \tag{5.46}$$

To solve the inverse kinematics, we must solve the equations that are obtained when all elements of matrix $\mathbf{A}$ are equated to the respective elements of matrix $\mathbf{B}$. Equating elements $(1,4)$, $(2,4)$, and $(3,4)$ of both matrices yields the following equations:

$$y_A s_{\varphi_{1A}} + t c_{\theta_A} c_A = R_{11} y_B s_{\varphi_{1B}} + R_{12} y_B c_{\varphi_{1B}} + p_x \tag{5.47}$$
$$y_A c_{\varphi_{1A}} - t c_{\theta_A} s_A = R_{21} y_B s_{\varphi_{1B}} + R_{22} y_B c_{\varphi_{1B}} + p_y \tag{5.48}$$
$$-t s_{\theta_A} = R_{31} y_B s_{\varphi_{1B}} + R_{32} y_B c_{\varphi_{1B}} + p_z \tag{5.49}$$

Equating elements $(1,2)$, $(2,2)$, $(3,1)$, $(3,3)$, and $(3,2)$ gives the following equations:

$$s_A = s_B R_{11} + c_B R_{12} \tag{5.50}$$
$$c_A = s_B R_{21} + c_B R_{22} \tag{5.51}$$
$$s_{\theta_A} = R_{33} s_{\theta_B} - c_{\theta_B}(c_B R_{31} - s_B R_{32}) \tag{5.52}$$
$$c_{\theta_A} = R_{33} c_{\theta_B} + s_{\theta_B}(c_B R_{31} - s_B R_{32}) \tag{5.53}$$
$$0 = s_B R_{31} + c_B R_{32} \tag{5.54}$$

It can be shown [136] that, if Equations (5.50) to (5.54) are satisfied, then elements $(1,1)$, $(1,3)$, $(2,1)$, and $(2,3)$ of $\mathbf{A}$ and $\mathbf{B}$ are equal. Hence, the equations that are obtained when equating these elements of both matrices must not be included since they depend on Equations (5.50) to (5.54), and we only need to solve Equations (5.47) to (5.54). These equations are solved next, distinguishing two cases.

5.4.3.1 Case 1: $(R_{31}, R_{32}) \neq (0,0)$

In this case, $R_{31}^2 + R_{32}^2 \neq 0$ and Equation (5.54) can be solved together with the condition $s_B^2 + c_B^2 = 1$, obtaining two solutions for s_B and c_B:

$$s_B = \sin(\varphi_{1B} - \varphi_{2B}) = \sigma_1 \frac{R_{32}}{\sqrt{R_{31}^2 + R_{32}^2}} \tag{5.55}$$

$$c_B = \cos(\varphi_{1B} - \varphi_{2B}) = \sigma_1 \frac{-R_{31}}{\sqrt{R_{31}^2 + R_{32}^2}} \tag{5.56}$$

where $\sigma_1 \in \{-1, 1\}$. Once s_B and c_B are known, the difference $\varphi_{1B} - \varphi_{2B}$ can be calculated as follows:

$$\varphi_{1B} - \varphi_{2B} = \mathrm{atan2}(s_B, c_B) \tag{5.57}$$

where atan2 is the well-known inverse tangent function that considers the signs of the sine and the cosine and computes the angle in the correct quadrant. Equation (5.57) gives φ_{2B} in terms of φ_{1B}. After calculating s_B and c_B, s_A and c_A are calculated from Equations (5.50) and (5.51), respectively. Then, the difference $\varphi_{1A} - \varphi_{2A}$ is obtained as follows:

$$\varphi_{1A} - \varphi_{2A} = \mathrm{atan2}\left(s_A, c_A\right) \tag{5.58}$$

which provides φ_{2A} in terms of φ_{1A}. Once s_A and c_A are known, Equations (5.47) to (5.49) constitute a system of three equations in five unknowns: $\{\varphi_{1B}, \varphi_{1A}, y_B, y_A, \theta_A\}$. Hence, three unknowns must be solved in terms of the other two, which must be assumed to be known. From the form of these equations, it is evident that the solution is straightforward if y_B and φ_{1B} are chosen as the known variables, since in that case the right-hand side of the equations is known. Particularly, Equation (5.49) yields $\sin\theta_A$:

$$s_{\theta_A} = -\frac{R_{31} y_B s_{\varphi_{1B}} + R_{32} y_B c_{\varphi_{1B}} + p_z}{t} \tag{5.59}$$

Note that the right-hand side of Equation (5.59) must be in $[-1, 1]$ in order to obtain a real value for θ_A. Knowing the value of s_{θ_A}, two solutions are obtained for $\cos\theta_A$:

$$c_{\theta_A} = \sigma_2 \sqrt{1 - s_{\theta_A}^2} \tag{5.60}$$

where $\sigma_2 \in \{-1, 1\}$. Once s_{θ_A} and c_{θ_A} are known, s_{θ_B} and c_{θ_B} are calculated inverting Equations (5.52) and (5.53), which can always be inverted [136]. The solution for s_{θ_B} and c_{θ_B} is:

$$s_{\theta_B} = R_{33} s_{\theta_A} + c_{\theta_A}(c_B R_{31} - s_B R_{32}) \tag{5.61}$$

$$c_{\theta_B} = R_{33} c_{\theta_A} - s_{\theta_A}(c_B R_{31} - s_B R_{32}) \tag{5.62}$$

Then, θ_A and θ_B are obtained as follows:

$$\theta_j = \mathrm{atan2}\left(s_{\theta_j}, c_{\theta_j}\right), \quad j \in \{A, B\} \tag{5.63}$$

After c_{θ_A} has been calculated, Equations (5.47) and (5.48) can be rewritten as follows:

$$y_A s_{\varphi_{1A}} = \Omega_x \tag{5.64}$$

$$y_A c_{\varphi_{1A}} = \Omega_y \tag{5.65}$$

where the terms Ω_x and Ω_y, on the right-hand side of the equations, are known quantities:

$$\Omega_x = R_{11} y_B s_{\varphi_{1B}} + R_{12} y_B c_{\varphi_{1B}} + p_x - t c_{\theta_A} c_A \tag{5.66}$$
$$\Omega_y = R_{21} y_B s_{\varphi_{1B}} + R_{22} y_B c_{\varphi_{1B}} + p_y + t c_{\theta_A} s_A \tag{5.67}$$

Squaring Equations (5.64) and (5.65) and adding them together eliminates φ_{1A} and yields the following solution for y_A:

$$y_A = \pm\sqrt{\Omega_x^2 + \Omega_y^2} \tag{5.68}$$

After calculating y_A, φ_{1A} is obtained as follows:

$$\varphi_{1A} = \text{atan2}\left(\Omega_x / y_A, \Omega_y / y_A\right) \tag{5.69}$$

which is valid only if $y_A \neq 0$. Note, however, that the case $y_A \leq 0$ will be impossible in practice due to mechanical interference between different links of leg A. Hence, we will be interested only in strictly positive solutions of y_A, given by the positive sign in Equation (5.68). In that case, y_A can be removed from Equation (5.69) because an angle is not affected if its sine and cosine are multiplied by the same positive constant (whereas multiplying them by a negative constant is equivalent to adding π rad to the angle). Hence, φ_{1A} can be directly obtained as follows:

$$\varphi_{1A} = \text{atan2}\left(\Omega_x, \Omega_y\right) \tag{5.70}$$

With the obtained value of φ_{1A}, φ_{2A} is calculated using Equation (5.58). Finally, φ_{2B} is calculated from φ_{1B} (which is assumed to be known) using Equation (5.57).

Note that intermediate joint coordinates y_{1A} and y_{1B} remain undetermined after solving all the equations. Hence, the values of these variables must also be fixed in order to complete the solution to the inverse kinematics. Assuming that the values of these variables are known, the solution to the inverse kinematics can be summarized as follows:

$$\begin{gathered} \{y_A, \varphi_{1A}, \varphi_{2A}, \varphi_{2B}, \theta_A, \theta_B\} = \mathbf{f}_1\left(\varphi_{1B}, y_B\right) \\ \{\varphi_{1B}, y_B, y_{1A}, y_{1B}\} = \text{free parameters} \end{gathered} \tag{5.71}$$

where the symbol $\mathbf{f}_1$ indicates that the variables on the left-hand side of Equation (5.71) are obtained from $\{\varphi_{1B}, y_B\}$ as described in this section.

5.4.3.2 Case 2: $R_{31} = R_{32} = 0$

Since $R_{31}^2 + R_{32}^2 + R_{33}^2 = 1$, the condition $R_{31} = R_{32} = 0$ is equivalent to $R_{33}^2 = 1$, which means that the Z axes of the coordinate frames attached to the feet are parallel. This situation appears in many simple movements of the robot when exploring a structure, such as in concave and convex transitions between different planes (e.g., see Figures 5.4 and 5.5).

This case admits a similar analysis to the previous one: some unknowns must be solved in terms of the others and, in particular, we can choose to solve all unknowns in terms of variables φ_{1B} and y_B to deal with simpler equations. However, since $R_{31} = R_{32} = 0$, a restriction is lost because Equation (5.54) is identically satisfied. Thus, the difference $\varphi_{1B} - \varphi_{2B}$ is no longer fixed and can adopt any value, which means that the value of variable φ_{2B} must also be decided. Except for this difference, the resolution procedure is exactly as described in the previous section: we fix the values of $\{y_B, \varphi_{1B}, \varphi_{2B}\}$, compute $s_B = \sin(\varphi_{1B} - \varphi_{2B})$ and $c_B = \cos(\varphi_{1B} - \varphi_{2B})$, and use Equations (5.58) to (5.70) to calculate the rest of the variables.

As in Case 1, the values of variables y_{1A} and y_{1B} must also be fixed since they remain undetermined after solving the equations. Thus, in this case, the solution to the inverse kinematics can be summarized as follows:

$$\begin{aligned} \{y_A, \varphi_{1A}, \varphi_{2A}, \theta_A, \theta_B\} &= \mathbf{f}_2\left(\varphi_{1B}, \varphi_{2B}, y_B\right) \\ \{\varphi_{1B}, \varphi_{2B}, y_B, y_{1A}, y_{1B}\} &= \text{free parameters} \end{aligned} \tag{5.72}$$

where the symbol $\mathbf{f}_2$ indicates that the variables on the left-hand side of Equation (5.72) are obtained from $\{\varphi_{1B}, \varphi_{2B}, y_B\}$ as described in this section.

5.4.4 Geometric Interpretation: Self-motion Manifolds

For non-redundant robots, the solution to the inverse kinematics for a desired position and orientation is a finite set of isolated points in the space of joint coordinates (joint space). However, for redundant robots (like the HyReCRo robot studied in this thesis), the values of the joint coordinates that permit the robot to reach a desired position and orientation lie on positive-dimensional manifolds (curves, surfaces, etc.) in the joint space. These are the so-called *self-motion manifolds*, because varying the joint coordinates along them modifies the posture of the robot without affecting the position and orientation of its end-effector, which remains motionless [22].

For this robot, the solutions to the inverse kinematics can be interpreted as self-motion manifolds in the ten-dimensional space of intermediate joint coordinates. In Case 1, the solutions lie on four-dimensional manifolds because the solution is parameterized in terms of four parameters, and Equation (5.71) provides a possible parameterization of these 4D self-motion manifolds. In Case 2, the dimension of self-motion manifolds is five, since five parameters must be fixed in order to determine all intermediate joint coordinates. In this case, a possible parametrization of these 5D self-motion manifolds is given by Equation (5.72).

Note that Equations (5.71) and (5.72) are not the only valid parametrizations of self-motion manifolds. There are many other parametrizations depending on the redundant variables that are assumed to be known when solving the equations of the inverse kinematics. For example, instead of choosing $\{y_B, \varphi_{1B}\}$, we could have chosen $\{\varphi_{1A}, \varphi_{1B}\}$ or $\{y_A, y_B\}$ as the known variables when solving three unknowns from Equations (5.47) to (5.49). In these cases, however, it is evident that the solution to these equations is not as straightforward as it is when $\{y_B, \varphi_{1B}\}$ are chosen as the known variables.

Although self-motion manifolds encode all the information regarding the solution to the inverse kinematics of redundant robots, the self-motion manifolds obtained for this robot are not practical due to their high dimension. Next, we propose a more useful representation of the solutions to the inverse kinematics of the HyReCRo robot.

5.4.5 Regions of Feasible Solutions

In previous subsections, it has been shown that the solution to the inverse kinematics can be written in terms of some free parameters or *decision variables* whose values must be decided: $\{\varphi_{1B}, y_B, y_{1A}, y_{1B}\}$ (and also φ_{2B}, when $R_{33}^2 = 1$). This section discusses how to choose appropriate values for these decision variables.

First, note that the chosen values should satisfy the joint limits of the parallel modules. As shown in Section 5.4.1, these joint limits impose restrictions on variables (φ_{ij}, y_{ij}) of the parallel modules. In this aspect, it will be assumed that the valid pairs (φ_{ij}, y_{ij}) of each parallel module belong to regions WS_{pm} of the form of Equation (5.39).

Taking the joint limits into consideration, the best representation of the solutions to the inverse kinematics of this robot would be the combinations of the decision variables $\{\varphi_{1B}, y_B, y_{1A}, y_{1B}\}$ (and also φ_{2B}, if $R_{33}^2 = 1$) that yield a desired position and orientation between the feet while satisfying the joint limits of parallel modules. This representation would consist of regions of feasible combinations of these variables in a four or five-dimensional space. However, as we will show next, it is possible to encode the most relevant information of these feasible regions using projections on only two or three dimensions.

Note that the posture of the robot is not affected equally by all decision variables. According to Equations (5.71) and (5.72), fixing all decision variables except $\{y_{1A}, y_{1B}\}$ fixes the remaining intermediate joint coordinates, which determine the posture of the robot. Once y_j has been fixed, varying y_{1j} only modifies the position of the core link of leg j along this leg, without affecting the posture of the robot. This is because a variation in y_{1j} will be compensated by the opposite variation in variable y_{2j} to conserve the value of y_j, which has already been fixed (see Figure 5.2).

Since variables $\{y_{1A}, y_{1B}\}$ do not affect the overall posture of the robot (except for the position of the core links along the legs), their concrete values are not important, as long as they satisfy the joint limits of the parallel modules. Thus, it is sufficient to focus only on the decision variables $\{\varphi_{1B}, \varphi_{2B}, y_B\}$, which determine the overall posture of the robot and are important for planning its movements (e.g., for choosing the postures that yield a desired position and orientation avoiding some collisions).

For Case 1 ($R_{33}^2 \neq 1$), we are interested in the regions (areas) R_f of plane (φ_{1B}, y_B) for which we can find values for the other decision variables (y_{1A} and y_{1B})

that permit the robot to reach a desired position and orientation satisfying the joint limits. A discrete approximation of these areas can be obtained using Algorithm 1. This algorithm picks N_p random points in plane (φ_{1B}, y_B), computes the intermediate joint coordinates $\{y_A, \varphi_{1A}, \varphi_{2A}, \varphi_{2B}, \theta_A, \theta_B\}$ for each point using the equations of Section 5.4.3.1, and discards all points for which the intermediate joint coordinates do not satisfy the joint limits of the parallel modules. According to Section 5.4.1, it is necessary that angles $\{\varphi_{1B}, \varphi_{2B}, \varphi_{1A}, \varphi_{2A}\}$ be in $[\varphi_{min}, \varphi_{max}]$ for a point to satisfy the joint limits. Moreover, y_j $(j \in \{A, B\})$ must be in $[\underline{y_j}, \overline{y_j}]$, where the bounds $\underline{y_j}$ and $\overline{y_j}$ are calculated using Equation (5.40) together with the bounds of the variables y_{1j} and y_{2j}, which depend on φ_{1j} and φ_{2j}, respectively.

Algorithm 1 Algorithm to compute the set R_f of feasible solutions to the inverse kinematics, which satisfy the joint limits of the parallel modules.

```
1:  $R_f = \emptyset$ (empty set)
2:  for $k = 1$ to $N_p$ do
3:      Sample $\varphi_{1B}$ uniformly on $[\varphi_{min}, \varphi_{max}]$
4:      Compute $\varphi_{2B}$ using Equation (5.57)
5:      if $\varphi_{2B} \in [\varphi_{min}, \varphi_{max}]$ then
6:          $\underline{y_B} = \underline{y_{1B}}(\varphi_{1B}) + \underline{y_{2B}}(\varphi_{2B}) - h$
7:          $\overline{y_B} = \overline{y_{1B}}(\varphi_{1B}) + \overline{y_{2B}}(\varphi_{2B}) - h$
8:          Sample $y_B$ uniformly on $[\underline{y_B}, \overline{y_B}]$
9:          Compute $\varphi_{1A}$ using Equation (5.70)
10:         if $\varphi_{1A} \in [\varphi_{min}, \varphi_{max}]$ then
11:             Compute $\varphi_{2A}$ using Equation (5.58)
12:             if $\varphi_{2A} \in [\varphi_{min}, \varphi_{max}]$ then
13:                 Compute $y_A$ using Equation (5.68)
14:                 $\underline{y_A} = \underline{y_{1A}}(\varphi_{1A}) + \underline{y_{2A}}(\varphi_{2A}) - h$
15:                 $\overline{y_A} = \overline{y_{1A}}(\varphi_{1A}) + \overline{y_{2A}}(\varphi_{2A}) - h$
16:                 if $y_A \in [\underline{y_A}, \overline{y_A}]$ then
17:                     Store the point $(\varphi_{1B}, y_B)$ in $R_f$
```

For Case 2 ($R_{33}^2 = 1$), we are interested in the regions (volumes) R_f of space $(\varphi_{1B}, \varphi_{2B}, y_B)$ for which we can find values of the decision variables $\{y_{1A}, y_{1B}\}$ that permit the robot to reach a desired position and orientation satisfying the joint limits. Algorithm 1 can also be used to compute a discrete approximation of these volumes. The only difference is in line 4 of the algorithm: instead of calculating φ_{2B} from Equation (5.57), the value of φ_{2B} must be sampled uniformly on $[\varphi_{min}, \varphi_{max}]$, because φ_{2B} cannot be calculated in terms of φ_{1B} in Case 2. Moreover, the points that are stored in R_f in line 17 have three coordinates, instead of two: $(\varphi_{1B}, \varphi_{2B}, y_B)$.

Note that the solution to the inverse kinematics in Case 1 involves two binary variables $\sigma_1, \sigma_2 \in \{-1, 1\}$. The solution in Case 2 involves only the binary variable σ_2. Thus, regions R_f computed by Algorithm 1 will be different for different choices of these binary variables. In both Cases 1 and 2, it is necessary to execute the algorithm for all combinations of the binary variables to avoid missing feasible solutions of the inverse kinematics.

All points calculated by Algorithm 1 guarantee the existence of values for the decision variables y_{1A} and y_{1B} that satisfy the joint limits, as demonstrated in the next subsection 5.4.5.1.

As an alternative to the numerical algorithm presented in this section, one may try to compute the regions R_f or redundant solutions analytically. To obtain analytic representations of these regions, it suffices to obtain analytically the curves (or surfaces, in Case 2) that enclose these regions. This can be done solving Equations (5.58) to (5.70) to find analytic expressions of all intermediate joint coordinates in terms of φ_{1B} and y_B (and also φ_{2B}, in Case 2). Then, equating each intermediate joint coordinate to either its upper or lower limit yields an equation that defines a curve in plane (φ_{1B}, y_B), or a surface in space $(\varphi_{1B}, \varphi_{2B}, y_B)$, and these curves/surfaces constitute the boundaries of regions R_f. Note that the lower and upper limits of φ_{ij} are constants: φ_{min} and φ_{max}, respectively. However, according to Algorithm 1, the bounds of y_j depend on φ_{ij}, which, in turn, must be written in terms of $\{\varphi_{1B}, \varphi_{2B}, y_B\}$.

Unfortunately, obtaining analytically all intermediate joint coordinates and the bounds of y_j in terms of $\{\varphi_{1B}, \varphi_{2B}, y_B\}$ can be very difficult or even unfeasible, even using computer algebra systems, due to the large size and complexity of the analytic expressions that must be handled in the calculations. For instance, Figure 5.12 shows an example of region R_f computed using Algorithm 1, and it also exhibits some of the curves that delimit this region. Only the boundaries that could be obtained analytically are shown; the analytic expressions of the remaining boundaries of R_f could not be obtained due to the complexity and large size of the terms involved in their calculation. Thus, for this robot, an analytic approach to compute the sets of redundant solutions of the inverse kinematics is intractable, and a numerical algorithm like the one proposed in this section stands as the best choice.

5.4.5.1 Choosing feasible values of y_{1A} and y_{1B}

The regions R_f of feasible solutions computed by Algorithm 1 only reflect the valid combinations of variables φ_{1B} and y_B (and also φ_{2B}, in Case 2). However, the values of variables y_{1A} and y_{1B} must also be decided in order to complete the solution of the inverse kinematics.

As discussed in the previous section, since variables $\{y_{1A}, y_{1B}\}$ do not affect the posture of the robot once $\{y_A, y_B\}$ are fixed, their values are not crucial as long as they satisfy the joint limits. Next, it will be shown that, for every point in R_f, we can always find non-empty intervals of possible values for variables y_{1A} and y_{1B} that satisfy the joint limits of the parallel modules. This will be demonstrated for a generic leg y_{1j} $(j \in \{A, B\})$.

According to Equation (5.39), for a given $\varphi_{1j} \in [\varphi_{min}, \varphi_{max}]$, y_{1j} must be in $[\underline{y_{1j}}(\varphi_{1j}), \overline{y_{1j}}(\varphi_{1j})]$. Additionally, y_{1j} must also satisfy Eq (5.40):

$$y_{2j} = y_j + h - y_{1j} \tag{5.73}$$

Similarly, for a given $\varphi_{2j} \in [\varphi_{min}, \varphi_{max}]$, y_{2j} must satisfy:

$$\underline{y_{2j}}(\varphi_{2j}) \leq y_{2j} \leq \overline{y_{2j}}(\varphi_{2j}) \tag{5.74}$$

Combining the conditions given in Equations (5.73) and (5.74) yields the following additional bounds for y_{1j}:

$$y_j + h - \overline{y_{2j}}(\varphi_{2j}) \leq y_{1j} \leq y_j + h - \underline{y_{2j}}(\varphi_{2j}) \tag{5.75}$$

Hence, the valid values of y_{1j} lie in the intersection of two intervals $I_1 = [\underline{y_{1j}}(\varphi_{1j}), \overline{y_{1j}}(\varphi_{1j})]$ and $I_2 = [y_j + h - \overline{y_{2j}}(\varphi_{2j}), y_j + h - \underline{y_{2j}}(\varphi_{2j})]$. Denoting the limits of these intervals by $I_1 = [a_0, a_1]$ and $I_2 = [b_0, b_1]$, their intersection will be empty if and only if [129]:

$$b_1 - a_0 < 0 \quad \text{OR} \quad b_0 - a_1 > 0 \tag{5.76}$$

Then, applying De Morgan's laws to the previous condition, we have that the intersection $I_1 \cap I_2$ will be non-empty iff:

$$b_1 - a_0 \geq 0 \quad \text{AND} \quad b_0 - a_1 \leq 0 \tag{5.77}$$

First, we check the first condition of Equation (5.77):

$$\begin{aligned} & b_1 - a_0 = y_j + h - \underline{y_{2j}}(\varphi_{2j}) - \underline{y_{1j}}(\varphi_{1j}) \geq \\ & \underline{y_{1j}}(\varphi_{1j}) + \underline{y_{2j}}(\varphi_{2j}) - h + h - \underline{y_{2j}}(\varphi_{2j}) - \underline{y_{1j}}(\varphi_{1j}) = 0 \end{aligned} \tag{5.78}$$

where it has been considered that $y_j \geq \underline{y_{1j}}(\varphi_{1j}) + \underline{y_{2j}}(\varphi_{2j}) - h$, which is guaranteed by Algorithm 1. Next, we check the second condition of Equation (5.77):

$$\begin{aligned} & b_0 - a_1 = y_j + h - \overline{y_{2j}}(\varphi_{2j}) - \overline{y_{1j}}(\varphi_{1j}) \leq \\ & \overline{y_{1j}}(\varphi_{1j}) + \overline{y_{2j}}(\varphi_{2j}) - h + h - \overline{y_{2j}}(\varphi_{2j}) - \overline{y_{1j}}(\varphi_{1j}) = 0 \end{aligned} \tag{5.79}$$

where it has been considered that $y_j \leq \overline{y_{1j}}(\varphi_{1j}) + \overline{y_{2j}}(\varphi_{2j}) - h$, which is also guaranteed by Algorithm 1. Thus, it is guaranteed that, for every point of sets R_f computed by the proposed algorithm, we can always find ranges of the variables y_{1A} and y_{1B} that satisfy the joint limits of the parallel modules, completing the solution of the inverse kinematics of the HyReCRo robot.

5.4.6 Examples

This section presents some examples of the application of Algorithm 1 to obtain the region R_f of valid solutions of the inverse kinematics. Algorithm 1 was implemented in Matlab R2011b, and the examples were tested on a laptop Acer Travelmate 5720, with an Intel(R) Core(TM)2 Duo T7300 CPU @ 2 GHz, 2 GB in RAM, and a 32-bit Operating System (Windows 7 Professional).

For the geometric parameters of the parallel modules $\{b, p, \rho_0, \Delta\rho\}$, the values indicated in Section 5.4.1 were used. For the parameters $\{h, t\}$, the following values were used: $h = 16$ cm and $t = 15.6$ cm.

5.4.6.1 Example 1

It is assumed that foot A is fixed, and the following position and orientation is desired for foot B:

$$\mathbf{T}_{B/A} = \begin{bmatrix} 0 & 0 & -1 & -20 \text{ cm} \\ \sin(\pi/3) & \cos(\pi/3) & 0 & 5 \text{ cm} \\ \cos(\pi/3) & -\sin(\pi/3) & 0 & 30 \text{ cm} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.80}$$

This example is represented in Figure 5.11a. Since $(R_{31}, R_{32}) \neq (0,0)$, two binary variables $\sigma_1, \sigma_2 \in \{-1, 1\}$ are involved in the calculation of the intermediate joint coordinates, as explained in Section 5.4.3.1. Hence, in order to obtain all feasible solutions, Algorithm 1 must be executed four times, once for each combination of these binary variables. The set R_f of feasible solutions depends on the choice of the binary variables.

For this example, Algorithm 1 was run using $N_p = 10^4$ and $N_p = 5 \cdot 10^4$ random points. In the first case, the average time of execution of the algorithm was 0.455 s, with a standard deviation (SD) of 0.022 s. For $N_p = 5 \cdot 10^4$, the average execution time was 2.35 s, with a SD of 0.12 s. In both cases, 20 experiments were performed.

In both cases, these are the times necessary for running the algorithm four times, once for each combination of σ_1 and σ_2. The only combination of σ_1 and σ_2 that yields a non-empty set of feasible solutions for this example is $\sigma_1 = \sigma_2 = -1$. The set R_f corresponding to this combination is shown in Figure 5.12. Figure 5.12 represents only the solution obtained for $N_p = 5 \cdot 10^4$ since the density of points is higher, providing a better approximation of the shape of the region of feasible solutions.

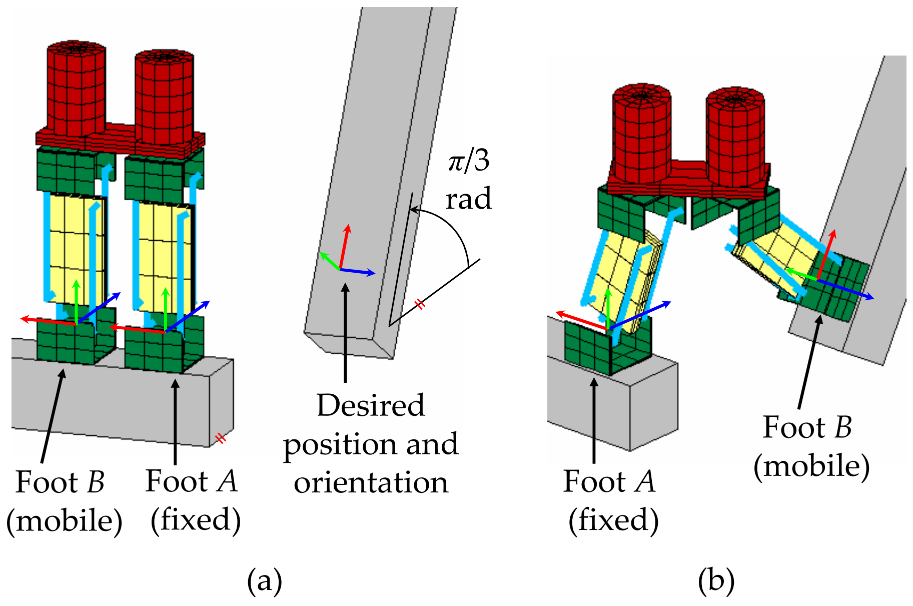


Figure 5.11: Desired position and orientation in Example 1.

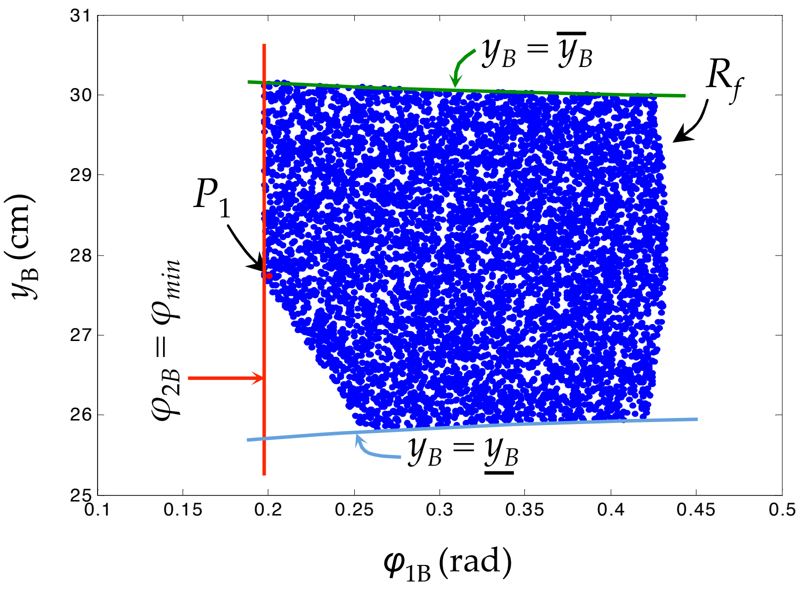


Figure 5.12: Discrete approximation of the set R_f of feasible solutions to inverse kinematics in Example 1, for $N_p = 5 \cdot 10^4$ points.

As shown in the previous section, any point of region R_f guarantees the existence of ranges for y_{1A} and y_{1B} that allow the robot to reach the position and orientation of Equation (5.80) satisfying the joint limits of the parallel modules. For example, point $P_1 = (0.2, 27.75)$ indicated in Figure 5.12 yields $\theta_A = -2.51$ rad and $\theta_B = -0.94$ rad. For this point, the valid ranges of y_{1A} and y_{1B} are $[20.21, 20.31]$ cm and $[21.79, 21.80]$ cm, respectively (the values are rounded to two decimal places). These ranges are very narrow because P_1 is close to the boundary of R_f. Choosing $y_{1A} = 20.21$ cm and $y_{1B} = 21.79$ cm gives the following values for the actuated joint coordinates of the parallel modules:

$$\begin{aligned} &l_{1A} = 21.42,\ r_{1A} = 19.00,\ l_{2A} = 21.52,\ r_{2A} = 19.10 \\ &l_{1B} = 21.00,\ r_{1B} = 22.59,\ l_{2B} = 24.99,\ r_{2B} = 19.01 \end{aligned} \tag{5.81}$$

where the values are in cm and rounded to two decimal places. Note that some lengths are close to the joint limits (19 and 25 cm), which agrees with the fact that P_1 is close to the boundary of R_f. This solution yields the posture shown in Figure 5.11b.

5.4.6.2 Example 2

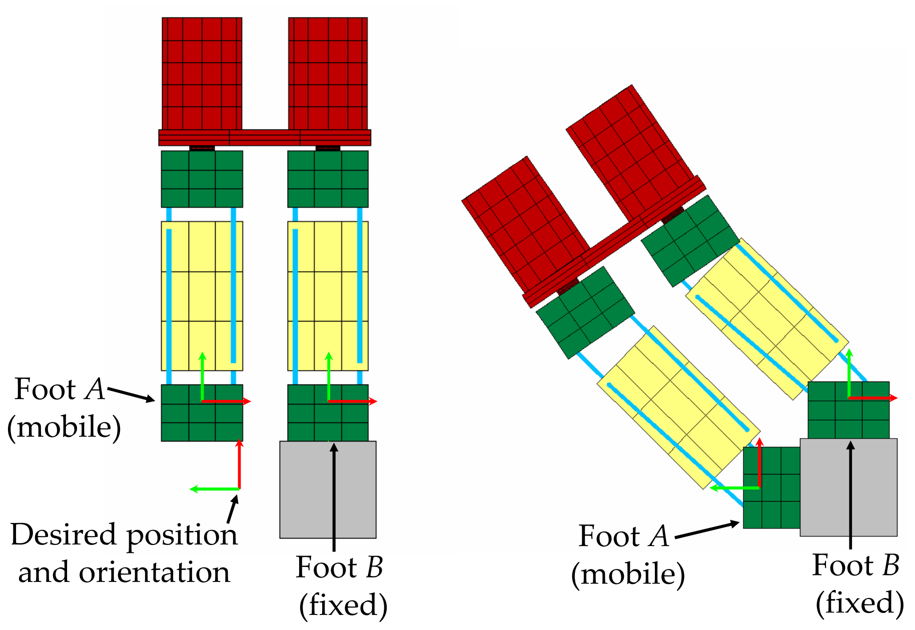
In this case, foot B is fixed and foot A must reach the following position and orientation to perform a convex transition between two adjacent faces of the same beam (see Figure 5.13):

$$\mathbf{T}_{A/B} = \begin{bmatrix} 0 & -1 & 0 & -11 \\ 1 & 0 & 0 & -11 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbf{T}_{B/A} = \begin{bmatrix} 0 & 1 & 0 & 11 \\ -1 & 0 & 0 & -11 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.82}$$

where the position is given in cm. First, matrix $\mathbf{T}_{A/B}$ must be inverted to obtain $\mathbf{T}_{B/A}$, which is the input to the inverse kinematic problem as formulated in Section 5.4.3.

According to Section 5.4.3.2, since $R_{31} = R_{32} = 0$ in this example, only the binary variable σ_2 is involved in the calculation of the intermediate joint coordinates. Thus, Algorithm 1 must be executed twice to obtain all the feasible solutions, once for each value of $\sigma_2 \in \{-1, 1\}$.

In this example, the algorithm was tested for $N_p = 10^4$ and $N_p = 5 \cdot 10^4$ random points, obtaining the following respective average execution times for 20 experiments in each case: 0.919 s (SD $= 0.003$ s) and 4.47 s (SD $= 0.20$ s). These are the times necessary for running the algorithm twice: for $\sigma_2 = -1$ and $\sigma_2 = 1$. In this example, only $\sigma_2 = 1$ yields a non-empty set of feasible solutions, which is shown in Figure 5.14 for $N_p = 5 \cdot 10^4$ points. The projections of the 3D solution set on the coordinate planes are also shown in Figure 5.14 to facilitate the 3D visualization of the shape of R_f.



(a) (b)

Figure 5.13: Desired position and orientation in Example 2, where the robot performs a convex transition between two planes.

For example, choosing point $P_3 = (-0.8, -0.2, 26)$ shown in Figure 5.14, yields the following ranges for y_{1A} and y_{1B}: $[21.78, 22.16]$ cm and $[21.84, 22.10]$ cm, respectively. Choosing $y_{1A} = y_{1B} = 22$ cm gives the following lengths:

$$\begin{aligned} l_{1A} &= 19.22, \ r_{1A} = 24.84, \ l_{2A} = 23.63, \ r_{2A} = 22.11 \\ l_{1B} &= 24.90, \ r_{1B} = 19.17, \ l_{2B} = 20.80, \ r_{2B} = 19.21 \end{aligned} \tag{5.83}$$

where the values are in cm. For this solution, the robot adopts the posture shown in Figure 5.13b. For this example, the rotations of the hip are: $\theta_A = \theta_B = 0$. Note that

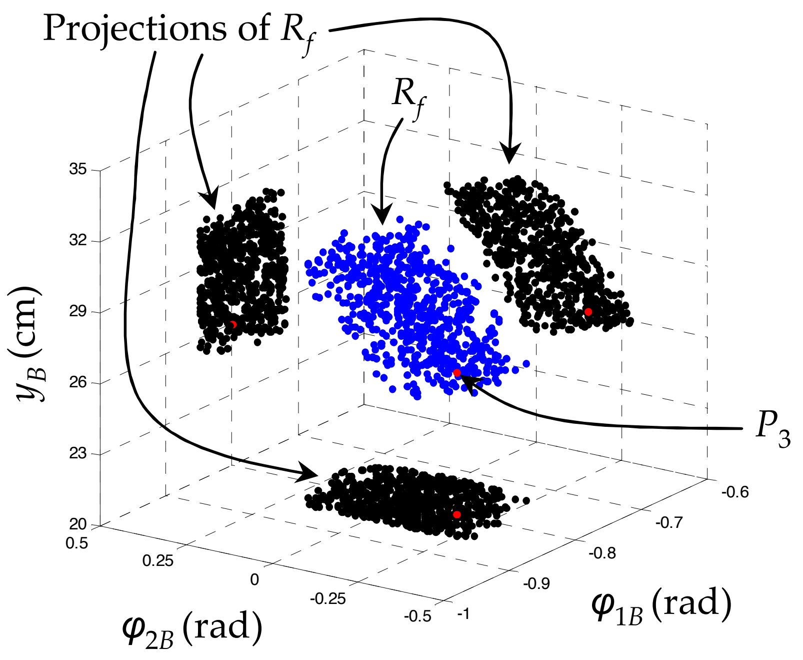


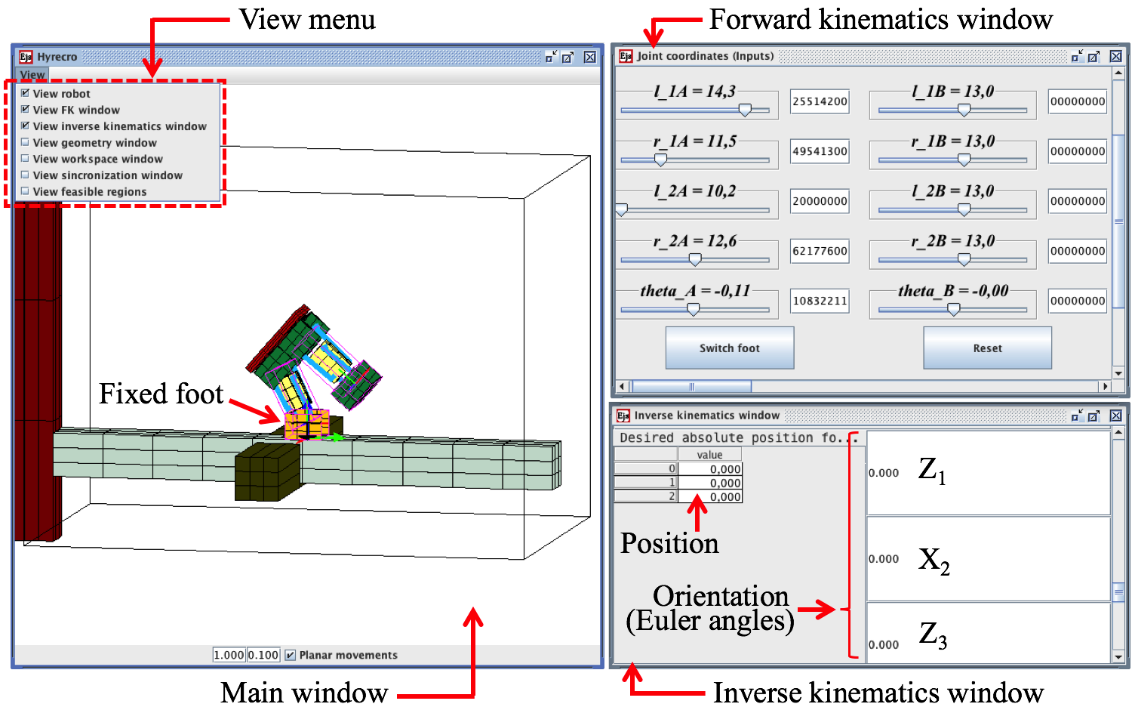
Figure 5.14: Set R_f of feasible solutions to the inverse kinematics in Example 2, for $N_p = 5 \cdot 10^4$ points.

the rotations of the hip do not depend on the decision variables when $R_{31} = R_{32} = 0$, as it can be observed in Equations (5.59) to (5.62).

5.5 A Simulator of the HyReCRo Robot

In order to facilitate the kinematic analysis of the HyReCRo robot, a graphical simulator has been developed using Easy Java Simulations. This simulator is part of the PaRoLa virtual laboratory presented in the previous chapter, and is accessible at `http://arvc.umh.es/parola`. In this chapter, we will demonstrate how the developed simulator can be used for simulating the forward and inverse kinematic problems of the HyReCRo robot, whereas in the next chapters we will demonstrate how to use this simulator for analyzing its workspace.

The developed simulator is shown in Figure 5.15. This simulator has a main window, which includes a schematic three-dimensional representation of the HyReCRo robot on a three-dimensional structure. The upper part of this window has a "view" menu, by means of which the user can activate or deactivate several other windows which can be used for performing some kinematic analyses of this robot. In the next subsections, we will focus on the use of this simulator for simulating the forward and inverse kinematics.


View menu
Forward kinematics window
Hyrecro
View
View robot
View FK window
View inverse kinematics window
View geometry window
View workspace window
View sincronization window
View feasible regions
Joint coordinates (Inputs)
Switch foot
Reset
Fixed foot
Inverse kinematics window
Desired absolute position fo...
value
Position
Orientation (Euler angles)
Main window
Planar movements

l_1A = 14,3 | 25514200
l_1B = 13,0 | 00000000
r_1A = 11,5 | 49541300
r_1B = 13,0 | 00000000
l_2A = 10,2 | 20000000
l_2B = 13,0 | 00000000
r_2A = 12,6 | 62177600
r_2B = 13,0 | 00000000
theta_A = -0,11 | 10832211
theta_B = -0,00 | 00000000

	value
0	0,000
1	0,000
2	0,000

0.000 Z_1
0.000 X_2
0.000 Z_3

1.000 0.100 ☑ Planar movements

Figure 5.15: Simulator developed for studying the forward and inverse kinematic problems of the HyReCRo robot.

5.5.1 Simulating the Forward Kinematics

As Figure 5.15 shows, the simulator of the HyReCRo robot has a window for simulating its forward kinematics. This window has sliders and numeric boxes for modifying the values of the ten actuated joint coordinates of the robot: lengths l_{ij} and r_{ij} of the parallel modules, and rotations θ_j of the hip ($i \in \{1,2\}$, $j \in \{A,B\}$). When modifying any of these actuated joint coordinates, the simulator solves the forward kinematics as described in Section 5.2, and the robot moves as a consequence.

When varying the lengths of the linear actuators of the parallel modules (variables l_{ij} and r_{ij}), the simulator solves the forward kinematics of these modules as explained in Section 5.2.1. As explained in Section 3.3, these parallel modules have four real solutions to their forward kinematic problem, but only one of them is feasible for the HyReCRo robot (the other three solutions imply some type of collision): the only valid solution is the one with highest value for y_{ij}. Accordingly, when the simulator solves the forward kinematics of the parallel modules (this problem is solved via elimination) and obtains the four real solutions, only the solution with highest y_{ij} is used. In this way, the simulator shows only one solution for the forward kinematics of the complete HyReCRo robot, which is the solution shown in the main window of the simulator.

At all times, one of the feet of the robot is attached to the structure: the fixed foot is indicated in orange color (see Figure 5.15). When simulating the forward kinematics, the robot moves considering that this foot is fixed. The user can switch the fixed foot by clicking on the button “Switch foot” in the window for simulating the forward kinematics (top-right window of Figure 5.15). When pressing this button, the fixed and free feet swap their roles. In this way, by simulating the forward kinematics and switching the adhesion of the feet to the structure, it is possible to simulate the motion of the robot along the structure, as explained in [133].

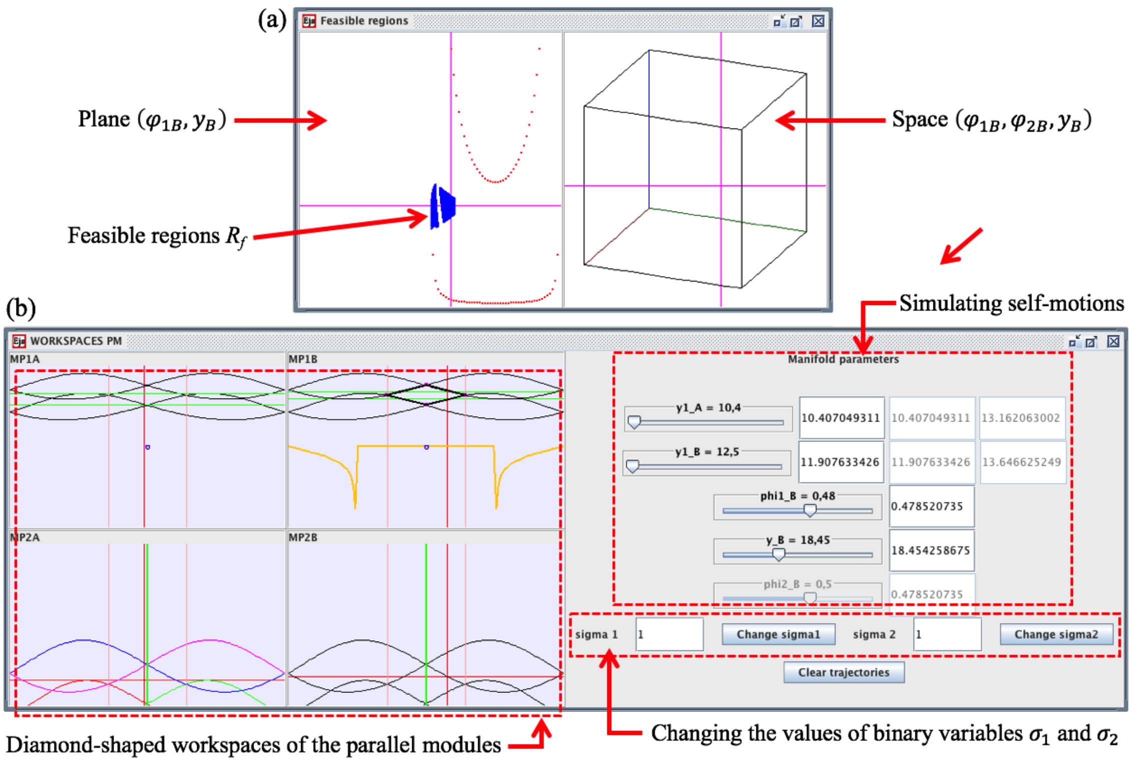
It should be mentioned that, in the current version of the simulator, it is possible for the HyReCRo robot to “walk on the air”, i.e.: the simulator does not implement the restriction that the feet should be in contact with the structure in order to adhere them to it. In the future, this will be corrected and the dynamics will also be included (effects of gravity), in order to provide more realistic simulations.

5.5.2 Simulating the Inverse Kinematics

Using the developed simulator, it is also possible to simulate the inverse kinematics. To that end, one must specify the desired position and orientation (pose) for the free foot of the robot. The desired pose must be specified with respect to a world reference frame which is attached to the structure, instead of referring the desired pose with respect to the fixed foot. This is because the pose of the fixed foot will vary as the robot moves along the structure, so it is simpler to always specify the desired pose using absolute coordinates. When specifying the desired absolute pose for the free foot, the simulator automatically computes the matrix $\mathbf{T}_{B/A}$ that encodes the desired relative pose between the feet, which is the input of the inverse kinematic problem as formulated in Section 5.4.3. The desired absolute pose for the free foot is specified in the inverse kinematics window of the simulator (bottom-right window in Figure 5.15). The orientation must be indicated as a triplet of $Z_1X_2Z_3$ Euler angles.

After introducing the desired absolute pose for the free foot, the simulator solves the inverse kinematics as explained in Section 5.4.3. As a result, the simulator displays the feasible regions R_f defined in Section 5.4.5. The displayed regions will be planar or three-dimensional volumes, depending whether the desired relative orientation between the feet satisfies $R_{33}^2 \neq 1$ or not, respectively. The user can enable the visualization of these feasible regions using the “view” menu at the top of the main window, as indicated in Figure 5.15. When doing this, a window with two panels pops-up: the left panel shows plane (φ_{1B}, y_B), whereas the right panel shows the (φ_{1B}, φ_{2B}, y_B) space (see Figure 5.16a).

When simulating the inverse kinematics, Algorithm 1 is executed for computing these feasible regions, which are displayed in only one of these panels, depending on the considered case of the inverse kinematics (Case 1 or 2). Moreover, recall from Section 5.4.3 that two binary variables ($\sigma_1, \sigma_2 \in \{-1, 1\}$) intervene in the computation of these feasible regions, such that the shape of these regions will vary for different values of these binary variables. The user can change the values of these binary variables in the panel shown in Figure 5.16b, which also displays the workspaces of the four parallel modules of the HyReCRo robot, so that the user can confirm that these workspaces have the diamond-like shapes assumed throughout Section 5.4 (i.e., that they have the form of Equation 5.39, as required by Algorithm 1). When varying the value of these


(a)
Feasible regions
Plane (φ_{1B}, y_B)
Space $(\varphi_{1B}, \varphi_{2B}, y_B)$
Feasible regions R_f
(b)
Simulating self-motions
WORKSPACES PM
MP1A
MP1B
MP2A
MP2B
Manifold parameters
Change sigma1
Change sigma2
Clear trajectories
Diamond-shaped workspaces of the parallel modules
Changing the values of binary variables σ_1 and σ_2

y1_A = 10,4 10.407049311 10.407049311 13.162063002
y1_B = 12,5 11.907633426 11.907633426 13.646625249
phi1_B = 0,48 0.478520735
y_B = 18,45 18.454258675
phi2_B = 0,5 0.478520735
sigma 1 1
sigma 2 1

Figure 5.16: (a) Representation of the feasible regions R_f in the simulator. (b) Performing self-motions in the simulator.

binary variables, the feasible regions shown in the panels of Figure 5.16a will change (and in some cases, they will be empty).

Using the windows shown in Figure 5.16, it is also possible to simulate self-motions. For example: in the window shown in Figure 5.16b, the user can modify the values of parameters $\{\varphi_{1B}, y_B, y_{1A}, y_{1B}\}$ (and also φ_{2B}, in Case 2) by means of sliders and numeric boxes: as explained in Section 5.4.4, these are the parameters used in this thesis for parameterizing the self-motion manifolds of the HyReCRo robot. When varying these parameters, the robot performs self-motions: the posture of the robot changes, but the relative position and orientation between its feet always remains constant. It is also possible to simulate self-motions by dragging the coordinates of $\{\varphi_{1B}, y_B\}$ (and also φ_{2B}, in Case 2) in the panel shown in Figure 5.16a: in that case, the values of the remaining parameters $\{y_{1A}, y_{1B}\}$ are automatically chosen by the simulator, which chooses the minimum allowed values for these parameters (recall from Section 5.4.5.1 that, for both these parameters, we can always find intervals of allowed values).

The simulation of the inverse kinematics is very useful for determining if some poses will be reachable for a given geometric design of the robot. For example, we may be interested in knowing if a given design of the HyReCRo robot will be able to perform convex transitions between adjacent faces of a beam, as in Figure 5.13b. To that end, one only needs to introduce the pose corresponding to this convex transition, and solve the inverse kinematics in the simulator, obtaining the feasible regions. If these feasible regions are empty, then this means that the robot cannot perform such transitions and it is necessary to modify the geometric parameters of the robot until these regions are not empty, as in Figure 5.14. For example, according to the analysis presented in Section 5.3.2, it would be a good idea to vary parameter p of the parallel modules, since the shape of the workspace seems to be very sensitive to variations in this parameter.

In essence, this is what was done in this thesis for determining appropriate values for the geometric design parameters of the prototype presented in Chapter 8, so that this prototype can perform convex transitions. This analysis, based on the inverse kinematics, can be completed with a workspace analysis, which provides more global information. As we will see in next chapter, a workspace analysis not only allows us to determine if a concrete individual pose is attainable (which is the information provided by an inverse kinematic analysis), but also allows us to know if complete regions of the workspace of the robot will be reachable.

5.6 Conclusions

In this chapter, we have solved the forward and inverse kinematic problems of the HyReCRo robot. First, the forward problem has been solved (Section 5.2), departing from the solution of the forward kinematics of the 2R$\underline{\text{P}}$R-PR parallel modules that make up the legs of this robot (Section 5.2.1). A total of 256 solutions have been identified for the forward kinematic problem of the HyReCRo robot, but only one of them is valid in practice due to collision constraints.

After this, the inverse kinematic problem has been addressed, solving first a simplified version of this problem which considers only planar and symmetric postures (Section 5.3.1). This problem is easier to solve than the general inverse kinematics and can be useful for planning many of the movements necessary for performing plane transitions in structures. The PSIK workspace, i.e., the workspace composed of planar and symmetric postures, has also been investigated in order to determine the sensitivity of this workspace with respect to the geometric design parameters of the robot (Section 5.3.2). From this sensitivity analysis, it has been found that the shape and size of the PSIK workspace are most sensitive to the width p of the feet of the robot and the stroke $\Delta\rho$ of the linear actuators that drive the parallel modules.

After solving the simplified PSIK problem, the general inverse kinematic problem of the HyReCRo robot has been addressed (Section 5.4). In order to facilitate the resolution of this problem, some intermediate joint coordinates have been defined, which allow us to study the HyReCRo robot as an equivalent serial robot. When solving these intermediate joint coordinates from the loop-closure equations of this robot, two cases have been identified: in Case 1, in which the feet of the robot are not parallel, the self-motion manifolds of this kinematically-redundant robot are four-dimensional. In Case 2, which is a singular and frequent situation in which the feet of the robot are parallel, these manifolds are five-dimensional. In both cases, simple parameterizations of these manifolds have been obtained, such that all intermediate joint coordinates can be easily solved in terms four (or five, in Case 2) of them. Then, in order to graphically visualize such high-dimensional manifolds, they have been projected to two- and three-dimensional subspaces, obtaining the so-called feasible regions R_f that constitute a compact and simple graphical representation of the solutions of the inverse kinematics of the HyReCRo robot.

Finally, a graphical simulator of the HyReCRo robot has also been developed, which implements the solutions of the forward and inverse kinematic problems as solved in this chapter.

5.7 Publications Related to this Chapter

The main results presented in this chapter are related to the following publications:

- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematic analysis and simulation of a hybrid biped climbing robot. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 24–34, 2015 [132].
    - This paper presents the solution of the forward kinematics of the complete HyReCRo robot, as well as the resolution of the Planar Symmetric Inverse Kinematic problem. Also, this paper presents the first version of the simulator of the HyReCRo robot, presented in Section 5.5.
- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematics, simulation, and analysis of the planar and symmetric postures of a serial-parallel climbing robot. In Joaquim Filipe, Kurosh Madani, Oleg Gusikhin, and Jurek Sasiadek, editors, *Informatics in Control, Automation and Robotics 12th International Conference, ICINCO 2015 Colmar, France, July 21-23, 2015 Revised Selected Papers*, pages 115–135. Springer International Publishing, 2016 [133].
    - This paper is a revised and extended version of the previous one, which was selected among some of the best papers presented in the ICINCO 2015 international conference. This paper studies the sensitivity of the PSIK workspace of the HyReCRo robot with respect to the geometric design parameters of this robot (Section 5.3.2).
- A. Peidro, A. Gil, J.M. Marin, and O. Reinoso. Inverse kinematic analysis of a redundant hybrid climbing robot. *International Journal of Advanced Robotic Systems*, 12(11):163, 2015 [136] **(SCI-JCR Impact Factor: 0.615, Q4)**.
    - This paper presents the solution of the general inverse kinematic problem of the complete HyReCRo robot (Section 5.4).

6 Boundaries of the Workspace

This chapter analyzes the workspace of the HyReCRo robot. First, section 6.1 presents a literature review of methods for computing workspaces of robot manipulators, in order to choose the most suitable method for the HyReCRo robot. Considering the complexity of this robot (redundancy and serial-parallel architecture), Monte Carlo methods turn out to be the most appropriate ones for computing its workspace. Classical Monte Carlo methods are reviewed in section 6.2, where they are used for performing some preliminary computations and analyses of the workspace of the HyReCRo robot. Then, section 6.3 analyzes the accuracy problems of classical Monte Carlo methods, demonstrating that they generally yield inaccurate boundaries of the workspace (even when sampling very large amounts of points). Then, section 6.4 proposes a new improved Monte Carlo method, which makes use of normal distributions for efficiently "growing" and generating workspaces. Through several examples based on the HyReCRo robot, section 6.5 compares the proposed method with classical Monte Carlo methods, demonstrating that the proposed method can obtain workspace boundaries much more accurately than classical methods, requiring the same or less computation time, which demonstrates its higher efficiency.

6.1 Review of Methods for Computing Workspaces

The workspace of a robot manipulator can be defined as the set of poses (positions and/or orientations) that its end-effector can reach, subject to the loop-closure constraints imposed by the architecture of the robot (as well as to other additional kinematic constraints, such as joint limits or avoidance of collisions). Knowing the workspace is very important for designing robot manipulators, as well as for planning their motion. Thus, calculating the workspace has been the objective of much research work during the last decades, which has resulted in the development of many methods for computing robot workspaces. Most of the existing methods can be classified into one of three main classes [116]: geometrical methods, singularity-based methods, and sampling methods.

Geometrical methods are exact and very fast, but their scope is limited since they are tailored to specific classes of robots. These methods usually rely on Computer Aided Design tools and can be used to compute constant-orientation workspaces of parallel robots as the intersection of the workspaces of all limbs of the robot [9], which are simple geometric shapes such as solid tori [96], annuli [118, 18], or spherical shells [64]. Geometric constructions have also been used to obtain analytic descriptions of the boundaries of reachable and other workspaces of parallel robots [4, 118], as well as maximal singularity-free areas [79]. Geometrical methods can handle kinematic constraints such as joint limits and even self-collisions, but only in relatively simple cases [115].

Singularity-based methods directly obtain the boundaries of the workspace, both for redundant and non-redundant robots. At these boundaries, $\mathbf{\Phi_z}$ becomes rank-deficient, where $\mathbf{\Phi_z}$ is a Jacobian matrix of derivatives of all kinematic constraints with respect to all variables involved in the problem (excluding the variables that parameterize the pose of the end-effector). The condition of $\mathbf{\Phi_z}$ being rank-deficient yields a system $\mathcal{S}$ of equations whose solution set contains the boundaries of the workspace. System $\mathcal{S}$ may be analytically solved in some cases [1, 2], but in general it must be solved numerically [71, 19]. In [71], $\mathcal{S}$ is solved via continuation, obtaining planar slices of workspace boundaries. Bohigas et al. [19] identified problematic situations in which continuation methods fail, such as degenerate boundaries and multicomponent workspaces. Alternatively, Bohigas et al. [19] manage to rewrite $\mathcal{S}$ as a quadratic system and solve it using a linear relaxation method which is robust to the aforementioned problematic situations. An important limitation of singularity-based methods is that all kinematic constraints must be written as equalities, which is not always possible. For example, joint limits (which are usually modeled as inequalities) can be easily rewritten as equalities [71], even as quadratic equations [19], by introducing auxiliary variables. But this is not easy in general for more complex constraints, like the prohibition of mechanical interferences (collisions).

Sampling methods generate many configurations of the robot, and check if each configuration belongs to the workspace satisfying all kinematic constraints. Thus, these methods are very flexible and can easily handle complex kinematic constraints, because one simply needs to check if all constraints are satisfied for each concrete configuration *after generating it* [10]. Configurations are typically generated by sampling points from the joint or task[1] spaces, following regular or random patterns (Monte Carlo methods [68, 10]). For serial robots, configurations are usually generated by sampling points from the joint space and solving afterwards the Forward Kinematics (FK-based methods [156, 24]), which is simpler than the Inverse Kinematics (IK) for these robots.

[1]The task space (or Cartesian space) is the space of poses of the end-effector.

Conversely, for parallel robots the IK is simpler than the FK. Thus, their configurations are often generated by sampling points from the task space and solving the IK for each point (IK-based methods [106, 45, 174]). When using IK-based methods with redundant robots, since their inverse kinematic problem admits infinitely many solutions for a given task point, it is sufficient to find only one solution satisfying all constraints to guarantee that the considered task point belongs to the workspace [147].

Considering the limitations and characteristics of the previously exposed methods, it is evident that the most suitable methods for computing the workspace of the HyReCRo robot are sampling methods. More specifically, we will use a FK-based Monte Carlo method, in which the actuated joint coordinates will be randomly sampled and the forward kinematic problem of the HyReCRo robot will be solved in order to obtain workspace points. This decision is based on the following reasons:

- The HyReCRo robot is too complex to use a geometrical method for computing its workspace. Geometrical methods are appropriate for far simpler robots than the HyReCRo robot.
- Although singularity-based methods are general and, in principle, should be able to obtain the boundaries of the workspace of the HyReCRo robot, this robot involves so many variables and kinematic constraints that it would take too long to compute its workspace using a method of this type.
- Moreover, these methods (geometrical and singularity-based) cannot easily handle no-collision constraints (i.e., the constraint that there should not exist mechanical interferences between different parts of the robot, or between the robot and obstacles of the environment in which it moves). On the contrary, sampling methods are very flexible and can easily accommodate the restriction that mechanical interferences should be prohibited, since one only needs to sample a configuration, check if it produces mechanical interferences, and discard it in that case.

For these reasons, we consider that it is more appropriate to use a sampling method to calculate the workspace of the HyReCRo robot. Of all sampling methods, it seems evident that, for the HyReCRo robot, a method based on the forward kinematics is more convenient than a method based on the inverse kinematics, due to the kinematic redundancy of this robot (as we have demonstrated in the previous chapter, the forward kinematic problem of the HyReCRo robot is far simpler than its inverse kinematic problem).

Finally, of all FK-based sampling methods for computing the workspace, we believe that it is more convenient to sample joint coordinates by following random patterns (i.e., Monte Carlo methods) instead of regular patterns (i.e., a regular grid in the joint space). This is because random patterns (distributions) are more general than regular patterns, which can be approximated by uniform random distributions. In fact, as we will demonstrate later in this chapter, sampling joint coordinates from uniform patterns is computationally inefficient, in the sense that increasing the number of samples does not translate into a significant improvement of the precision of the computed workspace.

Having said that, in the next section we will compute the workspace of the HyReCRo robot using a simple sampling FK-based Monte Carlo method. Later, we will improve this initial method in order to greatly and efficiently increase its accuracy, without increasing its computation time.

6.2 Computing the Workspace of the HyReCRo Robot via Classical Monte Carlo Methods

In this section, we will review the "classical" or "standard" Monte Carlo method for obtaining the workspace of robot manipulators, and then we will use this method to compute some workspaces of the HyReCRo robot.

Let $\mathbf{q} = [q_1, \ldots, q_d]^T$ denote the vector of joint coordinates of a robot with d degrees of freedom (DOF). Vector $\mathbf{q}$ may contain actuated joint coordinates or "intermediate" (not necessarily actuated) joint coordinates like those defined in section 5.4.2 for the HyReCRo robot. The Monte Carlo method consists in generating a large number of random vectors $\mathbf{q}$ and, for each of them, solving the forward kinematic problem to obtain the position $\mathbf{X} \in \mathbb{R}^3$ of the end-effector of the robot (in the case of the HyReCRo robot, the end-effector is its free foot). The components of each random vector $\mathbf{q}$ are randomly generated as follows:

$$q_k = q_k^{min} + (q_k^{max} - q_k^{min}) r_k, \quad k = 1, \ldots, d \tag{6.1}$$

where $\left\{q_k^{min}, q_k^{max}\right\}$ are the joint limits of joint coordinate q_k and r_k is a random variable in $(0, 1)$. After generating each random position of the robot, one should check if it satisfies other additional constraints that may exist (for example, different parts of the robot should not interfere, or the end-effector should have a desired orientation). If all constraints are satisfied, the generated point $\mathbf{X}$ is stored as a workspace point, and the set of all stored points constitutes a discrete approximation of the workspace of the manipulator.

The workspace generated in this way is a point cloud in $\mathbb{R}^3$ that can be represented graphically. However, for the practical use of the workspace (e.g. for path planning), it is necessary to build a database of the workspace using the generated random points [68]. To build this database, the Cartesian space is discretized with desired resolution, obtaining a set of cells in this space. Then, all cells which contain at least one workspace point are classified as "reachable", whereas the remaining cells are considered "unreachable" (see Figure 6.1a). The boundaries of the workspace can be approximated by the set of reachable cells which have at least one neighboring unreachable cell. In this chapter, and in the next chapter, the neighbors that will be considered in 3D are the 26-neighbors, whereas in 2D the 8-neighbors will be considered (see Figure 6.1b).

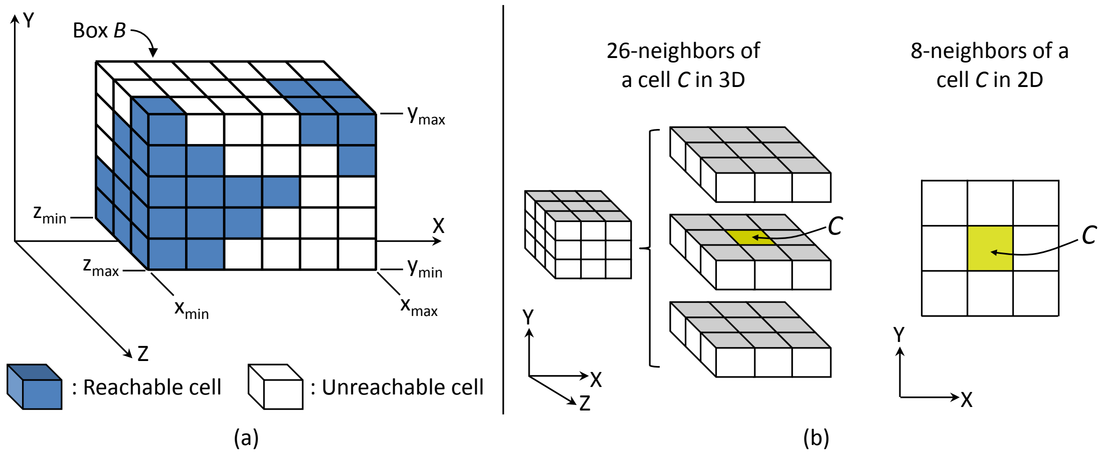


Figure 6.1: (a) A box $\mathcal{B}$ is defined and discretized into n_x, n_y and n_z cells along the X, Y, and Z axes, respectively (in this figure: $n_x = 6$, $n_y = 5$ and $n_z = 3$). (b) Illustration of the neighbors of a cell C considered in this chapter, in three and two dimensions.

The Monte Carlo method is a simple and widely used method especially suitable for computing the workspace of complex robots subject to complicated constraints, which have many degrees of freedom or are even kinematically redundant, like humanoid robots [68, 10, 24, 156, 6, 23, 191]. Next, we will use this method to compute the reachable and constant-orientation workspaces of the HyReCRo robot. For now, and for the sake of simplicity, we will only consider joint limits $\{q_k^{min}, q_k^{max}\}$. Collision constraints will be taken into account later in this chapter.

6.2.1 Reachable Workspace of the HyReCRo Robot

To generate the reachable workspace of the HyReCRo robot, we need the solution of its forward kinematic problem, which consists in computing the position and orientation of one foot of the robot with respect to the other, in terms of the actuated or intermediate joint coordinates. This problem was solved in section 5.2.2, and its solution is repeated next. From Equation (5.18), we obtain that the position $[p_x, p_y, p_z]$ of foot B relative to foot A has the following expression:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = y_A \begin{bmatrix} s_{\varphi_{1A}} \\ c_{\varphi_{1A}} \\ 0 \end{bmatrix} + y_B \begin{bmatrix} -c_\Theta c_{\Phi_A} s_{\varphi_{2B}} - s_{\Phi_A} c_{\varphi_{2B}} \\ c_\Theta s_{\Phi_A} s_{\varphi_{2B}} - c_{\Phi_A} c_{\varphi_{2B}} \\ s_\Theta s_{\varphi_{2B}} \end{bmatrix} + t \begin{bmatrix} c_{\theta_A} c_{\Phi_A} \\ -c_{\theta_A} s_{\Phi_A} \\ -s_{\theta_A} \end{bmatrix} \quad (6.2)$$

where $\Phi_j = \varphi_{1j} - \varphi_{2j}$. Similarly, from Equation (5.18) we also obtain the rotation matrix encoding the orientation of foot B relative to foot A:

$$\mathbf{R}_{B/A} = \begin{bmatrix} s_{\Phi_A} s_{\Phi_B} + c_\Theta c_{\Phi_A} c_{\Phi_B} & s_{\Phi_A} c_{\Phi_B} - c_\Theta c_{\Phi_A} s_{\Phi_B} & s_\Theta c_{\Phi_A} \\ c_{\Phi_A} s_{\Phi_B} - c_\Theta s_{\Phi_A} c_{\Phi_B} & c_{\Phi_A} c_{\Phi_B} + c_\Theta s_{\Phi_A} s_{\Phi_B} & -s_\Theta s_{\Phi_A} \\ -s_\Theta c_{\Phi_B} & s_\Theta s_{\Phi_B} & c_\Theta \end{bmatrix} \quad (6.3)$$

where $\Theta = \theta_A - \theta_B$. According to the previous equations, the position and orientation of foot B relative to foot A depend on the following eight intermediate joint coordinates, which were introduced in section 5.4.2: $\{\varphi_{1A}, \varphi_{2A}, \varphi_{1B}, \varphi_{2B}, y_A, y_B, \theta_A, \theta_B\}$. Thus, if we want to compute the reachable workspace of the HyReCRo robot, we only need to randomly sample all these eight variables and generate three-dimensional points $[p_x, p_y, p_z]$ using Equation (6.2). These three-dimensional points form a 3D point cloud in space that constitutes a discrete approximation of the workspace, which is a volume. For visualizing the obtained workspace, it is sufficient to visualize the boundary surfaces that delimit this volume workspace. To extract these boundary surfaces, we proceed as follows [6]: first, we define a 3D grid composed of n_x, n_y, and n_z cells along the X, Y, and Z axes, respectively (see Figure 6.1a). The cells that contain workspace points (i.e., reachable cells) are marked with "1", whereas the remaining cells are marked with "0". Then, the workspace boundary is composed of cells that are marked with "1" and have at least one neighboring cell marked with "0".

Before applying the Monte Carlo method, it is necessary to define the joint limits $\{q_k^{min}, q_k^{max}\}$ used in Equation (6.1) for each intermediate joint coordinate. Also, it is necessary to specify the random distribution from which r_k in Equation (6.1) will be sampled, for each intermediate joint coordinate.

6.2.1.1 Joint limits and random distribution for angles θ_j

We will assume that the legs of the HyReCRo robot can perform complete revolutions about the revolute axes of the hip. This means that the joint limits for angles θ_A and θ_B will be $q_k^{min} = -\pi$ rad and $q_k^{max} = \pi$ rad. Moreover, these angles will be uniformly sampled, i.e., r_k in Equation (6.1) will be a uniform number in $(0, 1)$. This choice of a uniform distribution for these angles will be better justified later in this chapter, but this is the main idea: if angle θ_j has no joint limits, then it can take any value from the interval $[-\pi, \pi]$ rad (one complete revolution). However, $-\pi$ and π are not true joint limits, since they correspond to the same angle. In other words, the limits of the interval $[-\pi, \pi]$ are "glued", so that θ_j actually takes values from the unit circle S^1 which, unlike an interval, has no limits (it is a closed curve).

6.2.1.2 Joint limits and random distribution for φ_{ij} and y_j

As in the previous chapter, we will assume that, due to joint limits, variables φ_{ij} and y_{ij} of the 2R$\underline{\text{P}}$R-PR parallel modules of the HyReCRo robot belong to diamond-shaped regions WS_{pm} like those defined in section 5.4.1. These regions were defined as follows:

$$WS_{pm} = \left\{(\varphi_{ij}, y_{ij}) : \varphi_{min} \leq \varphi_{ij} \leq \varphi_{max}, \underline{y_{ij}}(\varphi_{ij}) \leq y_{ij} \leq \overline{y_{ij}}(\varphi_{ij})\right\} \tag{6.4}$$

According to Equation (6.4), the joint limits for angles φ_{ij} will be: $q_k^{min} = \varphi_{min}$ and $q_k^{max} = \varphi_{max}$. To determine the joint limits of y_j, recall from the previous chapter (section 5.4.2) that for each leg j we have $y_j = y_{1j} + y_{2j} - h$. If $\varphi_{ij} \in [\varphi_{min}, \varphi_{max}]$ ($i \in \{1, 2\}$), then according to Equation (6.4), y_{ij} must satisfy:

$$\underline{y_{ij}}(\varphi_{ij}) \leq y_{ij} \leq \overline{y_{ij}}(\varphi_{ij}) \tag{6.5}$$

Since $y_j = y_{1j} + y_{2j} - h$, then variables y_j must verify:

$$\underline{y_{1j}}(\varphi_{1j}) + \underline{y_{2j}}(\varphi_{2j}) - h \leq y_j \leq \overline{y_{1j}}(\varphi_{1j}) + \overline{y_{2j}}(\varphi_{2j}) - h \tag{6.6}$$

which provides the joint limits for these intermediate joint coordinates.

Finally, after all joint limits necessary for the Monte Carlo computation of the workspace have been determined, it is necessary to specify the random distribution from which r_k will be sampled for variables y_j and φ_{ij}. These variables will be sampled from U-shaped beta distributions [24]. This is because the joint limits of these variables define the boundaries of the workspace, and sampling them from U-shaped distributions favors the generation of random points close to the boundaries of the workspace, which results in a better definition of these boundaries. Later in this chapter, it will become evident that sampling bounded joint coordinates (i.e., joint coordinates with *true* joint limits) from U-shaped beta distributions generally yields better results than sampling from uniform distributions.

6.2.1.3 Example: sensitivity analysis

After all joint limits and random distributions have been defined, the Monte Carlo calculation of the reachable workspace can be summarized in Algorithm 2. In this algorithm, N_r is the number of randomly sampled points.

Algorithm 2 Monte-Carlo calculation of the reachable workspace

```
1: $WS = \emptyset \rightarrow$ The reachable workspace is initialized as an empty set.
2: for $k = 1$ to $N_r$ do
3:     Sample $\theta_A$ and $\theta_B$ uniformly from $[-\pi, \pi]$
4:     Sample $\varphi_{1A}$, $\varphi_{2A}$, $\varphi_{1B}$, and $\varphi_{2B}$ from $[\varphi_{min}, \varphi_{max}]$ (U-shaped $\beta$ distr.)
5:     Compute the lower and upper limits for $y_j$ $(j \in \{A, B\})$:
6:         $\underline{y_j} = \underline{y_{1j}}(\varphi_{1j}) + \underline{y_{2j}}(\varphi_{2j}) - h$
7:         $\overline{y_j} = \overline{y_{1j}}(\varphi_{1j}) + \overline{y_{2j}}(\varphi_{2j}) - h$
8:     Randomly sample $y_j$ in $[\underline{y_j}, \overline{y_j}]$ $(j \in \{A, B\})$ (U-shaped $\beta$ distr.)
9:     Compute the position $\mathbf{P} = [p_x, p_y, p_z]^T$ of the free foot using Equation (6.2)
10:    Add point $\mathbf{P}$ to $WS$
```

As explained previously, after sampling all workspace points and obtaining the 3D point cloud, we can extract its boundaries as the set of cells of the Cartesian space that contain workspace points and have neighboring cells that do not contain workspace points.

Next, we will use Algorithm 2 to perform a sensitivity analysis similar to the one performed in section 5.3.2 of the previous chapter, i.e., we will vary all six geometric design parameters of the robot and we will study how the shape and size of the reachable workspace varies when changing the design of the robot. In all examples, we will generate $N_r = 2 \cdot 10^6$ random workspace points and we will extract the boundaries by discretizing the Cartesian space into $n_x = n_y = n_z = 50$ cells along each axis.

Consider a HyReCRo robot with the following default geometry: $\rho_0 = 19.5$, $\Delta\rho = 5$, $b = p = 4$, $t = 15.6$, $h = 16$ (all values in cm). The boundary of the reachable workspace for this geometry is shown in Figure 6.2 (center), which shows that the points above the fixed foot A cannot be reached by foot B. Next, we will vary the design parameters (one at a time, keeping the rest at their default values) to obtain a larger workspace in which the region above foot A is accessible.

Figure 6.2 shows that increasing h reduces the size of the reachable workspace, leaving its shape practically unaffected. If parameters t and ρ_0 are respectively varied in the intervals $[10, 20]$ cm and $[15, 25]$ cm, it can be checked that the size of the workspace increases with these parameters, but its shape hardly varies with them. Also, it can be checked that varying b in $(0, 10]$ cm hardly affects the shape or size of the reachable workspace. Thus, varying these four parameters generates workspaces where the points above foot A are still inaccessible. However, varying parameter p modifies noticeably the shape of the workspace, as shown in Figure 6.3. This figure shows that the reachable workspace opens as p increases. Thus, it is convenient to reduce p as shown in Figure 6.3(left) in order to eliminate the inaccessible region above foot A. It can be checked that varying $\Delta\rho$ in $[3, 6]$ cm produces a similar effect in the opposite direction: the workspace opens as $\Delta\rho$ decreases.

The previous analysis coincides with the results observed in the sensitivity analysis of the PSIK workspace analyzed in section 5.3.2 of the previous chapter, where we demonstrated that the shape of the PSIK workspace is most sensitive to parameters $\Delta\rho$ and p. Thus, these two parameters seem to be the most critical ones for designing the HyReCRo robot, since varying them can produce important changes in the shape of the workspace. Varying b alters little both the shape and size of the workspace. Finally, varying $\{h, t, \rho_0\}$ affects the size of the worksapce, leaving its shape almost unchanged.

Of the two critical design parameters $\Delta\rho$ and p, $\Delta\rho$ is the stroke of the linear actuators, and is therefore fixed and determined by the manufacturer of each actuator. This means that, in practice, we are only free to vary p if we want to produce important changes on the shape of the workspace (assuming that $\{b, t, h, \rho_0\}$ have little effect on this shape).

As we will see in the next example, for a given geometric design of the HyReCRo robot, the region of its workspace that allows it to perform concave transitions is generally larger than the region that allows it to perform convex transitions. For this reason, the ability to perform convex transitions is more critical when designing the HyReCRo robot, and p will need to be carefully chosen so that the robot has this ability.

6.2.2 Constant-orientation Workspace of the HyReCRo Robot

The constant-orientation workspace is the set of points that can be reached with a desired relative orientation between feet A and B. As in the previous chapter, the

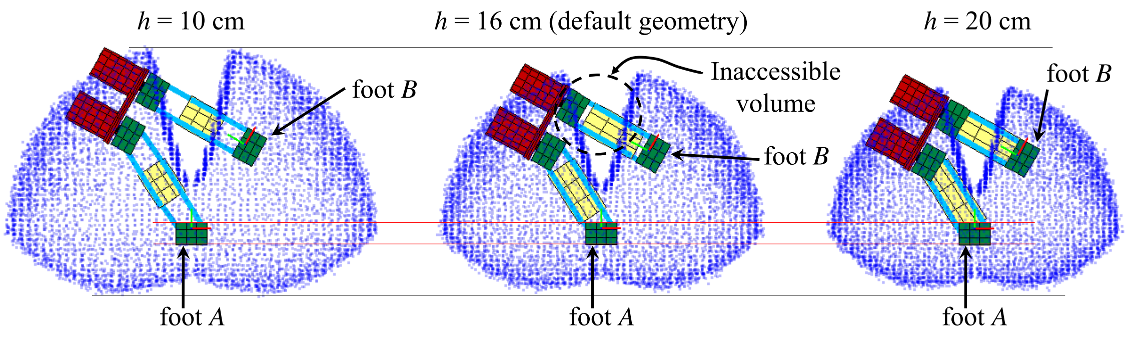


Figure 6.2: Variation of the reachable workspace when h is modified.

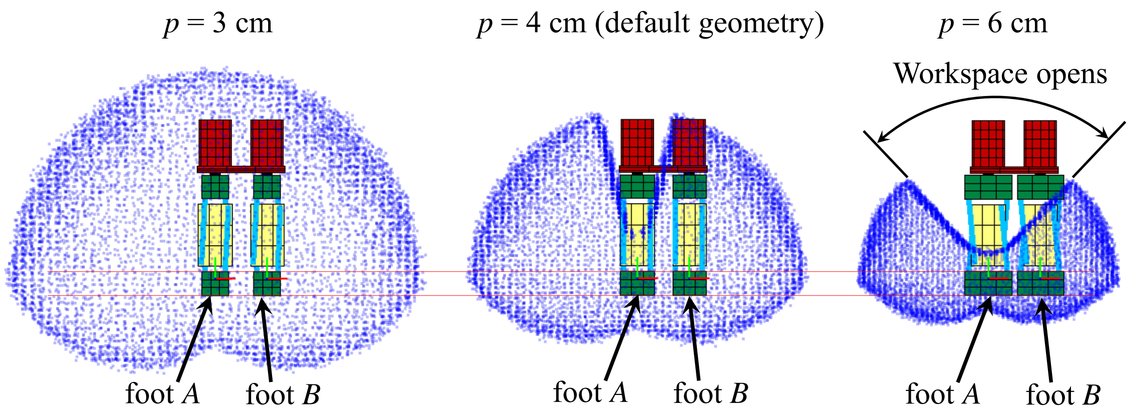


Figure 6.3: Variation of the reachable workspace when p is modified.

desired orientation can be specified as a known rotation matrix:

$$\mathbf{R}_{B/A} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \tag{6.7}$$

where R_{ij} are known quantities. Algorithm 2 can still be used to generate random points in the constant-orientation workspace. However, unlike in Algorithm 2, not all angles $\{\varphi_{ij}, \theta_j\}$ can be sampled independently now: these angles must satisfy certain relations to guarantee that the generated random points have the desired orientation. As in section 5.4.3 (solution of the inverse kinematics of the HyReCRo robot), we must distinguish two cases:

6.2.2.1 Case 1: $R_{33}^2 \neq 1$.

Equating element (3,3) of matrices (6.3) and (6.7) permits computing angle Θ as follows:

$$c_\Theta = R_{33} \longrightarrow s_\Theta = \sigma\sqrt{1 - R_{33}^2} \longrightarrow \Theta = \theta_A - \theta_B = \mathrm{atan2}(s_\Theta, c_\Theta) \tag{6.8}$$

where $\sigma \in \{-1, 1\}$. Once s_Θ is known, Equating elements (1,3), (2,3), (3,1) and (3,2) of Equations (6.3) and (6.7) allows for the calculation of Φ_A and Φ_B:

$$c_{\Phi_A} = R_{13}/s_\Theta, \quad s_{\Phi_A} = -R_{23}/s_\Theta \longrightarrow \Phi_A = \varphi_{1A} - \varphi_{2A} = \text{atan2}(s_{\Phi_A}, c_{\Phi_A}) \quad (6.9)$$

$$c_{\Phi_B} = -R_{31}/s_\Theta, \quad s_{\Phi_B} = R_{32}/s_\Theta \longrightarrow \Phi_B = \varphi_{1B} - \varphi_{2B} = \text{atan2}(s_{\Phi_B}, c_{\Phi_B}) \quad (6.10)$$

Note that Equations (6.8), (6.9), and (6.10) fix the differences $\theta_A - \theta_B$, $\varphi_{1A} - \varphi_{2A}$, and $\varphi_{1B} - \varphi_{2B}$, respectively. Thus, we cannot give random values to the six angles $\{\varphi_{1A}, \varphi_{2A}, \varphi_{1B}, \varphi_{2B}, \theta_A, \theta_B\}$ simultaneously. Instead, we can give values to angles $\{\theta_B, \varphi_{2A}, \varphi_{2B}\}$ and compute the other three angles using the previous equations to guarantee that the generated points have the desired orientation. Note that, after calculating $\{\varphi_{1A}, \varphi_{1B}\}$, these angles may not be in $[\varphi_{min}, \varphi_{max}]$, in which case the point must be discarded since it does not satisfy the joint limits.

6.2.2.2 Case 2: $R_{33}^2 = 1$.

In this case, Θ can be calculated from Equation (6.8), but Φ_A and Φ_B cannot be computed from Equations (6.9) and (6.10) since $s_\Theta = 0$. To compute these angles, we substitute $c_\Theta = R_{33}$ into elements (1,2) and (2,2) of Equation (6.3) and equate these elements to R_{12} and R_{22}:

$$\begin{bmatrix} R_{12} \\ R_{22} \end{bmatrix} = \begin{bmatrix} s_{\Phi_A} c_{\Phi_B} - R_{33} c_{\Phi_A} s_{\Phi_B} \\ c_{\Phi_A} c_{\Phi_B} + R_{33} s_{\Phi_A} s_{\Phi_B} \end{bmatrix} = \begin{bmatrix} \sin(\Phi_A - R_{33}\Phi_B) \\ \cos(\Phi_A - R_{33}\Phi_B) \end{bmatrix} \quad (6.11)$$

where the last equality is true because $R_{33} = 1$ or $R_{33} = -1$. In this case, Algorithm 2 can also be used with the following modification: $\{\varphi_{1B}, \varphi_{2B}, \varphi_{2A}\}$ are randomly sampled, whereas φ_{1A} is computed as $\varphi_{1A} = \varphi_{2A} + R_{33}\Phi_B + \text{atan2}(R_{12}, R_{22})$, discarding the point if $\varphi_{1A} \notin [\varphi_{min}, \varphi_{max}]$.

6.2.2.3 Example: constant-orientation workspace for concave and convex transitions

In this example, we assume that all design parameters are fixed at the default values used in the example of section 6.2.1.3, except for $\Delta\rho$, whose value must be chosen so as to allow the robot to perform a convex transition between different faces of a beam, as shown in Figure 6.4a. According to this figure, the desired position and orientation for foot B relative to the fixed foot A are given by the following matrices:

$$\mathbf{R}_{B/A} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} -11 \\ -11 \\ 0 \end{bmatrix} \text{ cm} \quad (6.12)$$

Doing the calculations explained above for this rotation matrix, and using $\Delta\rho = 5$ cm (default stroke for the linear actuators) yields the constant-orientation workspace of Figure 6.4b. As in section 6.2.1.3, here we have sampled $N_r = 2 \cdot 10^6$ random points and, in order to extract the boundaries, we have discretized the Cartesian space into $n_x = n_y = n_z = 50$ cells along each axis. Note that, for the default stroke, the

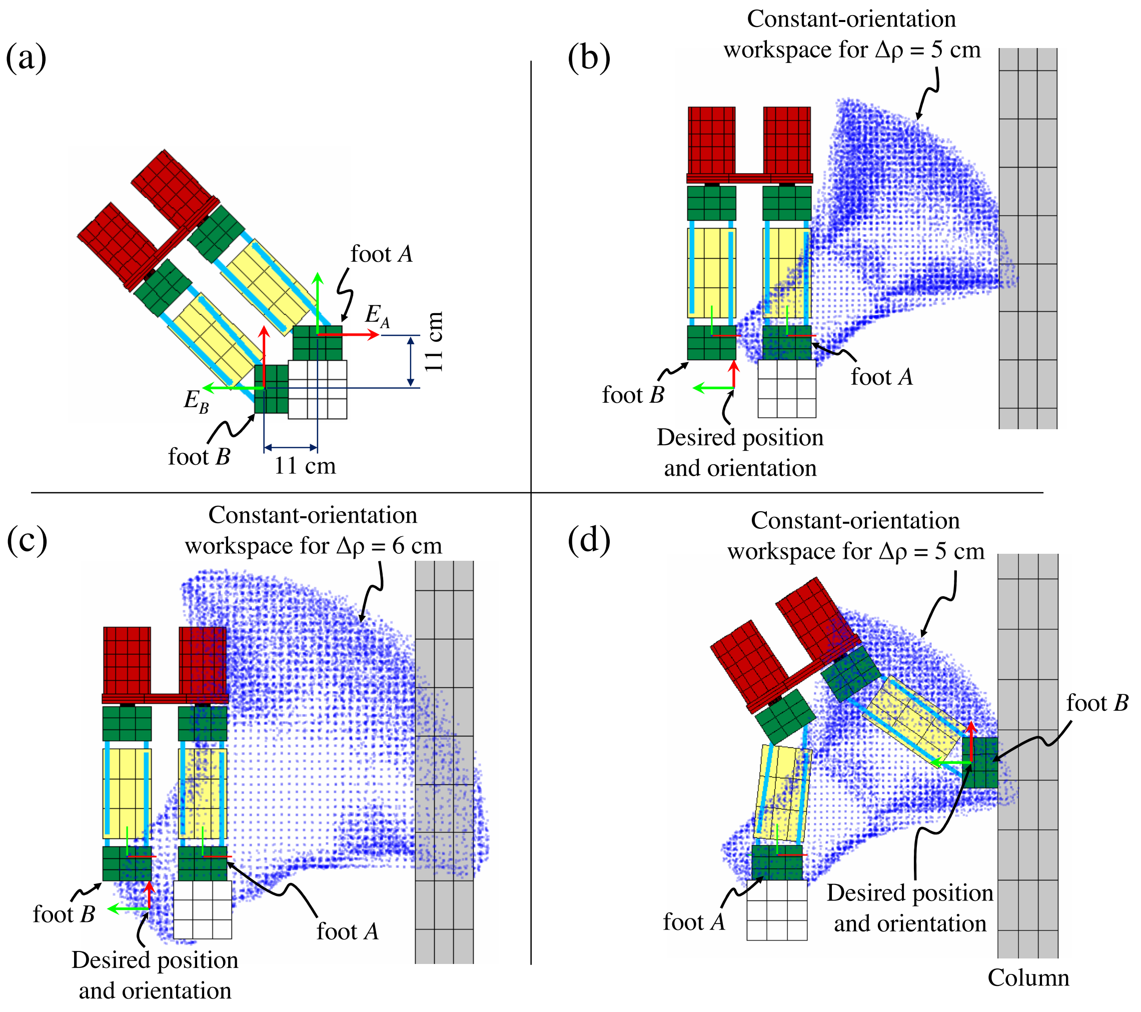

(a)
foot A
E_A
11 cm
E_B
foot B
11 cm
(b)
Constant-orientation workspace for Δρ = 5 cm
foot A
foot B
Desired position and orientation
(c)
Constant-orientation workspace for Δρ = 6 cm
foot B
foot A
Desired position and orientation
(d)
Constant-orientation workspace for Δρ = 5 cm
foot B
foot A
Desired position and orientation
Column

Figure 6.4: (a) Desired position and orientation to change between different faces of a beam. For $\Delta\rho = 5$ cm (b), the constant-orientation workspace for the desired orientation does not contain the desired point, but it contains the point for $\Delta\rho = 6$ cm (c). (d) Performing transitions between different beams using the default geometry.

desired point cannot be attained with the desired orientation because it lies outside the computed constant-orientation workspace. However, if the workspace is recalculated for $\Delta\rho = 6$ cm, we obtain the workspace of Figure 6.4c, which does contain the desired point. Thus, choosing a linear actuator with a stroke of 6 cm would permit the robot to change between different faces of the beam in this example.

Note that the orientation defined in Equation (6.12) is also necessary for attaching foot B to the right column by performing a concave transition, as indicated in Figure 6.4d. As shown in this figure, the constant-orientation workspace for $\Delta\rho = 5$ cm contains points that are near the left face of the column. Thus, it is possible to attach foot B to the column using the default geometry.

Note that, in the examples of Figures 6.2-6.4, one can approximately distinguish the boundaries of the workspace, but not very accurately. In fact, the obtained boundaries are quite noisy, so that it is difficult to exactly tell where is the real boundary

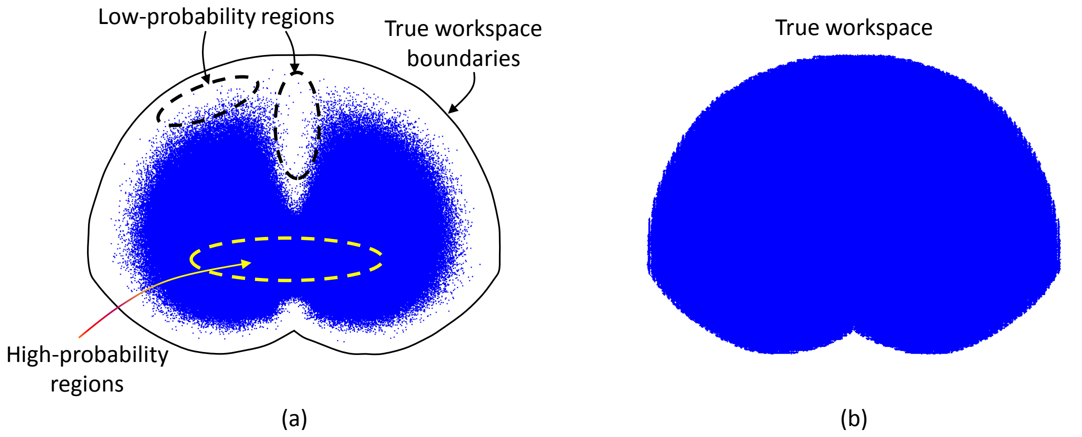


Figure 6.5: Comparison between the workspace obtained sampling the joint coordinates from uniform distributions (a) and the true workspace (b) of the HyReCRo robot. The workspace in (b) has been obtained using the new method proposed in Section 6.4. The time required to compute the workspaces (a) and (b) is the same.

of the workspace. This imprecision is even worse if all joint coordinates are sampled from uniform distributions, as we will demonstrate later. This is a drawback of classical Monte Carlo methods, a drawback that cannot be solved by simply increasing the number of randomly sampled points. We will analyze next this imprecision problem in more detail, and later we will propose an improved Monte Carlo method that solves it, being able to obtain more accurate workspaces without increasing the computation time.

6.3 Accuracy Problems of Classical Monte Carlo Methods

In the previous section, we have used a typical or "classical" Monte Carlo method for computing some workspaces of the HyReCRo robot. However, we have observed that the accuracy of the so obtained workspaces was not very good, in the sense that the boundaries of these workspaces were noisy and not very well defined. This section analyzes this accuracy problem in more detail.

Usually, variable r_k used in Equation (6.1) is a uniform random number in $(0, 1)$. However, as pointed out by Cao et al. [24], sampling from uniform distributions generally yields inaccurate and nonuniform workspaces, in which some regions are very dense and well-defined (regions populated by many workspace points) whereas other regions, especially those near the boundaries of the workspace, are too sparse (regions with comparatively much fewer points) and make it difficult to figure out the true shape of the workspace. For example, Figure 6.5a shows another example of the workspace of a HyReCRo robot, composed of $9 \cdot 10^6$ random workspace points obtained by sampling all joint coordinates from uniform distributions. Note that the true workspace of the robot, shown in Figure 6.5b, is bigger and has much better defined boundaries than the workspace obtained by sampling from uniform distributions, which has noisy and irregular boundaries.

The reason for this nonuniform density of the workspace is the nonlinearity of the forward kinematics transformation, which transforms joint coordinates $\mathbf{q}$ into position coordinates $\mathbf{X}$ of the end-effector. Although joint coordinates are distributed uniformly, this uniformity is not conserved by the nonlinearity of the transformation $\mathbf{q} \to \mathbf{X}$. As a result, $\mathbf{X}$ is distributed according to a nonuniform distribution, which has high-probability regions (regions in which workspace points are generated more often, like the internal regions indicated in Figure 6.5a) and regions of low probability (sparse regions in which points are hardly generated, like the workspace boundaries in Figure 6.5a). It should be noted that, although this nonuniform density may be undesirable for obtaining accurate workspaces, it is useful as a measure of the degree of redundancy of the robot across its workspace [23]. Indeed, the denser a region of the workspace is, the higher the redundancy is, because it means that the end-effector can be placed in that region with a wider variety of configurations.

To correct this accuracy problem and increase the density of points in sparse regions, one may try to increase the number of randomly generated points, increasing in this way also the computation time. However, this is not an efficient solution since most points still fall in high-probability regions [24]. Alternatively, to solve this problem and achieve higher accuracy (especially near workspace boundaries), Cao et al. [24] proposed using symmetric U-shaped beta distributions to sample the joint coordinates, instead of using uniform distributions. In that case, r_k in Equation (6.1) is a random variable with the following probability density function:

$$f(r_k, \beta_k) = K \left[r_k \left(1 - r_k\right)\right]^{\beta_k - 1} \tag{6.13}$$

where $0 < r_k < 1$, $0 < \beta_k \leq 1$ and K is a normalization constant such that $\int_0^1 f(r_k, \beta_k)\, \mathrm{d}r_k = 1$. This U-shaped distribution, shown in Figure 6.6 for different values of β_k, diverges to infinity at $r_k = 0$ and $r_k = 1$, and it is symmetric with respect to $r_k = 0.5$, where the minimum probability occurs. Parameter β_k determines the shape of the distribution: the smaller β_k is, the less probable the values around $r_k = 0.5$ are, and the more probable the values near the limits ($r_k = 0$ and $r_k = 1$) are. As β_k increases, the distribution adopts a more horizontal shape, and all values of $r_k \in (0, 1)$ acquire a more similar probability. The uniform distribution is a particular case of the beta distribution when β_k tends to 1 (see the case $\beta_k = 0.99$ in Figure 6.6).

As demonstrated in [24], using the beta distribution of Equation (6.13) to randomly sample the joint coordinates may yield more uniform workspaces, and with better defined boundaries, than using uniform distributions (generating in both cases the same number of random points). This is because, in many cases, the boundaries are typically attained when some joint coordinates reach their joint limits. If the beta distribution of Equation (6.13) is used, values of r_k near 0 and 1 will be generated more often than other values. Thus, according to Equation (6.1), more vectors of joint coordinates $\mathbf{q}$ will be generated near the joint limits. When transforming these vectors into workspace points, more points will be generated near the boundaries and, as a consequence, these boundaries will be better defined. For this reason, some intermediate joint coordinates were sampled from beta distributions in the examples of sections 6.2.1 and 6.2.2.

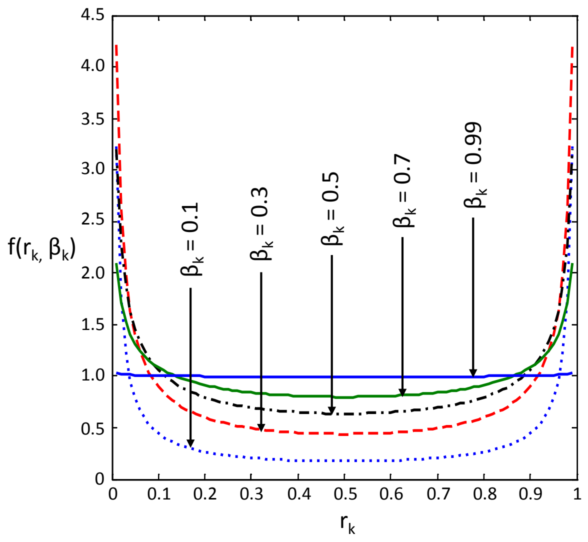


Figure 6.6: Symmetric U-shaped beta distribution.

Although using beta distributions may yield more accurate workspaces than uniform distributions, we have shown in sections 6.2.1 and 6.2.2 (and it will become even more evident after studying some more examples later) that this higher accuracy may still be insufficient. In the next section, we propose a new Monte Carlo method based on normal distributions to compute the workspace of robot manipulators. This new method will be compared with classical methods that use uniform or beta distributions, and we will demonstrate that the proposed method is able to obtain much more accuracy than previous Monte Carlo methods, requiring the same or less computation time than these methods.

6.4 An Improved Monte Carlo Method Based on Gaussian Growth

A drawback of classical Monte Carlo methods, in which joint coordinates are sampled from some random distribution (e.g., from uniform or beta distributions), is the fact that the random workspace points generated by these methods are distributed nonuniformly throughout the workspace. Thus, some regions are very dense and accurately defined whereas other regions are sparse and poorly defined. To densify these sparse regions and improve the accuracy of the workspace, it is necessary to increase the number of randomly generated points. However, that solution is not efficient because most of the newly generated points still fall in high-probability regions, i.e., much of the effort made to densify sparse regions is wasted in populating areas of the workspace that are already sufficiently populated.

Instead of using previous brute-force methods, in which more and more random points are generated in the whole workspace only to densify sparse regions, it would be more efficient to directly focus on densifying low-density regions until a uniform density is achieved throughout the workspace. In that case, all regions of the workspace would be equally well defined, including the boundaries. This is the objective of the method proposed in this section. Basically, the proposed method consists in generating an initial or *seed* imprecise workspace using a classical Monte Carlo method, and then growing and densifying low-density regions of this seed workspace using Gaussian (or normal) distributions. The method is divided into two stages, which are described next.

6.4.1 Stage 1: Generating a Seed Workspace

First, a classical Monte Carlo method is used for generating N_s workspace points, where N_s is much smaller than the number of points that will constitute the final dense workspace that will be obtained after both stages of the method have been completed. The objective at this point is to quickly generate, with little effort, an initial imprecise approximation of the workspace, whose points will be used as seeds around which an accurate and dense approximation of the workspace will be grown during the second stage of the method. Seed points can be generated sampling the joint coordinates from any random distribution, e.g., from uniform or beta distributions. Appropriate choices for this initial distribution will be briefly discussed in section 6.5.

To identify the regions of the workspace that have low density of points and need to be densified, it is necessary to discretize the 3D space into a set of cells with desired resolution along each dimension, and count the number of workspace points inside each cell. To this end, a box $\mathcal{B} = [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$ is first defined, which will enclose the densified workspace. Next, this box $\mathcal{B}$ is discretized into smaller cells, dividing its X, Y, and Z dimensions into n_x, n_y, and n_z equal parts, respectively (see Figure 6.1a).

Next, a database of cells is created, and the N_s seed points are stored in the corresponding cells of this database, storing up to N_c points in each cell (if the method attempts to store a point in a cell which already has N_c points, then that point is discarded). When storing each workspace point $\mathbf{X}^m$ in the cell in which it falls ($m = 1, \ldots, N_s$), we also store the vector of joint coordinates $\mathbf{q}^m$ that generated $\mathbf{X}^m$ (see Figure 6.7c). This is because $\mathbf{q}^m$ will be used in the second stage of the method to densify and grow the workspace. This step is illustrated in Figure 6.7 for the 2D case, for the sake of clarity.

After finishing the previous step, the cells containing N_c points are considered to be sufficiently dense, and the second stage of the method will not try to densify them. On the contrary, non-empty cells containing less than N_c points are included in a list PC of "pending cells", which is the set of cells that will be densified during the second stage of the method, since it is considered that they do not contain enough points to accurately define a workspace region.

6.4.2 Stage 2: Densifying and Growing the Seed Workspace

Upon the completion of the first stage, we have a list PC of non-empty cells which should be densified since they contain less than N_c workspace points. The second stage of the proposed method focuses on densifying all pending cells until PC is

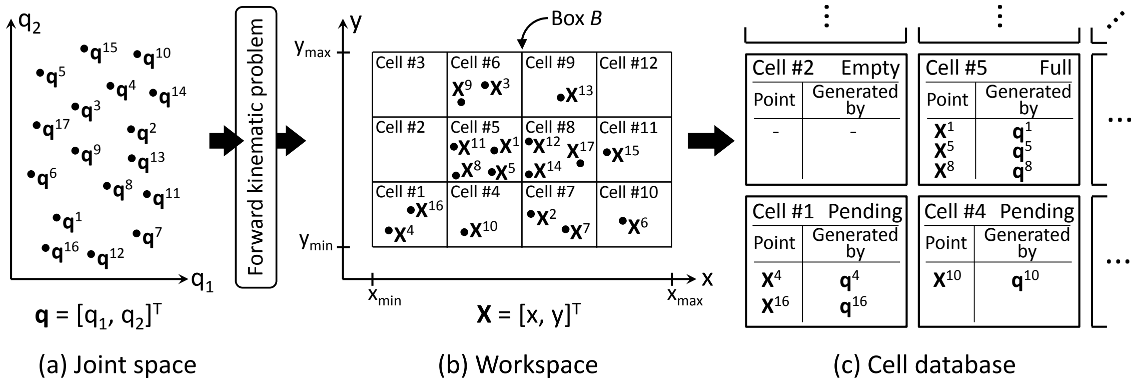


Figure 6.7: A 2D example of the first stage of the proposed GG method. (a) First, $N_s = 17$ seed joint coordinate vectors are randomly sampled in the joint space. (b) These vectors are mapped to the workspace solving the forward kinematic problem. The workspace is enclosed by a box $\mathcal{B}$, which is discretized into $n_x = 4$ and $n_y = 3$ cells along the X and Y axes, respectively. (c) A database of cells is created, and up to $N_c = 3$ points are stored in each cell. Cells $\{\#2, \#3, \#12\}$ are empty, cells $\{\#1, \#4, \#6, \#7, \#9, \#10, \#11\}$ are pending cells, and cells $\{\#5, \#8\}$ are full. Although cell #5 contains four points, only the first three points $\{\mathbf{X}^1, \mathbf{X}^5, \mathbf{X}^8\}$ are stored in the database since $N_c = 3$.

empty, growing also the workspace during this densification process. The steps of this stage are summarized in Algorihtm 3. Next, these steps will be detailed, omitting first some lines of Algorithm 3 for ease of exposition. The omitted lines will be described later.

The second stage of the method will not stop until list PC is empty (line 1 of Algorithm 3), i.e., until all pending cells have been densified. For each cell $C \in PC$, the algorithm will try to generate new random workspace points in C, until C contains N_c points (line 5). To generate a new point in C, one of the workspace points already stored in C is selected randomly (line 12). This randomly chosen workspace point of C is denoted by $\mathbf{X}^0$, and the vector of joint coordinates that generated $\mathbf{X}^0$ is denoted by $\mathbf{q}^0$ (line 13). Note that $\mathbf{q}^0$ is known because it was stored in the database created in the first stage, together with $\mathbf{X}^0$. Then, a new vector of joint coordinates $\mathbf{q}^*$ can be generated in the neighborhood of $\mathbf{q}^0$ using a multivariate normal distribution with mean $\mathbf{q}^0$ and appropriate covariance matrix. Alternatively, instead of sampling from a multivariate normal distribution, it may be simpler and sufficient to sample each joint coordinate q_k^* of $\mathbf{q}^*$ independently from a univariate normal distribution with mean q_k^0 and standard deviation σ_k ($k = 1, \ldots, d$), as shown in line 14.

If the newly generated vector of joint coordinates $\mathbf{q}^*$ satisfies the joint limits, the forward kinematic problem is solved to obtain the position $\mathbf{X}^*$ of the end-effector (lines 15 and 16). Next, it is checked if the generated position satisfies other *additional constraints* that may be considered, such as a desired orientation for the end-effector, or the absence of mechanical self-interferences or interferences with obstacles of the environment (line 17). If all constraints are satisfied, we proceed to the following steps.

Following, if point $\mathbf{X}^*$ is inside box $\mathcal{B}$, both $\mathbf{X}^*$ and $\mathbf{q}^*$ are stored in the cell C^* of $\mathcal{B}$ in which $\mathbf{X}^*$ falls, provided that C^* is not full (lines 18 to 25). Note that if $\mathbf{X}^*$ is the first point stored in cell C^* (i.e., the cell was empty prior to storing $\mathbf{X}^*$ in it), this cell is included in the list PC of pending cells which require densification (lines 21 and 22). Similarly, if $\mathbf{X}^*$ is the point that fills cell C^* (i.e. $\mathbf{X}^*$ is the N_c-th point stored in C^*), this cell is removed from the list of pending cells (lines 24 and 25).

Note that the cell C^* in which the generated point $\mathbf{X}^*$ falls may not be the current cell C that we are trying to densify, especially if the standard deviations used to generate $\mathbf{X}^*$ are too large (i.e., $\mathbf{q}^*$ is generated too far from $\mathbf{q}^0$). Thus, it may take too long to fill each pending cell C if the algorithm finds it difficult to generate points inside C. To avoid this, variable standard deviations are used instead of constant deviations. When beginning to densify each cell C, the standard deviations used to sample random joint coordinates from normal distributions are initialized to desired values σ_k^{ini} (line 3). Then, if the algorithm fails too often to generate points in C (because most randomly generated workspace points fall in cells other than C), the standard deviations are decreased so that the normal distributions centered at $\mathbf{q}^0$ become narrower and the probability of generating a point close to $\mathbf{q}^0$ (which *does* generate a point in C) increases.

The decrease of standard deviations is implemented as follows. Every time the algorithm fails to generate a point in current cell C, a counter variable n_f (which counts the number of successive failed attempts to generate a point in C) is increased by one unit (line 11). When n_f exceeds a threshold n_f^{max}, n_f is reset and each standard deviation σ_k is decreased dividing it by $\omega_k > 1$ (lines 8 to 10). Whenever the algorithm manages to generate a point in C, counter n_f is reset (lines 26 and 27).

The proposed method not only guarantees a uniform densification of the workspace, it also guarantees the growth of the workspace beyond the initial seed workspace obtained during the first stage of the method. When attempting to generate more points in each pending cell C, some of the generated points will fall in nearby cells due to the shape of the normal distribution (which has a decreasing but non-zero probability density as we move away from the mean). This means that, when trying to densify each cell C, the algorithm will also densify nearby cells, and in particular it will store points in nearby cells that were previously empty. These cells will be included in the list of pending cells, and when the algorithm tries to densify them the process will repeat: new cells around these will be populated, and this process will continue until the boundaries of the workspace are attained.

Finally, it should be noted that the algorithm does not necessarily have to completely densify all workspace cells to obtain an accurate workspace, i.e., it is not necessary to store exactly N_c points inside each cell, as justified next. When trying to densify an arbitrary pending cell C, if all its neighboring cells contain points, then we can conclude that C is not a boundary cell and we can guarantee that both C and its neighbors can be attained by the robot. Thus, there is no need to continue generating more points in C, which can be removed from list PC (lines 6 and 7). If C has empty neighboring cells, C may be a boundary cell and the algorithm should continue densifying C and trying to populate its neighbors, to ascertain whether the workspace has a boundary at C or, on the contrary, the workspace can be grown further beyond

C. Avoiding the complete densification of non-boundary cells saves computation time without affecting the accuracy of the final result.

After the second stage of the method has finished, the workspace is made up of all points that have been stored in cells of box $\mathcal{B}$, and the boundaries can be approximated by the set of non-empty cells that have empty neighboring cells.

Next, the performance of the method proposed in this section will be compared with classical Monte Carlo methods. The comparisons will be performed using the HyReCRo robot as a case study.

Algorithm 3 Gaussian densification and growth of the seed workspace

```
1:  while list $PC$ of pending cells is not empty do
2:      $C \leftarrow$ first pending cell of list $PC$
3:      Initialize standard deviations: $\sigma_k \leftarrow \sigma_k^{ini}$ $(k = 1, \dots, d)$
4:      $n_f \leftarrow 0$
5:      while $C$ contains less than $N_c$ points do
6:          if All neighboring cells of $C$ contain points then
7:              Remove $C$ from $PC$ and go back to line 1
8:          if $n_f > n_f^{max}$ then
9:              $n_f \leftarrow 0$
10:             Decrease standard deviations: $\sigma_k \leftarrow \sigma_k / \omega_k$ $(k = 1, \dots, d)$
11:         $n_f \leftarrow n_f + 1$
12:         $\mathbf{X}^0 \leftarrow$ workspace point randomly picked among those in $C$
13:         $\mathbf{q}^0 \leftarrow$ vector of joint coordinates that generated $\mathbf{X}^0$
14:         Sample $\mathbf{q}^*$ around $\mathbf{q}^0$: $q_k^* \leftarrow Normal(q_k^0, \sigma_k)$ $(k = 1, \dots, d)$
15:         if $\mathbf{q}^*$ satisfies joint limits then
16:             Solve the forward kinematic problem, obtaining $\mathbf{X}^*$ from $\mathbf{q}^*$
17:             if $\mathbf{X}^*$ satisfies *additional constraints* then
18:                 if $\mathbf{X}^* \in \mathcal{B}$ then
19:                     Find the cell $C^*$ of $\mathcal{B}$ in which $\mathbf{X}^*$ falls
20:                     if $C^*$ has less than $N_c$ points then
21:                         if $C^*$ is empty then
22:                             Add $C^*$ to list $PC$
23:                         Store $\mathbf{X}^*$ (and also $\mathbf{q}^*$) in cell $C^*$ of the database
24:                         if $C^*$ contains $N_c$ points then
25:                             Remove $C^*$ from list $PC$
26:                     if $C^* = C$ then
27:                         $n_f \leftarrow 0$
```

6.5 Examples, Comparative Analysis, and Discussion

In this section, we will apply the proposed Monte Carlo method based on Gaussian Growth (which we will denote by GG hereafter) to obtain some example workspaces of the HyReCRo robot, demonstrating the advantages of the proposed GG method.

In section 6.2, we found it more convenient to obtain the workspace of the HyReCRo robot by randomly sampling its intermediate joint coordinates, since this avoids having to solve the forward kinematics of the 2R$\underline{\text{P}}$R-PR parallel modules. However, in this section we will randomly sample the active joint coordinates (i.e., lengths l_{ij} and r_{ij}, as well as rotations θ_j) instead of the intermediate ones. This is because, for other general robots, "intermediate joint coordinates" analogous to those of the HyReCRo robot may not be always defined, whereas active/actuated joint coordinates are always defined for any robot manipulator, independently of its architecture (serial, parallel, hybrid...). Thus, in order to illustrate the proposed GG method with an example that can be representative, meaningful, and extrapolable to other robots (for which active joint coordinates are always well defined), in this section we will randomly sample the active joint coordinates of this robot, and vector $\mathbf{q}$ involved in the calculations described in previous section 6.4 will be: $\mathbf{q} = [l_{1A}, r_{1A}, l_{2A}, r_{2A}, l_{1B}, r_{1B}, l_{2B}, r_{2B}, \theta_A, \theta_B]^T$.

Note that, since we have decided to sample the active joint coordinates of the HyReCRo robot, we have to solve the forward kinematics of the 2R$\underline{\text{P}}$R-PR parallel modules that make up its legs, and this problem has four possible solutions for each parallel module. However, recall from the previous chapter that only one of these four solutions is valid for the HyReCRo robot: the one with highest value for y_{ij} (the remaining three solutions imply some mechanical interferences between different parts of the robot). Thus, when solving the forward kinematics of the parallel modules in the next examples, we will always use this only valid solution.

In the next examples, we will compute the reachable and constant orientation workspaces of the HyReCRo robot. To this end, and as we have done in section 6.2, we will compute positions of foot B relative to foot A using Equation (6.2). Also, we will use Equation (6.3) for computing constant orientation workspaces. For joint coordinates l_{ij} and r_{ij}, we will consider the following joint limits: $q_k^{min} = \rho_0$ and $q_k^{max} = \rho_0 + \Delta\rho$ (recall that ρ_0 is the minimum length of the linear actuators, whereas $\Delta\rho$ is their stroke).

For joint coordinates $\{\theta_A, \theta_B\}$, as argued in section 6.2.1.1, it will be assumed that they do not have joint limits, i.e., the legs can rotate freely with respect to the hip. However, since the configuration of the robot will not be affected if an integer multiple of 2π rad is added to angles $\{\theta_A, \theta_B\}$, we will restrict these angles to interval $[0, 2\pi]$. Although $\{\theta_A, \theta_B\}$ are restricted to this interval, since 0 and 2π are not true joint limits of these joint coordinates, the beta distribution should not be used to sample them. (Recall that the objective of using the beta distribution is to favor the generation of joint coordinates near their joint limits, since these generate workspace points that typically are near the boundaries of the workspace, which helps to better define these boundaries.) Therefore, when using classical Monte Carlo methods in the following examples, angles $\{\theta_A, \theta_B\}$ will always be uniformly sampled in $[0, 2\pi]$, even if the remaining joint coordinates (lengths $\{l_{ij}, r_{ij}\}$ of the eight linear actuators) are sampled from beta distributions.

To calculate the workspace of the HyReCRo robot in the next examples, the following values will be used for the design parameters: $b = p = 4$, $h = 16$, $t = 15.6$, $\rho_0 = 19$, and $\Delta\rho = 6$ (all values are in cm).

Also, it should be remarked that, in all the examples that follow, we will impose the additional condition that the legs of the robot should not collide. To check if legs collide, each leg is approximated by the union of two cuboids (one for the foot and another for the mechanism that connects the foot to the hip). Then, the Separating Axis Theorem [52] is used for checking if any cuboid of one leg intersects any cuboid of the other leg.

In the next subsections, we will present a comparison of different Monte Carlo methods to compute the workspace of the HyReCRo robot. The new Gaussian Growth (GG) method proposed in Section 6.4 will be compared with classical Monte Carlo methods (whose accuracy problems were discussed in Section 6.3). When using classical Monte Carlo methods, joint coordinates will be sampled from uniform or beta distributions. As explained above, when sampling from beta distributions, only the joint coordinates with true joint limits (i.e., lengths $\{l_{ij}, r_{ij}\}$ of the linear actuators) will be sampled from beta distributions; the rotations of the hip (θ_A and θ_B) will be uniformly sampled in $[0, 2\pi]$.

All examples shown in this section have been implemented in Java programming language and have been tested on a Mac Pro with a 3 GHz 8-Core Intel Xeon E5 processor and 16 GB RAM. Normally distributed random numbers were generated from uniformly distributed random numbers in $(0, 1)$ using Box-Muller's method (see [49], p. 235). Similarly, beta random numbers were generated from uniform random numbers in $(0, 1)$ using Johnk's method (see [49], p. 432), which is a fast method when the shape parameter β satisfies $0 < \beta < 1$ (which is our case).

In all the examples that follow, the seed workspace of the new proposed GG method will be generated by sampling the joint coordinates with joint limits from the beta distribution with $\beta = 0.1$ (θ_A and θ_B will be uniformly sampled). This choice is motivated by two facts that will be observed in the following experiments. In the first place, when generating the same number of workspace points using different values of β, computation time usually decreases with β. In the second place, when generating the same number of workspace points, these points are more diverse when β is smaller, whereas using a higher value of β (or a uniform distribution, in the limit $\beta \to 1$) generates points that are more similar, which is not good for the growing stage of the GG method. To begin growing the workspace in the GG method, it is more convenient to start from diverse and scattered seed points, than from very similar points concentrated in few regions.

6.5.1 Example 1: Reachable Workspace

In this example, we compute the reachable workspace, which is the set of points that can be reached by foot B with at least one orientation, i.e., we are not concerned about the orientation of foot B. To obtain this workspace, we only need to randomly sample the joint coordinates and solve the forward kinematic problem to obtain the position $\mathbf{X}$ of foot B [Equation (6.2)], checking for the absence of collisions between the legs of the robot.

First, the proposed GG method is used to calculate the workspace. A seed workspace of $N_s = 10,000$ points is generated by sampling all joint coordinates from beta distributions with $\beta = 0.1$, except for rotations θ_A and θ_B, which are uniformly sampled in $[0, 2\pi]$. Next, the second stage of the method is executed to grow and densify this seed workspace, defining the following box to enclose the workspace: $\mathcal{B} = [-70, 70] \times [-30, 70] \times [-45, 45]$ (all dimensions are in cm). This box is discretized into 100 cells along each axis, i.e., $n_x = n_y = n_z = 100$. The number of desired points in each cell is $N_c = 10$. The maximum number of consecutive failed attempts when trying to generate a point in each cell is $n_f^{max} = 10$. Whenever this maximum is exceeded, all standard deviations are divided by $\omega_k = 1.01$. The initial standard deviations of the joint coordinates when beginning to densify each cell are: $\sigma_k^{ini} = \Delta q_k/6$, where $\Delta q_k = \Delta\rho$ for the lengths $\{l_{ij}, r_{ij}\}$ of the linear actuators, and $\Delta q_k = 2\pi$ rad for $\{\theta_A, \theta_B\}$.

Executing the GG method with the previous parameters, it takes 17.76 minutes to generate a workspace of 3,527,664 points. This is the time necessary to execute the second stage of the method (densification and growth); the time required to generate the seed workspace is only about 0.04 minutes, which makes it negligible. Next, these points are assigned to the corresponding cells of box $\mathcal{B}$ and the boundary cells are extracted. Figure 6.8a shows the boundary surface of the calculated workspace, which is approximated by the centers of all boundary cells. The intersection between this boundary and the plane $z = 0.45$ cm is shown in Figure 6.9a. As these figures show, the reachable workspace has a single large connected component with a lens-shaped void inside. This internal void encloses foot A, and it originates from the condition of no-interference between the legs of the robot.

Next, classical Monte Carlo methods are used for calculating the workspace, generating the same number of points as the GG method (3,527,664 points). The boundaries of the workspaces obtained with classical Monte Carlo methods are shown in Figures 6.8b to 6.8f. Figure 6.8f shows the boundaries obtained when all joint coordinates are uniformly sampled, whereas Figures 6.8b to 6.8e show the boundaries obtained when sampling all joint coordinates from beta distributions with the indicated values of β (except for $\{\theta_A, \theta_B\}$, which are uniformly sampled in all these cases).

To facilitate the assessment of the precision of the obtained workspaces, Figure 6.9 shows the intersections of the boundaries of Figure 6.8 with the plane $z = 0.45$ cm. As these figures show, for the same number of random points, the proposed GG method generates a much more precise workspace than classical Monte Carlo methods, which yield noisy, thick and inaccurately defined workspace boundaries in all cases. The second best result after the GG method is obtained for $\beta = 0.1$. Note that the shape of the workspace becomes more distorted as β increases, and for $\beta \geq 0.5$ (approximately) the region above the internal void seems unreachable, which is false. The workspace obtained when sampling uniformly all joint coordinates is worthy of special mention, since it is the least accurate one (Figures 6.8f and 6.9f).

Figure 6.10 shows the time (in minutes) taken by each method to generate 3,527,664 random points. Note that the proposed GG method takes about 2.5 times

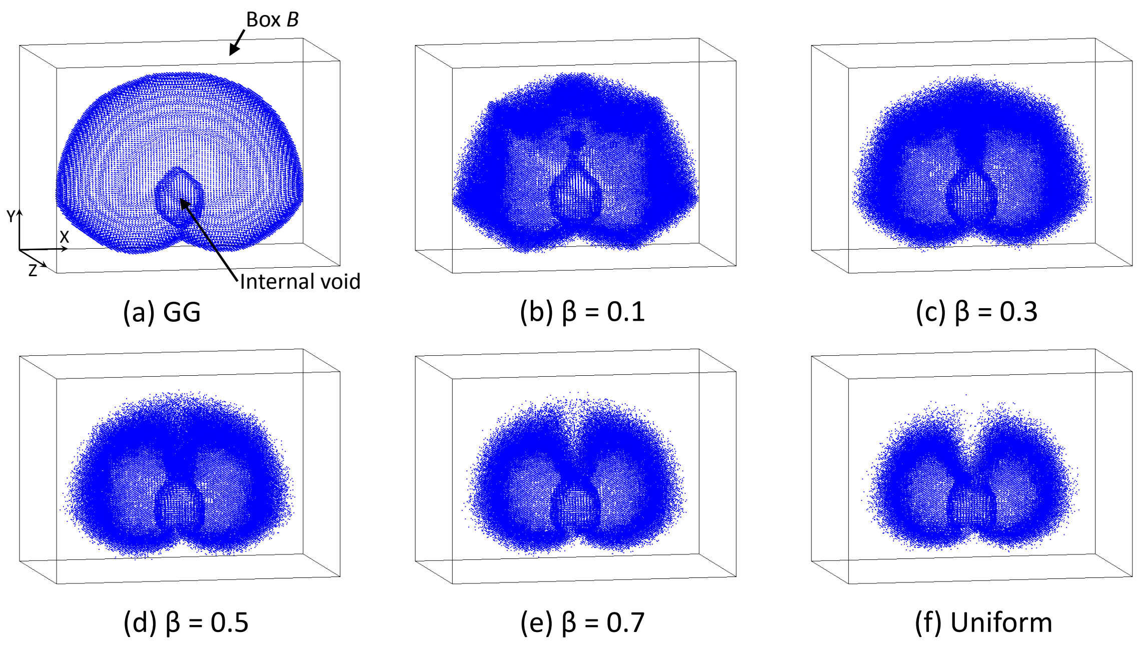


Figure 6.8: Boundaries of the reachable workspace, obtained with different Monte Carlo methods. The shown boundaries have been extracted from workspaces composed of 3,527,664 points each.

more time than the other methods, which is not surprising due to its higher precision. The question that arises now is: can the other methods generate as accurate workspaces as the GG method if the number of random points is increased until their computation times equal the time of the GG method? To answer this, the number of random points is increased for classical Monte Carlo methods, until their computation times equal approximately 17.76 minutes. Figures 6.11b to 6.11f show the intersections of the resulting workspace boundaries with the plane $z = 0.45$ cm, along with the number of random points and the actual computation time in each case.

As Figure 6.11 shows, the GG method is still much more accurate than the other methods for the same computation time. Note that, although the precision of the other methods has improved slightly, they generate boundaries that are still too noisy and inaccurately defined. This supports the idea that increasing the number of random workspace points is not an efficient solution for improving the accuracy of classical Monte Carlo methods [24].

6.5.2 Example 2: Workspace with Equality Constraints

In this example, we are interested in calculating a constant-orientation workspace, i.e. the set of points that can be attained with a desired orientation of foot B. The desired orientation is defined by the following rotation matrix, which is a rotation of $\pi/2$ rad about the Z axis:

$$\mathbf{R}^{desired} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.14}$$

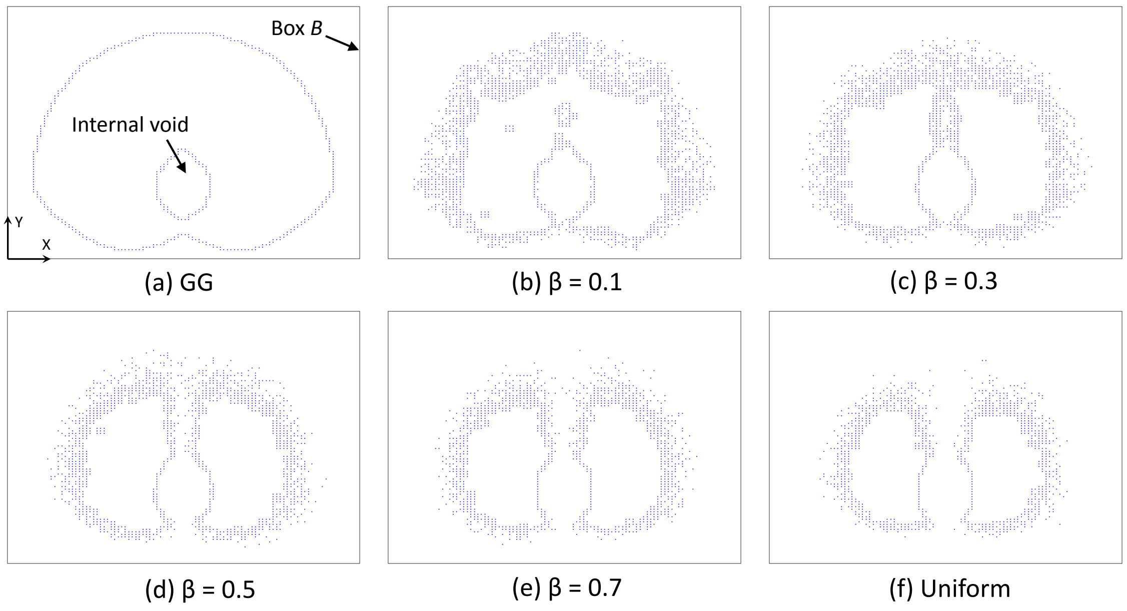


Figure 6.9: Intersections between the boundaries of Figure 6.8 and the plane $z = 0.45$ cm.

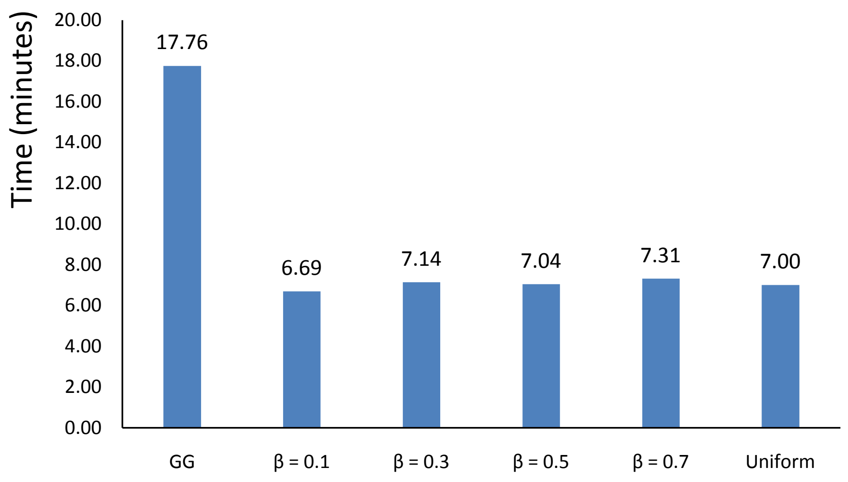


Figure 6.10: Time required to generate 3,527,664 workspace points with different Monte Carlo methods.

As in section 6.2.2.3, this orientation is necessary for performing concave or convex plane transitions in 3D structures. Furthermore, we are interested only in the intersection of this constant-orientation workspace with the plane $z = 0$, since the motions necessary for performing these transitions are planar. Equating the rotation matrices of Equations (6.3) and (6.14) yields the following two equations:

$$\cos(\theta_A - \theta_B) = 1 \quad \text{and} \quad \sin(\Phi_A - \Phi_B) = -1 \tag{6.15}$$

The previous two equations, together with the condition $z = 0$, impose three additional equality constraints to the problem of calculating the workspace. Next, these equality constraints will be handled following two different approaches.

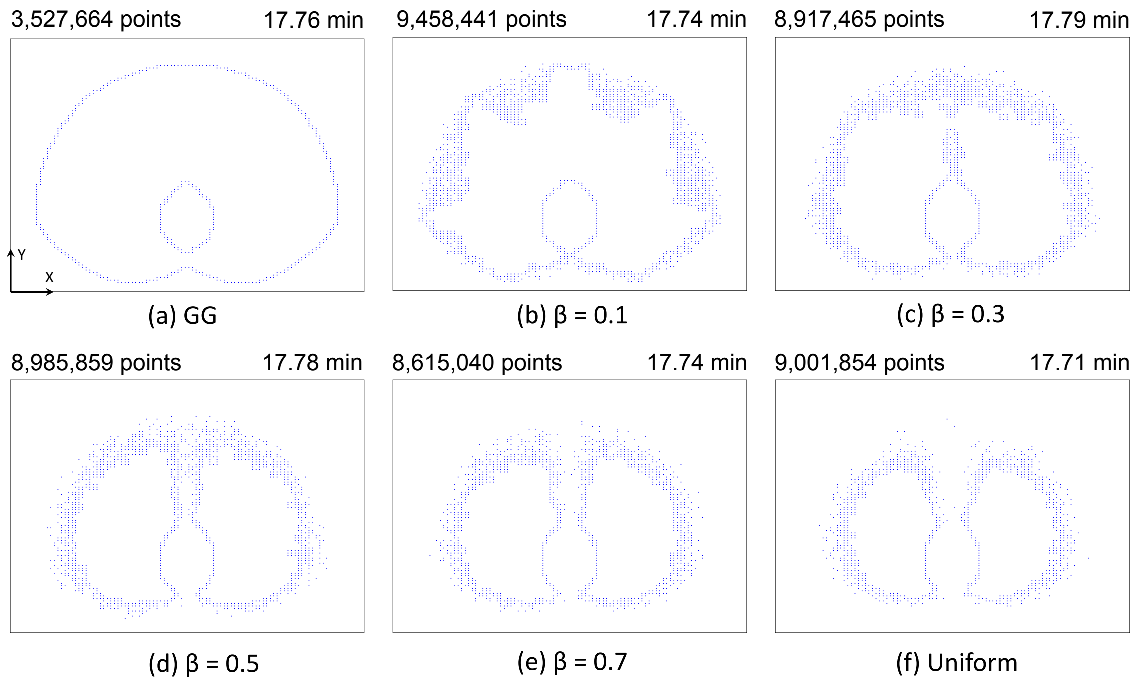


Figure 6.11: Slices at $z = 0.45$ cm of the workspaces obtained using different Monte Carlo methods (the time is the same in all cases). The number of points of the complete workspace (i.e., not only of the shown planar boundaries) is indicated in each case.

6.5.2.1 First approach: approximating equalities by inequalities

The constant-orientation workspace can be calculated following the same procedure followed to calculate the reachable workspace in the previous example, using any Monte Carlo method described in this chapter. The only difference is that, in addition to checking the condition of avoiding self-interferences, one should also check that the three aforementioned equality constraints are satisfied for each randomly generated workspace point. However, due to the numerical and random nature of Monte Carlo methods, it is practically impossible that a randomly generated workspace point satisfies *exactly* all three equalities simultaneously. Thus, it is necessary to transform the equalities into inequalities [68], which are easier to satisfy. In this example, the three equality constraints can be approximated by the following inequalities:

$$|z| < \epsilon_1, \quad |\cos(\theta_A - \theta_B) - 1| < \epsilon_2, \quad |\sin(\Phi_A - \Phi_B) + 1| < \epsilon_3 \tag{6.16}$$

where ϵ_i are sufficiently small. Evidently, the approximation will be better when ϵ_i are smaller, but it will also be more difficult to satisfy these inequalities and the computation time will increase (more random points will need to be sampled until we obtain a sufficiently high number of points that satisfy all these narrow inequalities). Next, we will compute the constant-orientation workspace using different Monte Carlo methods, including the inequality restrictions of Equation (6.16) in the calculation with the following thresholds: $\epsilon_1 = 0.5$ cm, $\epsilon_2 = 0.05$, and $\epsilon_3 = 0.001$.

First, the GG method is used. A seed workspace of 1,000 points is generated by sampling all joint coordinates from beta distributions with $\beta = 0.1$, except for

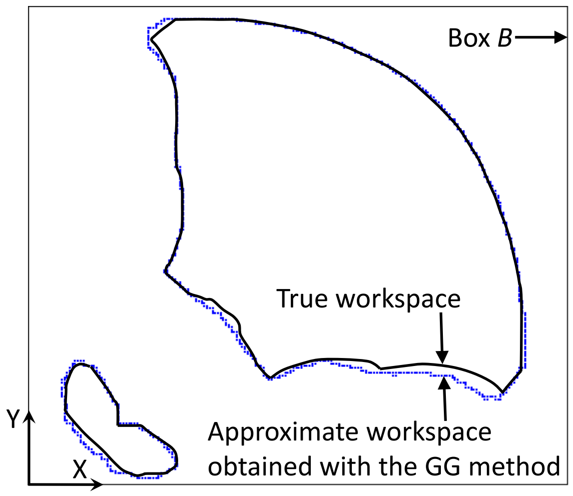


Figure 6.12: Planar constant-orientation workspace, obtained approximating all equality constraints by narrow inequalities.

$\{\theta_A, \theta_B\}$, which are uniformly sampled in $[0, 2\pi]$. The time required for generating the seed workspace in this case is 10.42 minutes. This increase in time with respect to the previous example is due to the fact that the workspace points must satisfy the narrow inequalities of Equation (6.16) in addition to the condition of avoiding self-interferences. After generating the seed workspace, the densification and growth stage of the GG method is executed. To enclose the workspace, the following box is defined: $\mathcal{B} = [-25, 65] \times [-20, 60] \times [-0.5, 0.5]$ (values in cm). This box is divided into $n_x = 200$, $n_y = 200$, and $n_z = 1$ cells along the X, Y, and Z axes, respectively (since the workspace is planar, the box is not discretized along the Z axis). The number of desired points in each cell is $N_c = 50$, and the maximum number of successive failed attempts is $n_f^{max} = 100$. Whenever this maximum is exceeded, all standard deviations are divided by $\omega_k = 1.01$, and the initial standard deviations of the joint coordinates when beginning to densify each cell are again: $\sigma_k^{ini} = \Delta q_k/6$.

Running the second stage of the GG method with the previous parameters, it takes 4.01 minutes to densify and grow the seed workspace. The workspace obtained after the second stage has finished is composed of 352,330 points. Thus, the net time necessary to execute the GG method in this example is 14.43 minutes, which is the sum of the time necessary for generating the seed workspace (10.42 minutes, not negligible in this case) and the time necessary for growing and densifying the seed workspace (4.01 minutes). The boundaries of the resulting planar workspace are shown in Figure 6.12, along with the true boundaries, which are shown in continuous line. Note that the obtained workspace does not exactly coincide with the true one. This is because the equality constraints have been approximated by the inequalities of Equation (6.16).

If we tried to generate the workspace with the same number of points (352,330) using classical Monte Carlo methods, the computation time would increase noticeably. For example, as shown immediately above, it takes 10.42 minutes to generate 1,000 seed points when sampling the joint coordinates from beta distributions with

$\beta = 0.1$ (except $\{\theta_A, \theta_B\}$, which are uniformly sampled). If all joint coordinates are sampled from uniform distributions, the time necessary to generate 1,000 workspace points increases to 39.47 minutes. Extrapolating, it is easy to conclude that the time necessary to generate 352,330 points using classical methods would be several hours (or even days), whereas the GG method only needs about 15 minutes to create the seed workspace and grow/densify it.

This example demonstrates that the proposed GG method is advantageous over classical Monte Carlo methods when narrow constraints are present, such as those obtained when approximating equalities by inequalities. This is because classical Monte Carlo methods generate points in the whole workspace without restriction or guidance, which makes it difficult to generate points that satisfy all narrow inequalities. On the contrary, the GG method focuses on generating points near those that already satisfy the narrow inequalities, i.e., it is a more directed search.

6.5.2.2 Second approach: solving the equality constraints

Although the first approach is simple, the drawback of approximating equality constraints by narrow inequalities is that the computation time can be high since it is difficult to generate points that satisfy all narrow inequalities, and the result is always approximate (see Figure 6.12). Wang et al. [191] proposed an alternative way of dealing with equality constraints in Monte Carlo methods, which is explained next.

First, the joint coordinates $\mathbf{q}$ are divided into two classes: independent joint coordinates $\mathbf{q}^{ind}$ and dependent joint coordinates $\mathbf{q}^{dep}$. Next, only the independent joint coordinates are randomly sampled, whereas the dependent joint coordinates are solved from the equality constraints in terms of $\mathbf{q}^{ind}$. Then, if $\mathbf{q}^{dep}$ satisfies the joint limits, the method continues as usual: the forward kinematic problem is solved and $\mathbf{X}$ is obtained. Since the dependent joint coordinates satisfy the equality constraints, this guarantees that vector $\mathbf{X}$ also satisfies these constraints exactly. In essence, this is the same strategy that we followed in section 6.2.2, in which some intermediate joint coordinates were solved from the constant-orientation equations in terms of other intermediate joint coordinates.

This approach is faster and more accurate than the one of Section 6.5.2.1. However, it is only feasible if the dependent joint coordinates $\mathbf{q}^{dep}$ can be easily solved from the constraints, preferably if we can obtain $\mathbf{q}^{dep}$ analytically in terms of $\mathbf{q}^{ind}$. In the example studied in this section, it can be shown that, if the joint coordinates are partitioned as $\mathbf{q}^{ind} = [l_{1A}, l_{2A}, r_{2A}, l_{1B}, r_{1B}, l_{2B}, r_{2B}]^T$ and $\mathbf{q}^{dep} = [\theta_A, \theta_B, r_{1A}]^T$, then it is possible to solve analytically $\mathbf{q}^{dep}$ in terms of $\mathbf{q}^{ind}$ from the equality constraints. Thus, the method described in [191] can be used in this example.

Once the joint coordinates have been divided into independent and dependent joint coordinates, both the classical and GG Monte Carlo methods described in this chapter can be applied with only two modifications. In the first place, instead of randomly sampling the vector of all joint coordinates $\mathbf{q}$ from some random distribution (uniform, beta, or normal) in any of the previous methods, only the vector of independent joint coordinates $\mathbf{q}^{ind}$ must be sampled (similarly, $\mathbf{q}^{ind}$ instead of $\mathbf{q}$ must be stored in the cell database in the GG method). In the second place, after randomly sampling $\mathbf{q}^{ind}$, $\mathbf{q}^{dep}$ must be solved from the equality constraints and it must be checked if $\mathbf{q}^{dep}$ satisfies the joint limits.

Next, the GG method will be used with the modifications described in the previous paragraphs to compute the planar constant-orientation workspace solving the three aforementioned equality constraints. The parameters of the densification and growth stage of the GG method have the same values as in Section 6.5.2.1. First, a seed workspace of 1,000 points is generated sampling the independent joint coordinates $\mathbf{q}^{ind} = [l_{1A}, l_{2A}, r_{2A}, l_{1B}, r_{1B}, l_{2B}, r_{2B}]^T$ from beta distributions with $\beta = 0.1$, which takes about 0.02 minutes. Then, the densification and growth stage of the GG method is executed, which takes 2.95 minutes and generates a planar constant-orientation workspace consisting of 521,212 points. The boundaries of the resulting workspace, which has two connected components, are shown in Figure 6.13a. In this case, the time necessary for generating the seed workspace is negligible compared to the time of the densification and growth stage.

Next, classical Monte Carlo methods are used for generating 521,212 workspace points, sampling all independent joint coordinates from uniform or beta distributions, and solving the dependent joint coordinates $\mathbf{q}^{dep} = [\theta_A, \theta_B, r_{1A}]^T$ from the equality constraints. The boundaries of the obtained workspaces are shown in Figure 6.13b-f, and the times required for generating 521,212 points in each case are shown in Figure 6.14.

As Figure 6.13 shows, the most accurate workspace is obtained using the GG method, while the other methods produce noisy and inaccurate boundaries. Again, the worst results are obtained when sampling from uniform distributions, and smaller values of β yield better results than higher values.

Moreover, Figure 6.14 shows that, in this example, the GG method is the fastest method for generating the same number of random workspace points (521,212), unlike in the example of Section 6.5.1. Thus, we can conclude that the GG method can obtain more precise workspaces than classical Monte Carlo methods even requiring less computation time, as this example shows.

6.5.3 Discussion

The previous experiments demonstrate that the GG method can calculate more accurate workspaces than previous classical Monte Carlo methods (which use uniform and/or beta distributions) requiring the same calculation time. Moreover, if additional constraints are imposed to the calculation of the workspace (e.g., a desired orientation for the end-effector), the GG method may even require less time than previous Monte Carlo methods, attaining also higher accuracy.

These experiments have also confirmed two features of classical Monte Carlo methods observed previously by Cao et al. [24], namely: using beta distributions may

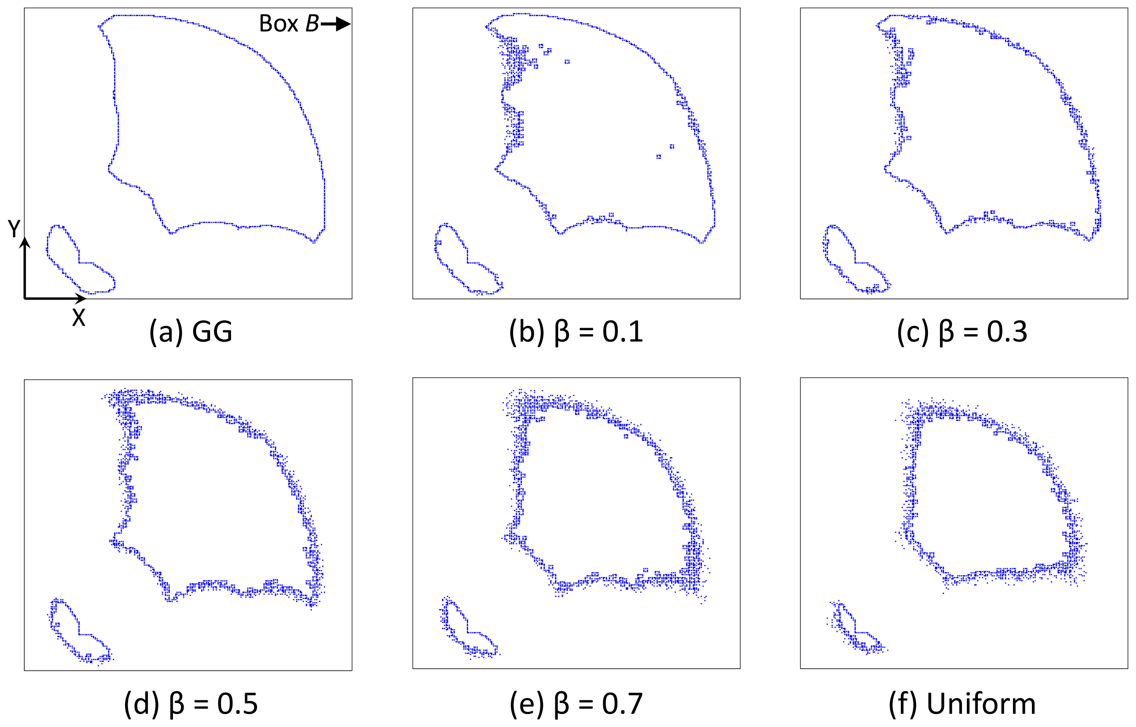


Figure 6.13: Boundaries of the planar constant-orientation workspace, obtained with different Monte Carlo methods. The shown boundaries have been extracted from workspaces composed of 521,212 points each.

yield better results than using uniform distributions, and increasing the number of random points is not an efficient solution for improving the accuracy. Besides, although the uniform distribution is the most widely used one for calculating the workspace of robots using Monte Carlo methods, the performed experiments discourage using this distribution since it yields poor results when compared with other methods. Still, although not detailed in this chapter, we have identified some cases in which uniform distributions may perform quite well. For example, comparing the GG and uniform sampling methods in robots containing only unbounded revolute joints (i.e., without joint limits), like a general 7R serial arm, reveals that sampling from uniform distributions might generate workspaces that are almost as accurate as those obtained with the proposed GG method, requiring the same computation time. Nevertheless, when joint limits are imposed to these revolute robots, the GG method outperforms again both beta and uniform sampling methods, obtaining higher accuracy requiring the same computation time.

Finally, it may be possible that the proposed method misses some small components of the workspace when it is composed of some disjoint components, as in the example of Figure 6.13. Since the GG method is based on the growth of workspace regions from the seed workspace, if the first stage of the method does not generate at least one seed point in a given component of the workspace, then the method may not grow and densify that component. There are several solutions to this problem, which may also be combined. For example, the simplest solution consists in increasing

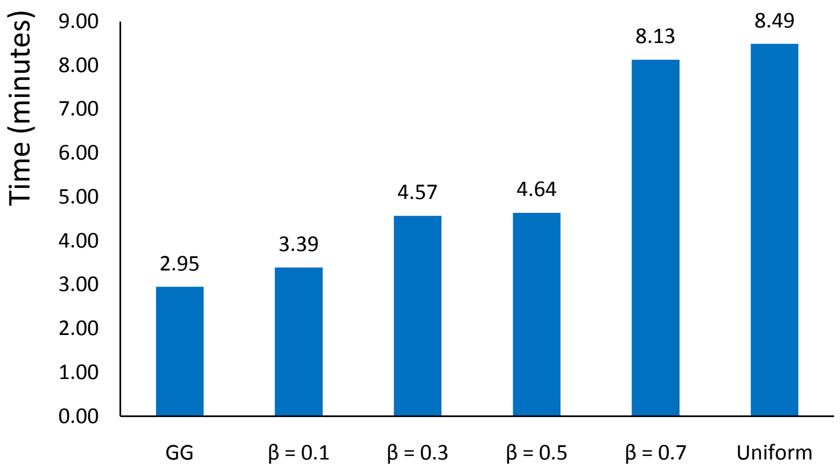


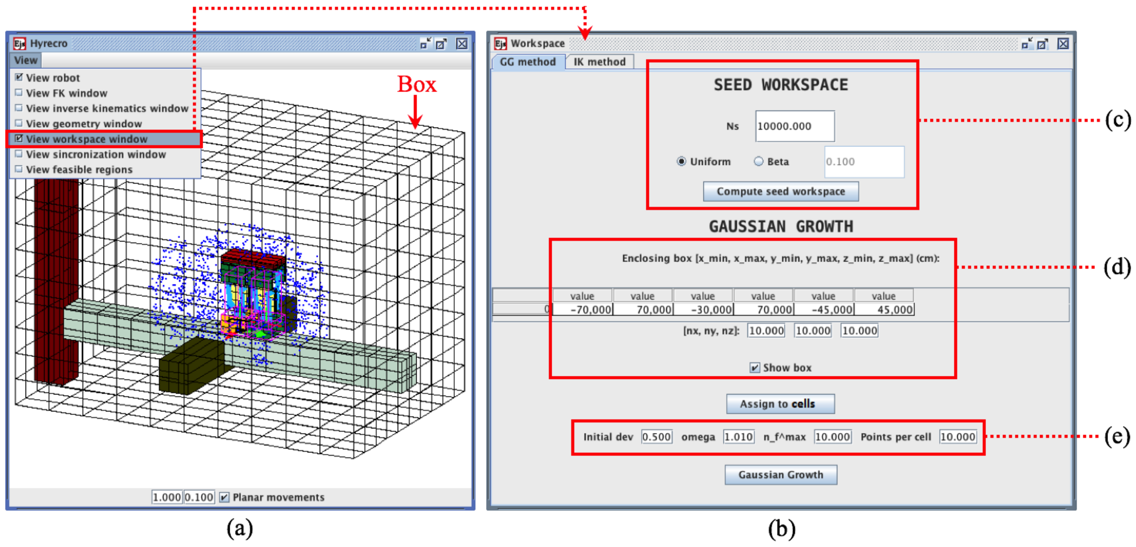
Figure 6.14: Time required to generate 521,212 workspace points with different Monte Carlo methods, when solving the equality constraints.

the number N_s of seeds generated during the first stage of the algorithm, with the purpose of increasing the probability of generating at least one seed in every component. Note that, in the example of Section 6.5.2.2, a densified workspace of 521,212 final points was grown from a seed workspace of 1,000 points, which constitutes less than 0.2% of the number of final points. Such a negligible ratio suggests that we may increase N_s by at least one order of magnitude to try to generate seeds in all components of the workspace, without practically affecting the overall performance of the method. Another solution may consist in choosing an appropriate distribution for the generation of the seed workspace, a distribution that favors the creation of diverse and scattered points in all components of the workspace. As discussed at the beginning of the present section 6.5, beta distributions may be useful for this purpose, but other random distributions may also be explored.

6.6 GG Method in the Simulator of the HyReCRo Robot

The simulator of the HyReCRo robot presented in Section 5.5 can also be used for studying the workspace of this robot using the GG method developed in this chapter. To that end, the user must activate the workspace window through the "view" menu at the top of the main window of the simulator, as illustrated in Figure 6.15a. When doing this, the window shown in Figure 6.15b pops up. This window has two tabs: "GG method" and "IK method". The second tab will be analyzed in the next chapter.

As for the content of the first tab "GG method" shown in Figure 6.15b: in this tab, the user can configure the GG method proposed in this chapter in order to obtain the workspace of the HyReCRo robot. To that end, first the user must generate the seed workspace (Figure 6.15c), indicating the number N_s of seed points to be randomly generated, as well as the distribution from which the actuated joint coordinates $\{l_{ij}, r_{ij}\}$ of the parallel modules should be sampled for generating these random points. These joint coordinates can be sampled from uniform or beta distributions,



1.000 0.100 ☑ Planar movements

Ns 10000.000

[✓ Uniform] Beta 0.100

Enclosing box [x_min, x_max, y_min, y_max, z_min, z_max] (cm):

	value	value	value	value	value	value
0	-70,000	70,000	-30,000	70,000	-45,000	45,000

[nx, ny, nz]: 10.000 10.000 10.000

☑ Show box

Initial dev 0.500 omega 1.010 n_f^max 10.000 Points per cell 10.000

Figure 6.15: Computing the workspace of the HyReCRo robot using the GG method in the developed simulator.

whereas θ_A and θ_B are always sampled from uniform distributions when generating seed workspaces, due to the reasons exposed earlier in this chapter. After this, pressing the button "Compute seed workspace" will generate the initial seed workspace from which the GG method will grow a more precise workspace.

Following, the second stage of the method (densifying and growing) must be executed. First, the user must define the box $\mathcal{B}$ that will enclose the workspace (Figure 6.15d), indicating the minimum and maximum coordinates of this box along each axis (x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}) and the number of cells into which each axis of this box will be discretized (n_x, n_y, n_z). The visualization of this box in the main window (Figure 6.15a) can be enabled or disabled by means of the tick-box entitled "Show box" in Figure 6.15d (it is advisable to disable the visualization of this box when finely discretizing it; otherwise, this box will have too many lines and it will be impossible to distinguish the robot or the workspace in the simulator). After defining box $\mathcal{B}$ in panel of Figure 6.15d, it is necessary to assign the seed workspace points to the cells of this box by pressing the button "Assign to cells", which initializes the cell database illustrated in Figure 6.7c.

Next, the user must tune the parameters of the proposed GG method, which can be done in the panel shown in Figure 6.15e. In this panel, the user must specify the value of the initial standard deviations (actually, the user specifies a factor that, multiplied by $\Delta q_k/3$, gives the initial standard deviations), the divider factor $\omega_k > 1$, the number n_f^{max} of maximum consecutive failures when attempting to generate points in each cell, and the number N_c of points to house in each cell.

After this, the user must click the button "Gaussian Growth" so that the method proposed in this chapter starts running. After some time computing (which depends on the configuration parameters of the method), the method finishes and shows the densified workspace in the main window of the simulator, together with the robot (as in Figure 6.15a). Note that in the concrete example of Figure 6.15a the obtained workspace is not very accurate, this is because a coarse discretization of box $\mathcal{B}$ was used on purpose, in order to better illustrate this functionality of the developed simulator. Actually, the proposed GG method is able to compute workspaces much more accurately, as we have demonstrated through the examples of Section 6.5.

Finally, in the simulator it is also possible to modify the geometric design parameters $\{b, p, t, h, \rho_0, \Delta\rho\}$ of the HyReCRo robot. This can be done in the window that appears when activating the button "View geometry window" in the "view" menu at the top of the main window (Figure 6.15a). This is useful for studying how the workspace is modified when varying the values of the geometric parameters of this robot, as we have done in Section 6.2.1.3.

6.7 Conclusions

In this chapter, the boundaries of the workspace of the HyReCRo robot have been analyzed. First, methods for computing workspaces have been reviewed in Section 6.1, in order to choose suitable methods for computing the workspace of the HyReCRo robot. Due to the complexity of this robot, which is serial-parallel and kinematically redundant, it has been concluded that Monte Carlo methods seem the most suitable ones. Accordingly, classical Monte Carlo methods have been used for computing reachable and constant-orientation workspaces of the HyReCRo robot (Section 6.2). The sensitivity of these workspaces with respect to the geometric design parameters of the robot has been investigated, obtaining that these workspaces are most sensitive to variations in the width p of the feet of the robot and in the stroke $\Delta\rho$ of the linear actuators. This coincides with the sensitivity analysis of the PSIK-workspace, performed in the previous chapter.

After this preliminary analysis of the workspace of the HyReCRo robot, based on classical Monte Carlo methods, the shortcomings of these methods have been discussed in Section 6.3, highlighting the low accuracy of the workspace boundaries obtained by these methods. This accuracy problem cannot be efficiently solved by simply increasing the number of randomly sampled points. Thus, an improved Monte Carlo has been proposed in Section 6.4 in order to alleviate this accuracy problem of classical Monte Carlo methods. The proposed method initially generates an imprecise workspace using classical Monte Carlo methods, and then grows uniformly this initial imprecise workspace using Guassian distributions, until the boundaries of the workspace are reached. Through several experiments presented in Section 6.5, it has been demonstrated that the proposed improved method can obtain much more accurate workspaces than classical Monte Carlo methods, requiring the same or less computation time than them. The proposed method has been implemented in the developed simulator of the HyReCRo robot (Section 6.6).

6.8 Publications Related to this Chapter

The main results presented in this chapter are related to the following publications:

- A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, L. Payá, and O. Reinoso. Monte-carlo workspace calculation of a serial-parallel biped robot. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 157–169, 2016. Springer International Publishing [131].
  - This paper presents the computation of the workspace of the HyReCRo robot using classical Monte Carlo methods, in order to determine the sensitivity of the shape and size of the workspace with respect to the geometric design parameters of this robot.
- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots. *Engineering Applications of Artificial Intelligence*, 64:197 – 207, 2017 [145] **(SCI-JCR Impact Factor: 2.819, Q1)**.
  - This paper presents the improved Monte Carlo method proposed in Section 6.4, which obtains more accurate workspaces than previous Monte Carlo methods requiring at most the same computation time as them.

7 Interior Barriers of the Workspace

In the previous chapter, we have computed and analyzed the workspace of the HyReCRo robot, using different Monte Carlo algorithms. However, in the previous chapter we have focused only on the outermost boundaries of the workspace, omitting important information about its internal structure, which includes interior barriers of the workspace that cannot be crossed by the robot. In contrast, this chapter analyzes the internal structure of the workspace of redundant robot manipulators. After reviewing methods for computing the interior barriers of the workspace under different conditions (section 7.1), section 7.2 presents a new sampling method for obtaining these barriers under joint limits and general collision constraints (previously existing methods for obtaining these interior barriers cannot easily accommodate general collision constraints). The proposed method is based on the property that, when self-motion manifolds of redundant robots vanish, then interior barriers occur (Section 7.1.1). After demonstrating the capabilities and usefulness of the proposed method through several examples of redundant parallel robots in section 7.3, section 7.4 proposes a variant of this method that may be applied to the HyReCRo robot.

7.1 Interior Barriers of the Workspace

In the previous chapter, we have presented a new Monte Carlo method based on Gaussian Growth which is able to obtain the workspace of robot manipulators more accurately than previously existing Monte Carlo methods, requiring the same or less computation time than them. In order to compare the accuracy attained by different Monte Carlo methods, in the previous chapter we obtained and compared the *boundaries* of the workspace obtained by different methods. However, once we obtain the boundaries of the workspace, it does not mean that we have obtained all the information regarding the workspace. These boundaries only provide information about the external shape and volume of the workspace, but they omit important information regarding its internal structure.

Usually, most of the methods reviewed in section 6.1 for obtaining the workspace only obtain its boundaries, omitting highly valuable information about its internal structure. It is well known that inside the boundaries of the workspace there may exist *interior barriers* (see Figure 7.1) which imply motion impediments for the robot [19, 1]. Knowing the distribution of such barriers inside the workspace is necessary for effectively planning trajectories in the task space, since a given trajectory that crosses one of these barriers may be unfeasible, depending on the values of the joint coordinates when approaching the barrier (see next section 7.1.1).

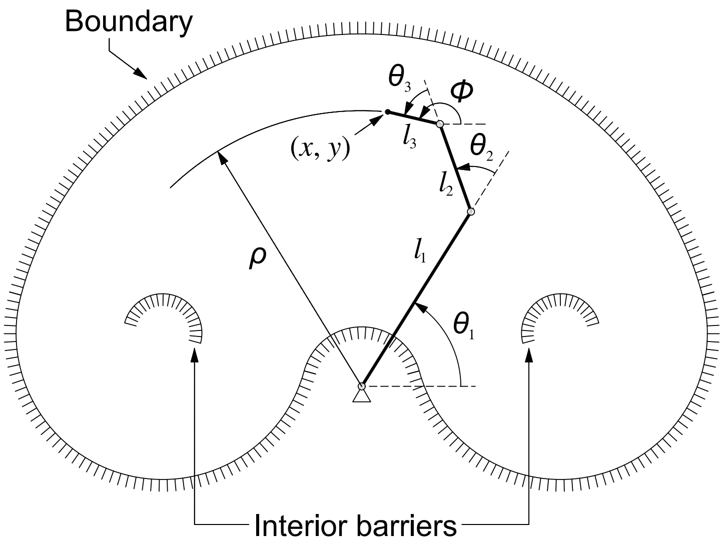


Figure 7.1: Workspace of a redundant 3R serial robot with $l_1 = 17.3$, $l_2 = 7.8$ and $l_3 = 4.5$. The first joint angle is subject to joint limits $\theta_1 \in [15, 165]°$; the second and third joints can freely rotate. This example is very similar to an example presented in [19].

Like the boundaries of the workspace, the interior barriers depend on the considered kinematic constraints, which typically are joint limits and collision constraints. While existing methods can easily handle joint limits, collision constraints are generally very difficult to model and accommodate by existing methods. However, collision constraints are important for the HyReCRo robot, which must not collide with itself or with the structure when climbing and exploring it. Therefore, the main objective of the present chapter will be to find a way to obtain interior barriers under collision constraints effectively, for redundant robots. To that end, we will begin by analyzing how we can obtain these interior barriers using the different families of methods reviewed in section 6.1, with the purpose of identifying the best strategy to attack this problem.

Singularity-based methods [19] naturally obtain the interior barriers among the solutions of system $\mathcal{S}$ mentioned in section 6.1. Geometrical methods may also be able to identify such interior barriers in simple cases [see the comments about reference [118] at the end of section 7.1.1 of the present chapter]. However, as argued in section 6.1, both these methods have the limitation that they cannot easily handle excessively complex (yet common) kinematic constraints, such as the prohibition of collisions.

In this context, interval analysis has proven useful for checking collisions [117] and obtaining collision-free generalized aspects [25] and workspaces [80] of non-redundant parallel robots. However, to the best of our knowledge, these methods have not been used for obtaining interior barriers in redundant robots under complex collision constraints.

In contrast to the previous methods, sampling methods seem promising for obtaining the interior barriers of the workspace under complex collision constraints, due to their ability to easily handle these constraints: one simply has to sample configurations of the robot, and discard all configurations that do not satisfy collision constraints. FK-based sampling methods might reveal the interior barriers in simple cases if joint coordinates are sampled from appropriate random distributions [156]. For example: in Figure 7.1, one can check that, if θ_1 is sampled from a U-shaped beta distribution [Equation (6.13)] in $[15°, 165°]$ and $\{\theta_2, \theta_3\}$ are uniformly sampled in $[0, 360°]$, then the density of randomly generated task points will be higher near the interior barriers, revealing them. However, this method is not completely robust nor predictable, and not easy to generalize.

Therefore, the last available option is to detect interior barriers using IK-based sampling methods. In non-redundant robots, the solutions of the inverse kinematics for a given task-space point generically are a finite number of isolated points of the joint space. Thus, in non-redundant robots, interior barriers can be easily found at those task-space points where the number of different inverse kinematic solutions changes. In redundant robots, like the HyReCRo robot, the solutions of the inverse kinematics for a given task-space point generically are a finite number of disjoint positive-dimensional manifolds in the joint space (*self-motion manifolds*). One may try to identify interior barriers with task-space points at which the number of such manifolds changes, but in general (as we show in this chapter) this does not necessarily imply the occurrence of interior barriers.

This chapter elaborates on the previous ideas and presents an IK-based sampling method to obtain the boundaries and interior barriers of the workspace of redundant robots, considering joint limits and collision constraints. The proposed method identifies the barriers of the workspace (both interior barriers and external boundaries) with the vanishing of self-motion manifolds, and consists of three stages: *sampling*, *clustering*, and *matching*. Firstly, self-motion manifolds are densely sampled by solving the equations of the inverse kinematics, discarding the samples that do not satisfy joint limits or collision constraints. Secondly, the obtained samples are clustered to identify disjoint self-motion manifolds. Then, a third *matching* stage monitors the transformations suffered by each identified manifold as a result of perturbing the task variables, to determine if any of these manifolds vanishes when varying the task variables. If the vanishing of manifolds is detected when the task variables move between two sufficiently close task points, this means that there exists a barrier between these two points.

In this chapter, we will assume the following conditions and notation: consider a redundant robot under joint limits and/or collision constraints. Assume that the kinematic chain of the robot (which can be serial, parallel, or hybrid, as in the HyReCRo robot) has n degrees of freedom (DOF) and the dimension of its task space is $m < n$. This means that n actuated joint coordinates $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]$ are used to control m task variables $\mathbf{t} = [t_1, \ldots, t_m]$. Typically, θ_j denote the lengths of linear actuators or the rotated angles of revolute actuators, whereas t_i define the position and/or orientation of some output link of the robot which is of interest for a given task. The workspace can be defined as the set of values that vector $\mathbf{t}$ can attain subject to the kinematics of the robot (i.e., the mathematical relationship between $\boldsymbol{\theta}$ and $\mathbf{t}$) and to other kinematic constraints (e.g., joint limits or avoidance of collisions).

Note that the analysis presented in this chapter is valuable not only for studying purely redundant robots (like the HyReCRo robot), but also when analyzing certain workspaces of non-redundant robots. For example, the workspace of a (not necessarily redundant) 6-DOF robot is the 6D set of spatial translations and orientations attainable by its end-effector. Since 6D sets cannot be represented, other 3D representations are necessary for visually analyzing the workspace. One may fix the orientation of the end-effector and represent only the attainable translations, obtaining the constant-orientation workspace (see section 6.2.2). Alternatively, one may project the 6D workspace to the 3D subspace of translations, obtaining the reachable workspace (i.e., the set of positions attainable by the end-effector with at least one orientation, see section 6.2.1.3). Reachable workspaces, which are widely studied, are examples of "redundant workspaces", because they only retain information related to the translation of the end-effector, omitting its orientation.

Next, before describing the proposed method in detail, we will analyze the relationship between barriers (either interior barriers or boundaries) of the workspace of redundant robot manipulators and their self-motion manifolds, in order to demonstrate that, when connected components of these manifolds vanish, then such barriers occur. This is the key property exploited by the method proposed in Section 7.2 for calculating the workspace barriers induced by collision constraints.

7.1.1 Relationship Between Self-motion Manifolds and Interior Barriers

In redundant robots, the inverse kinematic problem has infinitely many different solutions for a given task-space point. Generically, these solutions lie on a finite number of disjoint positive-dimensional manifolds in the n-dimensional joint space. These manifolds are called *self-motion manifolds* because moving the robot along them modifies the joint configuration $\boldsymbol{\theta}$ of the robot without affecting the values of the task variables $\mathbf{t}$.

Self-motion manifolds were introduced by Burdick [22], together with some topological notions which are of paramount importance for the global kinematics of redundant manipulators. These notions are: *c-bundles*, *w-sheets*, *Jacobian surfaces*, and *co-regular surfaces*. Although these notions were originally introduced for robots with serial architecture, the concept of self-motion manifold as the solution of the inverse kinematic problem can be naturally extended to redundant manipulators with other architectures (i.e., parallel or hybrid, like the HyReCRo robot) [123]. Next, we will introduce these notions with an example.

Consider a hypothetical redundant robot with joint coordinates (θ_1, θ_2) and task variable t_1. Therefore, its degree of redundancy is 1, which means that its self-motion manifolds are curves. Assume that its configuration space (i.e., the set of triplets $[t_1, \theta_1, \theta_2]$ that satisfy the kinematic constraints of the robot) is the surface represented in Figure 7.2a. Seven task-space points $\{a, b, c, d, e, f, g\}$ are identified in this figure.

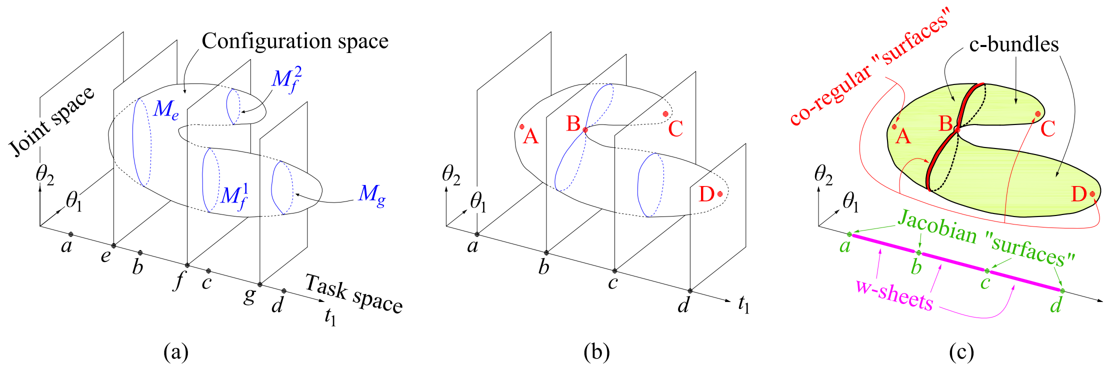


Figure 7.2: (a) Self-motion manifolds for task points $\{e, f, g\}$ (in blue). (b) Singularities (in red). (c) C-bundles (in yellow), co-regular "surfaces" (in red), Jacobian "surfaces" (in green), and w-sheets (in magenta). Note: in this example, Jacobian and co-regular "surfaces" actually are points and curves.

Let M_e denote the curve obtained when intersecting the configuration space with plane $t_1 = e$ (with $a < e < b$, Figure 7.2a). M_e is a 1-dimensional (self-motion) manifold such that all its points are mapped to e when projected to the task space (e is the image of M_e under this projection map). Therefore, M_e is the preimage or solution of the inverse kinematic problem for task point e. Similarly, the intersection between the configuration space and plane $t_1 = f$ (with $b < f < c$) has two connected components M_f^1 and M_f^2, which are the preimage manifolds for task point f. Usually, self-motion manifolds are projected to and visualized in the joint space of $\boldsymbol{\theta}$ coordinates, but for the following analysis it will be more convenient to visualize them on the configuration space, as in Figure 7.2.

Let $\mathbf{J}$ denote the Jacobian matrix that maps joint velocities to task velocities (i.e., $\dot{\mathbf{t}} = \mathbf{J}\dot{\boldsymbol{\theta}}$). The points of the configuration space at which $\mathbf{J}$ is not full rank are singularities. In the example of Figure 7.2b, singularities are the points {A, B, C, D} at which the plane tangent to the configuration space is perpendicular to axis

t_1. The projections of singularities on the task space are *Jacobian surfaces*, which divide the workspace into disjoint regions called *w-sheets*. In the example of Figure 7.2c, the workspace is segment ad (i.e., the projection of the whole configuration space on the task space), which is divided into three w-sheets by points $\{a, b, c, d\}$ (which are Jacobian "surfaces" in this example). The preimages of Jacobian surfaces are *co-regular surfaces*, which divide the configuration space into disjoint components called *c-bundles*, such that all self-motion manifolds on a given c-bundle share the same topology.

Jacobian surfaces, which divide the workspace into several w-sheets, can be either motion barriers or traversable singularities [19]. At the same time, barriers can be (exterior) boundaries or interior barriers of the workspace. Unlike traversable singularities, barriers may impede the motion of the robot across them, depending on the configuration of the robot when approaching the barriers. To determine whether a Jacobian surface is a traversable singularity or a barrier, one can analyze the changes in the topology of self-motion manifolds when crossing it, and this will be the essence of the method proposed in this chapter to determine the barriers induced by collision constraints and joint limits.

Next, in order to illustrate the idea behind the proposed method, the Jacobian surfaces of the example of Figure 7.2 will be classified into barriers or traversable singularities based on the analysis of the changes of topology of self-motion manifolds. Assume first that the task coordinates of the robot lie at task point e between a and b. The concrete configuration of the robot will lie somewhere on the self-motion manifold M_e corresponding to task point e. When task coordinate t_1 crosses point b while traveling from e to f, self-motion manifold M_e splits into two manifolds M_f^1 and M_f^2 after crossing the (8-shaped) co-regular surface which is the preimage of task point b. Note that, regardless of where the concrete configuration of the robot lies before crossing b, the robot will successfully cross point b since its configuration will lie on any of the two manifolds M_f^1 or M_f^2 into which M_e splits after crossing this Jacobian surface. Therefore, b is a traversable singularity.

Now imagine that the robot, starting at task point f, tries to cross Jacobian surface c to reach point g. If the configuration of the robot lies on manifold M_f^1 when approaching c, the robot will successfully cross this Jacobian surface since manifold M_f^1 will continuously deform until it transforms into M_g, which is the preimage manifold of task point g. However, the robot will be unable to cross c if its configuration originally lies on manifold M_f^2, since this manifold progressively reduces its size when approaching task point c, until it eventually shrinks to singularity C and then vanishes. Thus, c is an interior barrier of the workspace, which will impede the motion of the robot if its configuration lies on the c-bundle adjacent to singularity C (see Figure 7.2c).

A similar analysis can be repeated when trying to cross task points a and d in the negative and positive directions of axis t_1, respectively. In these cases, self-motion manifolds become progressively smaller until they degenerate into singularities A and D, and eventually vanish.

These examples illustrate the key property that will be exploited in this chapter to obtain the barriers of the workspace of redundant robots under joint limits and collision constraints: *when self-motion manifolds vanish, workspace barriers occur (either boundaries or interior barriers)*.

Note that, in the previous example, all self-motion manifolds vanish after degenerating into point singularities. However, it is possible to find examples of barriers at which a positive-dimensional (everywhere singular) manifold suddenly vanishes without shrinking to a point first. This is what occurs at the boundary of the workspace of the positioning arm of the 8-DOF AAI robot studied in [100].

Finally, it is also important to note that the relationship between workspace barriers and vanishing self-motion manifolds can be indirectly identified in earlier research works. In particular, Merlet et al. [118] proposed a geometrical method to obtain the reachable workspace of the planar 3R$\underline{\text{P}}$R parallel robot. Studying the reachable workspace of this robot is equivalent to regarding the robot as a redundant manipulator, since this workspace contains the planar positions ($m = 2$ task variables) that can be attained by the end-effector, which is controlled by $n = 3$ linear actuators. At each point of this workspace, the admissible values for the orientation ϕ of the end-effector lie in a finite number of disjoint intervals in $[0, 2\pi]$. In some cases, the aforementioned geometrical method yields arcs of sextic curves (called *separating arcs*) inside the workspace. Merlet et al. [118] noted that, when crossing such arcs, one of the mentioned disjoint intervals for angle ϕ vanishes, and the robot cannot cross these arcs if its orientation belongs to the vanishing interval. Actually, these separating arcs are interior barriers of the workspace, and the vanishing of these disjoint intervals implies the vanishing of self-motion manifolds. This is so because the orientation ϕ can be used to parameterize the self-motion manifolds of the 3R$\underline{\text{P}}$R robot, which are curves in the joint space of that robot.

7.1.2 Introducing Kinematic Constraints

The only kinematic constraints that we have considered in the previous subsection are those that constrain joint and task coordinates to lie on the configuration space of the robot (i.e., the surface represented in Figure 7.2). In the present subsection, we will analyze what happens when introducing additional kinematic constraints that can be modeled as forbidden regions of the configuration space. Typical examples of these constraints are joint limits and the prohibition of collisions.

Lück and Lee [100] studied how the global kinematics of redundant robots is altered when introducing kinematic constraints that can be modeled as forbidden regions $\mathcal{K}$ of the configuration space. Tangency points between the boundaries $\partial\mathcal{K}$ of these regions and self-motion manifolds are *semi-singularities* [102], which are unidirectional singularities at which the robot is unable to generate task velocities in some direction, being able to generate them in the opposite direction. Like singularities, semi-singularities generate new Jacobian and co-regular surfaces, which can drastically modify the distribution of w-sheets and c-bundles. Likewise, Jacobian surfaces generated by semi-singularities can be traversable singularities, boundaries, or interior barriers of the workspace. If we can find semi-singularities, then we can obtain the Jacobian surfaces generated by them, and classify these surfaces in a similar way to the example of Figure 7.2.

There are two approaches to finding semi-singularities [101]: the tangency approach (which exploits the fact that semi-singularities occur when self-motion manifolds and $\partial\mathcal{K}$ are tangent) and the Jacobian column approach (which analyzes the range space of a linear mapping with constrained domain). Both approaches require searching for semi-singularities along the boundaries $\partial\mathcal{K}$ of the kinematic constraints, for example, via steepest descent [101]. Although this search is feasible when the kinematic constraints are joint limits (in that case $\partial\mathcal{K}$ consists of known planes in the configuration space), in the case of more complex kinematic constraints (e.g., collisions between arbitrarily-shaped bodies with arbitrary relative pose) the boundary $\partial\mathcal{K}$ is not known in advance, it has a much more complicated shape, and lacks an analytic description that can be used in the computations. Thus, the search along $\partial\mathcal{K}$ under these conditions is unfeasible, and it is necessary to find an alternative approach able to take into account such complex kinematic constraints in the calculation of workspace barriers. As argued at the beginning of the present section 7.1, the best way to cope with complex kinematic constraints is using a sampling method, since in that case one only needs to check whether the sampled configurations satisfy the constraints or not, *after sampling them*.

In this context, it is worth mentioning the sampling method proposed by DeMers and Kreutz-Delgado [47] to identify w-sheets and c-bundles in serial redundant robots. Their method begins by randomly and densely sampling joint coordinate points $\boldsymbol{\theta}_i$ from the joint space, generating the corresponding task positions $\mathbf{t}_i$ by solving the forward kinematics for each $\boldsymbol{\theta}_i$. Then, the task space is swept following some pattern, testing some task query points $\mathbf{t}_q$. For each query point $\mathbf{t}_q$, its preimage self-motion manifolds are approximated by the set $\mathcal{M}(\mathbf{t}_q)$ of all sampled joint coordinate points $\boldsymbol{\theta}_i$ that have generated task positions $\mathbf{t}_i$ near $\mathbf{t}_q$ when solving the forward kinematics during the previous randomly-sampling stage. Then, all joint samples contained in $\mathcal{M}(\mathbf{t}_q)$ are clustered by solving a Minimum Spanning Tree problem [46], which allows for the identification of disjoint self-motion manifolds. DeMers [48] used this method to identify the Jacobian surfaces that separate neighboring w-sheets in the workspace of serial 3R redundant robots, in the absence of kinematic constraints (although the method may easily accommodate such constraints since it is a sampling method). To this end, radial coordinate ρ (Figure 7.1) was swept and self-motion manifolds were estimated at some discrete values of ρ, following the procedure described above. Then, Jacobian surfaces were identified with the values of ρ at which the number of disjoint self-motion manifolds identified by the clustering method changed.

In the absence of kinematic constraints, the changes in the number of disjoint self-motion manifolds can be used to reliably identify Jacobian surfaces [22]. However, the focus of the present chapter is on barriers, and the changes in the number of manifolds when moving along the task space do not necessarily imply the vanishing of manifolds (i.e., the existence of barriers). For example, the number of disjoint manifolds can decrease due to fusions between some of them, without any manifold necessarily vanishing (e.g., this happens when crossing point b in the negative direction of axis t_1 in Figure 7.2). Or, on the contrary, some manifolds may vanish (which decreases the number of manifolds) at the same time that other manifolds split (which increases the number of manifolds), in which case the net change in the number of disjoint manifolds may not reflect the vanishing of manifolds, which would mask the occurrence of barriers (see the example of Figure 7.4, discussed in subsection 7.2.3). Hence, workspace barriers cannot be reliably detected by identifying changes in the number of disjoint self-motion manifolds; it is necessary to robustly identify the vanishing of these manifolds. To this end, in the next section we propose a new sampling method that monitors the transformations suffered by self-motion manifolds when moving along the task space, with the purpose of detecting when any of such manifolds vanishes.

7.2 A Method for Obtaining Interior Barriers Under Collision Constraints

This section proposes a new sampling method to obtain the barriers (both interior barriers and external boundaries) of the workspace of redundant robot manipulators, considering joint limits and collision constraints (prohibition of self-collisions or collisions with obstacles of the environment). The method presented here is similar to the aforementioned method proposed by DeMers and Kreutz-Delgado [47], in the sense that our method also involves sampling and clustering joint coordinate points $\boldsymbol{\theta}_i$ in the joint space. However, there are three main differences between the previous method [47] and our proposed method:

1. *Different sampling approach:* instead of approximating the set $\mathcal{M}(\mathbf{t}_q)$ (set of joint coordinate vectors that place the task variables at a given task point $\mathbf{t}_q$) by a collection of randomly sampled joint vectors that map (under the forward kinematics) sufficiently near $\mathbf{t}_q$, our proposed method directly solves the inverse kinematics to densely sample the self-motion manifolds at $\mathbf{t}_q$.

2. *Different clustering approach:* instead of clustering the points of the set $\mathcal{M}(\mathbf{t}_q)$ by solving a Minimum Spanning Tree in a graph, our method clusters these points using kd-trees, identifying in this way the disjoint components of the self-motion manifolds.

3. *Additional matching stage:* our method includes an additional third stage, which consists in matching neighboring self-motion manifolds to identify how manifolds transform when varying the task variables, and detect in this way the vanishing (or creation) of self-motion manifolds. This matching takes advantage of the very kd-trees used in the previous clustering stage, "recycling" in this way the previous calculations.

The following subsections 7.2.1, 7.2.2 and 7.2.3 describe in detail the three main stages of our method: *sampling*, *clustering* and *matching*. Then, subsection 7.2.4 describes the complete method, which combines these three stages.

7.2.1 Sampling Self-motion Manifolds

The first step to determine how self-motion manifolds transform when varying the task variables along the task space, consists in calculating such manifolds at each task point $\mathbf{t}$. This subsection presents a method to densely sample the self-motion manifolds at a given task point, such that sufficiently dense discrete approximations of these manifolds can be computed, to effectively use them in the following stages of the proposed method (clustering and matching).

First, consider a hypothetical redundant robot with two joint coordinates (θ_1, θ_2) and task variable t_1. The degree of redundancy is one; therefore, its self-motion manifolds are curves in plane (θ_1, θ_2). Assume that, for a given value of t_1, these 1-dimensional self-motion manifolds are defined as the solution sets of the following scalar equation (see Figure 7.3a):

$$f(\theta_1, \theta_2, t_1) = 0 \tag{7.1}$$

which does not include additional kinematic constraints like joint limits or prohibition of collisions. Algorithm 4 proposes a simple procedure (called SWEEP_THETA1) to sample the self-motion manifolds and obtain a discrete approximation $\mathcal{M}(t_1)$ of these manifolds at task point t_1.

Algorithm 4 Sampling the manifolds defined by Equation (7.1)

```
1:  procedure SWEEP_THETA1($\theta^{lim}$, $N_s$)
        Inputs:
            • $\theta^{lim} = \{\theta_1^{min}, \theta_1^{max}, \theta_2^{min}, \theta_2^{max}\}$ // joint limits
            • $N_s$ // number of discrete values for $\theta_1$
        Output:
            • $\mathcal{M}(t_1)$ // Discrete approximation of self-motion
            manifolds at task point $t_1$
2:      $\mathcal{M}(t_1) \leftarrow \emptyset$ // initialized as the empty set
3:      $\Delta\theta_1 \leftarrow \frac{\theta_1^{max} - \theta_1^{min}}{N_s - 1}$ // step along $\theta_1$ axis
4:      for $k = 0, \ldots, N_s - 1$ do
5:          $\theta_1^k \leftarrow \theta_1^{min} + \Delta\theta_1 \cdot k$
6:          $\theta_1 \leftarrow \theta_1^k$
7:          Solve $\theta_2$ from Equation (7.1), obtaining a finite set
            of solutions: $\{\theta_2^{k,1}, \theta_2^{k,2}, \ldots\}$
8:          for all solutions $\theta_2^{k,l}$ do
9:              if $\theta_2^{min} \leq \theta_2^{k,l} \leq \theta_2^{max}$ AND $(\theta_1^k, \theta_2^{k,l}, t_1)$
                satisfies collision constraints then
10:                 Add point $(\theta_1^k, \theta_2^{k,l})$ to $\mathcal{M}(t_1)$
11:     return $\mathcal{M}(t_1)$
```

Algorithm 4 is illustrated in Figure 7.3b, and it proceeds as follows. First, the interval $[\theta_1^{min}, \theta_1^{max}]$ between the joint limits is discretized into a grid of N_s points.

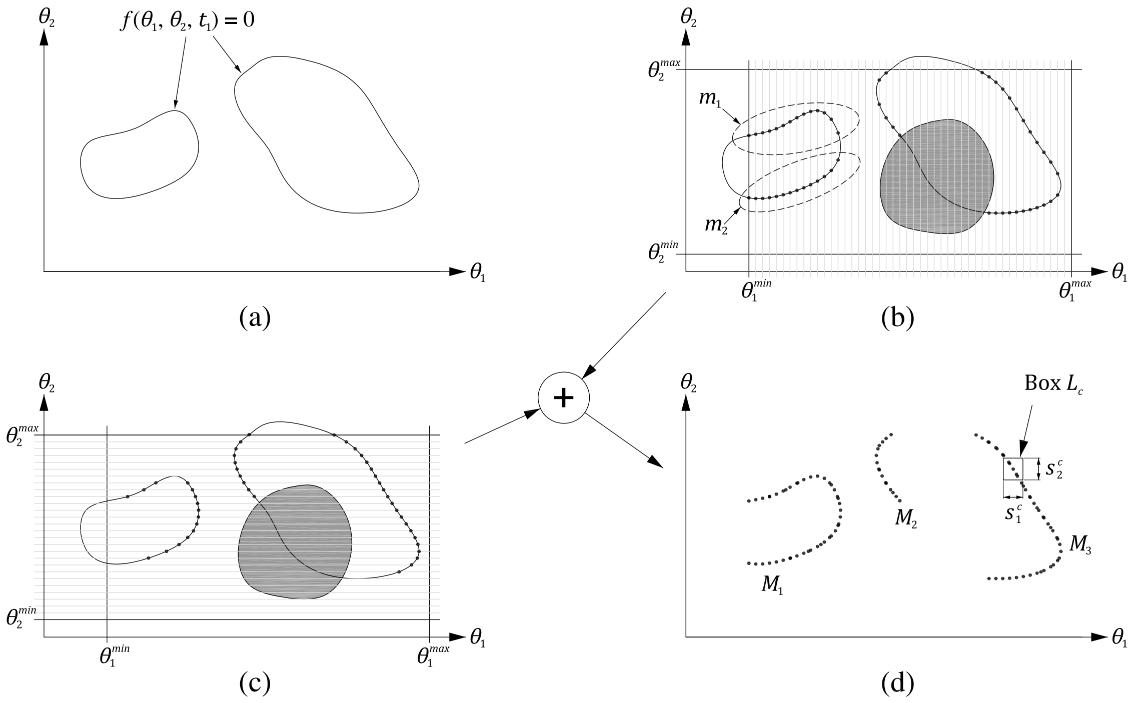


Figure 7.3: A method to densely sample one-dimensional self-motion manifolds, considering joint limits and collision constraints. In this example, the forbidden shaded region shown in (b) and (c) may represent configurations for which collisions occur.

For each point θ_1^k of this grid, θ_2 is solved from Equation (7.1), obtaining a finite set of different solutions: $\{\theta_2^{k,1}, \theta_2^{k,2}, \ldots\}$. These solutions correspond to the intersection points between the curves defined by Equation (7.1) and the vertical line $\theta_1 = \theta_1^k$ (see Figure 7.3b). Next (line 9 of Algorithm 4), it is checked if each obtained intersection point $\theta_2^{k,l}$ satisfies both the joint limits for θ_2 and collision constraints (if any). If that is the case, then point $(\theta_1^k, \theta_2^{k,l})$ is accepted and added to the set $\mathcal{M}(t_1)$, which constitutes a discrete approximation of the self-motion manifolds at task point t_1.

After Algorithm 4 finishes, the set $\mathcal{M}(t_1)$ can be viewed as a point cloud in the joint space, which approximates the self-motion manifolds at t_1. For subsequent stages of the proposed method, we need this point cloud to be sufficiently dense. However, in general this is not guaranteed by Algorithm 4: indeed, as Figure 7.3b shows, the density of sampled points is lower near the points where the self-motion manifolds have vertical tangent. These are the singular points of the parameterization of the self-motion manifolds which uses the coordinate θ_1 as parameter. Due to these low-density regions, a clustering method may erroneously identify two disjoint self-motion manifolds m_1 and m_2 (see Figure 7.3b) where there is actually a single connected self-motion manifold M_1 (see Figure 7.3d).

Note that the problematic low-density regions shown in Figure 7.3b can be sampled more densely if we sweep the angle θ_2 and compute θ_1 (instead of sweeping θ_1 and computing θ_2), which consists in parameterizing the self-motion manifolds via the coordinate θ_2. This can be achieved by swapping the roles of θ_1 and θ_2 in Algorithm

4, and the result is equivalent to intersecting the self-motion manifolds with a bundle of horizontal lines (see Figure 7.3c). By doing this, the poorly-defined low-density regions obtained previously (when sweeping θ_1) appear now more dense and better defined (Figure 7.3c). However, this improvement is at the expense of the density of samples near the points where the self-motion manifolds have horizontal tangent, which are now poorly defined. It is clear that none of these two parameterizations can be used alone to compute dense discrete approximations of the self-motion manifolds. Nevertheless, if the point clouds of Figures 7.3b and 7.3c are combined, we obtain a dense discretization of the self-motion manifolds in plane (θ_1, θ_2), which has no poorly-defined regions, since the areas where one parameterization fails are properly covered by the other parameterization (see Figure 7.3d). Note that, in the dense point cloud of Figure 7.3d, any appropriate clustering method would correctly identify three disjoint self-motion manifolds M_1, M_2 and M_3.

The previous example considered one-dimensional self-motion manifolds in a two-dimensional joint space. In the general case, the dimensions of the task space, joint space and self-motion manifolds are m, n and r, respectively (with $n > m$ and $r > 0$)[1]. Let $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^T$ and $\mathbf{t} = [t_1, \ldots, t_m]^T$ denote the vectors of actuated joint coordinates and task variables, respectively. Also, let $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_r]^T$ denote a vector of r passive variables which are neither classified as joint coordinates nor as task variables. Assume that, for a given task point $\mathbf{t}$, the self-motion manifolds at $\mathbf{t}$ are defined as the sets of values of $\boldsymbol{\theta}$ that satisfy the following system of n scalar equations for some value of $\boldsymbol{\psi}$:

$$\mathbf{f}(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\psi}) = \mathbf{0}_{n\times 1} \tag{7.2}$$

where $\mathbf{f} = [f_1(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\psi}), \ldots, f_n(\boldsymbol{\theta}, \mathbf{t}, \boldsymbol{\psi})]^T$. It is assumed that Equation (7.2) does not include kinematic constraints of inequality type (e.g., joint limits or mechanical interferences).

Let us generalize now the method illustrated in Figure 7.3 to the general case. To densely sample r-dimensional self-motion manifolds in an n-dimensional ambient (joint) space, we should solve Equation (7.2) when sweeping each of the $\binom{n}{r}$ different combinations of the n joint coordinates taken r at a time. In other words, we should generate or "plot" the self-motion manifolds by sweeping all their possible parameterizations that use as parameters r of the n coordinates of the ambient joint space in which these manifolds "live". By sweeping all these possible parameterizations, the self-motion manifolds can be approximated by sufficiently dense point clouds in the joint space. Such point clouds are free from the undesirable low-density regions because the regions where one particular parameterization is almost singular (which yields low-density regions) can be properly covered by other parameterizations.

Algorithm 5 summarizes this procedure. First, the interval between the joint limits of each joint coordinate θ_j is discretized and approximated by a set $\mathcal{I}_j$ of N_s^j

[1]Usually, we have $r = n - m$. However, this is not true if additional equality constraints are considered. For example, this happens when computing planar slices of higher-dimensional workspaces (see the examples of Section 7.3).

equally spaced points ($j = 1, \dots, n$) (lines 4-6). Then, for each possible combination p of r different joint coordinates (line 7), the following operations are performed. A regular grid $\mathcal{G}$ is defined in the r-dimensional subspace of the joint coordinates $\theta_j \in p$. This grid is obtained as the Cartesian product of the sets $\mathcal{I}_j$, for $\theta_j \in p$ (line 9). Next, for each point $\mathbf{g}$ of the grid $\mathcal{G}$, Equation (7.2) is solved to obtain $\boldsymbol{\psi}$ and the remaining $n - r$ joint coordinates (i.e., those $\theta_j \notin p$). Generally, solving Equation (7.2) will yield several solutions. For each of these solutions, it is checked if all kinematic constraints are satisfied (e.g., joint limits, avoidance of self-collisions, etc.). If that is the case, then all the joint coordinates θ_j of that solution ($j = 1, \dots, n$) are grouped in the n-dimensional vector $\boldsymbol{\theta}$, which is added to the set $\mathcal{M}(\mathbf{t})$. After Algorithm 5 finishes, $\mathcal{M}(\mathbf{t})$ represents a point cloud in the n-dimensional joint space, which densely approximates the self-motion manifolds at task point $\mathbf{t}$.

Algorithm 5 A combinatorial sweeping procedure to densely sample the manifolds described by Equation (7.2)

```
 1: procedure COMBINATORIAL_SWEEPING($\theta^{lim}$, $N_s$)
        Inputs:
            • $\theta^{lim} = \{\theta_1^{min}, \theta_1^{max}, ..., \theta_n^{min}, \theta_n^{max}\}$ // joint limits
            • $N_s = \{N_s^1, ..., N_s^n\}$ // No. of points for each axis $\theta_j$
        Output:
            • $\mathcal{M}(\mathbf{t})$ // Discrete and dense approximation of
            self-motion manifolds at task point $\mathbf{t}$
 2:     $\mathcal{M}(\mathbf{t}) \leftarrow \emptyset$ // initialized as the empty set
 3:     $\mathcal{Q} \leftarrow \{\theta_1, \dots, \theta_n\}$ // set of all the $n$ joint coordinates
 4:     for $j = 1, \dots, n$ do
 5:         $\Delta\theta_j \leftarrow \frac{\theta_j^{max} - \theta_j^{min}}{N_s^j - 1}$ // step along $\theta_j$ axis
 6:         $\mathcal{I}_j \leftarrow \{\, \theta_j^k : \theta_j^k = \theta_j^{min} + \Delta\theta_j \cdot k,$
                        $k = 0, \dots, N_s^j - 1\}$
 7:     for each subset $p \subset \mathcal{Q}$ such that $|p| = r$ do
 8:         $q \leftarrow Q \setminus p$
 9:         $\mathcal{G} \leftarrow \prod_{\theta_j \in p} \mathcal{I}_j$
10:         for each point $\mathbf{g} \in \mathcal{G}$ do
11:             Solve $\boldsymbol{\psi}$ and $\theta_j \in q$ from Equation (7.2)
12:             for each obtained solution do
13:                 if all kinematic constraints are satisfied then
14:                     Add $\boldsymbol{\theta}$ to $\mathcal{M}(\mathbf{t})$
15:     return $\mathcal{M}(\mathbf{t})$
```

The number of calculations to be performed in Algorithm 5 grows linearly with the number $\binom{n}{r}$ of combinations, and exponentially with the dimension r of the self-motion manifolds (due to the r-dimensional grid $\mathcal{G}$ obtained as a Cartesian product in line 9). Hence, if the dimension r of the self-motion manifolds is too large, the time necessary to execute Algorithm 5 may be prohibitive, depending on the number N_s^j of points used to discretize each axis of the joint space. However, for the most typical situations (excluding hyper-redundant manipulators [10]), the degree of redundancy is usually not too high: in practice, r usually equals 1 or 2, or at most 3. When $r = 1$, Algorithm 5 reduces to sweeping a 1-dimensional regular grid along each coordinate axis of the joint space, as in Figure 7.3. When $r = 2$, Algorithm 5 consists in sweeping a 2-dimensional regular grid along each coordinate plane of the joint space. As we will show throughout the examples of Section 7.3, in these cases of 1- and 2-dimensional self-motion manifolds, Algorithm 5 can be executed in quite reasonable times obtaining sufficiently accurate results.

The proposed Algorithm 5 requires solving Equation (7.2) for $\boldsymbol{\psi}$ and $n - r$ unknown joint coordinates. In essence, solving Equation (7.2) consists in solving the inverse kinematic problem of the redundant robot, assuming that r joint coordinates are known (so that the problem is no longer underdetermined and yields a finite set of solutions). For this reason, it would be convenient if the inverse kinematic problem could be solved relatively easily, preferably in closed form. Since the inverse kinematic problem is easier to solve for parallel robots (because each scalar equation typically involves only one unknown joint coordinate, which can be obtained by solving a quadratic equation - see the examples of Section 7.3), the proposed method is more suitable for manipulators with parallel architecture. However, in principle it can be applied to robots with any architecture, as long as Equation (7.2) can be easily solved. Note that parallel robots are precisely the type of robots that can benefit the most from the proposed method, since it is precisely this kind of robots that suffer the most from collision constraints due to their closed-chain architecture, which often results in different potentially-colliding links moving very closely to each other.

As we will explain in the examples of Section 7.3, in these examples Equation (7.2) will be solved via elimination, which reduces the system to a univariate polynomial equation that can be solved using root-finding techniques. Elimination methods are the preferred ones in the context of Algorithm 5 since they are simple, very fast [note that Equation (7.2) must be solved many times during the execution of this algorithm], and find all solutions (although complex and spurious solutions must be discarded). However, elimination methods are not always feasible or practical, especially when the degree of the mentioned univariate polynomial grows too large (e.g., a few tens). In these cases, it may be more convenient to solve Equation (7.2) using other general techniques for position analysis of manipulators, such as polynomial continuation [17] or branch-and-prune methods [154], although these techniques may increase the computation time of Algorithm 5.

7.2.2 Clustering Self-motion Manifolds

After the previous stage has finished, we have obtained a set $\mathcal{M}(\mathbf{t})$ of unordered points which constitutes a dense point cloud that approximates the self-motion manifolds in the joint space. The next step consists in clustering these points to identify the different disjoint self-motion manifolds at task point $\mathbf{t}$ (e.g., identifying the three manifolds M_1, M_2 and M_3 in Figure 7.3d). The outcomes of this clustering will be the number $\mathcal{N}(\mathbf{t})$

of identified disjoint self-motion manifolds and, for each point $\boldsymbol{\theta} \in \mathcal{M}(\mathbf{t})$, an integer $\mathcal{B}(\boldsymbol{\theta}, \mathbf{t}) \in \{0, \ldots, \mathcal{N}(\mathbf{t}) - 1\}$ that identifies the manifold to which $\boldsymbol{\theta}$ belongs.

To cluster the self-motion manifolds, a kd-tree $\mathcal{T}(\mathbf{t})$ is constructed first from the point cloud $\mathcal{M}(\mathbf{t})$. A kd-tree is a data structure useful to partition and organize point clouds in any dimension. The reader is referred to [155] for further information about the construction and use of kd-trees. Constructing a kd-tree from a point cloud permits performing very efficiently and quickly some operations, such as obtaining which points of the point cloud are closest to another query point, even if the query point does not belong to the constructed kd-tree. Next, this property of kd-trees will be used for identifying the disjoint self-motion manifolds. In the following subsection 7.2.3, we will use this property to determine how each self-motion manifold transforms when slightly varying the task variables.

Procedure Cluster of Algorithm 6 clusters all points of the set $\mathcal{M}(\mathbf{t})$ into disjoint self-motion manifolds. First, a set $\mathcal{P}$ of pending points is defined, which contains the points of the set $\mathcal{M}(\mathbf{t})$ that still have to be classified into some self-motion manifold. Initially, $\mathcal{P}$ coincides with the whole set $\mathcal{M}(\mathbf{t})$ (line 2). Next, an integer variable k is set to zero (line 3). k equals the number of disjoint self-motion manifolds identified by the algorithm at each iteration. Then, the recursive procedure described next is repeated until there are no more points to classify, i.e., until $\mathcal{P}$ is empty (line 4).

First, the first point $\boldsymbol{\theta}$ of the set $\mathcal{P}$ is selected (line 5), and it is passed to the sub-routine Classify$(\cdot)$ (line 6), whose definition starts at line 10. Inside this sub-routine, the current point (denoted by $\mathbf{x}$ inside the sub-routine) is classified into the disjoint manifold identified by the current value of the integer k, and $\mathbf{x}$ is removed from the set $\mathcal{P}$ of points to be classified (lines 11 and 12). Then, the kd-tree is searched to obtain the set $\mathcal{U}(\mathbf{x})$ of points *sufficiently close* to the current point $\mathbf{x}$ (line 13). Next, the unclassified neighbors of $\mathbf{x}$ are passed to the function Classify$(\cdot)$ (lines 14-16), to classify them into the same manifold as the current point (it is assumed that they should belong to the same manifold due to proximity). Note that this is a recursive algorithm: when passing each neighbor $\mathbf{y} \in \mathcal{U}(\mathbf{x})$ to the sub-routine Classify$(\cdot)$, the set $\mathcal{U}(\mathbf{y})$ of neighbors of $\mathbf{y}$ will also be classified into the same manifold. This recursive process will repeat until it completely covers the connected manifold to which the original point $\boldsymbol{\theta}$ belongs, i.e., until all the neighboring points of each point of the considered manifold are assigned the same label k.

After classifying all points of the current connected manifold, k is increased by one unit, indicating that a connected self-motion manifold has been completely identified (line 7). Then, the previous procedure is repeated until all points of $\mathcal{M}(\mathbf{t})$ have been classified into any of the disjoint self-motion manifolds, and no new manifolds are identified. Upon the complete execution of Algorithm 6, $\mathcal{N}(\mathbf{t})$ stores the number of identified disjoint manifolds (line 8), and $\mathcal{B}(\boldsymbol{\theta}, \mathbf{t})$ stores an integer that identifies the manifold to which $\boldsymbol{\theta}$ belongs, $\forall \boldsymbol{\theta} \in \mathcal{M}(\mathbf{t})$.

In Algorithm 6, we classify the set $\mathcal{U}(\mathbf{x})$ of points sufficiently close to a given point $\mathbf{x}$ into the same manifold to which $\mathbf{x}$ belongs. By *sufficiently close* points, we

Algorithm 6 Identifying disjoint self-motion manifolds

```
 1: procedure CLUSTER($\mathcal{M}(\mathbf{t})$, $\mathcal{T}(\mathbf{t})$)
        Inputs:
            • $\mathcal{M}(\mathbf{t})$ // point cloud to be clustered
            • $\mathcal{T}(\mathbf{t})$ // kd-tree of $\mathcal{M}(\mathbf{t})$
        Outputs:
            • $\mathcal{N}(\mathbf{t})$ // number of identified disjoint manifolds
            • $\mathcal{B}(\boldsymbol{\theta}, \mathbf{t})$ // integer identifier of the disjoint manifold
            to which each $\boldsymbol{\theta}$ belongs, $\forall \boldsymbol{\theta} \in \mathcal{M}(\mathbf{t})$
 2:     $\mathcal{P} \leftarrow \mathcal{M}(\mathbf{t})$ // Initialize the set of points to be classified
 3:     $k \leftarrow 0$ // Initialize the label that identifies each manifold
 4:     while $\mathcal{P}$ is not empty do
 5:         $\boldsymbol{\theta} \leftarrow$ first point of the set $\mathcal{P}$
 6:         CLASSIFY($\boldsymbol{\theta}$)
 7:         $k \leftarrow k + 1$
 8:     $\mathcal{N}(\mathbf{t}) \leftarrow k$
 9:     return $\mathcal{N}(\mathbf{t})$, $\mathcal{B}(\cdot, \mathbf{t})$

10: procedure CLASSIFY($\mathbf{x}$)
        Input:
            • $\mathbf{x} \in \mathcal{M}(\mathbf{t})$ // joint-space point to be classified
            into a disjoint self-motion manifold
        Output:
            • Classifies $\mathbf{x}$ and its neighbors into one of the
            identified disjoint self-motion manifolds
11:     $\mathcal{B}(\mathbf{x}, \mathbf{t}) \leftarrow k$
12:     Remove $\mathbf{x}$ from $\mathcal{P}$
13:     $\mathcal{U}(\mathbf{x}) \leftarrow$ set of points of $\mathcal{T}(\mathbf{t})$ sufficiently close to $\mathbf{x}$
14:     for all $\mathbf{y} \in \mathcal{U}(\mathbf{x})$ do
15:         if $\mathbf{y} \in \mathcal{P}$ then
16:             CLASSIFY($\mathbf{y}$)
17:     return
```

mean the points of the set $\mathcal{M}(\mathbf{t})$ which fall inside an n-dimensional box L_c centered at $\mathbf{x}$, with its main axes aligned with the coordinate axes of the joint space, and with appropriate size s_j^c along each axis θ_j $(j = 1, \ldots, n)$ (see Figure 7.3d for an example with $n = 2$). Obtaining the points of the point cloud $\mathcal{M}(\mathbf{t})$ that fall inside such a box is a task that can be performed very efficiently by searching the previously constructed kd-tree $\mathcal{T}(\mathbf{t})$.

The outcome of the clustering will depend on the choice of the size of the box L_c. On the one hand, if the box is too small, Algorithm 6 may erroneously identify different parts of the same connected manifold as different disjoint manifolds (in the limit, a zero-size box would identify each individual point of $\mathcal{M}(\mathbf{t})$ as a different self-motion manifold). On the other hand, too large a box may erroneously identify as a single connected manifold different manifolds which actually are disjoint. Moreover, if the sampling of the self-motion manifolds is not sufficiently dense, we will have low-density regions in a single manifold that may be erroneously detected as separations between different disjoint manifolds (e.g., manifolds m_1 and m_2 in Figure 7.3b), but this can be avoided if Algorithm 5 presented in previous subsection 7.2.1 is used to densely sample the self-motion manifolds.

If the self-motion manifolds have been densely sampled using Algorithm 5, then a good choice for the size of the box along each axis is $s_j^c = \sigma_c \cdot 2 \cdot \Delta\theta_j$ $(j = 1, \ldots, n)$, where $\Delta\theta_j$ is the step used to discretize the θ_j axis (line 5 of Algorithm 5). This is because the distance along the θ_j axis between two neighboring samples will be $\Delta\theta_j$ at most. The coefficient $\sigma_c \geq 1$ is the *clustering factor*, and its purpose is to increase the size of box L_c.

7.2.3 Matching Self-motion Manifolds

This subsection presents a method to determine how self-motion manifolds transform when the task coordinates move from a point $\mathbf{t}_a$ to another sufficiently near neighboring point $\mathbf{t}_b$. It is assumed that, for both task points, self-motion manifolds have been densely sampled according to Algorithm 5, obtaining non-empty point clouds $\mathcal{M}(\mathbf{t}_a)$ and $\mathcal{M}(\mathbf{t}_b)$. It is also assumed that two kd-trees $\mathcal{T}(\mathbf{t}_a)$ and $\mathcal{T}(\mathbf{t}_b)$ have been constructed from these point clouds, and that the clustering Algorithm 6 has been applied. Therefore, at each task point $\mathbf{t}_i$ $(i \in \{a, b\})$, we know the number $\mathcal{N}(\mathbf{t}_i)$ of disjoint self-motion manifolds and the integer index $\mathcal{B}(\boldsymbol{\theta}, \mathbf{t}_i)$ of the manifold to which each $\boldsymbol{\theta} \in \mathcal{M}(\mathbf{t}_i)$ belongs.

Using the previous information, Algorithm 7 matches the self-motion manifolds at $\mathbf{t}_a$ with the manifolds at $\mathbf{t}_b$, to determine how the self-motion manifolds at $\mathbf{t}_a$ transform into the manifolds at $\mathbf{t}_b$. First, for each disjoint self-motion manifold m_a at $\mathbf{t}_a$ $(m_a = 0, \ldots, \mathcal{N}(\mathbf{t}_a) - 1)$, we define $\Omega(m_a)$ as the subset of the manifolds at $\mathbf{t}_b$ into which m_a transforms when the task variables move from $\mathbf{t}_a$ to $\mathbf{t}_b$. The set $\Omega(m_a)$ is initialized as the empty set, $\forall m_a$ (lines 2-3). Next, for each $\mathbf{x} \in \mathcal{M}(\mathbf{t}_a)$, the kd-tree $\mathcal{T}(\mathbf{t}_b)$ is searched to efficiently obtain the set $\mathcal{U}(\mathbf{x})$ of points of $\mathcal{M}(\mathbf{t}_b)$ which are *sufficiently close* to $\mathbf{x}$. Then, due to proximity, it can be assumed that the self-motion manifold $\mathcal{B}(\mathbf{x}, \mathbf{t}_a)$ to which $\mathbf{x}$ belongs transforms into the manifolds $\mathcal{B}(\mathbf{y}, \mathbf{t}_b)$ to which each $\mathbf{y} \in \mathcal{U}(\mathbf{x})$ belongs (lines 6-8). Note that this matching Algorithm 7 reuses the very same kd-trees that were constructed and utilized in the clustering Algorithm 6, i.e., it is not necessary to perform new calculations to generate new information to match the self-motion manifolds.

Algorithm 7 Matching self-motion manifolds at two neighboring task points $\mathbf{t}_a$ and $\mathbf{t}_b$

```
1: procedure MATCH($\mathcal{N}(\mathbf{t}_a)$, $\mathcal{M}(\mathbf{t}_a)$, $\mathcal{T}(\mathbf{t}_b)$, $\mathcal{B}(\cdot,\mathbf{t}_a)$, $\mathcal{B}(\cdot,\mathbf{t}_b)$)
       Inputs:
           • $\mathcal{N}(\mathbf{t}_a)$ // number of manifolds at task point $\mathbf{t}_a$
           • $\mathcal{M}(\mathbf{t}_a)$ // dense point cloud approximating
           the self-motion manifolds at task point $\mathbf{t}_a$
           • $\mathcal{T}(\mathbf{t}_b)$ // kd-tree of point cloud $\mathcal{M}(\mathbf{t}_b)$
           • $\mathcal{B}(\boldsymbol{\theta},\mathbf{t}_i)$ // integer index of the manifold at $\mathbf{t}_i$ to
           which each $\boldsymbol{\theta} \in \mathcal{M}(\mathbf{t}_i)$ belongs ($i \in \{a,b\}$)
       Output:
           • $\Omega(m_a)$ // integer identifiers of the disjoint
           manifolds at $\mathbf{t}_b$ into which each manifold
           $m_a$ at $\mathbf{t}_a$ transforms
2:     for $m_a = 0,\ldots,\mathcal{N}(\mathbf{t}_a)-1$ do
3:         $\Omega(m_a) \leftarrow \emptyset$
4:     for all $\mathbf{x} \in \mathcal{M}(\mathbf{t}_a)$ do
5:         $\mathcal{U}(\mathbf{x}) \leftarrow$ set of points of $\mathcal{T}(\mathbf{t}_b)$ sufficiently close to $\mathbf{x}$
6:         for all $\mathbf{y} \in \mathcal{U}(\mathbf{x})$ do
7:             if $\mathcal{B}(\mathbf{y},\mathbf{t}_b) \notin \Omega(\mathcal{B}(\mathbf{x},\mathbf{t}_a))$ then
8:                 Add $\mathcal{B}(\mathbf{y},\mathbf{t}_b)$ to $\Omega(\mathcal{B}(\mathbf{x},\mathbf{t}_a))$
9:     return $\Omega(\cdot)$
```

Figure 7.4 shows an example to illustrate the application of Algorithm 7. In this example, the self-motion manifolds at task point $\mathbf{t}_a$ are approximated by a cloud of filled circles, whereas the self-motion manifolds at task point $\mathbf{t}_b$ (which is sufficiently near to $\mathbf{t}_a$) are approximated by a cloud of empty circles. These dense point clouds have been obtained applying Algorithm 5. Assume that the clustering Algorithm 6 has identified two disjoint self-motion manifolds $\{0_a, 1_a\}$ at $\mathbf{t}_a$, and other two disjoint self-motion manifolds $\{0_b, 1_b\}$ at $\mathbf{t}_b$. Assuming that the task coordinates move from $\mathbf{t}_a$ to $\mathbf{t}_b$, Algorithm 7 would identify the following matching:

$$\Omega(0_a) = \emptyset \text{ (empty set)}, \quad \Omega(1_a) = \{0_b, 1_b\} \tag{7.3}$$

which means that manifold 0_a vanishes (since it does not transform into any manifold at $\mathbf{t}_b$), whereas manifold 1_a is split into manifolds 0_b and 1_b. If the task coordinates move backwards from $\mathbf{t}_b$ to $\mathbf{t}_a$, then swapping $\mathbf{t}_a$ and $\mathbf{t}_b$ in Algorithm 7 would yield the following matching:

$$\Omega(0_b) = \{1_a\}, \quad \Omega(1_b) = \{1_a\} \tag{7.4}$$

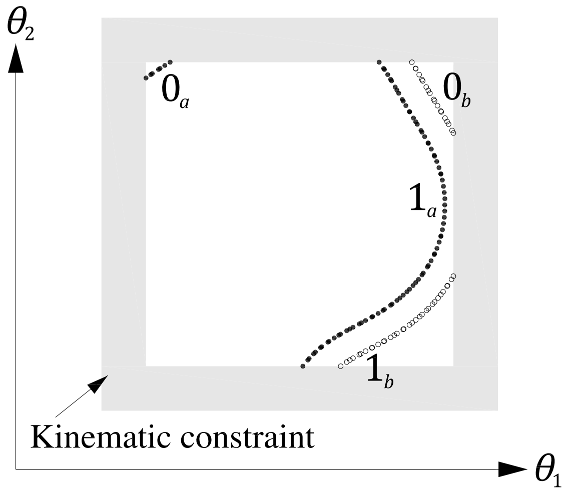


Figure 7.4: Matching self-motion manifolds $\{0_a, 1_a\}$ at $\mathbf{t}_a$ with self-motion manifolds $\{0_b, 1_b\}$ at $\mathbf{t}_b$.

which means that both manifolds 0_b and 1_b merge into manifold 1_a.

The previous example shows how Algorithm 7 can be used to identify the vanishing of self-motion manifolds when moving the task coordinates from an initial point $\mathbf{t}_a$ to a final point $\mathbf{t}_b$: a self-motion manifold vanishes if at least one of the sets $\Omega(m_a)$ is empty, for some disjoint manifold m_a at the initial point $\mathbf{t}_a$ [see the example in Equation (7.3)].

Note that the creation of self-motion manifolds when moving the task coordinates from an initial point $\mathbf{t}_b$ to a final point $\mathbf{t}_a$ implies the vanishing of the same manifolds when reversing the trajectory (from $\mathbf{t}_a$ to $\mathbf{t}_b$). Thus, the creation of self-motion manifolds also implies the existence of barriers somewhere between $\mathbf{t}_a$ and $\mathbf{t}_b$. The creation of self-motion manifolds when moving from an initial point $\mathbf{t}_b$ to a final point $\mathbf{t}_a$ can also be inferred from the matching obtained by Algorithm 7: in this case, the creation of manifolds can be identified by the fact that at least one manifold m_a (at final point $\mathbf{t}_a$) does not belong to any set $\Omega(m_b)$, for all manifolds m_b at initial point $\mathbf{t}_b$ (i.e., no manifold at $\mathbf{t}_b$ transforms into the newly created manifolds at $\mathbf{t}_a$). This is what occurs in the example of Equation (7.4): manifold 0_a cannot be found among the manifolds that are obtained by transforming any of the manifolds at $\mathbf{t}_b$ (manifolds 0_b and 1_b).

The example of Figure 7.4 also shows that, in general, a change in the number of disjoint self-motion manifolds is not related to the vanishing/creation of manifolds. In Figure 7.4, the number of self-motion manifolds is the same for both task points $\mathbf{t}_a$ and $\mathbf{t}_b$. In this case, although manifold 0_a vanishes when moving from $\mathbf{t}_a$ to $\mathbf{t}_b$, manifold 1_a also splits into two disjoint manifolds 0_b and 1_b, so the net change in the number of disjoint self-motion manifolds between both task points is zero.

Once again, it is necessary to precisely define the meaning of "*sufficiently close*" in line 5 of Algorithm 7. As in Algorithm 6, we define $\mathcal{U}(\mathbf{x})$ for the matching Algorithm

7 (line 5) as the set of points of $\mathcal{M}(\mathbf{t}_b)$ which fall inside an n-dimensional box L_m centered at $\mathbf{x}$, with its main axes parallel to the coordinate axes of the joint space and with appropriate size s_j^m along each axis ($j = 1, \ldots, n$). Next, we will describe how to obtain possible appropriate values for the sizes s_j^m. Instead of using a fixed-size box (as in the clustering Algorithm 6) for all $\mathbf{x} \in \mathcal{M}(\mathbf{t}_a)$, the size of the box L_m used in the matching algorithm will generally depend on $\mathbf{x}$.

First, Equation (7.2) is approximated by its first-order Taylor expansion about $\boldsymbol{\theta} = \mathbf{x} \in \mathcal{M}(\mathbf{t}_a)$ (i.e., the center of the box L_m), $\mathbf{t} = \mathbf{t}_a$ and $\boldsymbol{\psi} = \boldsymbol{\psi}_0$, where $\boldsymbol{\psi}_0$ is the (known) value of $\boldsymbol{\psi}$ that satisfies Equation (7.2) for $\boldsymbol{\theta} = \mathbf{x}$ and $\mathbf{t} = \mathbf{t}_a$. (Recall that passive variables $\boldsymbol{\psi}$ are obtained for each $\boldsymbol{\theta}$ when densely sampling the self-motion manifolds; see line 11 of Algorithm 5.) This first-order expansion yields the following linear system:

$$\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}} \delta\boldsymbol{\theta} + \frac{\partial \mathbf{f}}{\partial \mathbf{t}} \delta\mathbf{t} + \frac{\partial \mathbf{f}}{\partial \boldsymbol{\psi}} \delta\boldsymbol{\psi} = \mathbf{0}_{n\times 1} \tag{7.5}$$

where $\delta\boldsymbol{\theta} = \boldsymbol{\theta} - \mathbf{x}$, $\delta\mathbf{t} = \mathbf{t} - \mathbf{t}_a$ and $\delta\boldsymbol{\psi} = \boldsymbol{\psi} - \boldsymbol{\psi}_0$. The Jacobian matrices $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\theta}}$, $\frac{\partial \mathbf{f}}{\partial \mathbf{t}}$ and $\frac{\partial \mathbf{f}}{\partial \boldsymbol{\psi}}$ (with sizes $n \times n$, $n \times m$ and $n \times r$, respectively) are evaluated at $(\boldsymbol{\theta} = \mathbf{x}, \mathbf{t} = \mathbf{t}_a, \boldsymbol{\psi} = \boldsymbol{\psi}_0)$. Next, any appropriate subset of r equations of the linear system (7.5) are utilized to eliminate $\delta\boldsymbol{\psi}$, obtaining the following reduced linear system:

$$\mathbf{J}_{\boldsymbol{\theta}} \delta\boldsymbol{\theta} = \mathbf{J}_{\mathbf{t}} \delta\mathbf{t} \tag{7.6}$$

In this way, passive variables $\delta\boldsymbol{\psi}$ are eliminated, obtaining the input-output relationship (7.6), which yields the change $\delta\boldsymbol{\theta}$ in the joint coordinates due to the change in the task variables when moving from $\mathbf{t}_a$ to $\mathbf{t}_b$, i.e., $\delta\mathbf{t} = \mathbf{t}_b - \mathbf{t}_a$.

The sizes of $\mathbf{J}_{\boldsymbol{\theta}}$ and $\mathbf{J}_{\mathbf{t}}$ are $(n - r) \times n$ and $(n - r) \times m$, respectively. Thus, system (7.6) contains n unknowns $\delta\boldsymbol{\theta} = [\delta\theta_1, \ldots, \delta\theta_n]^T$ to be solved from $(n - r)$ equations, which yields infinitely many different solutions due to the redundancy. Of all these possible solutions, it is reasonable to choose the solution with minimum norm, since Equation (7.6) is a linear approximation only valid if variations $\delta\boldsymbol{\theta}$ and $\delta\mathbf{t}$ are sufficiently small. Thus, choosing the minimum-norm solution yields:

$$\delta\boldsymbol{\theta} = \mathbf{J}_{\boldsymbol{\theta}}^T (\mathbf{J}_{\boldsymbol{\theta}} \mathbf{J}_{\boldsymbol{\theta}}^T)^{-1} \mathbf{J}_{\mathbf{t}} \delta\mathbf{t} \tag{7.7}$$

Once $\delta\theta_j$ has been obtained ($j = 1, \ldots, n$), an appropriate value for the size s_j^m of the matching box L_m (centered at $\mathbf{x}$) along the θ_j axis is: $s_j^m = \sigma_m \cdot 2 \cdot |\delta\theta_j|$, where $\sigma_m \geq 1$ is the *matching factor*, whose purpose is to increase the size of the box L_m, if necessary.

7.2.4 Obtaining Workspace Barriers

After describing in the previous subsection a method to determine how self-motion manifolds transform when moving between two neighboring task points, this subsection presents a method to estimate the boundaries and interior barriers of the workspace inside a prescribed region T of the m-dimensional task space. This method is summarized in Algorithm 8 and explained next.

Algorithm 8 Obtaining the boundaries and interior barriers of the workspace in a region T of the task space

```
1: procedure FIND_BARRIERS($T$, $N_T$)
       Inputs:
         • $T = [t_1^{min}, t_1^{max}] \times ... \times [t_m^{min}, t_m^{max}]$ // region of interest
         • $N_T = \{N_{t_1}, ..., N_{t_m}\}$ // No. of points for each axis $t_i$
       Outputs:
         • $B$ // boundaries of the workspace inside $T$
         • $B_I$ // interior barriers of the workspace inside $T$
2:     for $i = 1, \dots, m$ do
3:         $\Delta t_i \leftarrow \frac{t_i^{max} - t_i^{min}}{N_{t_i} - 1}$ // step along $t_i$ axis
4:         $\mathcal{I}_i \leftarrow \{ t_i^k : t_i^k = t_i^{min} + \Delta t_i \cdot k,$
                   $k = 0, \dots, N_{t_i} - 1\}$
5:     $\mathcal{G} \leftarrow \mathcal{I}_1 \times \mathcal{I}_2 \times \dots \times \mathcal{I}_m$ // Cartesian product
6:     for each node $\mathbf{g} \in \mathcal{G}$ do
7:         Obtain $\mathcal{M}(\mathbf{g})$ (Algorithm 5)
8:         Cluster $\mathcal{M}(\mathbf{g})$ (Algorithm 6)
9:     $B \leftarrow \emptyset$ // initialized as the empty set
10:    $B_I \leftarrow \emptyset$ // initialized as the empty set
11:    $M_p \leftarrow \emptyset$ // initialized as the empty set
12:    for each node $\mathbf{g} \in \mathcal{G}$ do
13:        $\mathcal{U}(\mathbf{g}) \leftarrow$ set of neighboring nodes of node $\mathbf{g}$
14:        for each $\mathbf{h} \in \mathcal{U}(\mathbf{g})$ do
15:            if $\{\mathbf{g}, \mathbf{h}\} \notin M_p$ then
16:                Add $\{\mathbf{g}, \mathbf{h}\}$ to $M_p$
17:                if $\mathcal{M}(\mathbf{g}) = \emptyset$ XOR $\mathcal{M}(\mathbf{h}) = \emptyset$ then
18:                    Add $(\mathbf{g} + \mathbf{h})/2$ to $B$
19:                else if $\mathcal{M}(\mathbf{g}) \neq \emptyset$ AND $\mathcal{M}(\mathbf{h}) \neq \emptyset$ then
20:                    Match $\mathcal{M}(\mathbf{g})$ and $\mathcal{M}(\mathbf{h})$ (Algorithm 7)
21:                    if manifolds are created OR vanish then
22:                        Add $(\mathbf{g} + \mathbf{h})/2$ to $B_I$
23:    return $B$, $B_I$
```

First, an m-dimensional box T is defined as $T = [t_1^{min}, t_1^{max}] \times \ldots \times [t_m^{min}, t_m^{max}]$, which encloses the region of interest of the task space, in which we wish to estimate the workspace barriers. Next, the main axes of T are discretized into N_{t_i} equally spaced points along each task axis t_i, approximating T by a regular m-dimensional grid $\mathcal{G}$ (lines 2-5). For each task node $\mathbf{g} \in \mathcal{G}$, the self-motion manifolds are densely sampled, and the disjoint manifolds are identified (lines 6-8). Next, two sets B and B_I are defined (lines 9-10). Upon the complete execution of the algorithm, the sets B and B_I will contain task points that approximate the boundaries and interior barriers inside region T, respectively. Also, a set M_p is defined as the set of (unordered) pairs of neighboring nodes $\{\mathbf{g}, \mathbf{h}\}$ between which the algorithm has already checked whether there exist barriers or not (line 11).

Next, for each node $\mathbf{g} \in \mathcal{G}$, the set $\mathcal{U}(\mathbf{g}) \subset \mathcal{G}$ of neighboring nodes of $\mathbf{g}$ is obtained. For example, in a two-dimensional task space, $\mathcal{U}(\mathbf{g})$ can be defined as the set of 8 nodes (of the grid $\mathcal{G}$) surrounding the center node $\mathbf{g}$, as in Figure 6.1b. Then, the self-motion manifolds at node $\mathbf{g}$ are compared (matched) to the manifolds at each neighboring node $\mathbf{h} \in \mathcal{U}(\mathbf{g})$, to check if there exist barriers between both nodes $\mathbf{g}$ and $\mathbf{h}$. If one (and only one) of these two nodes yields empty self-motion manifolds, then that node is outside the workspace, whereas the other node belongs to the workspace. In that case, a workspace boundary exists somewhere between both nodes. For simplicity, it is assumed that the boundary occurs at the midpoint between these nodes, which is added to the set B (lines 17-18).

If both neighboring nodes $\mathbf{g}$ and $\mathbf{h}$ yield non-empty self-motion manifolds, then both nodes belong to the workspace and it is necessary to check if interior barriers exist between these nodes. To this end, the manifolds at $\mathbf{g}$ and $\mathbf{h}$ are matched as described in subsection 7.2.3 (line 20). If the matching algorithm identifies the vanishing and/or creation of disjoint self-motion manifolds when moving from node $\mathbf{g}$ to node $\mathbf{h}$, then an interior barrier exists between both nodes, which, for simplicity, can be located at the midpoint of these nodes (lines 21-22).

Note that, both for the identified boundaries and interior barriers, it is possible to know the forbidden direction of the barrier (i.e., the direction in which the motion of the robot is impeded) at each point of these barriers: the motion is always impeded in the direction in which self-motion manifolds vanish (or the opposite of the direction in which manifolds are created). Depending on the kinematics of the robot and on the complexity of the kinematic constraints, bidirectional barriers may be obtained, i.e., barriers where some disjoint self-motion manifolds are created while other manifolds vanish.

Finally, note that Algorithm 8 can be easily parallelized. For example, the m-dimensional grid $\mathcal{G}$ can be divided into N_p parts, with each part having the same number of nodes. Then, each of these N_p parts can be processed by an individual processor dedicated to search barriers among its nodes, following Algorithm 8.

7.2.5 Elusive Barriers

Obviously, when decreasing steps $\Delta\theta_j$ (used for sampling self-motion manifolds, line 5 of Algorithm 5) and Δt_i (used for discretizing the task space, line 3 of Algorithm 8), the proposed method will estimate the workspace barriers more accurately (at the expense of higher computation times). However, the proposed method may be unable to detect some *elusive barriers* independently of how small $\Delta\theta_j$ and Δt_i are. Next, we will analyze the mechanisms shown in Figure 7.5 to illustrate two types of elusive barriers that the proposed method may fail to detect.

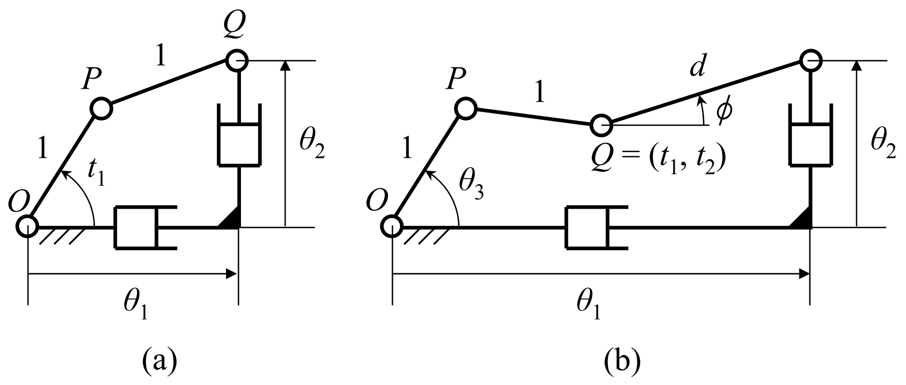


Figure 7.5: (a) Two-DOF and (b) three-DOF mechanisms used for illustrating elusive barriers. Links OP and PQ have unitary length: $\overline{OP} = \overline{PQ} = 1$.

7.2.5.1 Manifolds existing only at bidirectional elusive barriers

The first type of elusive barriers that we will analyze are generated by self-motion manifolds that only exist when the task coordinates are placed *exactly* on these barriers, such that these manifolds vanish when perturbing the task coordinates away from these barriers. In order to illustrate this situation, consider the 2-DOF mechanism of Figure 7.5a, which can be considered as a redundant robot with joint coordinates (θ_1, θ_2) (position of joint Q) and task variable t_1 (orientation of link OP). For a given t_1, the self-motion manifold of this robot in plane (θ_1, θ_2) is a circle C centered at $P = (\cos t_1, \sin t_1)$ and with radius 1. In the absence of additional constraints, C has constant topology $\forall t_1$. Thus, the workspace of task variable t_1 is the unit circle, without interior barriers or singularities.

Assume now that, in order to avoid collisions, point Q cannot be in the horizontal region H of plane (θ_1, θ_2) defined by: $1 < \theta_2 < 2$ (strict inequalities). In that case, the workspace of t_1 is still the unit circle, but it contains singularities now (see Figure 7.6a): two traversable singularities $T^{\pm}$ at $\cos t_1 = \pm 1$, and a bidirectional barrier B at $\sin t_1 = 1$. For t_1 in the lower semicircle ($\sin t_1 < 0$), there is a single self-motion manifold which is the circle C defined in previous paragraph (Figure 7.6d). For t_1 in the two upper quadrants defined by $\{\sin t_1 > 0 \text{ AND } \cos t_1 \neq 0\}$, there is a single self-motion manifold which is the part of circle C not falling in region H (Figure 7.6c). Traversable singularities $T^{\pm}$ are generated by the change of topology of C when invading region H, where circle C (Figure 7.6d) becomes an open arc of circle (Figure

7.6c). Finally, at the bidirectional barrier B, there are two disjoint manifolds (Figure 7.6b): a semicircle and isolated point $V = (0, 2)$ (which is out of the forbidden region H).

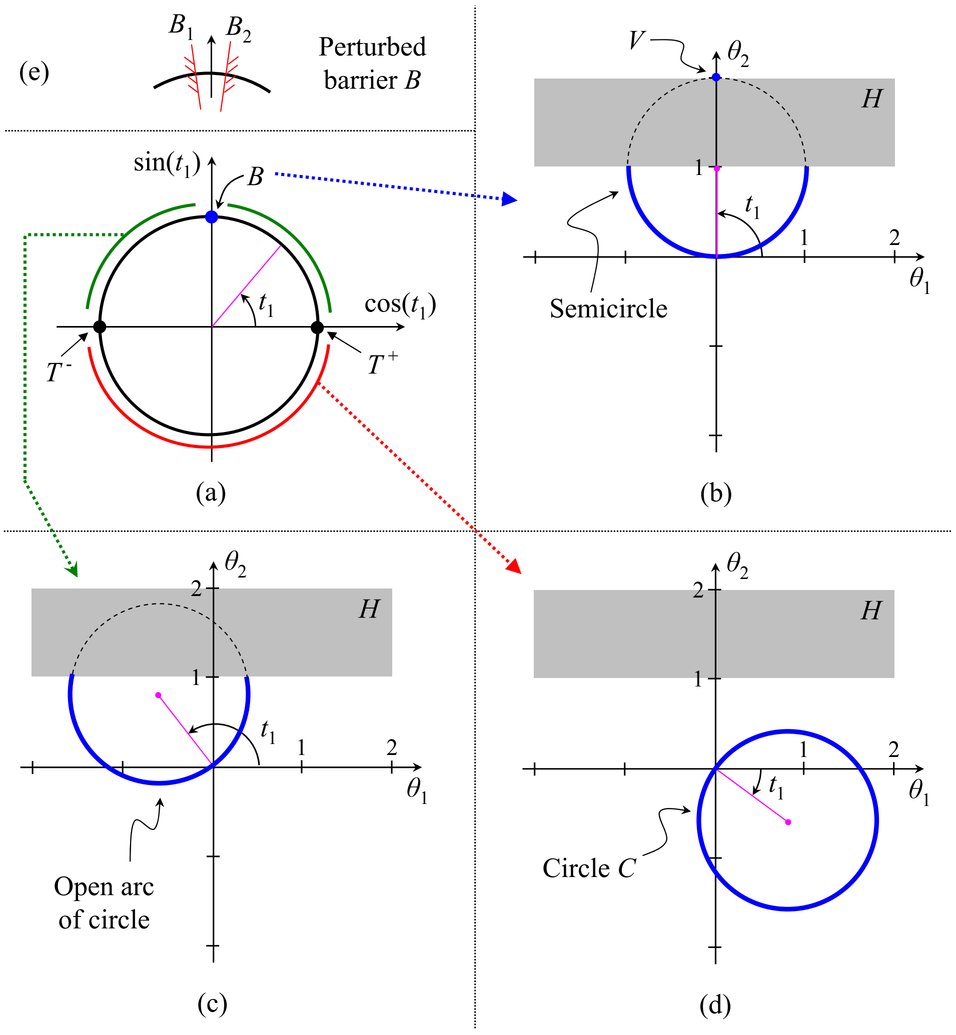


Figure 7.6: (a) Workspace of the robot of Figure 7.5a. Self-motion manifolds at different regions of this workspace are shown in (b), (c), and (d), in blue continuous line. The perturbation of elusive barrier B is shown in (e).

Note that manifold V only exists at $\sin t_1 = 1$: when task variable t_1 is slightly perturbed in any direction, V vanishes and a bidirectional barrier occurs at B. This is an example of elusive barrier that the proposed method would miss with probability one, independently of how small the discretization steps $\Delta\theta_j$ and Δt_i are. Indeed, independently of how small Δt_1 is, if task point B is not *exactly* one of the nodes of the grid $\mathcal{G}$ defined in line 5 of Algorithm 8 (in this example, grid $\mathcal{G}$ may be a regular discretization of interval $[-\pi/2, 3\pi/2]$), then Algorithm 8 will not be able to detect the vanishing of manifold V when matching self-motion manifolds between two neighboring nodes of this grid, since V only exists at B.

Elusive barriers of this type are unstable under small perturbations of the constraints or of the geometry of the robot, since in that case these barriers disappear or become detectable by the proposed method. For example, barrier B disappears when redefining region H as $\{1 < \theta_2 \leq 2\}$ (H now includes line $\theta_2 = 2$), since in this case $V \in H$, and the only manifold for $\sin t_1 = 1$ would be the semicircle shown in Figure 7.6b. On the contrary, if H is redefined as $\{1 < \theta_2 < 1.99\}$, then barrier B is split into two unidirectional barriers B_1 and B_2 that enclose the arc B_1B_2 of the unit circle for which $\sin t_1 > 0.99$ (see Figure 7.6e). For t_1 inside this arc, there exist two disjoint self-motion manifolds, which are the parts of circle C outside H. Elusive barrier B is the limit of arc B_1B_2 degenerating into a point. The perturbed barriers B_1 and B_2 would be detectable by the proposed method using a sufficiently small step Δt_1, since the arc enclosed by them has non-zero length and will contain nodes of grid $\mathcal{G}$.

7.2.5.2 Elusive barriers not generated by vanishing manifolds

In order to illustrate another type of elusive barrier, consider the mechanism of Figure 7.5b, which is obtained from the mechanism of Figure 7.5a by adding another link of length d between joint Q and the vertical prismatic leg of length $|\theta_2|$. This new mechanism has three DOF, and it can be considered as a redundant robot with joint coordinates $(\theta_1, \theta_2, \theta_3)$ and task variables (t_1, t_2) (position coordinates of joint Q relative to fixed joint O). The orientation angle ϕ of link d is a passive variable. For a given task point $Q = (t_1, t_2) \neq (0, 0) = O$, the self-motion manifolds in the 3D joint space $(\theta_1, \theta_2, \theta_3)$ are two circles obtained as the intersection between cylinder $\{(\theta_1 - t_1)^2 + (\theta_2 - t_2)^2 = d^2\}$ and two planes $\{\theta_3 = K_1\}$ and $\{\theta_3 = K_2\}$, where K_1 and K_2 are the two solutions of θ_3 in $[-\pi, \pi]$ that satisfy the following equation:

$$(t_1 - \cos\theta_3)^2 + (t_2 - \sin\theta_3)^2 = 1 \tag{7.8}$$

Fixed joint $O = (0, 0)$ in this example is an interior barrier of the workspace, as justified next. Assume that joint Q coincides with O, as illustrated in Figure 7.7a. In that case, the robot can only generate task velocities in the direction perpendicular to link OP. If one wishes to move joint Q along the direction L of link OP (for describing a linear trajectory across O, for example), links OP and PQ must be rotated first an angle of 90°, so that they remain perpendicular to L (Figure 7.7b). After that, velocities or displacements along L can be generated (Figure 7.7c).

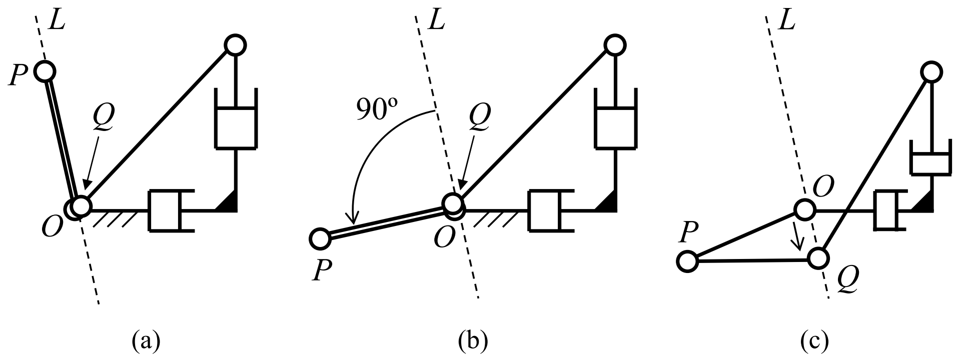


Figure 7.7: Maneuvering for traversing a point elusive barrier.

This occurs for any orientation θ_3 of link OP, since when Q coincides with O, link OP can freely rotate without modifying the position of Q (this is because links OP and PQ have the same length). This can also be observed in Equation (7.8), which is satisfied $\forall\theta_3$ when $Q = (t_1, t_2) = (0, 0) = O$. Since θ_3 is not constrained, the self-motion manifolds at O are not two circles but a single cylinder defined by $\{\theta_1^2 + \theta_2^2 = d^2\}$. Note that self-motion manifolds do not vanish at this special barrier, but their dimension instantaneously changes: two circles, which are one-dimensional manifolds obtained by intersecting a cylinder with two planes, transform into the complete cylinder, which is a two-dimensional manifold.

Note also that barrier O in this example, which is an isolated point barrier, is another elusive barrier that the proposed method cannot detect for two reasons. Firstly: in order for Algorithm 8 to detect point barrier O, this point should coincide exactly with one of the nodes of the planar grid $\mathcal{G}$ used for scanning the task plane (t_1, t_2) in this example (and this occurs in general with zero probability, as argued for the elusive barrier B of Figure 7.6a). Secondly, and more importantly: the proposed method identifies barriers with the vanishing of self-motion manifolds, but these manifolds do not vanish at O (they experience a change in their dimension).

Finally, like the elusive barrier B of Figure 7.6a, the elusive barrier O of this example is also unstable: when slightly perturbing the geometry of the robot (so that links OP and PQ do not have exactly the same length), point barrier O transforms into a circular barrier centered at O and with radius $|\overline{OP} - \overline{PQ}|$. This perturbed circular barrier can be detected by the proposed method if sufficiently small steps Δt_1 and Δt_2 are used, since self-motion manifolds vanish inside this circular barrier (the region enclosed by this circular barrier is not reachable).

7.3 Examples

This section presents some illustrative and meaningful examples to demonstrate the viability and usefulness of the method proposed in Section 7.2. All examples shown in this section have been implemented in Java and have been tested on a computer with a 3 GHz 8-Core Intel Xeon E5 processor and with 16 GB of RAM memory. All examples shown in this section have been obtained by parallelizing the calculations over 8 cores as suggested at the end of subsection 7.2.4, using MPJ Express for the parallelization [170].

Table 7.1 summarizes the main data of the examples detailed in this section, including the CPU time required to obtain all barriers in each example and the numbers of the figures in which the computed barriers can be found. According to Table 7.1, CPU times vary between 10 minutes and almost 2 hours, depending on the degree of redundancy r (dimension of self-motion manifolds) or the considered constraints (collisions checked or omitted), among other parameters. These CPU times are reasonable considering the attained precision (see next examples) and the amount of information obtained (interior barriers are also computed, not only workspace boundaries). Also, these CPU times are similar to those required by other methods that obtain workspace

Table 7.1: Summary of the six experiments performed, indicating the dimensions of the task space (m), joint space (n), and self-motion manifolds (r).

Experiment number	1	2	3	4	5	6
Robot	Stewart platform				3$\underline{\text{R}}$RR	
(m, n, r)	$(2, 6, 1)$		$(2, 6, 2)$		$(2, 6, 1)$	
$(N_{t_i}, N_s^j, \sigma^c, \sigma^m)$	$(400, 400, 1.5, 15)$		$(160, 60, 1.5, 15)$		$(200, 200, 1.5, 2)$	
Collisions checked	no	yes	no	yes	no	yes
CPU time (minutes)	9.5	10.1	116	110	27	17
Figure No.	7.9	7.10	7.11	7.12	7.14	7.15

barriers, like the times reported in [19] (although this is only an orientative comparison since both methods were not tested on the same computer, and considered different constraints and robots). Anyway, the CPU times reported in Table 7.1 suggest that our method is more suitable for offline computations: workspace barriers can be pre-computed offline first, and then used later with design or offline path planning objectives.

In the context of online path planning, for example, it may be necessary to know if the robot will get blocked at workspace barriers (e.g., due to collisions) when following the current trajectory. In that case, it is not necessary to do the time-consuming task of computing all barriers in a given region of the task space, but it would be more useful to know if some self-motion manifold will vanish when moving from current task point $\mathbf{t}_k$ to the next one $\mathbf{t}_{k+1}$. The proposed method may still be useful for this purpose, since sampling/clustering the manifolds at only two task points and matching them is relatively fast (e.g., about 0.25 seconds when considering collisions in the 3$\underline{\text{R}}$RR example of Table 7.1). Although these times may still be too large for real-time control (they can always be reduced by decreasing N_s^j), they may allow for collision control algorithms during online path planning. However, the detailed analysis of this application is beyond the scope of the present chapter.

7.3.1 Stewart Platform with 1D Self-motion Manifolds

In this section, we will use the proposed method to analyze the workspace of the Stewart platform, considering that the degree of redundancy is $r = 1$. Consider a Stewart platform as shown in Figure 7.8, used for machining (assume that the Z axis is a tool, e.g. a drill). In this robot, six linear actuators or legs of type U$\underline{\text{P}}$S are used to control the position and orientation of a frame Σ attached to the mobile platform. The position and orientation of frame Σ with respect to the base frame W is defined by the (x, y, z) coordinates of point P of the mobile platform and by the XYZ Euler angles (α, β, γ).

Note that, in machining applications, the rotation γ of the mobile platform about the tool axis (which coincides with the Z axis of frame Σ) is not relevant, since the tool is continuously rotating about its own axis. Therefore, if γ is not relevant, then we can consider the robot of Figure 7.8 as a redundant robot in which 6 linear

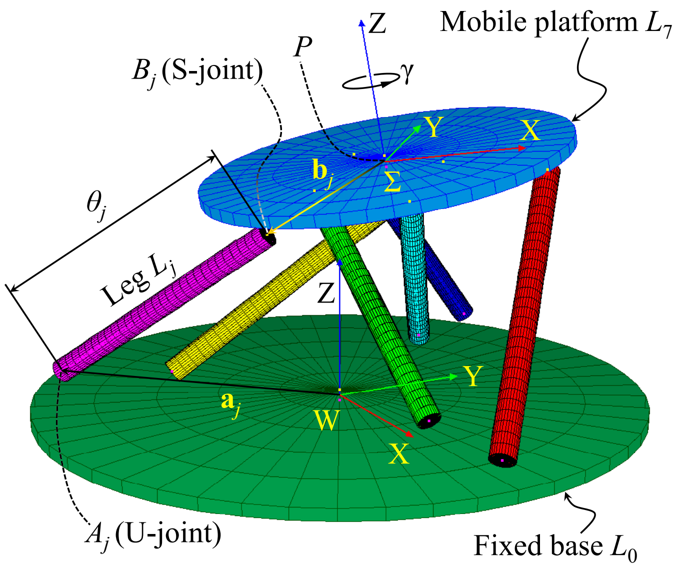


Figure 7.8: Stewart platform and associated notation.

actuators are used to control 5 task variables: the position (x, y, z) and angles (α, β). The degree of redundancy is therefore $r = 1$, so self-motion manifolds are curves in the six-dimensional joint space (which is the space of lengths θ_j of the linear actuators, $j = 1, \ldots, 6$).

In this example, the task space is 5-dimensional. Although Algorithm 8 can be applied in principle to obtain the barriers in task spaces with any dimension m, since the number of calculations of this algorithm grows exponentially with the dimension of the task space, the computation times would become prohibitive for $m = 5$. Therefore, to illustrate the proposed algorithm, we will analyze the 2-dimensional (y, z) task subspace, keeping constant the values of the remaining task variables (x, α, β). This is equivalent to analyzing a planar slice of such a 5-dimensional workspace.

Next, we will describe how Algorithm 8 can be particularized to obtain the barriers of the workspace in a specified region of the task plane (y, z) in this example, considering the following kinematic constraints:

- The length θ_j of each linear actuator should be between a minimum length $\theta_j^{min} > 0$ and a maximum length $\theta_j^{max} > \theta_j^{min}$ $(j = 1, \ldots, 6)$.
- Different links/bodies of the robot should not collide. Eight links are considered: the six linear actuators, the base, and the mobile platform. These eight links will be modeled as cylinders, as explained later in more detail.

In this example, the joint coordinates are the lengths θ_j of the six linear actuators: $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_6]^T$ $(n = 6)$. The task variables are $\mathbf{t} = [y, z]^T$ $(m = 2)$. The only passive variable in this example is $\boldsymbol{\psi} = [\gamma]^T$ $(r = 1)$. Note that, since (x, α, β)

have been fixed in order to analyze a 2-dimensional slice of the task space, these three variables can be considered as geometric parameters of the robot in our formulation. For each linear actuator $j \in \{1, \ldots, 6\}$, we impose the condition that its length must equal θ_j:

$$\left\| \mathbf{R}_{\alpha x} \mathbf{R}_{\beta y} \mathbf{R}_{\gamma z} \mathbf{b}_j + [x, y, z]^T - \mathbf{a}_j \right\|^2 - \theta_j^2 = 0 \tag{7.9}$$

where (see Figure 7.8): $\mathbf{b}_j$ are the coordinates of the centers B_j of the spherical joints S referred to frame Σ, $\mathbf{a}_j$ are the coordinates of the centers A_j of the universal joints U referred to frame W, and $\mathbf{R}_{vw}$ denotes a 3×3 rotation matrix of angle v about axis w. If the left-hand side of Equation (7.9) is denoted by f_j, then particularizing j for $\{1, \ldots, 6\}$ yields the constraint function $\mathbf{f} = [f_1, \ldots, f_6]^T$ of Equation (7.2), which defines the self-motion manifolds in this example.

To densely sample the self-motion manifolds at each task point $\mathbf{t} = [y, z]^T$ following Algorithm 5, we proceed as follows. First, since the degree of redundancy is $r = 1$, Algorithm 5 reduces to sweeping each joint coordinate axis θ_j independently ($j = 1 \ldots, 6$), between θ_j^{min} and θ_j^{max}. As each axis θ_j is swept, both γ (the passive variable) and the remaining unknown joint coordinates θ_k ($k \neq j$) are calculated in terms of θ_j as explained next.

Given θ_j (i.e., the joint coordinate that is being swept) and $\mathbf{t} = [y, z]^T$ (i.e., the task point at which self-motion manifolds are being densely sampled) the only unknown in Equation (7.9) is the passive angle γ, which can be easily solved from Equation (7.9) since this equation has the following form: $C \cdot \cos\gamma + S \cdot \sin\gamma + I = 0$, where $\{C, S, I\}$ are known constants [using Weierstrass substitution, this reduces to a quadratic equation in $\tan(\gamma/2)$]. After computing γ, Equation (7.9) is particularized for each of the remaining unknown joint coordinates θ_k ($k \neq j$), which are obtained as follows:

$$\theta_k = \left\| \mathbf{R}_{\alpha x} \mathbf{R}_{\beta y} \mathbf{R}_{\gamma z} \mathbf{b}_k + [x, y, z]^T - \mathbf{a}_k \right\| \tag{7.10}$$

Note that the right-hand side of Equation (7.10) is now completely known because γ has already been solved from Equation (7.9) particularized for the joint coordinate that is swept.

To illustrate the proposed method with 1-dimensional self-motion manifolds in the Stewart platform, the following parameters are used: $x = 0.3$, $\alpha = 0.63$ rad, $\beta = 0$. The minimum and maximum lengths of linear actuators are: $\theta_j^{min} = 0.55$ and $\theta_j^{max} = 1$ ($j = 1, \ldots, 6$). For the geometry of the robot (positions $\{\mathbf{a}_j, \mathbf{b}_j\}$ of joints), the *double-ring* design shown in Table 7.2 is chosen, since this arrangement of joints is known to imply some risks of interference between different legs [115].

The existence of mechanical interferences between different links is checked using the SOLID library [188] through Java Native Interface. Eight links L_j are considered: the six legs, the base, and the mobile platform. All these eight links are modeled as cylinders, as detailed next. Each leg L_j ($j = 1, \ldots, 6$) is modeled as a cylinder with radius 0.025 and axis $A_j B_j$, where A_j and B_j (i.e., the centers of the universal and spherical joints) are the centers of its top and bottom sections (see Figure

Table 7.2: Geometry of a double-ring Stewart platform.

j	$\mathbf{a}_j$	$\mathbf{b}_j$
1	$\begin{bmatrix} 0.6\cos(0) \\ 0.6\sin(0) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.34\cos(0) \\ 0.34\sin(0) \\ -0.05 \end{bmatrix}$
2	$\begin{bmatrix} 0.36\cos(0) \\ 0.36\sin(0) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.17\cos(180°) \\ 0.17\sin(180°) \\ -0.05 \end{bmatrix}$
3	$\begin{bmatrix} 0.6\cos(120°) \\ 0.6\sin(120°) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.34\cos(120°) \\ 0.34\sin(120°) \\ -0.05 \end{bmatrix}$
4	$\begin{bmatrix} 0.36\cos(120°) \\ 0.36\sin(120°) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.17\cos(300°) \\ 0.17\sin(300°) \\ -0.05 \end{bmatrix}$
5	$\begin{bmatrix} 0.6\cos(-120°) \\ 0.6\sin(-120°) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.34\cos(-120°) \\ 0.34\sin(-120°) \\ -0.05 \end{bmatrix}$
6	$\begin{bmatrix} 0.36\cos(-120°) \\ 0.36\sin(-120°) \\ 0.05 \end{bmatrix}$	$\begin{bmatrix} 0.17\cos(-300°) \\ 0.17\sin(-300°) \\ -0.05 \end{bmatrix}$

7.8). The fixed base L_0 is also modeled as a cylinder centered at the origin of frame W, with its axis coincident with the Z axis of frame W, with radius 0.65 and height 0.025. Similarly, the mobile platform L_7 is modeled as a cylinder centered at the origin of frame Σ, with its axis coincident with the Z axis of frame Σ, with radius 0.4 and height 0.025. The geometry described in this paragraph and in the previous one is precisely the geometry that can be observed in Figure 7.8.

To apply the method described in Section 7.2 and obtain the workspace interior barriers and boundaries in a region of the task plane (y, z), the following parameters are used. Barriers and boundaries are searched in task box $T = [-0.41, 0.15] \times [0.41, 0.81]$. Each axis of this box is discretized into $N_{t_i} = 400$ equally spaced points. Therefore, box T is approximated by a regular grid $\mathcal{G}$ of $1.6 \cdot 10^5$ task nodes in which to compute the 1-dimensional self-motion manifolds. The task of computing, clustering, and matching manifolds in all these nodes is distributed over $N_p = 8$ processors working in parallel (the distribution of the workload is done partitioning the y axis, as indicated in Figure 7.9). To densely sample the self-motion manifolds according to Algorithm 5, each joint axis θ_j is discretized into $N_s^j = 400$ equally spaced points. Finally, the chosen clustering and matching factors are: $\sigma_c = 1.5$ and $\sigma_m = 15$.

The precision of the result depends on factors σ_c and σ_m. Our tests suggest that acceptable results are usually obtained if both factors are between 1 and 2. If these factors are too small (i.e., they take values too close to 1), the results may become too noisy. On the contrary, if these factors are too large, then the results may be erroneous due to the identification of disjoint manifolds as the same connected manifold. In this example, acceptable results can be obtained using $\sigma_c = \sigma_m = 1.5$, although in that case the rightmost boundary B_N of Figure 7.9 becomes too noisy (in the sense that some of its parts are erroneously identified as bidirectional barriers). Increasing σ_m to 15 greatly reduces the noise of boundary B_N without affecting the remaining barriers.

Figure 7.9 shows the obtained barriers (both boundaries and interior barriers) for the example described in the previous paragraphs. In the following figures, we will represent workspace barriers following a convention similar to the one used in [19]. We will represent barriers in blue continuous line, and we will draw at each point of the barriers a small red vector, indicating that the robot cannot trespass the barriers toward the side where the red vector lies/points, due to the vanishing of self-motion manifolds. The direction of these vectors is obtained as follows: when Algorithm 8 (lines 17-22) detects that self-motion manifolds vanish (or are created) when moving from a given task node $\mathbf{g}$ to one of its neighbors $\mathbf{h}$, the represented barrier vector points from $\mathbf{g}$ to $\mathbf{h}$ (or from $\mathbf{h}$ to $\mathbf{g}$, respectively). For 2-dimensional task spaces such as the examples shown in the following figures, $\mathbf{g}$ and $\mathbf{h}$ are nodes of a planar grid, with $\mathbf{h}$ being one of the 8-neighbors of $\mathbf{g}$ (as explained in subsection 7.2.4). In that case, the red barrier vectors can only have eight possible orientations: those of the vectors connecting a node $\mathbf{g}$ to each of its 8-neighbors (and these orientations will be integer multiples of $45°$ if the grid has the same step along both axes). Although this representation is not as accurate as the ones used in [71] and [19] (where these vectors are drawn orthogonal to the barriers), it is sufficient for easily visualizing which side of each barrier will be forbidden.

The workspace shown in Figure 7.9 is obtained omitting collisions between different links, i.e., different links are allowed to mechanically interfere. The result shown in Figure 7.9 is obtained in about 9.5 minutes using the computer described at the beginning of Section 7.3. As Figure 7.9 shows, some of the identified barriers/boundaries have noisy regions, but these can be reduced by modifying the parameters of the proposed method (e.g., increasing the matching factor σ_m).

Note that the boundaries and barriers of the workspace are composed of the union of smooth arcs, which are joined at non-smooth sharp points. Figure 7.9 indicates the conditions that generate each of these smooth arcs: at each arc, two joint coordinates simultaneously reach either their minimum or maximum joint limits. These conditions have been obtained by simulating and analyzing the complete configuration of the robot when placing its point P at the different identified arcs.

Next, to analyze the effect of collisions between different links on the boundaries and barriers of the workspace, the example of Figure 7.9 is re-computed again, but considering the additional kinematic constraint that different links cannot interfere, as described in previous paragraphs. If different links are not allowed to interfere, then the boundaries and barriers of Figure 7.9 transform into the ones shown in Figure 7.10 (in this case, Algorithm 8 takes about 10.1 minutes). When comparing Figures 7.9 and 7.10, it can be observed that the no-collision restriction alters both the outermost boundaries of the workspace (i.e., its shape) and the interior barriers. Some boundary/barrier arcs of the original workspace of Figure 7.9 are maintained when forbidding

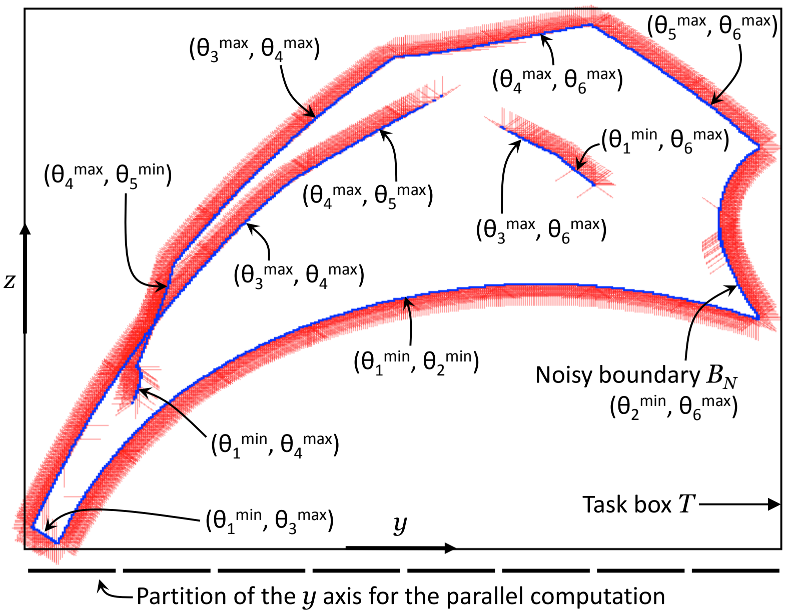

(θ_3^{max}, θ_4^{max})
(θ_5^{max}, θ_6^{max})
(θ_4^{max}, θ_6^{max})
(θ_1^{min}, θ_6^{max})
(θ_4^{max}, θ_5^{min})
(θ_4^{max}, θ_5^{max})
(θ_3^{max}, θ_6^{max})
(θ_3^{max}, θ_4^{max})
z
(θ_1^{min}, θ_2^{min})
Noisy boundary B_N
(θ_2^{min}, θ_6^{max})
(θ_1^{min}, θ_4^{max})
Task box T
(θ_1^{min}, θ_3^{max})
y
Partition of the y axis for the parallel computation

Figure 7.9: Workspace barriers of a Stewart platform with 1-dimensional self-motion manifolds ($\sigma_m = 15$). Mechanical interferences between different links are allowed.

collisions, but other new arcs are generated due to the prohibition of collisions. Other interior barriers of Figure 7.9, such as those defined by $(\theta_3^{max}, \theta_6^{max})$ and $(\theta_1^{min}, \theta_6^{max})$, counterintuitively disappear when considering collisions: this is because these barriers were generated by the vanishing of disjoint self-motion manifolds that do not even exist when forbidding collisions. Figure 7.10 indicates the conditions that give rise to some arcs. In all cases, each arc is still defined by two simultaneous kinematic constraints: some joint coordinate reaches a joint limit and/or different links are about to collide (if two links L_j and L_k collide, this is indicated in Figure 7.10 as "L_jL_k").

7.3.2 Stewart Platform with 2D Manifolds

Next, the method proposed in Section 7.2 will be applied to an example where the degree of redundancy is $r = 2$, to demonstrate the feasibility of the proposed method with redundant robots in which self-motion manifolds are surfaces in the joint space. Also, comparing this example with the previous one will allow us to observe how the workspace of a redundant robot is modified when increasing the degree of redundancy.

To this end, consider the example of the redundant Stewart platform analyzed in previous subsection 7.3.1 (where rotation γ was considered irrelevant), but assume now that the Euler angle β (rotation about the mobile Y axis) is also irrelevant for the task to be performed. In that case, the joint and task coordinates still are $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_6]^T$ and $\mathbf{t} = [y, z]^T$, respectively, but now there are $r = 2$ passive variables: $\boldsymbol{\psi} = [\beta, \gamma]^T$. Therefore, to densely sample the self-motion manifolds at each task point according to Algorithm 5, we must sweep all the $\binom{6}{2} = 15$ coordinate planes of the 6-dimensional joint space.

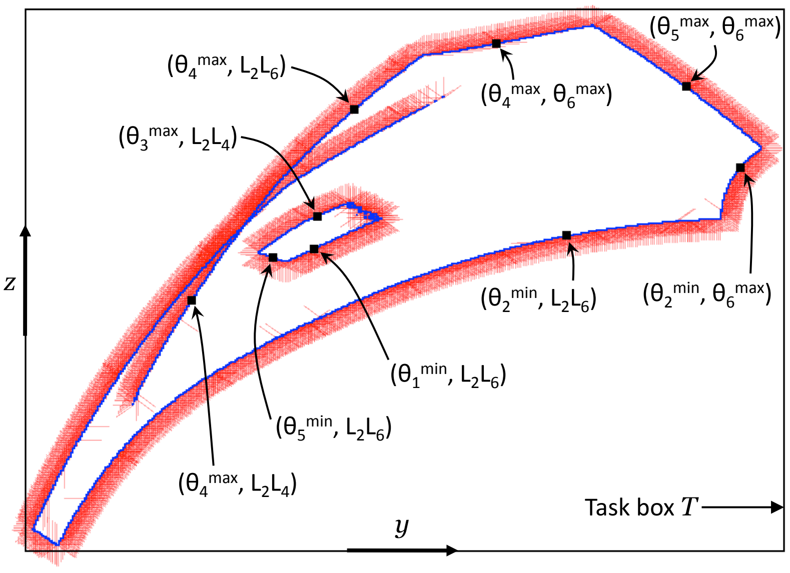


Figure 7.10: Barriers obtained when forbidding mechanical interferences between different links in the example of Figure 7.9.

When sweeping any two joint coordinates $(\theta_{j_1}, \theta_{j_2})$ as required by Algorithm 5 for $r = 2$, the two (unknown) passive variables (β, γ) and the remaining four (unknown) joint coordinates must be solved from Equation (7.2). All these six unknowns can be solved in terms of $(\theta_{j_1}, \theta_{j_2})$ following the elimination procedure explained next. Since the swept joint coordinates $(\theta_{j_1}, \theta_{j_2})$ are known, then particularizing Equation (7.9) for these two joint coordinates yields the following system of two trigonometric equations (7.11)-(7.12), in which the only unknowns are angles β and γ:

$$\left\|\mathbf{R}_{\alpha x}\mathbf{R}_{\beta y}\mathbf{R}_{\gamma z}\mathbf{b}_{j_1} + [x, y, z]^T - \mathbf{a}_{j_1}\right\|^2 - \theta_{j_1}^2 = 0 \tag{7.11}$$

$$\left\|\mathbf{R}_{\alpha x}\mathbf{R}_{\beta y}\mathbf{R}_{\gamma z}\mathbf{b}_{j_2} + [x, y, z]^T - \mathbf{a}_{j_2}\right\|^2 - \theta_{j_2}^2 = 0 \tag{7.12}$$

To solve (β, γ) from system (7.11)-(7.12), the Weierstrass substitution is used to transform these two trigonometric equations into two polynomial equations of degree two in the unknowns $\tan(\beta/2)$ and $\tan(\gamma/2)$. Using resultants, these two equations can be reduced to a single univariate polynomial equation of degree eight in $\tan(\beta/2)$ [or in $\tan(\gamma/2)$], which can be easily solved using root finding techniques. After solving β and γ from system (7.11)-(7.12), these two passive variables can be substituted into Equation (7.10) particularized for the four unknown joint coordinates θ_k (where $\theta_k \neq \theta_{j_1}$ and $\theta_k \neq \theta_{j_2}$), to obtain the values of these unknown joint coordinates.

After describing how to apply Algorithm 5 to densely sample the 2-dimensional self-motion manifolds in this case, the proposed method will be applied to a concrete example next. For the following example, workspace boundaries and barriers will be obtained in the region of the task space inside box $T = [-0.46, 0.38] \times [0.28, 0.83]$. Each axis of this box is discretized into $N_{t_i} = 160$ equally spaced points, thus discretizing box T into a planar regular grid $\mathcal{G}$ consisting of 25600 nodes. The task of sampling, clustering, and matching the self-motion manifolds at all these nodes is again distributed over $N_p = 8$ processors along the horizontal y axis, as indicated in Figure 7.9. Each joint axis is discretized into $N_s^j = 60$ equally spaced points. The clustering and matching factors are again: $\sigma_c = 1.5$ and $\sigma_m = 15$. The remaining parameters have the same values as in subsection 7.3.1: $x = 0.3$, $\alpha = 0.63$ rad, the joint limits are $\theta_j^{min} = 0.55$ and $\theta_j^{max} = 1$, and the geometric design of the robot is given in Table 7.2. Unlike in subsection 7.3.1, now β is not fixed but a passive variable.

Figure 7.11 shows the barriers and boundaries obtained using the proposed method with the previous parameters. The barriers shown in Figure 7.11 omit collisions between different links, i.e., different links are allowed to mechanically interfere. In this case, since self-motion manifolds are 2-dimensional, there is a significant increase in the time required to execute Algorithm 8 with respect to the previous case of 1-dimensional manifolds: running on the computer described at the beginning of this section, Algorithm 8 takes about 116 minutes to obtain the barriers presented in Figure 7.11. As in Figures 7.9 and 7.10, Figure 7.11 indicates the conditions that give rise to some of the identified barriers. In this case, as Figure 7.11 shows, barriers occur when three joint coordinates simultaneously reach their minimum or maximum joint limits. The occurrence of joint limits at barriers is abbreviately denoted in Figure 7.11 as follows: $\theta_j^{min} \equiv \underline{j}$, $\theta_j^{max} \equiv \overline{j}$.

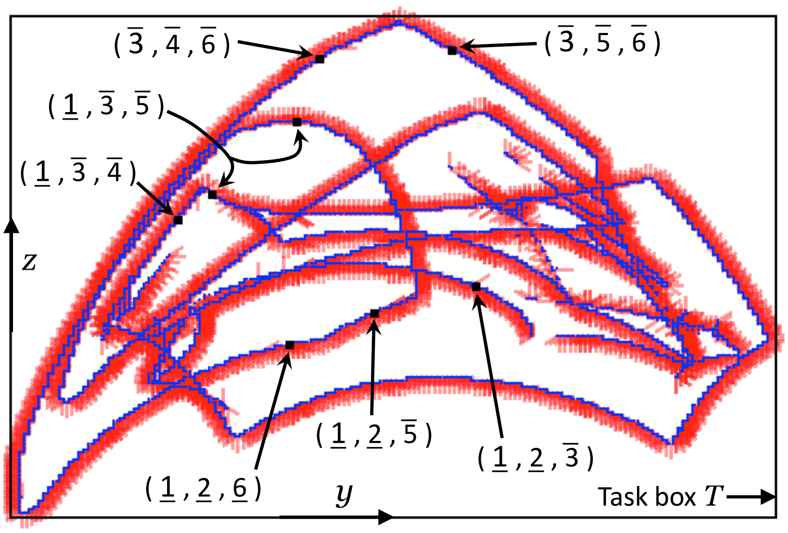


Figure 7.11: Workspace barriers of a Stewart platform with 2-dimensional self-motion manifolds. Mechanical interferences are allowed.

If we repeat the calculation but considering the additional constraint that different links cannot interfere, then we obtain the result shown in Figure 7.12 (in this case, Algorithm 8 takes about 110 minutes). When comparing Figures 7.11 and 7.12, it can be observed that including the condition of no-interference between different links can importantly alter both the boundaries and interior barriers of the workspace, although some boundaries are conserved. Following the same notation used in Figure 7.10 to indicate imminent collisions, Figure 7.12 indicates the conditions that give rise to some of the identified barriers: in all the indicated cases, the barrier is originated by three kinematic constraints occurring simultaneously (joint limits and/or imminent collisions between different links). The noise indicated in Figure 7.12 can be reduced by increasing N_s^j (i.e., sampling more finely the 2-dimensional self-motion manifolds), but this increases the computation time. Alternatively, increasing σ_m reduces this noise without increasing the computation time, but this degrades some parts of the barriers, which "lose" density (e.g., part V in Figure 7.12) and may erroneously vanish.

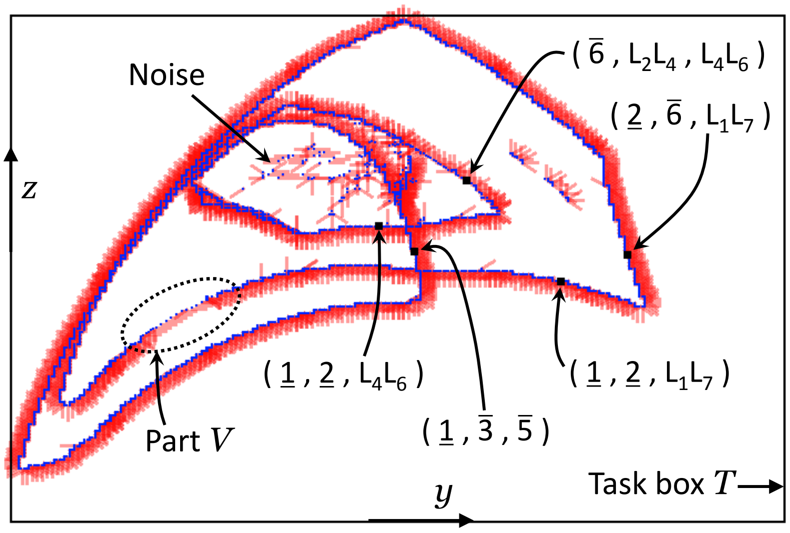


Figure 7.12: Barriers obtained when forbidding collisions in the example of Figure 7.11.

In the examples shown in present subsection 7.3.2 and in previous subsection 7.3.1, both the interior barriers and the external boundaries obtained when omitting collisions are altered when forbidding these collisions between different links. In the following subsection, we will show an example where forbidding collisions does not affect the external boundaries of the workspace (i.e., the shape of the workspace), but drastically modifies its interior barriers (i.e., its internal structure).

7.3.3 3$\underline{\text{R}}$RR Parallel Robot with 1D Manifolds

In this subsection, the proposed method will be applied to obtain the barriers of the reachable workspace of the planar 3$\underline{\text{R}}$RR parallel robot shown in Figure 7.13. This robot can be considered as a redundant robot if we actuate angles $(\alpha_1, \alpha_2, \alpha_3)$ to control the position (x, y) of the center of the end-effector (which is an equilateral triangle $B_1B_2B_3$ of side h), without caring about its orientation ϕ. In that case, self-motion manifolds are curves in the 3-dimensional joint space of actuated angles $(\alpha_1, \alpha_2, \alpha_3)$.

However, the method proposed in Section 7.2 cannot be directly applied to angular joint coordinates. This is because angles undergo wrapping (i.e., two angles differing by an integer multiple of 2π rad can be considered as the same angle *if there are not joint limits* [100]), which impedes clustering (Section 7.2.2) and matching (Section 7.2.3) self-motion manifolds using kd-trees. Kd-trees require arranging and sorting the coordinates of points along intervals [155], but due to wrapping, angular coordinates cannot be arranged along intervals but along circles. However, the wrapping problem of

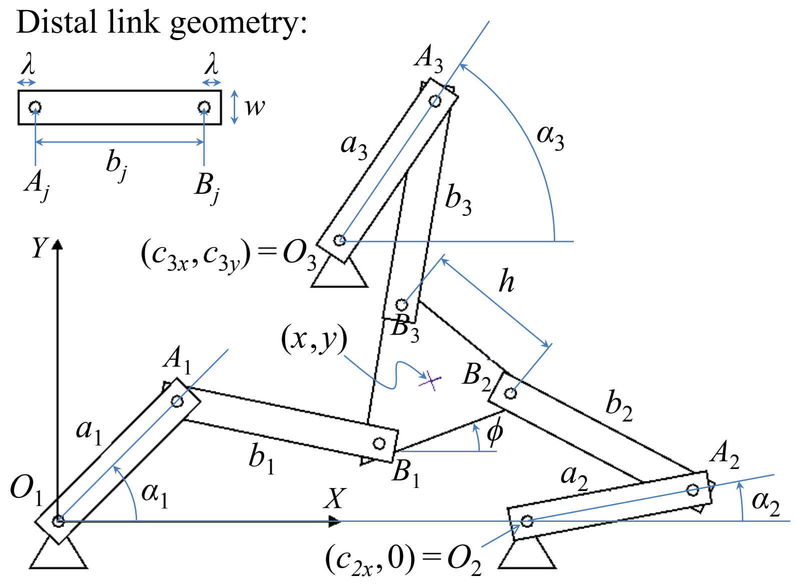


Figure 7.13: Planar 3$\underline{\text{R}}$RR parallel robot, and rectangular geometry of each distal link A_jB_j.

angles α_i can be easily avoided if we define the joint coordinates as the sines and cosines of these angles, i.e., we will consider that, in this robot, there are six "augmented" joint coordinates $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_6]^T$ defined as follows:

$$\begin{aligned} \theta_1 &= \cos\alpha_1 \quad & \theta_3 &= \cos\alpha_2 \quad & \theta_5 &= \cos\alpha_3 \\ \theta_2 &= \sin\alpha_1 \quad & \theta_4 &= \sin\alpha_2 \quad & \theta_6 &= \sin\alpha_3 \end{aligned} \tag{7.13}$$

These "augmented" joint coordinates will be subject to the following additional constraints:

$$\theta_1^2 + \theta_2^2 - 1 = 0 \tag{7.14}$$

$$\theta_3^2 + \theta_4^2 - 1 = 0 \tag{7.15}$$

$$\theta_5^2 + \theta_6^2 - 1 = 0 \tag{7.16}$$

In this way, we have doubled the dimension of the original joint space (which is now 6-dimensional), but the wrapping problem is now avoided and the proposed method can be applied exactly as described in Section 7.2. In this case, self-motion manifolds are curves in the 6-dimensional joint space $(\theta_1, \ldots, \theta_6)$. The task variables are the position coordinates $\mathbf{t} = [x, y]^T$ of the end-effector, and the passive variable is angle $\boldsymbol{\psi} = [\phi]^T$. The following restrictions can be derived from Figure 7.13:

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - a_1 \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} - \frac{h}{\sqrt{3}} \begin{bmatrix} \cos(\phi + \frac{\pi}{6}) \\ \sin(\phi + \frac{\pi}{6}) \end{bmatrix} \right\|^2 - b_1^2 = 0 \tag{7.17}$$

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_{2x} \\ 0 \end{bmatrix} - a_2 \begin{bmatrix} \theta_3 \\ \theta_4 \end{bmatrix} - \frac{h}{\sqrt{3}} \begin{bmatrix} \cos(\phi + \frac{5\pi}{6}) \\ \sin(\phi + \frac{5\pi}{6}) \end{bmatrix} \right\|^2 - b_2^2 = 0 \tag{7.18}$$

$$\left\| \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} c_{3x} \\ c_{3y} \end{bmatrix} - a_3 \begin{bmatrix} \theta_5 \\ \theta_6 \end{bmatrix} - \frac{h}{\sqrt{3}} \begin{bmatrix} \cos(\phi - \frac{\pi}{2}) \\ \sin(\phi - \frac{\pi}{2}) \end{bmatrix} \right\|^2 - b_3^2 = 0 \tag{7.19}$$

where a_j are the lengths of proximal links O_jA_j, b_j are the lengths of distal links A_jB_j, and $\{c_{2x}, c_{3x}, c_{3y}\}$ determine the positions of joints O_2 and O_3 as illustrated in Figure 7.13. Equations (7.14)-(7.19) constitute the system (7.2) defining the self-motion manifolds in this particular example.

Since in this example self-motion manifolds are 1-dimensional, Algorithm 5 reduces to sweeping each of the six axes θ_j of the joint space independently, and calculating all five unknown joint coordinates (and passive angle ϕ) in terms of the swept joint coordinate in each case. Note that joint coordinates in this example must be swept between $\theta_j^{min} = -1$ and $\theta_j^{max} = 1$, since the "augmented" joint coordinates are sines or cosines. Next, we will describe how to solve all five unknown joint coordinates and ϕ from Equations (7.14)-(7.19) in terms of the joint coordinate that is swept in each iteration of Algorithm 5 (line 7), for given values of the task variables (x, y).

Consider first the case when θ_1 (or θ_2) is swept. If θ_1 (or θ_2) is known, we can easily solve θ_2 (respectively, θ_1) from Equation (7.14). Then, since we know the values of (θ_1, θ_2), the only unknown in Equation (7.17) is the passive angle ϕ. Equation (7.17) has the following form: $C \cdot \cos\phi + S \cdot \sin\phi + I = 0$, which can be easily solved by transforming it into a quadratic equation in $\tan(\phi/2)$, as explained earlier in this section. After solving ϕ from this quadratic equation, the positions of joints B_2 and B_3 are known. This allows us to solve analytically the (well-known) inverse kinematic problem of the non-redundant 2-DOF serial revolute chains $O_2A_2B_2$ and $O_3A_3B_3$, obtaining angles α_2 and α_3. Knowing (α_2, α_3), the remaining four unknown joint coordinates $(\theta_3, \theta_4, \theta_5, \theta_6)$ can be directly computed using Equation (7.13). Following this procedure, ϕ and all joint coordinates are obtained in terms of the swept joint coordinate (θ_1 or θ_2).

Since all three legs of the 3$\underline{\text{R}}$RR robot have the same topology, it is straightforward to extend the procedure described in the previous paragraph to the cases where other joint coordinates are swept. For example, when sweeping θ_3 (or θ_4), θ_4 (resp. θ_3) is solved from Equation (7.15). Then, ϕ is solved from Equation (7.18), which is quadratic. Next, the IK problems of serial chains $O_1A_1B_1$ and $O_3A_3B_3$ are solved, obtaining (α_1, α_3). Finally, $(\theta_1, \theta_2, \theta_5, \theta_6)$ are computed from Equation (7.13). It is easy to extend this procedure to the cases where θ_5 or θ_6 are swept.

For the examples that will be illustrated next, we will consider that there are not joint limits and that distal links A_1B_1, A_2B_2, and A_3B_3 move in the same plane and, therefore, their mechanical interference should be forbidden. For collision testing, we will consider that distal links have rectangular shape with the dimensions indicated in Figure 7.13: their length is $(b_j + 2\lambda)$, and their width is w. The collision test between a pair of distal links (rectangles) can be easily performed using the Separating Axis Theorem.

Next, we will apply the proposed method to an example of a 3$\underline{\text{R}}$RR robot with the following parameters: $a_j = 0.18$, $b_j = 0.22$, $h = 0.15$, $c_{2x} = 0.5$, $c_{3x} = c_{3y} = 0.3$, $2\lambda = w = 0.035$ (this geometry precisely corresponds to the robot depicted in Figure 7.13). The method described in Section 7.2 is applied to obtain the barriers of the reachable workspace of this robot in the following box of the task space: $T = [0, 0.5] \times [-0.19, 0.41]$. The following parameters are used to execute the proposed method: $N_s^j = N_{t_i} = 200$, $\sigma_c = 1.5$, $\sigma_m = 2$. The calculation of barriers in this example is again distributed over $N_p = 8$ processes working in parallel. In this case, box T is divided into eight equal parts along the vertical y axis (as indicated in Figure 7.14), such that each part is assigned to a different process.

Figure 7.14 shows the boundaries and barriers of the reachable workspace for the considered example, when omitting collisions between distal links (i.e., when allowing their mechanical interference). Algorithm 8 takes about 27 minutes to obtain the result shown in Figure 7.14. As it can be observed in Figure 7.14, if collisions between distal links are omitted, then there are eight disjoint interior barriers inside the reachable workspace. At these interior barriers, two serial limbs $O_jA_jB_j$ are simultaneously completely stretched or folded, as indicated in Figure 7.14.

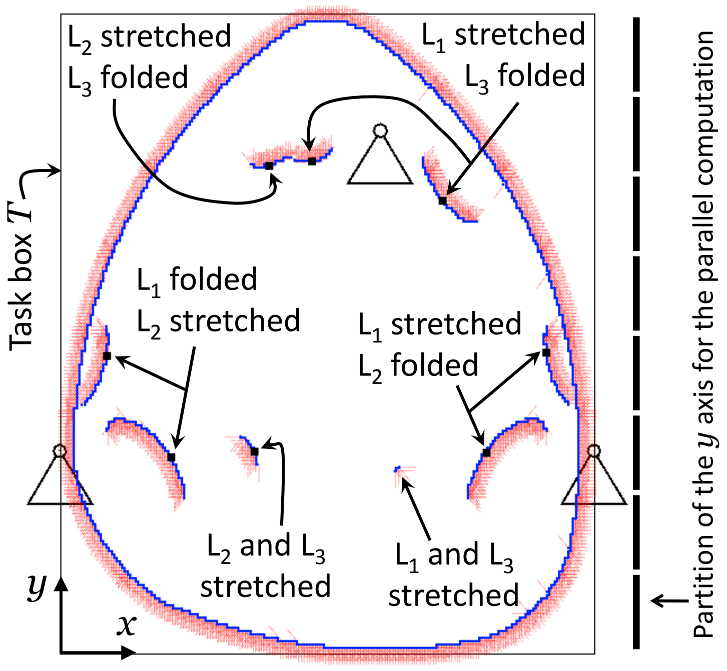


Figure 7.14: Reachable workspace of a 3$\underline{\text{R}}$RR parallel robot, including interior barriers, when omitting collisions. Fixed joints of the robot are represented for reference. Each serial limb $O_jA_jB_j$ is denoted by L_j.

Next, the computation is repeated but forbidding mechanical interferences between all three distal links. In that case, Algorithm 8 requires about 17 minutes to obtain the result shown in Figure 7.15. As Figure 7.15 shows, in this example forbidding the collisions between distal links does not affect the boundaries of the reachable workspace, but interior barriers change drastically. When allowing distal links to interfere, there are only a few small interior barriers (Figure 7.14). However, forbidding their interference generates many large and intricate interior barriers (Figure 7.15).

This example reveals an interesting problem related to the measurement of the "efficiency" of a workspace when considering collisions (e.g., with the purpose of

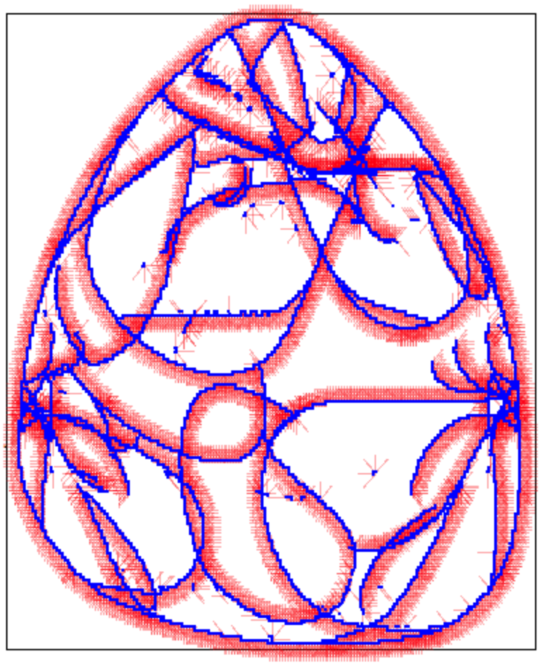


Figure 7.15: When forbidding collisions between distal links, interior barriers of the reachable workspace shown in Figure 7.14 change drastically.

comparing different robot designs). One possible way to measure the efficiency η of a workspace involving collisions consists in dividing the area $\mathcal{A}_c$ of the workspace obtained when considering collisions over the area $\mathcal{A}_t$ of the theoretical workspace obtained when omitting collisions, i.e.: $\eta = \mathcal{A}_c / \mathcal{A}_t$ [45], where $0 \leq \eta \leq 1$. Since in this example the boundary (shape) of the workspace is not affected when considering collisions (and the collision workspace of Figure 7.15 contains no holes inside its boundaries), we obtain $\eta = 1$. This may erroneously suggest that the workspace of the design analyzed in this section is not affected by collisions between distal links, but this is because the previous metric ignores the internal structure of the workspace. In this case, a more sophisticated metric should be defined, a metric which takes into account not only the area of the workspace but also the distribution of its interior barriers. However, the definition of such a metric is left for future research.

7.4 A First Approach to the Interior Barriers of the HyReCRo Robot

As previous examples have demonstrated, the computation time of the proposed method drastically increases when increasing the dimension of self-motion manifolds. For example: according to table 7.1, the computation of the barriers of the Stewart platform with 2D self-motion manifolds is ten times higher than the necessary time to compute the barriers when having 1D manifolds. This means that the method presented in previous sections is not suitable for the HyReCRo robot, since this robot has 4D manifolds in a 10D ambient space of active joint coordinates (see section 5.4.4): in that case, in order to densely sample these 4D manifolds, Algorithm 5 would need to sweep *four-dimensional grids* for $\binom{10}{4}$=*210 different parameterizations*, which clearly implies a huge amount of calculations.

Although the method presented in previous sections is not suitable for obtaining the interior barriers of the HyReCRo robot, we still can devise an alternative algorithm to obtain a first approximation of the interior barriers of the HyReCRo robot, as explained next. The method presented in previous sections identifies interior barriers with the vanishing of self-motion manifolds. Recall from chapter 5 that we can obtain and represent the solutions of the inverse kinematics of the HyReCRo robot as feasible regions R_f in two- or three-dimensional spaces. Actually, these regions R_f are projections of the self-motion manifolds of the HyReCRo robot on lower-dimensional spaces. Thus, we can still identify some interior barriers with the vanishing of some of these projections R_f.

Note that this alternative method (based on studying the lower-dimensional projections of self-motion manifolds) is not completely rigorous, since we may lose information when projecting self-motion manifolds on lower-dimensional spaces (e.g., the projections of large manifolds may mask the projections of smaller manifolds whose vanishing may not be detected due to this masking). Nevertheless, in this section we will admit this possible loss of information in exchange of obtaining a first approximation of the interior barriers of the HyReCRo robot.

7.4.1 Workspace Barriers and Empty Feasible Regions R_f

In this section, we will describe how the solution of the inverse kinematic problem, described in section 5.4, can be used to obtain the workspace of the HyReCRo robot, including its external boundaries and interior barriers. In addition to joint limits (which are automatically taken into account in the algorithm presented in section 5.4.5 for computing the feasible regions R_f), we will also impose the condition that mechanical interferences between the legs of the robot should be forbidden. As in the previous chapter, we will check the occurrence of mechanical interferences between the legs of the HyReCRo robot using the Separating Axis Theorem.

According to section 5.4, the solution of the inverse kinematic problem of the HyReCRo robot can be summarized as follows. Given a desired relative position and orientation between the feet of the robot, encoded by the homogeneous transformation matrix $\mathbf{T}_{B/A}$ defined in Equation (5.41), the solution of the inverse kinematics depends on R_{33}, which is the third element of the third row of $\mathbf{T}_{B/A}$:

- If the desired relative orientation satisfies $R_{33}^2 \neq 1$, then the solution to the inverse kinematics can be represented by a 2D feasible region R_f in plane (φ_{1B}, y_B), such that any point of this region corresponds to a posture that allows the robot to attain the desired position and orientation $\mathbf{T}_{B/A}$ satisfying the joint limits of the linear actuators and guaranteeing that the legs of the robot do not interfere. Moreover, there exist four different branches for the solution, i.e. four different feasible regions R_f corresponding to the four possible combinations of binary variables σ_1 and σ_2 (section 5.4.3.1).

- If the desired relative orientation satisfies $R_{33}^2 = 1$, the solution to the inverse kinematics can be represented by a 3D feasible region R_f in space $(\varphi_{1B}, \varphi_{2B}, y_B)$, such that any point of R_f yields a posture that allows the robot to attain the desired position and orientation $\mathbf{T}_{B/A}$ satisfying the joint limits of the linear actuators and guaranteeing that the legs of the robot do not interfere. In this case, there exist two different branches for the solution, i.e. two different feasible regions R_f, one region per each value of the binary variable σ_2 (section 5.4.3.2).

Once the solution to the inverse kinematics is available (including all branches of this solution), the workspace boundaries and barriers can be obtained using a discretization algorithm explained next, which is a (very) simplified version of Algorithm 8.

The workspace of the HyReCRo robot can be defined as the set of positions and orientations that foot B can attain with respect to foot A. Such a workspace is a six-dimensional set, since the position and orientation of foot B relative to foot A can be represented by three translations $\mathbf{p} = [p_x, p_y, p_z]^T$ and three rotations $\mathbf{r} = [\alpha, \beta, \gamma]^T$ (where α, β and γ are Euler angles). To calculate the workspace, the six-dimensional space of variables $\{p_x, p_y, p_z, \alpha, \beta, \gamma\}$ is discretized into a regular grid of nodes. For example, we may approximate each axis of this six-dimensional space by n_d nodes regularly distributed between two limits, which yields an overall grid of n_d^6 nodes. Then, for each node of this grid, the inverse kinematic problem is solved to check if the node is attainable by each branch of the solution of the inverse kinematics.

For each branch i of the solution to the inverse kinematics, we create a list WS_i of the nodes that can be reached using that branch. After the algorithm has checked all the nodes of the grid, the list WS_i is an approximation of the workspace associated to branch i, since it contains all positions and orientations that can be reached with that branch. Then, the boundaries of the workspace associated to the i-th branch can be approximated by the nodes contained in WS_i which have at least one unreachable neighboring node (i.e., a neighboring node not contained in WS_i). After obtaining the boundaries of the workspaces associated to the different branches, the boundaries of all branches can be joined to obtain the boundaries and barriers of the complete workspace.

To determine if an arbitrary node of the grid is attainable using a given branch i of the inverse kinematics, the feasible region R_f associated to that branch is calculated using the Monte Carlo Algorithm 1 described in section 5.4.5. If the region R_f is empty, then it is considered that the node is not attainable by the i-th branch, and the node is not included in list WS_i (see Figure 7.16). Algorithm 1 generates the region R_f by randomly sampling points in plane (φ_{1B}, y_B) [or in space $(\varphi_{1B}, \varphi_{2B}, y_B)$, if $R_{33}^2 = 1$]. If the posture generated by each randomly sampled point satisfies the joint limits of the linear actuators, and if the legs do not interfere, then that point is stored as a point of feasible region R_f. In this way, a discrete approximation of the region R_f can be obtained by sampling a large number of points in plane (φ_{1B}, y_B) [or in space $(\varphi_{1B}, \varphi_{2B}, y_B)$].

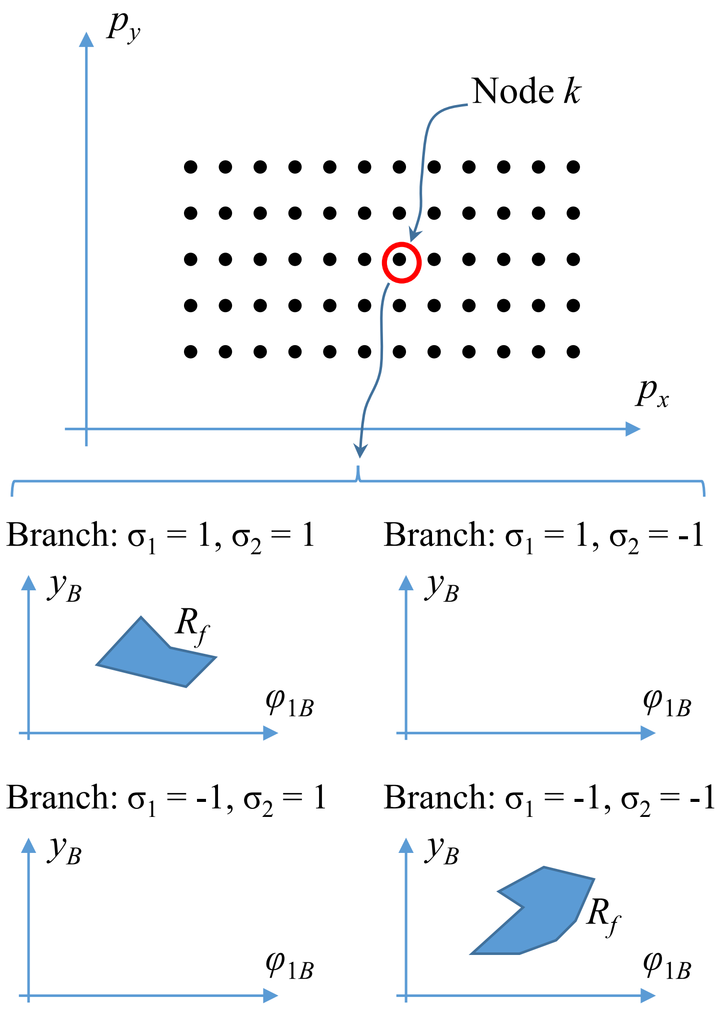

p_y
Node k
p_x
Branch: $\sigma_1 = 1, \sigma_2 = 1$
y_B
R_f
φ_{1B}
Branch: $\sigma_1 = 1, \sigma_2 = -1$
y_B
φ_{1B}
Branch: $\sigma_1 = -1, \sigma_2 = 1$
y_B
φ_{1B}
Branch: $\sigma_1 = -1, \sigma_2 = -1$
y_B
R_f
φ_{1B}

Figure 7.16: A 2D example that illustrates the process to determine whether a given node k of the potential workspace is reachable or not. For a given node k, the feasible regions R_f are calculated using all branches of the solution of the inverse kinematic problem. In this case, the node k belongs only to the workspaces of branches $(\sigma_1 = 1, \sigma_2 = 1)$ and $(\sigma_1 = -1, \sigma_2 = -1)$ since these are the only branches leading to non-empty feasible regions R_f.

Note that it is sufficient to find a single point belonging to R_f to guarantee that R_f is not empty and classify the corresponding node of the workspace as attainable. However, checking if R_f is empty (and classifying the corresponding node as unattainable by the i-th branch of the inverse kinematics) is not that easy, since one would need to explore exhaustively plane (φ_{1B}, y_B) [or space $(\varphi_{1B}, \varphi_{2B}, y_B)$] in order to guarantee that this plane (space) does not have points satisfying all constraints (i.e., to guarantee that R_f is empty). Since it is not feasible to perform such an exhaustive search to check if R_f is empty, this problem is practically solved by establishing a maximum number n_a of attempts to generate a point in R_f. Then, Algorithm 1 begins to randomly sample points in plane (φ_{1B}, y_B) [or in space $(\varphi_{1B}, \varphi_{2B}, y_B)$, if $R_{33}^2 = 1$]. If it finds a point that satisfies all constraints (joint limits and no-interference), Algorithm 1 stops: the region R_f is not empty (it contains at least one point) and the node is classified as attainable. If, on the contrary, Algorithm 1 has sampled n_a random points and none of them satisfies the constraints, it is considered that R_f is empty, which means that the corresponding node cannot be attained by the considered branch of the inverse kinematics. Obviously, this method will be more accurate when n_a increases, but the computation will also take more time, so a compromise between precision and computational cost is necessary.

It should be remarked that the method described in this section to compute the workspace of the HyReCRo robot is very computer-intensive if we try to discretize and compute the six-dimensional workspace of variables $\{p_x, p_y, p_z, \alpha, \beta, \gamma\}$, since the number of nodes to check is n_d^6 (and, for each node, the feasible regions R_f must be obtained for the different branches of the inverse kinematics). Moreover, a six-dimensional workspace cannot be represented graphically. For these reasons, and to decrease the computational cost, we will fix some of these six variables to obtain lower-dimensional workspaces that can be represented graphically and are more easy to understand, such as the constant-orientation workspace. In the following section, we will illustrate the algorithm described in the present section using some examples.

7.4.2 Examples

This section presents some examples of the application of the method described above to compute different workspaces of the HyReCRo robot, including some interior barriers. For the next examples, the geometric design parameters of the robot are: $b = p = 4$, $h = 16$, $t = 15.6$, $\rho_0 = 19$, and $\Delta\rho = 7.5$ (all in cm). In all examples of this section, we will discretize the workspace into $n_d = 200$ points along each axis. Moreover, we will sample a maximum of $n_a = 5000$ random points before deciding that the feasible region R_f (of valid postures that yield a given position and orientation between the feet) is empty. In all examples, we will assume that foot A is firmly attached to the structure, and we will compute the set of positions that foot B (which is free to move) can reach.

7.4.2.1 Example 1

In this example, we are interested in obtaining a constant-orientation workspace, i.e. the set of attainable points by foot B when the relative orientation between the feet is constant. More specifically, we will study the workspace obtained when both feet have the same orientation, which means that the rotation submatrix of $\mathbf{T}_{B/A}$ is the identity matrix. Furthermore, we will be interested only in the intersection of such constant-orientation workspace with plane $p_z = 0$, to study the planar motions of the robot inside this constant-orientation workspace. Note that the robot needs to perform motions of this type to travel along a beam of a structure, as shown in Figure 7.17. The desired position and orientation will have the following form for this example:

$$\mathbf{T}_{B/A} = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.20}$$

To apply the algorithm described in the previous section, we will define a box $\mathcal{B}$ that encloses the workspace in plane (p_x, p_y), and we will discretize this box into a grid of 40000 nodes regularly distributed (200 nodes per each axis). Then, for each node, we will solve the inverse kinematic problem using the matrix of Equation (7.20) as the input, to check if each node is attainable by the different branches of the inverse kinematics. The chosen box for this example is $\mathcal{B} = \{(p_x, p_y) : -80 \text{ cm} \leq p_x \leq 80 \text{ cm}, -50 \text{ cm} \leq p_y \leq 50 \text{ cm}\}$. Running the algorithm described in the previous section for these parameters, the workspace shown in Figure 7.17 is obtained. Note that, since in this case $R_{33}^2 = 1$, according to section 5.4.3 the solution to the inverse kinematic problem in this case has two branches: one for $\sigma_2 = 1$ and other for $\sigma_2 = -1$. The workspaces associated to these branches are shown in Figure 7.17 with different colors.

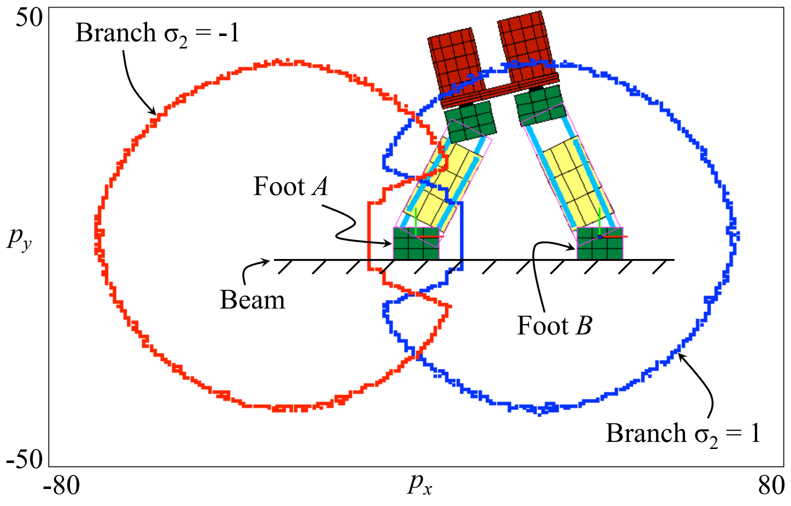


Figure 7.17: Planar constant-orientation workspace that provides the points at which foot B can be placed with the same orientation as foot A. This constant-orientation workspace is useful for planing longitudinal movements along the direction of the beam. The workspaces associated with the branches $\sigma_2 = -1$ and $\sigma_2 = 1$ are represented in red and blue colors, respectively.

Note that this workspace is split into the components associated with the two branches of the solution to the inverse kinematic problem. The complete workspace (i.e. the union of the two components) has a void around foot A, which is necessary for avoiding interferences between the legs. According to Figure 7.17, the robot cannot move the foot B from one side of the workspace (e.g. the right half of the workspace, associated with branch $\sigma_2 = 1$) to the other side (e.g. the left half, associated with branch $\sigma_2 = -1$) keeping constant the orientation between the feet, since the workspaces associated with both branches have boundaries in the middle of the workspace, both above and below foot A. This is illustrated in Figure 7.18, where the robot starts at an initial point in the workspace associated with branch $\sigma_2 = 1$, and tries to describe a trajectory towards the left half of the workspace (see Figure 7.18a). However, the trajectory cannot be completed because the robot cannot cross the boundaries of the component of the workspace in which it moves. This boundary is originated from the fact that the legs cannot interfere: as Figure 7.18b shows, when foot B is close to the mentioned boundary, both legs are about to collide.

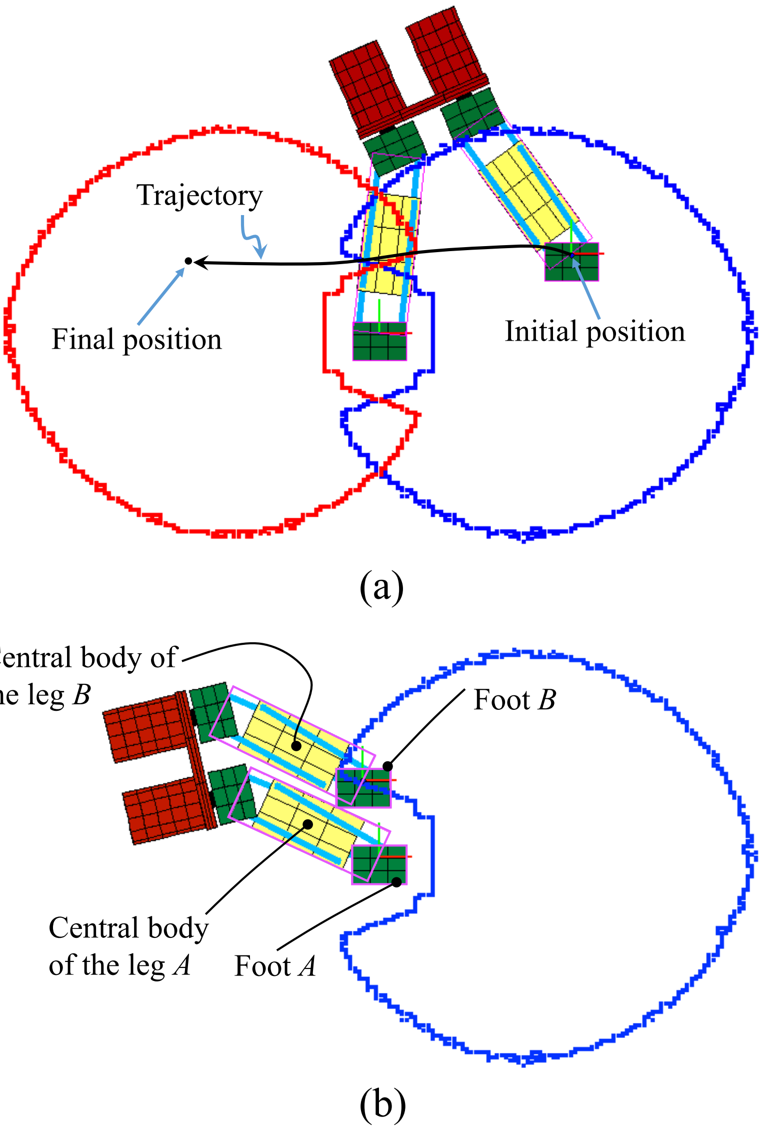


Figure 7.18: (a) A trajectory between both components of the workspace. (b) The robot cannot reach the left component because it cannot cross a boundary of the component in which it moves (the right component). When approaching the boundary, the legs are about to collide: foot B is almost touching the central body of leg A. To cross the boundary, an interference between these two bodies would be necessary.

As in the previous chapter, the Separating Axis Theorem [52] is used for checking the interference between the legs of the HyReCRo robot. Each leg can be approximated by the union of two *cuboids* (also known as *rectangular parallelepipeds*): one cuboid encloses the foot, and the other cuboid encloses the central body of the leg, including the linear actuators (these cuboids are represented in magenta in Figure 7.18b). Then, the two legs will interfere if one of the cuboids of one leg intersects one of the cuboids of the other leg. Since the cuboids are convex shapes, the Separating Axis Theorem can easily be used to check if they intersect.

7.4.2.2 Example 2

In this example, the objective is to find the planar constant-orientation workspace defined by the following homogeneous transformation matrix between the feet of the robot:

$$\mathbf{T}_{B/A} = \begin{bmatrix} 0 & -1 & 0 & p_x \\ 1 & 0 & 0 & p_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.21}$$

This orientation is a rotation of 90° about the Z axis, which is necessary for performing a transition between two perpendicular beams in a structure, as shown in Figure 7.19. Again, we are interested only in the intersection of this constant-orientation workspace with the plane $p_z = 0$, since the motion necessary to perform such a transition is planar.

Next, the algorithm described in Section 7.4.1 is applied, discretizing the following box $\mathcal{B}$ into a grid of 40000 nodes (200 nodes per axis): $\mathcal{B} = \{(p_x, p_y) : -40 \text{ cm} \leq p_x \leq 80 \text{ cm}, -40 \text{ cm} \leq p_y \leq 80 \text{ cm}\}$. The resulting workspace is shown in Figure 7.19. Again, since the desired orientation satisfies $R_{33}^2 = 1$, the solution to the inverse kinematic problem has two branches, one for $\sigma_2 = 1$ (shown in blue in Figure 7.19) and other for $\sigma_2 = -1$ (shown in red in Figure 7.19).

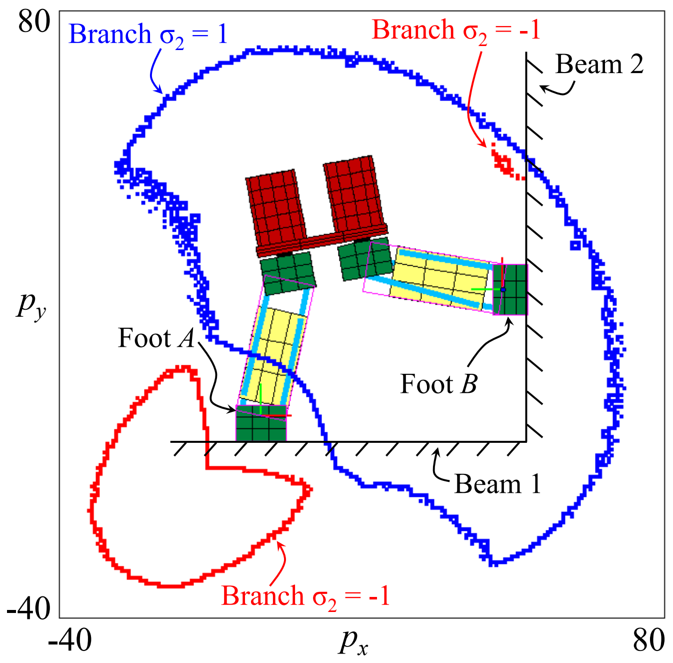


Figure 7.19: Planar constant-orientation workspace containing all the points of the plane $p_z = 0$ that can be reached with foot B rotated 90° about the Z axis with respect to foot A.

In this case, the workspace attainable using the branch $\sigma_2 = -1$ of the solution to the inverse kinematic problem is smaller than the workspace associated with branch $\sigma_2 = 1$. Note that the workspace associated with branch $\sigma_2 = -1$ has two components:

a big one, which is close to foot A in Figure 7.19, and a smaller one, which is close to beam 2 in the same figure. Actually, the shape of the smaller component cannot be appreciated very well in Figure 7.19. However, a more precise approximation of that component can be obtained if the algorithm described in Section 7.4.1 is executed again discretizing a smaller box $\mathcal{B}$ that encloses only the area around the mentioned small component of the workspace, instead of using a big box that contains all components as shown in Figure 7.19.

The posture of the robot shown in Figure 7.19, with foot B placed on beam 2, is obtained using the branch $\sigma_2 = 1$ of the solution to the inverse kinematics. However, according to the same figure, it would be also possible to place foot B on beam 2 using branch $\sigma_2 = -1$, since this branch yields a small component of the workspace near beam 2 (i.e., the small component described in the previous paragraph). This is checked in Figure 7.20, which shows a posture of the robot that places foot B at a point of the smallest of the two workspace components associated with branch $\sigma_2 = -1$. As this figure shows, using branch $\sigma_2 = -1$, foot B can also be effectively placed on beam 2 with the desired orientation. However, in this posture, the hip of the robot intersects the beams, so this would not be a feasible solution in practice. This unfeasibility can be easily detected by the presented method if we include the condition that no part of the robot should intersect the obstacles of the environment, using a similar procedure to the method used for checking if different legs intersect, described in subsection 7.4.2.1.

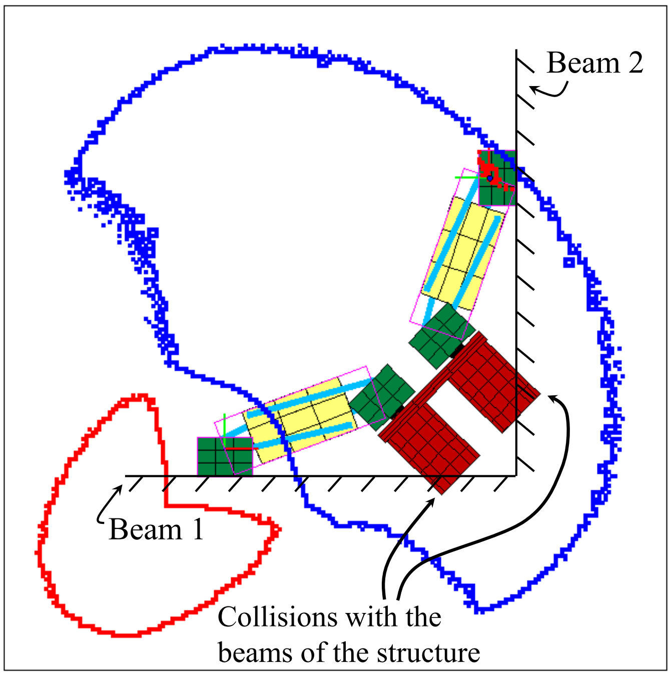


Figure 7.20: A posture which places foot B on beam 2 with the desired orientation, using branch $\sigma_2 = -1$. In this posture, the robot collides with the structure.

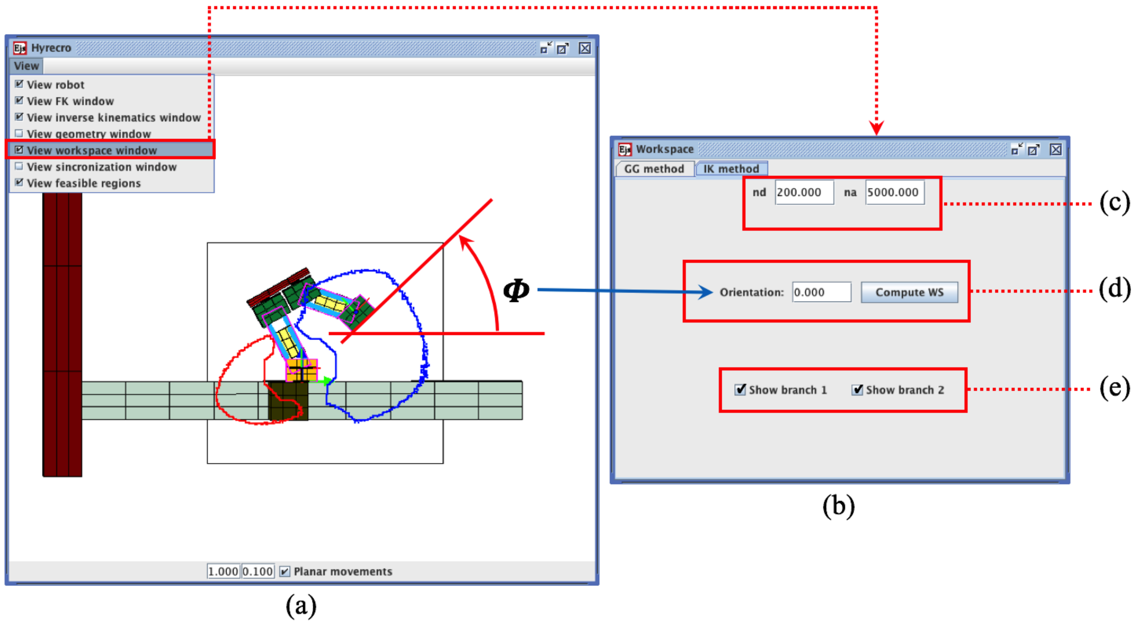


Figure 7.21: Computing the workspace of the HyReCRo robot in the developed simulator, using the method proposed in Section 7.4.

7.5 Simulator of the HyReCRo Robot: Interior Barriers

The simulator presented in Section 5.5 can also be used for obtaining the workspace of the HyReCRo robot using the method described in Section 7.4. To that end, the user must click the "View workspace window" option in the "view" menu at the top of the main window of the simulator (see Figure 7.21a). When doing this, the window shown in Figure 7.21b pops up. This window has two tabs: "GG method" (which was analyzed in Section 6.6) and "IK method". In this section, we will focus on the content of the second tab.

By means of the window shown in Figure 7.21b, the user can compute and visualize the workspace of the HyReCRo robot using the method presented in Section 7.4. When applying this method in the simulator, it is assumed that the robot adopts planar postures with the relative orientation between its feet satisfying $R_{33}^2 = 1$, i.e., Case 2 of the inverse kinematic problem of this robot is considered. This is because many of the important postures necessary for exploring structures (i.e., those postures required for performing plane transitions) are planar postures satisfying $R_{33}^2 = 1$, as we have observed during the previous chapters of this thesis. Moreover, the computed workspaces are constant-orientation workspaces, considering that the orientation Φ of the free foot is constant (angle Φ is indicated in Figure 7.21a).

For computing the constant-orientation workspace according to the method described in Section 7.4, the user must first define the box $\mathcal{B}$ in which the workspace will be computed. This box is defined in tab "GG method", as described in Section 6.6. Note that, since in the present section we are dealing with planar workspaces, the algorithm will only consider the dimensions of box $\mathcal{B}$ along axes X and Y. Next, the user must define the number n_d of nodes into which the X and Y axes of box $\mathcal{B}$ will be discretized, as well as the maximum number n_a of points which Algorithm 1 will attempt to generate before considering that the feasible region R_f at a given workspace point is empty. n_d and n_a are introduced by the user as indicated in Figure 7.21c.

After this, the user specifies the desired orientation Φ (in rad) for the free foot, as indicated in Figure 7.21d. Next, pressing the button "Compute WS" will launch the execution of the method described in Section 7.4, which will represent in Figure 7.21a the computed workspaces after some time (typically, some minutes, depending on the values chosen for n_d and n_a). Since the computed constant-orientation workspaces consider that the orientation between the feet satisfy the condition $R_{33}^2 = 1$ (Case 2 of the inverse kinematics, which has two solution branches due to the binary variable $\sigma_2 \in \{-1, 1\}$), we will obtain two workspaces, one for $\sigma_2 = 1$ (in blue) and another for $\sigma_2 = -1$ (in red). Using the tick-boxes shown in Figure 7.21e, the user can activate or deactivate the visualization of these two workspaces.

7.6 Conclusions

This chapter has investigated the interior barriers of the workspaces of redundant manipulators, like the HyReCRo robot. These barriers imply motion impediments for the robot and depend on the considered kinematic constraints, such as joint limits or collision constraints. Collision constraints are important for the HyReCRo robot, since this robot should not collide with itself or with the beams of a structure when climbing it. After reviewing previously existing methods for computing the interior barriers of redundant robots (Section 7.1), it has been concluded that existing methods cannot easily accommodate collision constraints, and a need for designing a new method able to easily handle such constraints has been identified. In order to design this new method, the relationship between self-motion manifolds and interior barriers has been investigated, concluding that, when disjoint components of these manifolds vanish, the robot encounters interior barriers.

Based on this property (manifold vanishes $\Rightarrow$ workspace barrier occurs), a new method has been proposed in Section 7.2 for obtaining the interior barriers of the workspace of redundant robots under arbitrarily complex collision constraints. This method consists of three stages: first, manifolds are densely sampled, discarding samples that do not satisfy collision constraints. Then, non-discarded samples are clustered using kd-trees, in order to identify disjoint self-motion manifolds. Finally, disjoint manifolds identified at neighboring workspace points are compared and matched, in order to detect if some manifold vanishes when traveling between these two workspace points, in which case a workspace barrier is identified between these points. As the tests conducted in Section 7.3 have demonstrated, the proposed method is feasible for robots whose self-motion manifolds are one- and two-dimensional, and collision constraints alter drastically the distribution of interior barriers inside the workspace. However, the proposed method is not completely appropriate for robots with higher-dimensional manifolds, like the HyReCRo robot, since in those cases the execution of the method is very computer-intensive and may take prohibitive computation times.

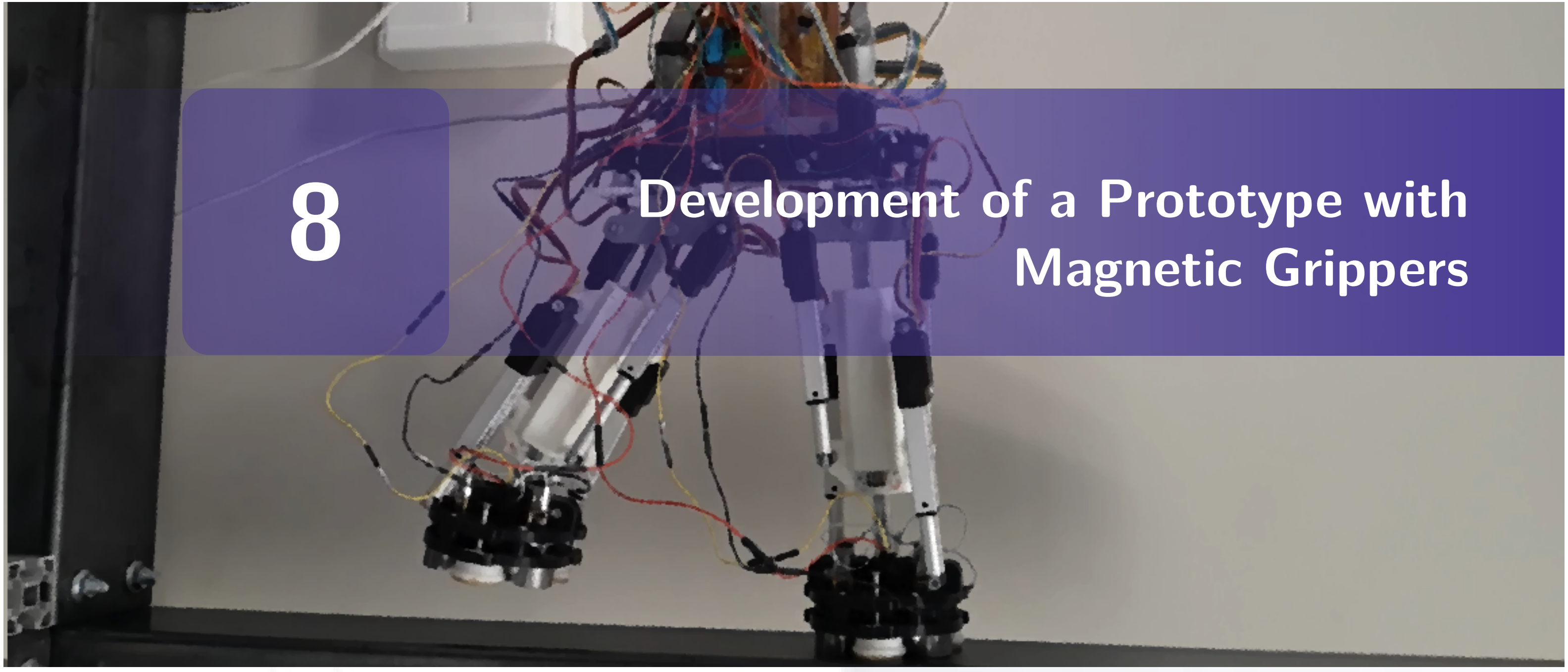
In order to try to extend the proposed method to the HyReCRo robot, whose self-motion manifolds are four- or five-dimensional (depending on the relative orientation between the feet), an approximate variant of the previous method has been proposed in Section 7.4, which consists in analyzing lower-dimensional projections of these self-motion manifolds (these projections are the feasible regions R_f defined in Section 5.4.5). These lower-dimensional projections allow for the computation of some interior barriers of the workspace of the HyReCRo robot, although it is not guaranteed that all interior barriers of this robot are detected using this method. Thus, this method based on lower-dimensional projections of self-motion manifolds will have to be refined in the future, in order to avoid missing barriers.

Finally, the approximate method proposed in Section 7.4 has been implemented in the simulator of the HyReCRo robot (Section 7.5).

7.7 Publications Related to this Chapter

The main results presented in this chapter are related to the following publications:

- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. *Mechanism and Machine Theory*, 128:84 – 109, 2018 [146] **(SCI-JCR Impact Factor: 2.796, Q1)**.
  - This paper presents the method proposed in Section 7.2 for obtaining the interior barriers of the workspace of redundant manipulators under collision constraints.
- A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Calculation of the boundaries and barriers of the workspace of a redundant serial-parallel robot using the inverse kinematics. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 412–420, 2016 [147].
  - This paper presents the method proposed in Section 7.4, which constitutes a first approach to the computation of the interior barriers of the HyReCRo robot.



8 Development of a Prototype with Magnetic Grippers

The design of a climbing robot comprises two main parts: design of the *grippers* with which the robot adheres to the climbed structure, and design of the kinematic chain or *manipulator* that connects these grippers and moves them to the different attachment points of the structure. The *manipulator* of the HyReCRo robot has been thoroughly analyzed during the previous chapters of this thesis. Therefore, in this chapter we will focus on the design of the *grippers* of the HyReCRo robot.

This chapter begins by presenting a functional prototype of the HyReCRo robot, which is able to perform the basic movements necessary for exploring three-dimensional steel structures (Section 8.1). The objective of this chapter is to design magnetic grippers for this prototype, so that it can adhere to steel structures and climb them. The magnetic grippers to be designed will be based on the technology of *switchable magnets* (SM), which is introduced in Section 8.2. Next, Section 8.3 reviews the basic movements necessary for climbing and exploring three-dimensional structures, and from these basic movements we derive the postures that will be used for designing the grippers. Later, Section 8.4 presents a conceptual design of these grippers, identifying their main design parameters.

This is followed by the design process of the grippers, which considers two criteria to guarantee the stability of the adhesion and avoid the fall of the robot:

- On the one hand, the notion of Zero Moment Point (ZMP) is used to guarantee that the grippers will not detach from the structure, which would produce the tip-over and fall of the robot (Sections 8.5 and 8.6).

- On the other hand, even if the grippers never detach from the structure, they may slip due to insufficient friction. Therefore, Section 8.7 presents a static friction analysis of the grippers, in order to determine the necessary static friction coefficient for avoiding slippage.

Finally, Section 8.8 presents some experiments performed with the prototype of the HyReCRo robot including the developed grippers. These experiments, in which the robot climbs a real steel structure, show that the developed grippers offer a stable adhesion even in the worst design postures.

8.1 A Prototype of the HyReCRo Robot

In previous chapters, we have performed the kinematic and workspace analyses of the HyReCRo robot. The next step in this thesis is to use these analyses to design and develop a functional prototype of the HyReCRo robot. To that end, the first step is to design the geometry of the prototype, i.e., to determine the necessary values of the six geometric design parameters of the robot $(b, p, h, t, \rho_0, \Delta\rho)$ so that the prototype can perform the necessary movements for exploring structures.

After a trial-and-error process aided by the developed simulator of the HyReCRo robot, described at the end of Chapters 5-7, the following geometric design was chosen: $b = 25$, $p = 31.5$, $t = 110$, $h = 70$, $\rho_0 = 100$, $\Delta\rho = 50$ (values in mm).

The chosen design allows the robot to perform concave transitions between different beams, as well as convex transitions between different faces of the same beam. By combining these two movements with the longitudinal advance along a beam, and with the two rotations of the hip (θ_A and θ_B), the HyReCRo robot can completely explore three-dimensional structures (see Section 8.3).

A prototype, shown in Figure 8.1, was developed with these geometric design parameters. The main characteristics of this prototype are listed next:

- The dimensions of the robot at its home configuration are: 250 $\times$ 500 $\times$ 120 (dimensions in mm).
- Its weight, excluding the magnetic grippers to be designed in this chapter, is $m_M = 1.55$ kg.
- Most parts of the robot are PLA 3D-printed parts, including also a few parts made of aluminum.
- Linear actuators used in the 2R$\underline{\text{P}}$R-PR parallel mechanisms of the legs are from manufacturer Actuonix [3] (model no. L12-50-210-12-P).
- DC Motors used in the hip are from manufacturer Maxon [112] (model A-max 22 with 590:1 gear box). These motors control rotations θ_A and θ_B, as indicated in Figure 8.1.

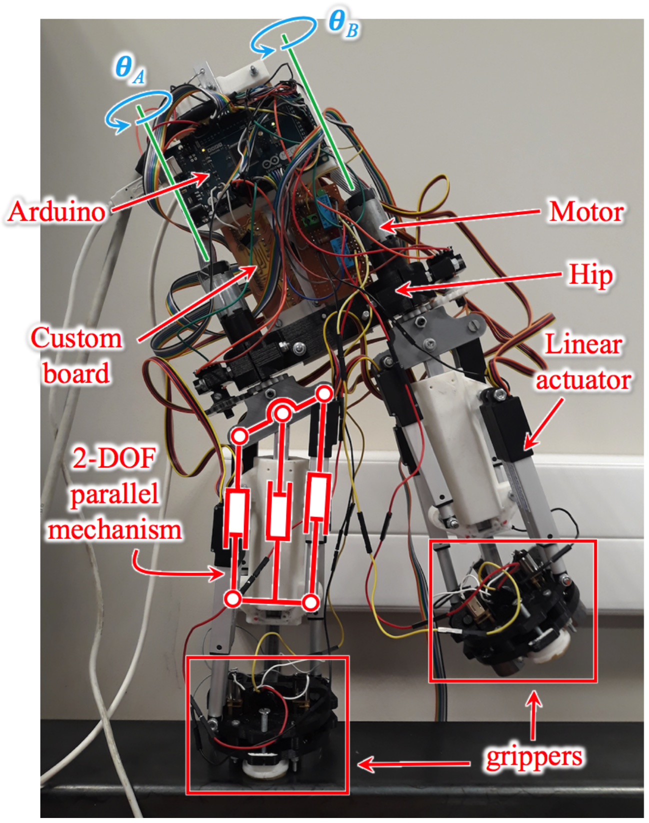


Figure 8.1: Prototype of the HyReCRo robot.

- The robot is controlled through a control unit mounted on the hip, which includes an Arduino Mega 2560 board and a custom-made amplification and filtering board interfacing between the Arduino board and all actuators. This custom-made board is powered at ± 12 V through a cable (currently, the robot is umbilical, without batteries).
- Currently, the robot is tele-operated through a gamepad. Through this gamepad, the user inputs desired incremental changes $\Delta\mathbf{t}$ of the position and/or orientation of the free gripper of the robot, and the necessary increments of the actuated joint coordinates $(\Delta\mathbf{q})$ are solved from the Jacobian relationship: $\Delta\mathbf{t} = \mathbf{J}\Delta\mathbf{q}$, which is solved through a pseudo-inverse solution since $\mathbf{J}$ is not square (the robot is redundant). The necessary increments $\Delta\mathbf{q}$ are passed to Proportional-Integral-Derivative (PID) control loops that control each actuator.
- Alternatively, the simulator of the HyReCRo robot described in previous chapters can also be used to control the prototype of the HyReCRo robot. For example, from this simulator it is possible to tune the gains of the PID controllers or set desired lengths for the linear actuators, as well as specify Cartesian trajectories for the free gripper of the prototype. However, this functionality of the developed simulator (trajectory planning) still needs to be polished, and the experiments at the end of this chapter will be conducted by directly moving the free gripper of the prototype with the gamepad.

To complete this prototype, the last step is to develop its magnetic grippers, which will be attached to the feet of the robot and will allow it to adhere to real steel structures and climb them. The objective of the present chapter is to design such magnetic grippers, which will be based on the technology of switchable magnets.

8.2 Switchable Magnets in Robotics

As explained in chapter 2, step-by-step structure-climbing robots like the HyReCRo robot (and general multi-legged climbing robots) must alternate the role of their grippers as they explore structures: during some phases of their motion, one of the grippers must be firmly attached to the structure, whereas the other gripper, which is moved by the manipulator to the next attachment point, must be free. Then, in other phases of the motion these roles are inverted and the previously free gripper is attached to the structure, whereas the previously attached gripper must be released in order to move it to the next anchor point of the environment. Therefore, it is necessary to use grippers with the ability to activate and deactivate their adhesion to the environment as required during each motion phase.

Traditionally, many climbing robots have relied on coils and electromagnets for activating and deactivating adhesion forces when climbing ferromagnetic structures [8, 89, 5, 97]. However, during the last years, an increasing number of climbing robots have been substituting electromagnets by switchable magnets, due to their advantages. Generally, a switchable magnet device is a magnetic circuit that includes permanent magnets such that, by moving some part of the circuit or these permanent magnets, it is possible to redirect the magnetic flux so that most of it either traverses the ferromagnetic substrate or is internally recirculated through the device. In the first case, it is said that the state of the switchable magnet is ON, and the switchable magnet strongly adheres to the ferromagnetic substrate. In the second case, the switchable magnet is OFF and the adhesion force to the ferromagnetic structure is negligible.

Under this working principle, it is possible to devise many typologies of switchable magnets [161], although some of the most widely used types are the so-called H-type SM devices, like the one illustrated in Figure 8.2a. The switchable magnet of Figure 8.2a is composed of a ferromagnetic housing and two cylindrical permanent magnets with diametrical magnetization. The lower magnet is firmly attached to the ferromagnetic housing (with its poles oriented along the principal axis of the housing), whereas the upper magnet can rotate with respect to the lower one. When the poles of the upper magnet are oriented anti-parallel to the poles of the lower magnet, the magnetic field is mainly closed between the two magnets, through the ferromagnetic housing (Figure 8.2c), and little magnetic flux circulates through the ferromagnetic substrate. In that case, the adhesion force between the ferromagnetic substrate and the switchable magnet is negligible and the device is at OFF state. When rotating the upper magnet by 180°, the poles of both magnets coincide and most of the magnetic flux circulates through the ferromagnetic substrate (Figure 8.2d), so the switchable magnet adheres to it (ON state).

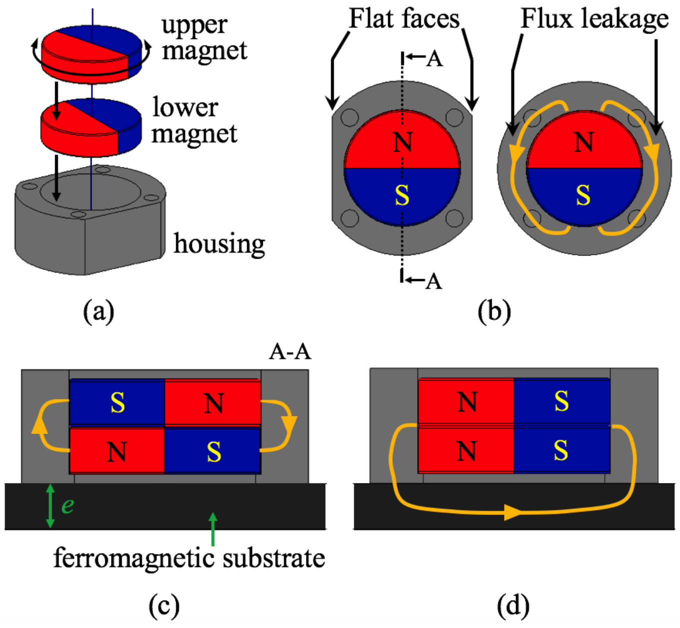


Figure 8.2: (a) Parts of a switchable magnet. (b) Flat faces reduce flux leakage. (c) Switchable magnet at OFF state. (d) Switchable magnet at ON state.

The adhesion force between the switchable magnet and the ferromagnetic substrate depends on many factors, such as [180]: the material of the permanent magnets, the air gap between the housing and the substrate, the thickness e of the substrate (the adhesion force increases with e), the material and shape of the ferromagnetic housing, etc. Regarding the influence of the shape of the housings on the adhesion force, as it can be observed in Figure 8.2b, housings are usually manufactured with two flat faces instead of having a cylindrical shape. The purpose of these faces is to minimize the magnetic flux leakage through the housing during the ON state, which would reduce the adhesion force (Figure 8.2b), enforcing most of the magnetic flux to traverse the substrate [180].

Switchable magnets have traditionally been used as holding devices in machining workshops and industries. In that case, the upper magnet of Figure 8.2a can be manually rotated by means of a knob. If this knob is substituted by a small servomotor, then one can automate the rotation of the upper magnet, which allows for the use of switchable magnets as variable adhesion devices in robotics, as many researchers have done during the recent years.

Schempf et al. [167] used a switchable magnet to allow the *Neptune* robot to climb and inspect storage tanks. The biped robot *MagGIE* [55] also uses SM to climb steel structures. Gilpin et al. [61] presented the *Miche* robots, which are identical robotic modules that can assemble by means of SM, forming arbitrary shapes. The *Hand-Bot* [20] climbs vertical structures by combining grippers with a rope which is anchored to the ceiling through a switchable magnet. Reference [70] presents the innovative design of a permanent-magnet wheel for a wall climbing robot; the magnetic flux can be conducted through the wheel or the ferromagnetic wall by inserting or removing an induction pin. Yao and Li [201] optimized the design of H-type switchable magnets, attaining high adhesion forces and ratios between adhesion forces at ON and OFF states. The inchworm robot *Tubulo II* [162] also uses SM to adhere to the inner walls of boiler tubes, in order to explore and visually inspect them. Rochat et al. [160] also present the *TREMO* robot, a modular inchworm robot which makes use of anchors having three SM symmetrically placed 120° apart. Magnenat et al. [109] present the *marXbot*, a mobile robot with an SM-based gripper to grasp ferromagnetic objects. Chen et al. [29, 31] designed and optimized a switchable magnetic device based on the linear Halbach array, and used it on a wheeled robot for climbing ferromagnetic walls [30]. Romão et al. [163] designed the *InchwormClimber*, a lightweight 1-degree-of-freedom climbing robot whose adhesion relied on switchable magnets combined with static (non-switchable) permanent magnets for increased safety. Finally, the omnidirectional climbing robot *OmniClimber* [177] uses SM in order to adhere to steel structures and perform transitions between different working planes with the help of an articulated arm.

Switchable magnets present several advantages over electromagnets used in climbing robots, namely [180]: they offer higher adhesion force per unit mass, lower power consumption (SM only consume power when rotating the mobile magnet for switching between the ON/OFF states), they are simpler to manufacture, and offer higher safety (in the event of power loss, adhesion will not be lost and the robot will not fall). Considering these advantages, in this chapter we will design magnetic grippers based on SM for the HyReCRo climbing robot. Before proceeding with the design, in next section we will recall the basic postures and movements which are necessary for exploring three-dimensional structures, which were introduced in previous chapters of this thesis.

8.3 Basic Movements for Exploring Structures

This section reviews the basic movements necessary for exploring three-dimensional structures. From these basic movements, we derive the postures that will be used for designing the magnetic grippers in later sections.

A three-dimensional metallic structure can be defined as a network of interconnected bars or beams. In order to navigate a structure, a climbing robot needs to perform three basic movements, which will be denoted by $\{L, E, I\}$ in this chapter:

- L: Longitudinal displacement along a beam.
- E: Exterior[1] (or convex) transition between two adjacent faces of the same beam.

[1] Note that, in previous chapters, "exterior" and "interior" transitions were called "convex" and "concave" transitions, respectively. However, in this chapter we will prefer to use the terms "exterior" and "interior" transitions (which are used by some researchers as synonyms of convex and concave transitions [44]) since they can be abbreviately and intuitively denoted by different letters (E, I), whereas both terms "convex" and "concave" begin with the same letter (C).

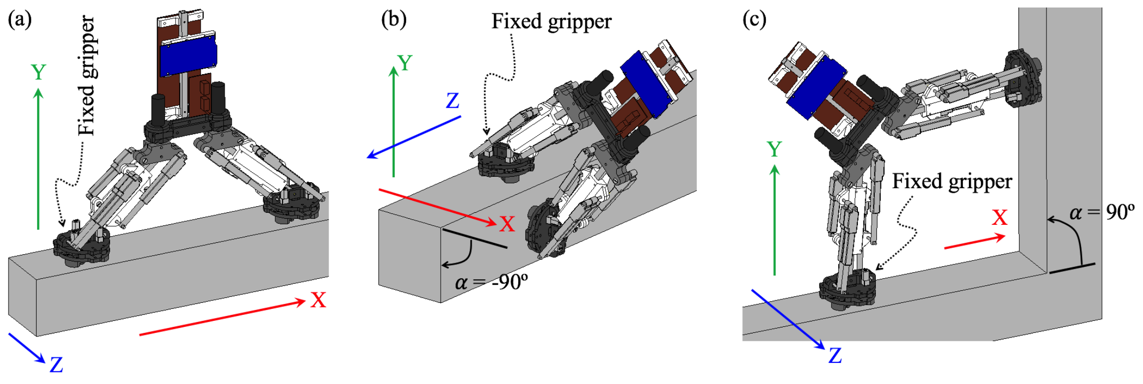


Figure 8.3: Basic movements to explore structures: longitudinal displacement along a beam (a), exterior transition between faces of a beam (b), and interior transition between beams (c).

- I: Interior (or concave) transition between two different adjacent beams.

As demonstrated in [132], by combining the three basic movements $\{L, E, I\}$ with the two rotations of the hip $\{\theta_A, \theta_B\}$, the HyReCRo robot can completely explore three-dimensional structures.

Figure 8.3 illustrates the HyReCRo robot performing these three basic movements. The longitudinal displacement along a beam can be accomplished by different gaits [67], although in this chapter we will prefer to use an inchworm-like gait, in which the robot alternatively extends or retracts its legs while adhering and releasing its grippers as required (see Figure 5.7). This gait is simpler to execute with the HyReCRo robot, and may be more energy-efficient than other gaits [14].

Figures 8.3b and 8.3c illustrate the HyReCRo robot performing exterior and interior transitions, respectively. For simplicity, in this chapter we will consider that exterior and interior transitions are orthogonal, i.e., we will consider transitions between planes that meet at $\pm 90°$. As demonstrated in earlier chapters, the motion capabilities of the HyReCRo robot are not limited to these two cases only, but it can attain a wider range of orientations between its grippers. However, in this chapter we will simplify our analysis to the three cases of Figure 8.3, in which the relative orientation α between the grippers is either $0°$ (Figure 8.3a), $-90°$ (Figure 8.3b), or $+90°$ (Figure 8.3c). In real structures, planes may meet at arbitrary angles α other than these three, but even in that case, α will lie between these three extreme cases or near them. Therefore, it is reasonable to assume that, if the grippers are designed to resist the basic extreme cases of Figure 8.3, then they will resist any other intermediate posture.

For the design of magnetic grippers in next sections, we will consider 18 scenarios or design cases, which are derived from the three basic postures of Figure 8.3: for each posture, we consider that gravity can act along the two possible directions (positive and negative directions) of each coordinate axis (coordinate axes are represented in Figure 8.3). We will identify each case by a string of three characters "UVW", where $U \in \{L, E, I\}$ denotes each posture of Figure 8.3, $V \in \{X, Y, Z\}$ denotes the axis along which gravity acts, and $W \in \{+,-\}$ denotes the sign of gravity along this axis. Using this notation, for example, "$LZ-$" identifies the scenario in which the robot is advancing longitudinally along a beam (L) with gravity acting along the negative $(-)$ direction of Z axis. In this way, the enumeration of all 18 cases is as follows: $LX+$, $LX-$, $LY+$, $LY-$, $LZ+$, $LZ-$, $EX+$, $EX-$, $EY+$, $EY-$, $EZ+$, $EZ-$, $IX+$, $IX-$, $IY+$, $IY-$, $IZ+$, $IZ-$.

Note that these 18 cases include the most typical situations found when climbing and exploring three-dimensional structures. For example:

- Cases "$LX\pm$" correspond to the situations in which the robot is climbing up or down a vertical beam, with gravity acting along the direction of the beam.
- Case "$LY+$" corresponds to the scenario in which the robot is hanging from the ceiling upside down, with gravity trying to directly detach the robot from the ceiling.
- Case "$IX-$" corresponds to the scenario in which the robot is performing a plane transition from wall to ceiling.

Similarly, other cases can be identified with other typical movements performed while climbing structures.

8.4 Conceptual Design of Magnetic Grippers

This section describes the basic structure and geometry of the magnetic grippers that will be designed in this chapter in order to firmly attach the HyReCRo robot to ferromagnetic structures in all design cases described in the previous section.

The first parameter to decide in the design of the grippers is the number of switchable magnets that each gripper should carry. If we equip each gripper with only one or two switchable magnets, both the net adhesion force of the gripper and the convex hull of the contact areas of these magnets will be small, which will compromise the stability of the adhesion (the robot will tip-over more easily). On the contrary, if we equip each gripper with many switchable magnets, then both the overall adhesion force and contact area will increase, which will improve the stability of the adhesion. However, a higher number of SM will also increase the weight of the gripper (especially due to the ferromagnetic housings), which will induce a higher detaching torque on the fixed gripper when the robot extends and the free gripper is far from the fixed gripper, as in the postures of Figures 8.3a,c. This increased detaching torque will require more and stronger switchable magnets, which in turn will increase again the weight of the gripper.

Thus, it is necessary to reach a balance between stability of the adhesion (given by the adhesion force and contact area) and the weight of the gripper. Therefore, and as Figure 8.4 depicts, the gripper to be designed will carry three switchable magnets

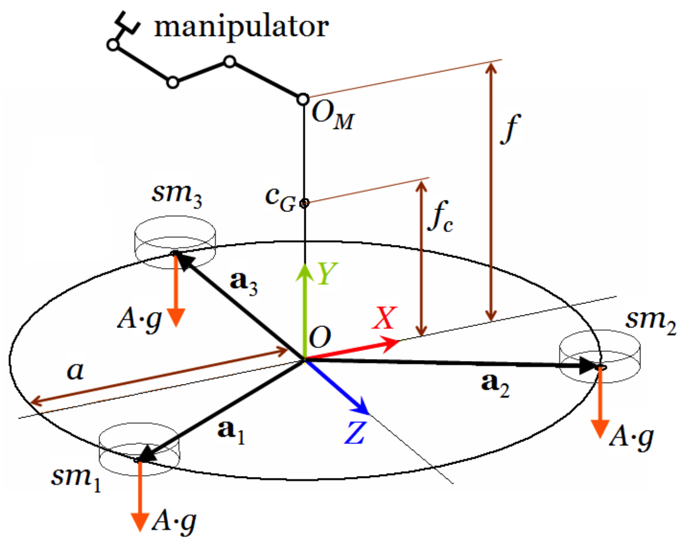


Figure 8.4: Geometry of the grippers to be designed. The XYZ axes shown in this figure are parallel to the same axes depicted in Figures 8.3a,b,c.

$\{sm_1, sm_2, sm_3\}$ evenly distributed on a circle of radius a, placed $120°$ apart, as in [160]. With three magnets distributed in this way, the gripper will be able to better resist three-dimensional external forces and torques like those occurring in the 18 design cases defined in the previous subsection.

For all calculations performed in this chapter, we will consider a reference frame $OXYZ$ centered at the base of the gripper, where plane OXZ is the plane of contact between the gripper and the ferromagnetic structure to which the robot must be attached (see Figure 8.4). The projections of the centers of the three switchable magnets on plane OXZ are defined by the following three vectors $\mathbf{a}_i$:

$$\mathbf{a}_1 = a \begin{bmatrix} \sin(-60°) \\ 0 \\ \cos(-60°) \end{bmatrix}, \; \mathbf{a}_2 = a \begin{bmatrix} \sin(60°) \\ 0 \\ \cos(60°) \end{bmatrix}, \; \mathbf{a}_3 = \begin{bmatrix} 0 \\ 0 \\ -a \end{bmatrix} \tag{8.1}$$

The adhesion force between each switchable magnet and the ferromagnetic substrate will be denoted by A, which will be measured in kg for later convenience (defining A in kg will allow us to drop the gravity acceleration later, to directly compare the adhesion force and the mass of the robot). Accordingly, due to magnetic adhesion, the gripper will be subject to three forces $A \cdot g$ [N] acting along the negative direction of Y axis, as depicted in Figure 8.4 ($g = 9.81m/s^2$ is the acceleration of gravity).

Other parameters of importance for the design of the grippers, besides the adhesion force A, are the following:

- m_G: mass of each gripper.
- f_c: distance between the base of the gripper and the center of mass c_G of the gripper.

- f: distance between the base of the gripper and the point O_M at which the gripper is connected to the manipulator.

The objective of next sections will be to determine the conditions that the design parameters $\{A, m_G, f_c, f\}$ must satisfy in order for the grippers to keep the robot firmly attached to the ferromagnetic structure in all 18 design cases identified in the previous subsection. In order to obtain these design conditions, we will analyze the statics of the complete robot (i.e., the manipulator equipped with both grippers) in all 18 design cases, considering the two possible failure modes affecting climbing and walking robots [208]:

- **Tip-over:** the gripper detaches when contact between some of the switchable magnets and the ferromagnetic structure is lost, and the robot falls.
- **Slippage:** contact between the gripper and the ferromagnetic structure is not lost, but the robot slips down due to insufficient friction at the contact.

The design of the grippers based on the statics rather than the dynamic analysis of the robot is justified for two reasons. Firstly, like most step-by-step climbing robots, the motion of the HyReCRo robot is slow, with low velocities and accelerations that allow us to consider quasi-static scenarios. Secondly, analyzing the statics instead of the dynamics will allow us to work with simpler symbolic design equations involving explicitly all unknown design parameters, which will also facilitate the identification of the worst-case design scenarios. If a dynamic analysis was performed instead of a static one, then trajectories of the robot should be simulated (instead of static postures), and this would require iterating numerical values of the design parameters until a feasible solution was found.

8.5 Preventing Tip-over

In this section, the concept of Zero Moment Point [190] will be used to obtain the design conditions that the magnetic gripper must satisfy in order to prevent the robot from tipping-over. According to this concept, the gripper will not detach (and the robot will not tip-over) if the position $\mathbf{p}$ of the Zero Moment Point, which can be computed from Equation (8.2), lies within the convex hull of the contact area between the gripper and the structure.

$$\mathbf{p} = \frac{\mathbf{n} \times \boldsymbol{\tau}}{\mathbf{n} \bullet \mathbf{f}} \tag{8.2}$$

where $\mathbf{n}$ is the normal unit vector to the contact plane ($\mathbf{n} = [0, 1, 0]^T$ according to Figure 8.4), $\mathbf{f}$ is the net external force acting on the robot, and $\boldsymbol{\tau}$ is the net external torque acting on the robot with respect to origin O indicated in Figure 8.4. Symbols "$\times$" and "$\bullet$" denote the cross and dot products, respectively. In our analysis, $\mathbf{f}$ and $\boldsymbol{\tau}$ are due to gravity and adhesion forces, excluding contact wrenches. The external force

$\mathbf{f}$ is the sum of the magnetic adhesion forces (indicated in Figure 8.4) and the weights of the manipulator and grippers:

$$\mathbf{f} = \begin{bmatrix} 0 \\ -3\,A\,g \\ 0 \end{bmatrix} + (m_M + 2\,m_G)\; g\,\mathbf{u}_g \tag{8.3}$$

where $m_M = 1.55$ kg is the mass of the manipulator (excluding grippers), m_G is the mass of each gripper (which is a design parameter), and $\mathbf{u}_g$ is the unit vector of gravity (which will vary for the 18 considered cases). Similarly, $\boldsymbol{\tau}$ can be derived from Figures 8.4 and 8.5 as follows:

$$\begin{aligned}\boldsymbol{\tau} = \begin{bmatrix} 0 \\ f_c \\ 0 \end{bmatrix} \times (m_G\,g\,\mathbf{u}_g) + \begin{bmatrix} x_M \\ y_M \\ 0 \end{bmatrix} \times (m_M\,g\,\mathbf{u}_g) + \\ + \begin{bmatrix} x_G \\ y_G \\ 0 \end{bmatrix} \times (m_G\,g\,\mathbf{u}_g) + \sum_{i=1}^{3} \left(\mathbf{a}_i \times \begin{bmatrix} 0 \\ -3\,A\,g \\ 0 \end{bmatrix} \right)\end{aligned} \tag{8.4}$$

where (x_M, y_M) are the coordinates of the center of mass c_M of the manipulator, whereas (x_G, y_G) are the coordinates of the center of mass c_G of the free gripper (see Figure 8.5). The positions of these centers of mass depend on the posture of the manipulator and on design parameters f and f_c, as observed in Figure 8.5. Note that the last sum in Equation (8.4) vanishes since the three switchable magnets are symmetrically distributed along a circle: $\mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3 = 0$. Thus, the adhesion forces exert no net moment with respect to O.

Substituting Equations (8.3) and (8.4) into (8.2) yields:

$$\mathbf{p} = \frac{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \times \left\{ \left(m_G \begin{bmatrix} 0 \\ f_c \\ 0 \end{bmatrix} + m_M \begin{bmatrix} x_M \\ y_M \\ 0 \end{bmatrix} + m_G \begin{bmatrix} x_G \\ y_G \\ 0 \end{bmatrix} \right) \times \mathbf{u}_g \right\}}{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \bullet \left\{ \begin{bmatrix} 0 \\ -3\,A \\ 0 \end{bmatrix} + (m_M + 2\,m_G)\;\mathbf{u}_g \right\}} \tag{8.5}$$

where the acceleration of gravity g has been canceled between numerator and denominator (this is possible because the adhesion force A was defined in [kg] in Section 8.4).

According to Equation (8.5), the position $\mathbf{p}$ of the ZMP will depend on the posture of the robot (which affects $\{x_M, y_M, x_G, y_G\}$) and on the direction $\mathbf{u}_g$ of gravity, and these parameters will be different for each of the 18 cases considered. In order for the gripper to remain attached to the ferromagnetic structure, $\mathbf{p}$ should belong to the convex hull of the contact area of the gripper. The contact area will depend on radius a (Figure 8.4) and the shape of the housings of the switchable magnets, whose footprints approximately are circles with two flat faces, as in Figure

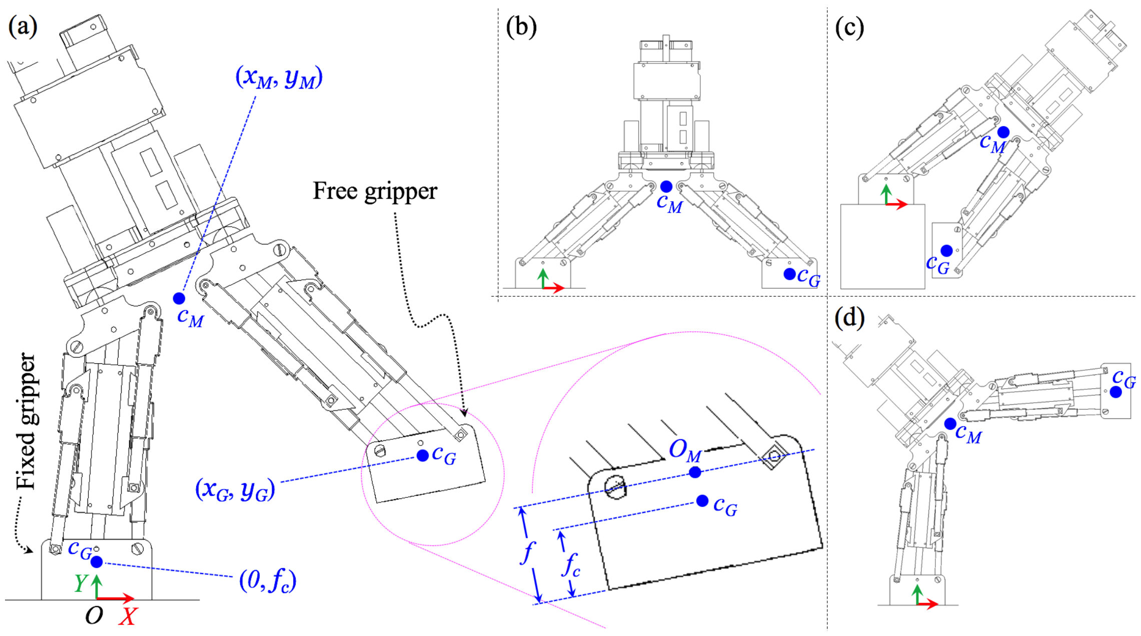


Figure 8.5: Positions of the centers of mass of the grippers (c_G) and manipulator (c_M) in general and for the three basic postures.

8.2b(left). Independently of the concrete shape of this convex hull, it is evident from Figure 8.4 that it will be an approximately triangular shape centered at O. Thus, it will be convenient that $|\mathbf{p}|$ is as small as possible to guarantee that the ZMP is within the convex hull. For this reason, next we will study which cases (out of the 18 design cases identified in Section 8.3) yield higher values of $|\mathbf{p}|$, since these worst cases will impose the most restrictive conditions on the design of the grippers.

8.5.1 Gravity Along Axes X or Z

First, consider the case when gravity is directed along axes X ($\mathbf{u}_g = [\pm 1, 0, 0]^T$) or Z ($\mathbf{u}_g = [0, 0, \pm 1]^T$). In that case: $\mathbf{n} \bullet \mathbf{u}_g = 0$, and Equation (8.5) becomes:

$$\mathbf{p} = \frac{m_G f_c + m_M y_M + m_G y_G}{3\,A}\mathbf{u}_g \tag{8.6}$$

Note that $|\mathbf{p}|$ will be larger when the y_M and y_G coordinates of the centers of mass of both the manipulator and free gripper are larger, which clearly occurs for the posture of Figure 8.5d. Thus, the cases when the robot is performing interior transitions between two beams with gravity directed along the X or Z axes will be among the worst scenarios. Notably, this includes one of the most typical climbing movements, which consists in performing plane transitions between a vertical wall and a ceiling/floor (cases $IX\pm$).

8.5.2 Gravity Along Axis Y

Consider now that gravity acts along axis Y, i.e.: $\mathbf{u}_g = \sigma[0,1,0]^T$, where $\sigma = \pm 1$. In that case, Equation (8.5) becomes:

$$\mathbf{p} = \sigma \frac{m_M x_M + m_G x_G}{\sigma(m_M + 2m_G) - 3\,A}[1,0,0]^T \tag{8.7}$$

According to Equation (8.7), $|\mathbf{p}|$ will be higher when $\sigma = 1$, since in that case the denominator will have a smaller absolute value. This means that the gripper will be more prone to detach while the robot is hanging (i.e., when gravity acts in the positive direction of axis Y) than while resting on a beam (i.e., when gravity acts along $Y-$), which is reasonable. Moreover, $|\mathbf{p}|$ will be larger when the robot is completely stretched while advancing longitudinally along a beam, as in Figure 8.5b, since in that case both x_M and x_G are larger. Therefore, the situation in which the robot is hanging and advancing longitudinally along the lower face of a beam will be among some of the worst-case scenarios for the design (case $LY+$).

Note that, in principle, it is not possible to predict which of the worst cases identified until now will be more demanding for the grippers (i.e., which case will yield a higher magnitude $|\mathbf{p}|$). This is because the worst cases of Equation (8.6) involve y-coordinates of centers of mass, whereas the worst case of Equation (8.7) involves their x-coordinates, which cannot be easily compared with y-coordinates in Figure 8.5 (furthermore, the denominators in these two equations are different). Thus, the worst scenario will be identified only after concrete numerical values are substituted for these x- and y-coordinates, which will be done in the next section.

8.6 Developed Magnetic Gripper

This section presents the development of magnetic grippers which guarantee that the ZMP remains in the convex hull of the contact area of the gripper for all 18 design cases, including the worst cases identified in the previous section. Each magnetic gripper carries three switchable magnets.

Figure 8.6 shows a detailed exploded view of the developed gripper, which is composed of two 3D-printed PLA plates (upper and lower plates) that are rigidly connected. The upper plate is directly connected to the manipulator through three revolute joints, and it carries three Pololu 12V DC motors (gear ratio 1000:1). Each of these motors switches the state of one switchable magnet. The lower plate is rigidly connected to the housings of the three switchable magnets. The weight of each gripper is $m_G = 0.32$ kg. Thus, the mass of the complete prototype (manipulator + two grippers) is 2.19 kg. The dimensions of the gripper along the XYZ axes are: $92 \times 60 \times 90$ mm.

As Figure 8.6 illustrates, the gap between both plates houses the mechanism that switches the state of the switchable magnets. Each SM is independently actuated by one of such mechanisms, which consists of a cam driven by each Pololu motor, and

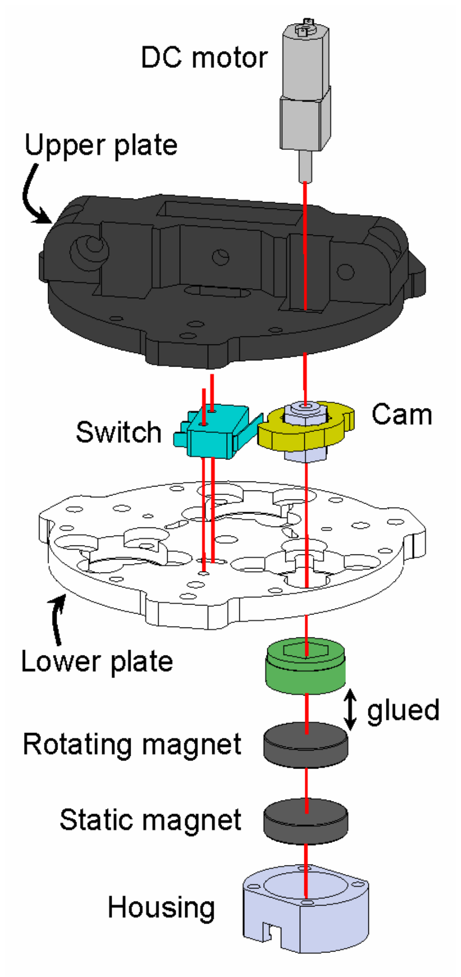


Figure 8.6: Exploded view of the developed gripper. The electric circuit is not represented. The figure only illustrates the switching mechanism for one of the switchable magnets; this mechanism is repeated for the other two switchable magnets.

a normally-closed switch. The objective of the cam is to detect the change of state of the switchable magnet, i.e., detect when the magnet rotates by $180°$.

This is achieved by means of the electric circuit shown in Figure 8.7: according to this circuit, when a short 12V pulse is received from the Arduino board (through a relay), all three motors simultaneously begin to rotate. After each motor starts to rotate, the corresponding switch remains closed because the cam no longer presses it, and the motor continues rotating since it is connected to a 12V source. Then, after completing half a turn, the cam presses again the switch, which opens the circuit of its corresponding motor and interrupts its rotation. In this way, although all motors simultaneously start to rotate when commanded to do so, the rest of the rotation occurs independently for each motor, until it completes exactly half a revolution.

The adhesion force of each *individual* switchable magnet is 16 kg at 3mm-thick steel plates (which is the thickness that will be assumed for the rest of the chapter), although this force decreases to 11 kg per switchable magnet when all three SM are

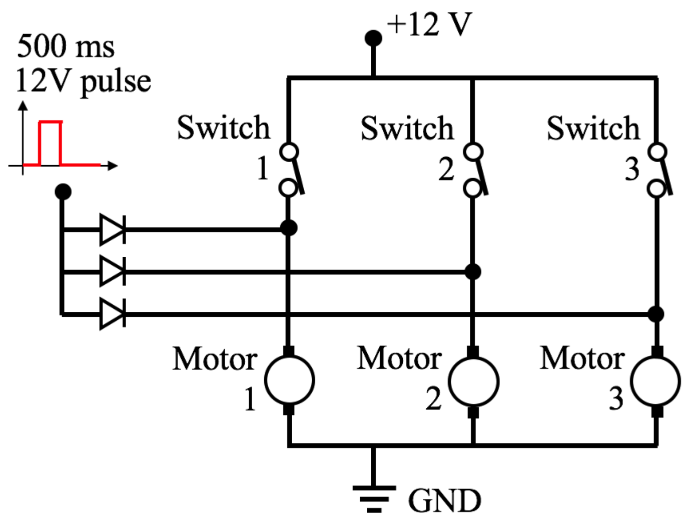


Figure 8.7: Electric circuit of each gripper.

Table 8.1: Specifications of the cylindrical permanent magnets used.

Material	NdFeB
Magnetization	Diametral
Diameter × height	20 mm × 5 mm
Material/grade	N48
Coating	Nickel
Max. operation temperature	80°
Flux density inside magnet	1.3799 Tesla
Holding force on a steel plate (unspecified thickness)	15.89 Newton
Weight which the magnet can lift:	1.62 kg
Dead weight:	11.72 g

mounted on the gripper and simultaneously turned ON (i.e., the individual force of each SM decreases due to the proximity and interaction with the magnetic fields generated by the other two SM). Therefore, the effective value of A to be used in all calculations in the remaining of this chapter is $A = 11$ kg.

The dimensions of the housings are detailed in Figure 8.8a, whereas the used permanent magnets are cylindrical with height 5 mm and diameter 20 mm. These are grade N48 NdFeB magnets from HKCM manufacturer [75] (see their specifications as provided by the manufacturer in Table 8.1), whereas housings are made of AISI 1018 steel. Instead of using glue, the lower magnet is firmly immobilized to the housing by means of a small headless screw (see Figure 8.8b), which facilitates the assembly and dis-assembly of switchable magnets (one may need to occasionally dis-assemble a switchable magnet in order to clean it and remove iron shavings that may hinder the rotation of the cylindrical permanent magnets).

As detailed in Figure 8.8a, the housings of the switchable magnets have 5×6 mm notches, whose objective is to reduce lateral magnetic shortcircuits and minimize

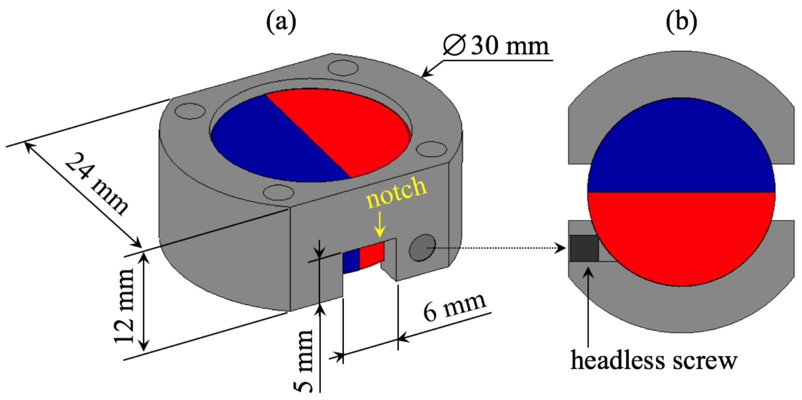


Figure 8.8: Dimensions and details of the housings.

the internal leakage of magnetic flux through the housing when the SM are ON (Figure 8.2b). When the SM are ON, it is desirable that most of the magnetic flux traverses the ferromagnetic substrate to which the SM should adhere, in order to increase the adhesion force.

These notches were made to the housings in order to increase the adhesion force and the stability of the designed magnetic grippers. This is because we developed first a preliminary functional design of the grippers which used housings without these notches (as in Figure 8.2a), but they were on the verge of detaching for some extreme postures of the HyReCRo robot (due to insufficient adhesion force). The adhesion force of each individual switchable magnet before realizing these notches was 10 kg (at 3mm-thick steel plates), whereas this individual force increases to 16 kg when realizing the notches (as reported above in this section). Thus, realizing the notches to reduce magnetic shortcircuits increases the adhesion force by a factor of 1.6.

However, this increase of adhesion force comes at the expense of a higher torque necessary for rotating the upper permanent magnet inside the housing, due to an increase of the magnetic repulsion when trying to align the poles of both magnets to turn ON the switchable magnet. To overcome this higher repulsion, the rotation of each switchable magnet must be done by an independent Pololu DC motor (as in Figure 8.6), since a single motor is unable to rotate all three switchable magnets simultaneously (as our preliminary design of the grippers did). One may think that the addition of more DC motors for rotating the switchable magnets may imply an undesirable increase of the overall mass of the grippers, but such an increase of mass is negligible since the used Pololu DC motors are very lightweight: each motor weighs only 10.5 g, which is small compared to the mass $m_G = 320$ g of each gripper (also, one should take into account the reduction of mass of the housings when removing some steel to realize the notches). Therefore, the overall effect of realizing the notches is positive, since the adhesion force increases by a factor of 1.6 without practically increasing the mass of the whole gripper.

Regarding the remaining design parameters f and f_c, their values for the developed grippers are: $f = 39$ mm and $f_c = 17.7$ mm. Inserting these values into Equation

Table 8.2: Positions of the centers of mass of the manipulator and free gripper, in mm.

Posture	x_M	y_M	x_G	y_G
Longitudinal	190.0	166.0	380.0	17.7
Exterior transition	179.4	107.4	88.7	-71.0
Interior transition	48.5	281.4	311.3	329.0

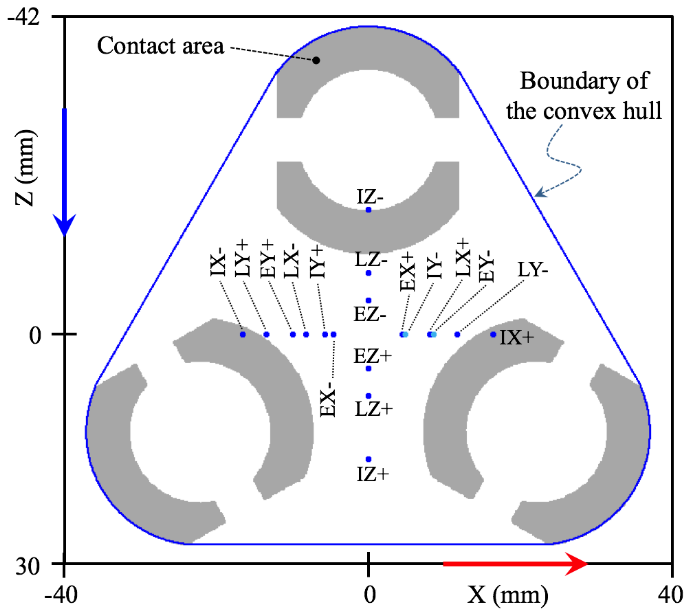


Figure 8.9: Positions of the ZMP for all design cases considered.

(8.5), together with $m_G = 0.32$ kg, $A = 11$ kg, and the positions of the centers of mass shown in Table 8.2, allows us to compute the positions of the ZMP for all 18 design cases. The positions of the ZMP are represented for all 18 design cases in Figure 8.9, which also represents the six U-shaped contact areas of the switchable magnets (in shaded) and the boundary of the convex hull of these contact areas (in continuous line). As this figure shows, the position of the ZMP is in the convex hull and quite far from its boundaries for all 18 design cases, which suggests a stable adhesion. Figure 8.9 also confirms that the worst design cases coincide with the candidates identified in Section 8.5: the distance $|\mathbf{p}|$ between the center of the gripper and the ZMP is maximal for cases $IX\pm$ and $IZ\pm$. The other candidate to worst case identified in Section 8.5 was $LY+$, and it presents the second higher distance to the center of the gripper, as it can be observed in Figure 8.9.

The grippers developed in this section were connected to both feet of the prototype of the HyReCRo robot and were tested for all 18 design cases on a real steel structure that will be described in Section 8.8. In these preliminary tests, it was found that the fixed gripper firmly adhered to the structure, so that strong forces were required in order to detach it in all cases. However, although the gripper never detached from the structure, the adhesion was not stable in all cases due to insufficient friction. In the cases when gravity acted along axis Z, gravity exerted too high torsional mo-

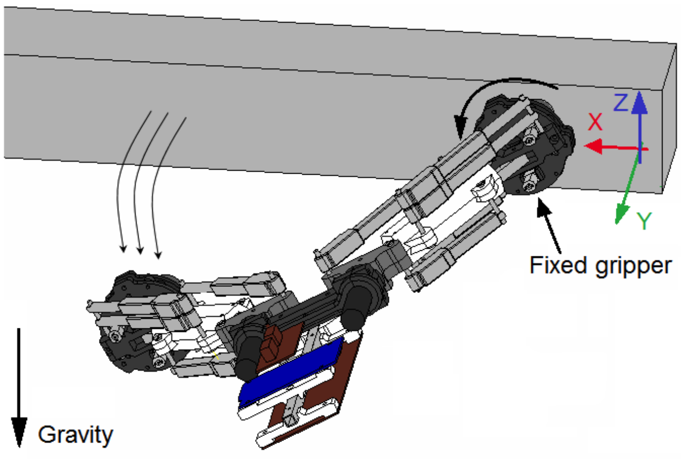


Figure 8.10: Slippage of the fixed gripper due to insufficient friction.

ments about the Y axis perpendicular to the contact surface, which resulted in the fixed gripper slipping down as illustrated in Figure 8.10. Therefore, it was necessary to slightly modify the developed grippers in order to increase friction and prevent slippage, as explained in the next section.

8.7 Preventing Slippage

In the previous section, we have presented the developed magnetic grippers, which prevent the robot from tipping-over, guaranteeing that the base of the gripper and the ferromagnetic structure are always in contact. However, after performing some preliminary experiments, it was found that the grippers slipped for some postures, due to insufficient friction. To avoid this, it is necessary to increase friction between the developed grippers and the structure to which the robot is attached.

In this section, we will modify the developed gripers by adding a removable friction accessory that fills the gaps between the switchable magnets, as illustrated in Figure 8.11a. As this figure shows, this friction accessory consists of a 3D-printed part shaped like a fidget spinner toy, with three friction pads uniformly distributed along a circle (i.e., placed 120° apart). These friction pads are made of rubber and provide a higher friction coefficient between the grippers and the ferromagnetic structure. Since the friction accessory will be very lightweight, we assume that its mass is negligible compared to the mass of the gripper, so that the calculations of previous sections are not affected by the incorporation of this accessory. The objective of the present section is to determine the necessary static friction coefficient μ of the friction pads so that the grippers will not slip as in Figure 8.10.

When adding the friction accessories to the grippers, it is important to be able to finely adjust the separation ϵ between the base of the lower plate of the gripper and the friction accessory. In order to finely adjust this separation, three screws were added to the grippers, as illustrated in Figure 8.11b (one screw for each friction pad). If ϵ is too small, the friction pads will not make contact with the structure (Figure 8.11b1)

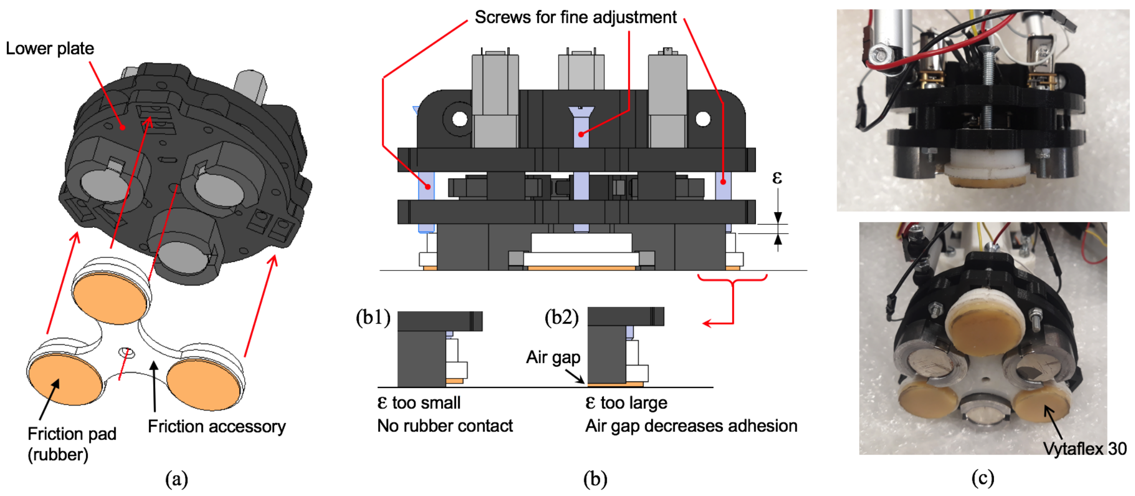


Figure 8.11: (a) Removable friction accessory to prevent slippage. (b) Adjusting the position of the friction accessory. (c) Photographies of the developed accessory.

and friction will still be too low, so the robot will continue slipping as in Figure 8.10. On the contrary, if ϵ is too large, then an air gap will appear between the structure and the switchable magnets (Figure 8.11b2), which will produce an undesirable decrease of the magnetic adhesion force [171] that may compromise the stability of the adhesion, leading to the fall of the robot. Using these screws, ϵ can be finely adjusted so that the air gap is practically zero (so that adhesion forces A do not decrease) at the same time that we guarantee that the grippers make contact with the ferromagnetic structure through the rubber pads, instead of through the switchable magnets (i.e., both undesirable situations of Figures 8.11b1 and b2 are avoided).

Through this fine adjustment, we can consider that the gripper makes contact with the ferromagnetic structure only through the three friction pads. The true area of contact will be unknown and will depend on the deformation of the rubber pads due to the pressing of the fine-adjustment screws. For simplicity, we will assume that the contacts between these rubber pads and the ferromagnetic structure are punctual, at the positions where the fine-adjustment screws press the friction pads against the structure. In this way, we can consider that now the contact between the gripper and the structure is through three punctual supports $\{fp_1, fp_2, fp_3\}$ symmetrically distributed along a circle of radius $b = 42.5$ mm, as illustrated in Figure 8.12 (b is the distance between the center of the gripper and the axes of the fine-adjustment screws). This approximation will simplify the following friction calculations, which would yield much more complex expressions if positive-dimensional contact areas were considered, instead of point contacts [41].

The objective of next subsections will be to determine the necessary static friction coefficient μ between the rubber friction pads and the ferromagnetic structure to prevent the slippage depicted in Figure 8.10. To that end, we will analyze all cases in which slippage may occur (of all 18 design cases considered in this chapter).

8.7.1 Purely Translational Slippage

According to Figure 8.5, when gravity acts along axis X, the whole robot will tend to slip along that axis as if it was a point mass. In that case, the gripper will not slip if μ satisfies:

$$m_M + 2m_G \leq \mu \cdot 3A \tag{8.8}$$

substituting $m_M = 1.55$ kg, $m_G = 0.32$ kg, and $A = 11$ kg, Equation (8.8) yields: $\mu \geq 0.066$. We measured the static friction coefficient between the steel structure and the steel housings of the switchable magnets and obtained $\mu = 0.15$, which is greater than the minimal required value (0.066). This explains why the grippers did not slip when gravity acted along axis X during the preliminary experiments described in the previous section, in which the friction accessory was not needed.

8.7.2 Mixed Roto-translational Slippage

When gravity acts along the Z axis, the whole robot tends to both translate along this axis and rotate about axis Y, due to the torsional torque exerted by gravity, which is parallel to Y. Thus, the robot cannot be treated as a point mass as in Equation (8.8), but it is necessary to analyze it as a rigid body tending to roto-translate in the XZ plane, which complicates the static friction analysis.

For the following static analysis, consider the free-body diagram of Figure 8.12, which represents schematically the fixed gripper adhered to the ferromagnetic structure. The fixed gripper is subject to the three adhesion forces $A{\cdot}g$ of the switchable magnets, as well as to the normal N_i and friction R_i reaction forces acting at the friction pads fp_i. Also, the fixed gripper is subject to its own weight ($m_G{\cdot}g$), the weight of the free gripper ($m_G{\cdot}g$), and the weight of the manipulator ($m_M{\cdot}g$). As indicated in Figure 8.12, the effects of these three weights can be reduced to a force F along axis $Z+$, a detaching torque D along axis $X+$, and a torsional torque T along axis $Y-$ (all three passing through origin O). The expressions of $\{F,\ D,\ T\}$ in terms of the weights of the manipulator and grippers can be easily derived from Figure 8.5 considering that gravity acts along the Z axis:

$$F = \sigma(m_M + 2m_G)g \tag{8.9}$$

$$D = \sigma(f_c m_G + y_M m_M + y_G m_G)g \tag{8.10}$$

$$T = \sigma(x_M m_M + x_G m_G)g \tag{8.11}$$

where $\sigma = 1$ if gravity acts along the positive direction of the Z axis, whereas $\sigma = -1$ if gravity acts along the negative direction.

Dahmen et al. [42] analyzed the static equilibrium of a very similar problem to the one illustrated in Figure 8.12 (but without adhesion forces), and determined the critical combinations of F and T for which the gripper starts to slip (assuming that $D = F{\cdot}h$, for a constant distance h). In our case, we will follow the approach of [42], but we will solve the opposite problem: for given $\{F,\ D,\ T\}$, we will determine the critical static friction coefficient μ for which the gripper starts to slip.

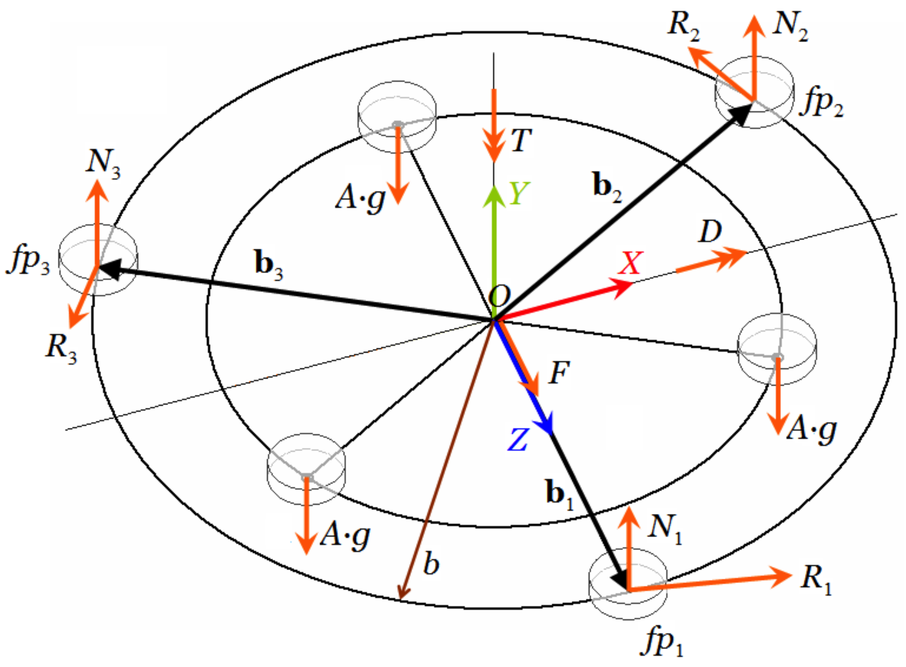


Figure 8.12: Free-body diagram of the fixed gripper when gravity acts along axis Z. Torques are represented by double-headed arrows.

To that end, let us assume that the friction forces at all three friction pads have reached their maximum value ($R_i = \mu{\cdot}N_i$ for $i = 1, 2, 3$) and the gripper is about to slip along plane XZ as a rigid body [42]. It is well known that planar rigid body motions can be instantaneously considered as pure rotations about a point known as Instantaneous Center of Rotation (ICR). Let us denote the coordinates of the ICR by $\mathbf{c} = [c_x, 0, c_z]^T$ (where c_x and c_z are unknowns to be determined). Then, as indicated in Figure 8.13, when the gripper starts to slip, it performs an infinitesimal rotation $\delta\omega$ about the ICR, which induces an infinitesimal displacement $\mathbf{d}_i$ in each friction pad fp_i [42]:

$$\mathbf{d}_i = \begin{bmatrix} 0 \\ \delta\omega \\ 0 \end{bmatrix} \times (\mathbf{b}_i - \mathbf{c}) \tag{8.12}$$

where $\mathbf{b}_i$ are the positions of the friction pads (see Figure 8.12):

$$\mathbf{b}_1 = \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix}, \; \mathbf{b}_2 = b \begin{bmatrix} \sin(120°) \\ 0 \\ \cos(120°) \end{bmatrix}, \; \mathbf{b}_3 = b \begin{bmatrix} \sin(-120°) \\ 0 \\ \cos(-120°) \end{bmatrix} \tag{8.13}$$

The vector $\mathbf{R}_i$ of the friction force at each friction pad will be [42]:

$$\mathbf{R}_i = -\mu\, N_i \frac{\mathbf{d}_i}{|\mathbf{d}_i|} \tag{8.14}$$

where the negative sign "$-$" is due to the fact that static friction forces oppose to the impending motion. Let us operate Equation (8.12) and rewrite $\mathbf{d}_i$ as follows:

$$\mathbf{d}_i = \delta\omega \cdot [b_{iz} - c_z,\, 0,\, c_x - b_{ix}]^T \tag{8.15}$$

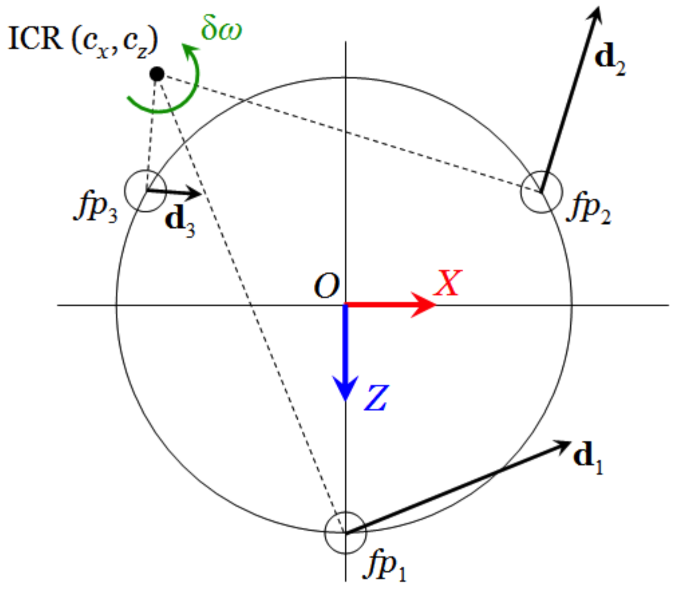


Figure 8.13: When the gripper starts to slip, it rotates about the ICR.

where b_{ix} and b_{iz} denote the x- and z-coordinates of $\mathbf{b}_i$, whose values are given in Equation (8.13). The magnitude of $\mathbf{d}_i$ can be written as $|\mathbf{d}_i| = |\delta\omega| \cdot d_i$, where d_i depends on the coordinates of the ICR, which are unknown:

$$d_i = d_i\,(c_x, c_z) = \sqrt{(b_{iz} - c_z)^2 + (c_x - b_{ix})^2} \tag{8.16}$$

Therefore, the normalization of $\mathbf{d}_i$ in Equation (8.14) yields:

$$\frac{\mathbf{d}_i}{|\mathbf{d}_i|} = \frac{\delta\omega \cdot [b_{iz} - c_z,\ 0,\ c_x - b_{ix}]^T}{|\delta\omega| \cdot d_i\,(c_x, c_z)} = \frac{\mathrm{sign}(\delta\omega)}{d_i\,(c_x, c_z)} \begin{bmatrix} b_{iz} - c_z \\ 0 \\ c_x - b_{ix} \end{bmatrix} \tag{8.17}$$

where $\mathrm{sign}(\delta\omega) = 1$ if $\delta\omega > 0$, and $\mathrm{sign}(\delta\omega) = -1$ if $\delta\omega < 0$. Thus, as one should expect, the value of the infinitesimal rotation $\delta\omega$ is irrelevant for the analysis, only its sign is important. The sign of $\delta\omega$ must be assumed, trying to guess the direction in which the gripper will rotate when starting to slip: this is equivalent to guessing the direction of the impending motion in undergraduate statics problems involving pulleys and blocks resting on ramps with friction. However, by comparing Figures 8.12 and 8.13, in our case it will be easy to pick the correct sign for $\delta\omega$ as follows: $\mathrm{sign}(\delta\omega) = -\mathrm{sign}(T) = -\sigma$. Anyway, if one failed to guess the correct sign of $\delta\omega$, the obtained friction coefficient μ would be negative, so one would simply invert the sign of μ. This is because, according to Equations (8.14) and (8.17), μ and $\mathrm{sign}(\delta\omega)$ will always appear as the product "$\mu \cdot \mathrm{sign}(\delta\omega)$".

After deriving the expression of friction forces, we must write the equilibrium equations of forces and moments, and solve μ (and other unknowns) from these equations. According to Figure 8.12, the equilibrium of forces translates into the following equation:

$$\sum_{i=1}^{3}\left(\mathbf{R}_i + \begin{bmatrix}0\\N_i\\0\end{bmatrix}\right) + \begin{bmatrix}0\\0\\F\end{bmatrix} + \begin{bmatrix}0\\-3{\cdot}A{\cdot}g\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix} \tag{8.18}$$

The equilibrium of moments about the origin O of Figure 8.12 yields:

$$\sum_{i=1}^{3}\left\{\mathbf{b}_i \times \left(\mathbf{R}_i + \begin{bmatrix}0\\N_i\\0\end{bmatrix}\right)\right\} + \begin{bmatrix}D\\-T\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix} \tag{8.19}$$

The moments of the adhesion forces do not appear in Equation (8.19) since their net moment with respect to O is zero due to symmetry, as in Equation (8.4).

Vector equations (8.18) and (8.19) constitute a system of six scalar equations in six unknowns: the three normal forces (N_1, N_2, N_3), the two coordinates (c_x, c_z) of the ICR in plane XZ, and the static friction coefficient μ. These equations can be solved by following the process described next, which is similar to the steps described in [42]. First, we pick the second scalar component of Equation (8.18), together with the first and third components of Equation (8.19). These are the equilibrium conditions of forces along axis Y, moments along axis X, and moments along axis Z, respectively, and they constitute a linear system which only involves the three normal forces $\{N_1$, N_2, $N_3\}$ as unknowns:

$$N_1 + N_2 + N_3 = 3{\cdot}A{\cdot}g \tag{8.20}$$

$$b_{1z}N_1 + b_{2z}N_2 + b_{3z}N_3 = D \tag{8.21}$$

$$b_{1x}N_1 + b_{2x}N_2 + b_{3x}N_3 = 0 \tag{8.22}$$

Therefore, the normal forces can be easily solved from this linear system. When solving it, one should obtain $N_i > 0$; otherwise, the gripper would detach from the structure since normal forces cannot be negative (the structure cannot pull from the gripper, only push). After solving the normal forces from these equations, the obtained values are substituted into the three remaining scalar components of Equations (8.18) and (8.19), which are the equilibrium conditions of forces along axis X, forces along Z, and moments along Y, respectively:

$$\left(\sum_{i=1}^{3}\frac{N_i\cdot(b_{iz}-c_z)}{d_i\left(c_x,c_z\right)}\right)\mu{\cdot}\mathsf{sign}(\delta\omega) = 0 \tag{8.23}$$

$$-\left(\sum_{i=1}^{3}\frac{N_i\cdot(b_{ix}-c_x)}{d_i\left(c_x,c_z\right)}\right)\mu{\cdot}\mathsf{sign}(\delta\omega) = F \tag{8.24}$$

$$-\left(\sum_{i=1}^{3}\frac{N_i\cdot(b_{ix}^2+b_{iz}^2-b_{ix}c_x-b_{iz}c_z)}{d_i\left(c_x,c_z\right)}\right)\mu{\cdot}\mathsf{sign}(\delta\omega) = T \tag{8.25}$$

These three equations, in which N_i are already known, only involve three unknowns: the friction coefficient μ and the coordinates (c_x, c_z) of the ICR. These three remaining unknowns can be solved as described next (this is where our resolution process mainly differs from the one described in [42], where $\{F, T\}$ were solved instead of $\{c_x, c_z\}$).

Firstly, note that, since $\mu \cdot \text{sign}(\delta\omega) \neq 0$, Equation (8.23) can only be satisfied if the first factor (which only involves c_x and c_z) is zero:

$$\sum_{i=1}^{3} \frac{N_i \cdot (b_{iz} - c_z)}{d_i\,(c_x, c_z)} = 0 \tag{8.26}$$

Moreover, the factor "$\mu \cdot \text{sign}(\delta\omega)$" can be easily eliminated between Equations (8.24) and (8.25), obtaining the following equation, which only involves c_x and c_z:

$$\left(\sum_{i=1}^{3} \frac{N_i \cdot (b_{ix}^2 + b_{iz}^2 - b_{ix}c_x - b_{iz}c_z)}{d_i\,(c_x, c_z)}\right) F = \left(\sum_{i=1}^{3} \frac{N_i \cdot (b_{ix} - c_x)}{d_i\,(c_x, c_z)}\right) T \tag{8.27}$$

The only unknowns in Equations (8.26) and (8.27) are c_x and c_z, and these two equations can be graphically interpreted as defining two curves in plane (c_x, c_z). Thus, by plotting these curves in plane (c_x, c_z) and finding their point of intersection, we can graphically solve Equations (8.26) and (8.27) and obtain the coordinates of the ICR.

After obtaining graphically the coordinates of the ICR, the static friction coefficient μ can be solved either from Equation (8.24) or (8.25). For example, using Equation (8.24) yields:

$$\mu = -\frac{F}{\text{sign}(\delta\omega)} \left(\sum_{i=1}^{3} \frac{N_i \cdot (b_{ix} - c_x)}{d_i\,(c_x, c_z)}\right)^{-1} \tag{8.28}$$

where the right-hand side is now completely known since c_x and c_z are already known. The value of μ computed from (8.28) will be the minimum required static friction coefficient to prevent slippage.

Until now, we have assumed that all three friction pads are about to slip, i.e., the ICR does not coincide with any of these friction pads. However, it may occur that, when graphically solving the coordinates of the ICR from Equations (8.26) and (8.27), the ICR coincided with, say, the k-th friction pad ($k \in \{1, 2, 3\}$), i.e.: $\mathbf{c} = \mathbf{b}_k$. In that case, the analysis presented in this section would not be valid since the infinitesimal displacement vector $\mathbf{d}_k$ of the friction pad coinciding with the ICR would be null. Therefore, the friction force $\mathbf{R}_k$ at the k-th pad cannot be computed as in Equation (8.14) since the normalization $\mathbf{d}_k / |\mathbf{d}_k|$ would result in the indetermination 0/0. This special case will be addressed next.

8.7.2.1 The ICR coincides with a friction pad

If the k-th friction pad fp_k ($k \in \{1, 2, 3\}$) coincides with the ICR, this means that fp_k is not about to slide, unlike the other two friction pads (which are on the verge of rotating about fp_k). If fp_k is not about to slide, then the friction force $\mathbf{R}_k$ at fp_k is still below its maximum value ($\mu{\cdot}N_k$) and should not be computed as in Equation (8.14). In that case, the friction force $\mathbf{R}_k$ is unknown both in magnitude and direction, and this force must be treated as an unknown $\mathbf{R}_k = [R_{kx},\, 0,\, R_{kz}]^T$ in Equations (8.18) and (8.19). Friction forces at the other two friction pads, which are about to slide, must still be computed using Equation (8.14). Note that, since the position of the ICR is known in this case (it coincides with the k-th friction pad), the directions $\mathbf{d}_i / |\mathbf{d}_i|$ of friction forces at the other two pads are known, which results in much simpler equations. Let us rewrite Equations (8.18) and (8.19) so as to clearly separate the friction force at the k-th friction pad from the friction forces at the other pads:

$$\begin{bmatrix} R_{kx} \\ N_k \\ R_{kz} \end{bmatrix} + \sum_{\substack{i\in\{1,2,3\} \\ i\neq k}} \left(\mathbf{R}_i + \begin{bmatrix} 0 \\ N_i \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 3{\cdot}A{\cdot}g \\ -F \end{bmatrix} \tag{8.29}$$

$$\mathbf{b}_k \times \begin{bmatrix} R_{kx} \\ N_k \\ R_{kz} \end{bmatrix} + \sum_{\substack{i\in\{1,2,3\} \\ i\neq k}} \left\{ \mathbf{b}_i \times \left(\mathbf{R}_i + \begin{bmatrix} 0 \\ N_i \\ 0 \end{bmatrix} \right) \right\} = \begin{bmatrix} -D \\ T \\ 0 \end{bmatrix} \tag{8.30}$$

where the $\mathbf{R}_i$ appearing inside the sums are the friction forces at the two pads which are about to slip and, therefore, must be computed using Equation (8.14). Equations (8.29) and (8.30) constitute again a system of six scalar equations in six unknowns, which are: μ, N_1, N_2, N_3, R_{kx}, and R_{kz}. However, these equations are much easier to solve now. First, one picks the second scalar component of (8.29), as well as the first and third components of (8.30). This results again in the linear system composed of Equations (8.20) to (8.22), from which the three normal forces are solved in a straightforward manner. Then, one substitutes these normal forces into the three remaining scalar components of (8.29) and (8.30), which make up another linear system in the remaining unknowns (μ, R_{kx}, R_{kz}):

$$R_{kx} - \left(\sum_{\substack{i\in\{1,2,3\} \\ i\neq k}} \frac{N_i \cdot (b_{iz} - c_z)}{d_i} \right) \mu{\cdot}\text{sign}(\delta\omega) = 0 \tag{8.31}$$

$$R_{kz} + \left(\sum_{\substack{i\in\{1,2,3\} \\ i\neq k}} \frac{N_i \cdot (b_{ix} - c_x)}{d_i} \right) \mu{\cdot}\text{sign}(\delta\omega) = -F \tag{8.32}$$

$$b_{kz}R_{kx} - b_{kx}R_{kz} - \left(\sum_{\substack{i\in\{1,2,3\} \\ i\neq k}} \frac{N_i \cdot (b_{ix}^2 + b_{iz}^2 - b_{ix}c_x - b_{iz}c_z)}{d_i} \right) \mu{\cdot}\text{sign}(\delta\omega) = T \tag{8.33}$$

Finally, μ, R_{kx}, and R_{kz} are solved from this linear system, which completes the calculation of the minimum friction coefficient necessary for avoiding slippage. After

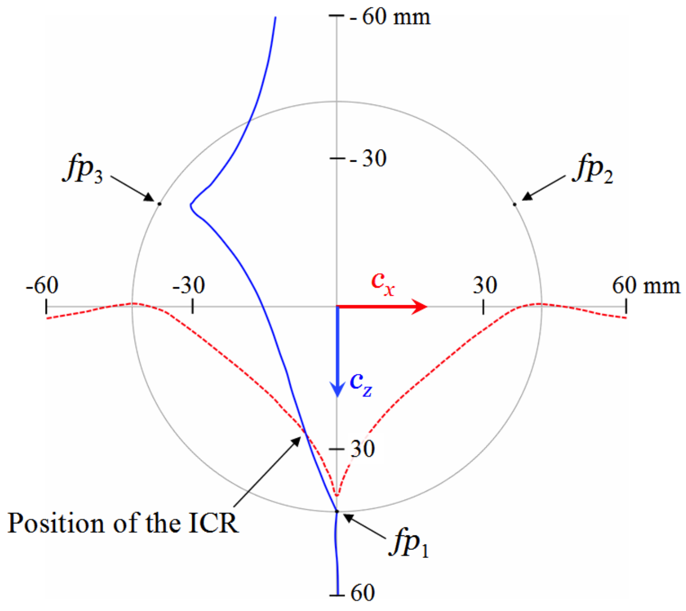


Figure 8.14: Graphical calculation of the ICR as the intersection of the curves defined by Equations (8.26) (red dashed curve) and (8.27) (blue continuous curve), for case LZ+ of Table 8.3.

this, one should check that the magnitude of $\mathbf{R}_k$ is smaller than the maximum available static friction force ($\sqrt{R_{kx}^2 + R_{kz}^2} < \mu{\cdot}N_k$), since the k-th friction pad is not about to slip.

8.7.3 Determining the Necessary Friction Coefficient

Next, the resolution process described in the previous subsection will be applied to the six design cases in which gravity acts along the Z axis, in order to determine the necessary friction coefficient to prevent slippage. The cases to be considered are: $LZ\pm$, $EZ\pm$, and $IZ\pm$.

The results of applying the previous procedure to these six cases are summarized in Table 8.3, which shows the normal forces resulting in each case (all of them are positive, as expected), together with the positions of the ICR and the resulting static friction coefficient μ. The position (c_x, c_z) of the ICR has been computed graphically, as previously explained. For example, Figure 8.14 illustrates the graphical computation of the ICR for case LZ+, in which the position of the ICR is computed as the intersection of the curves defined by Equations (8.26) and (8.27) in plane (c_x, c_z). According to Table 8.3, for cases LZ+ and EZ+, the ICR does not coincide with any of the friction pads. However, for the other four cases the ICR does coincide with one of the friction pads: for case IZ+ the ICR coincides with friction pad fp_1, whereas for cases {LZ-, EZ-, IZ-} the ICR coincides with friction pad fp_3. For these last four cases, it can be checked from Table 8.3 that the magnitude of the friction force at the ICR is smaller than $\mu{\cdot}N_k$, as one should expect since the friction pad coinciding with the ICR is not about to slip.

Table 8.3: Summary of the static friction analysis for all cases for which gravity acts along axis Z. In all cases: $N_2 = N_3$.

Case	LZ+	EZ+	IZ+
F (N)	21.5	21.5	21.5
D (N·m)	2.6352	1.4658	5.3672
T (N·m)	4.0819	3.0063	1.7147
sign($\delta\omega$)	-1	-1	-1
(N_1, $N_2 = N_3$) (N)	(149.25, 87.24)	(130.90, 96.41)	(192.10, 65.81)
ICR (c_x, c_z) (mm)	(-6.286, 26.783)	(-19.586, 3.193)	ICR $\equiv fp_1$
(R_{kx}, R_{kz}) (N)	—	—	(20.173, -21.5)
μ	0.3214	0.2354	0.1769
Case	LZ-	EZ-	IZ-
F (N)	-21.5	-21.5	-21.5
D (N·m)	-2.6352	-1.4658	-5.3672
T (N·m)	-4.0819	-3.0063	-1.7147
sign($\delta\omega$)	+1	+1	+1
(N_1, $N_2 = N_3$) (N)	(66.57, 128.58)	(84.92, 119.41)	(23.72, 150.01)
ICR (c_x, c_z) (mm)	ICR $\equiv fp_3$	ICR $\equiv fp_3$	ICR $\equiv fp_3$
(R_{kx}, R_{kz}) (N)	(19.558, -33.409)	(18.568, -19.369)	(4.025, -10.219)
μ	0.3392	0.2525	0.1959

Table 8.3 provides the minimum friction coefficient necessary to prevent slippage in each case. We observe that the highest friction coefficients are obtained for cases LZ±, which is reasonable since, for these cases, the robot is completely extended and this generates a higher torsional moment T that tries to rotate the robot. According to Table 8.3, the static friction coefficient must be higher than 0.3392 in order to prevent slippage in all cases. In order to provide sufficient friction, the friction pads were made of Vytaflex® 30 rubber, which provides a measured static friction coefficient of 0.438 in contact with steel. These Vytaflex pads were glued to the friction accessory, which is shown in the photography of Figure 8.11c. This friction accessory weighs 15 g, which is negligible compared to the weight of the gripper (320 g). Thus, adding the friction accessory does not invalidate the previous calculations, in which the weight of the friction accessory was omitted since it was unknown.

With this removable friction accessory, which can be easily added to the gripper or removed as required, the gripper remains static in all cases, including those in which gravity acts along axis Z. This will be demonstrated through some experiments in the next section.

8.8 Experiments

This section presents three experiments performed with the prototype of the HyReCRo robot carrying the developed grippers. These experiments demonstrate the stability of these grippers in different situations that include the worst scenarios identified in all

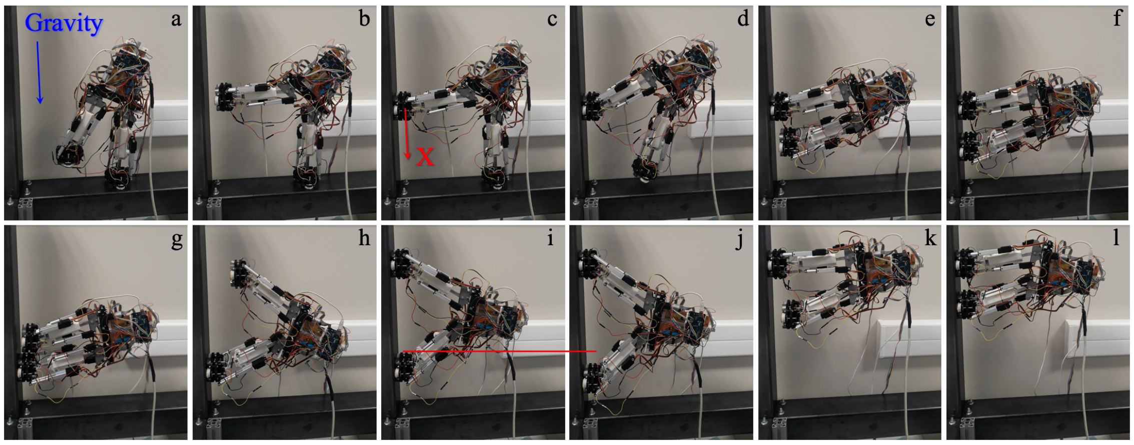


Figure 8.15: Video frames of the robot performing a concave plane transition and climbing up a beam.

previous sections, regarding both failure modes: tip-over/detaching and slippage. All experiments have been performed on a steel structure made of square tubular beams with 100×100 mm cross section and 3 mm-thick walls.

8.8.1 1st Experiment: Performing an Interior Transition and Climbing Up a Vertical Beam

In the first experiment, which is illustrated in Figure 8.15, the robot is initially resting on a horizontal beam, with its right gripper attached to it (Figure 8.15a). The objective is to perform an interior transition from the horizontal beam to a vertical one, and then climb up the vertical beam, overcoming gravity. Firstly, the robot is extended in order to place its left gripper on the vertical beam (Figure 8.15b), and then the left gripper is attached to this beam (Figure 8.15c). Following, the right gripper is detached from the horizontal beam (Figure 8.15d), and it is placed and adhered also to the vertical beam (Figures 8.15e and f). Note that this sequence of movements includes one of the worst design scenarios identified in Section 8.5, namely, case IX+: this case occurs between Figures 8.15c and d, when the left gripper is attached to the vertical beam and the robot performs an interior transition from the horizontal beam to the vertical one.

Once both grippers are attached to the vertical beam (Figure 8.15f), the robot proceeds to climb up this beam. To this end, the upper gripper must be released first in order to move it upward. When suddenly releasing the upper gripper, it falls and hits the lower gripper (compare Figures 8.15f and g) due to joint clearances present in the manipulator. However, despite this impact, the lower gripper remains firmly attached to the vertical beam. Next, the upper gripper is moved upward and is attached again to the vertical beam (Figures 8.15h and i). Following, it is necessary to release the lower gripper in order to lift it and continue climbing the beam. When the lower gripper is suddenly released, this gripper suddenly falls again due to clearances, and the robot is shaken as a consequence. However, the upper gripper remains firmly attached despite

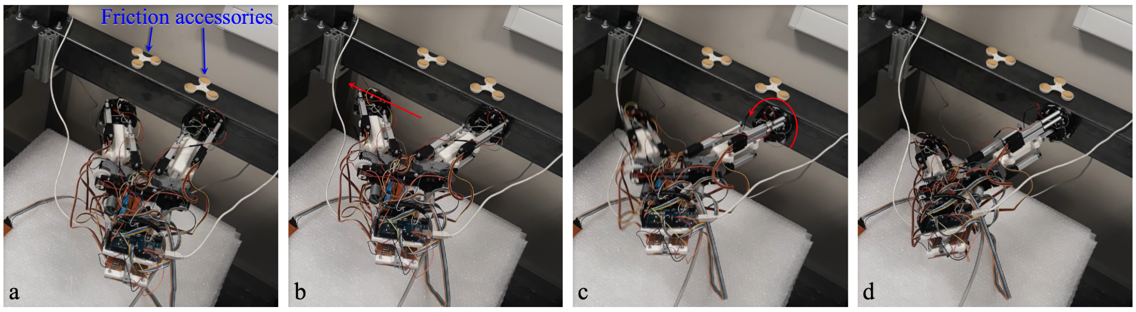


Figure 8.16: Video frames of the robot slipping due to insufficient friction when gravity acts along axis Z.

this shake. This falling of the released lower gripper due to clearances is highlighted by a horizontal red line spanning across Figures 8.15i and j.

Finally, after releasing the lower gripper, the robot lifts it and attaches it again to the vertical beam (Figures 8.15k and l). By repeating this inchworm-like gait, the robot would continue climbing up the vertical beam.

8.8.2 2nd Experiment: Slipping Due to Gravity Acting Along Z

In the second experiment, illustrated in Figure 8.16, the grippers do not carry the friction accessories. Thus, all friction is directly between the steel housings of the switchable magnets and the steel beam. Initially, the robot has one of its grippers attached to one of the vertical faces of a horizontal beam (Figure 8.16a), such that gravity acts along the negative direction of axis Z, perpendicular to the plane of motion of the robot. The objective is that the robot advances longitudinally along this horizontal beam, which requires extending the robot (Figure 8.16b). At the initial posture (Figure 8.16a), the net torsional moment T exerted by gravity is so small that the static friction between the housings and the beam is sufficient to prevent slippage. However, as the robot is extended (Figure 8.16b), T increases more and more until this steel-to-steel friction is insufficient and the fixed gripper rotates without detaching from the beam (Figure 8.16c), and the robot ends up at the oblique posture shown in Figure 8.16d, which prevents it from continuing to advance along the beam. Note that, according to Section 8.7, this is the worst scenario regarding the failure mode due to slippage. This can be avoided by incorporating the friction accessories to the grippers, as next experiment demonstrates.

8.8.3 3rd Experiment: Preventing Slippage with Friction Accessories

In the last experiment, illustrated in Figure 8.17, the robot must advance longitudinally along the vertical face of a horizontal beam and perform an interior transition between two orthogonal horizontal beams, with gravity being always perpendicular to the plane of motion of the robot. In this case, both grippers carry the friction accessories.

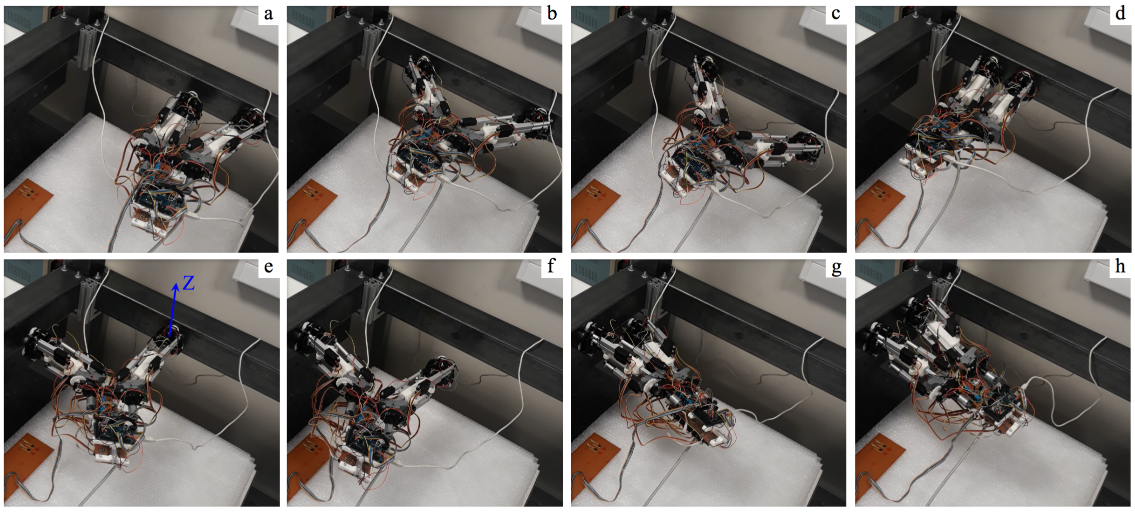


Figure 8.17: Video frames of the robot successfully performing a plane transition with gravity acting along axis Z. Slippage is avoided thanks to friction accessories.

The first part of the experiment (longitudinal advance along a horizontal beam) is the same as in the previous experiment. First, the robot extends completely and adheres its left gripper to the beam (Figures 8.17a and b). During this extension, and unlike in the previous experiment, the fixed gripper does not slip at all thanks to the friction accessories. Next, the right gripper is suddenly released, which shakes the robot as in the first experiment. In this case, due to this shake, the (left) fixed gripper slips slightly despite carrying the friction accessory. This slight slippage of the fixed gripper occurs between Figures 8.17b and c. Anyway, this slippage is small and can be corrected by detaching more smoothly the right gripper, instead of detaching it so abruptly (one of the advantages of switchable magnets is that their adhesion force can be varied smoothly by varying the relative angle between the two cylindrical permanent magnets [180]). After this, the robot retracts and the right gripper is attached again to the horizontal beam (Figure 8.17d).

Following, the robot must perform an interior transition between two horizontal beams, as illustrated in Figures 8.17e to h. This transition includes another of the worst scenarios identified in Section 8.5, which is the case when the robot performs an interior transition between two orthogonal beams with gravity acting along axis Z (Figure 8.17e). Also, this transition includes a couple of detachings of one of the grippers: one when performing the movement between Figures 8.17d and e, and another when performing the movement between Figures 8.17e and f. Both these detachings shake the robot, but none of them causes the fall of the robot or the slippage of the fixed gripper. Finally, the robot completes the interior transition, and both grippers remain attached to the new horizontal beam (Figure 8.17h).

8.9 Conclusions

In this chapter, we have presented a prototype of the HyReCRo robot, which is completely functional as it can climb real steel structures. The design of this prototype is based on the kinematic analyses and simulation tools presented in all previous chapters of this thesis: these analyses and tools have been used for determining appropriate values of the geometric design parameters of this robot in order to allow it to perform convex and concave plane transitions. This prototype weighs 2.19 kg (including its magnetic grippers), it is driven by DC motors and linear actuators, is controlled by an Arduino and a custom-made power board, and is teleoperated by means of a gamepad.

The design of the magnetic grippers of this prototype has been the main focus of the present chapter. The developed grippers are based on the technology of switchable magnets, which is safer and more energy-efficient than traditional electromagnets, since they only require power in order to switch their adhesion state (Section 8.2).

Based on the notion of the Zero Moment Point, we have derived the design conditions for preventing the detachment of the grippers and the fall of the robot while climbing structures (Section 8.5). These conditions have also revealed the worst scenarios occurring when climbing structures, which are those situations in which the grippers are more prone to detach. These worst scenarios occur when the robot is performing a concave plane transition between a vertical wall and a ceiling or a floor (cases $IX\pm$), and also when performing concave transitions between two vertical walls, with gravity acting along the direction perpendicular to the plane of motion of the robot (cases $IZ\pm$).

In order to remain attached to steel structures in these worst scenarios, two identical magnetic grippers, with three switchable magnets per gripper, have been developed (Section 8.6). Each developed magnetic gripper offers an adhesion force of 33 kg on 3mm-thick steel plates. The housings of the switchable magnets have notches, which reduce magnetic flux leakages and increase the adhesion force by a factor of 1.6.

Then, after guaranteeing that the developed grippers will not detach, a static friction analysis has been performed in order to determine the minimum static friction coefficient necessary to prevent slippage of the grippers, which is likely to occur when gravity acts along the direction perpendicular to the plane of movement of the robot. Departing from the resolution steps described by Dahmen et al. [42], a procedure has been proposed in Section 8.7 for graphically obtaining the coordinates of the Instantaneous Center of Rotation, about which the gripper will rotate when starting to slip. This procedure also yields the minimum static friction coefficient necessary for preventing slippage. Based on this static friction analysis, a friction accessory for the developed grippers has been designed, which consists of three friction pads made of rubber. This friction accessory increases the friction coefficient between the grippers and the steel structure, preventing slippage.

Finally, the experiments described in Section 8.8 have validated the developed grippers, which allow the robot to firmly adhere to real steel structures and climb them, even in the worst design scenarios identified above. The developed grippers have compact dimensions, which makes them especially suitable for climbing steel structures, since the beams of these structures usually are very narrow and impede using bulky magnetic grippers.

9 Conclusions and Future Work

After detailing in the previous chapters all the research work conducted under the framework of the present thesis, this final chapter summarizes the main contributions of this work. Also, Section 9.2 hints possible extensions and future work that can be derived from the different research lines presented in this document.

9.1 Contributions

This thesis has presented a comprehensive kinematic analysis of the HyReCRo robot, a serial-parallel and redundant robot for climbing and exploring three-dimensional metallic structures. This analysis has consisted in the forward and singularity analyses of the parallel modules that compose the legs of this robot, as well as the forward and inverse kinematic analyses of the complete HyReCRo robot. The presented analysis has been supported by specifically developed graphical tools for simulating parallel and other robots. Also, the workspace of the HyReCRo robot has been analyzed, both at external and internal levels. Finally, a prototype of the robot with magnetic grippers has been developed, for adhering to real steel structures and climb them. Next, all achievements and contributions of this thesis are summarized.

Chapter 3

- It has been demonstrated for the first time that 2R$\underline{P}$R-PR analytic parallel mechanisms with flat mobile platform and perpendicular passive slider always have special singularities which are fourfold solutions of the forward kinematic problem of these mechanisms. Also, it has been shown that these special singularities can be encircled in the actuated joint space in order to produce nonsingular transitions between different solutions of the forward kinematic problem of these analytic mechanisms.

- This phenomenon (nonsingular transitions when encircling special singularities which are fourfold solutions of the forward kinematics) has also been demonstrated in analytic 3R$\underline{\text{P}}$R planar parallel robots with non-similar and flat platforms.

- It has been shown that the behavior of the robot when encircling these fourfold singularities is conserved when slightly perturbing the geometry of the robot, in which case the special fourfold singularities transform into small deltoids.

- A method for predicting how isolated singularities (which include some of the fourfold singularities mentioned above) transform under perturbations of the geometric design of the robot has been proposed. This method is based on quadratic Taylor expansions.

Chapter 4

- A web-based virtual laboratory of parallel robots has been developed, with the purpose of facilitating the kinematic analysis of parallel and other robots, like the HyReCRo robot. This virtual laboratory consists of several Java applets that allow the user to simulate the forward and inverse kinematic problems of several parallel robots, as well as to visualize their workspace and singularities and analyze how these deform under changes in the design of these robots.

Chapter 5

- The forward kinematic problem of the complete HyReCRo robot has been solved, identifying the only feasible solution to this problem (all other 255 solutions are unfeasible due to collision constraints).

- A special and simplified case of the inverse kinematic problem which considers planar and symmetric postures has been solved. The workspace reachable using only these planar and symmetric postures has been investigated, in order to determine the relationship between the design parameters of the HyReCRo robot and the shape of this workspace.

- The general inverse kinematic problem of the complete HyReCRo robot has been solved, identifying the self-motion manifolds of this redundant robot. It has been found that these self-motion manifolds generically are four-dimensional, but they become five-dimensional in a special singular (and quite frequent) case in which the feet of the robot remain parallel. Simple parameterizations of these self-motion manifolds have been built. A method for obtaining two- and three-dimensional projections of these self-motion manifolds has been proposed. These projections, which are called "feasible regions" (denoted by R_f) in this thesis, provide compact and intuitive graphical representations of the solutions of the inverse kinematic problem of the HyReCRo robot. Also, these projections do not miss relevant information regarding the overall posture of the robot, which makes them useful for identifying postures that are collision-free.

Chapter 6

- Classical Monte Carlo methods have been used for performing a preliminary workspace analysis of the HyReCRo robot, identifying the influence of the design parameters of this robot on the shape and size of its workspace. It is found that the workspace is most sensitive to the stroke of the linear actuators and the width of the feet of this robot.
- A new improved Monte Carlo method for obtaining the boundaries of the workspace of complex robot manipulators has been proposed. This method is more efficient than previously existing Monte Carlo methods, since it can attain much higher accuracy than previous methods requiring the same or less computation time than them. The proposed method consists in uniformly "growing" an initial imprecise workspace by means of normal distributions, until the boundaries of the workspace are reached.

Chapter 7

- A method for effectively obtaining the boundaries and interior barriers of the workspace of redundant robots under collision constraints has been proposed. Previously existing methods find difficulties to compute interior barriers induced by arbitrarily complex collision constraints, since it is very difficult to model these constraints as equalities. The proposed method identifies the occurrence of interior barriers with the vanishing of disjoint components of the self-motion manifolds of the robot. To that end, self-motion manifolds are first densely sampled, discarding samples that do not satisfy collision constraints. Then, the samples are clustered using kd-trees in order to identify disjoint components of the self-motion manifolds. Finally, the self-motion manifolds at neighboring points of the workspace are compared and matched in order to determine if any disjoint self-motion manifold vanishes when moving between these neighboring workspace points, in which case an interior barrier is identified between these two points. Experiments with the proposed method demonstrate that this method is feasible for redundant robots whose self-motion manifolds are one- and two-dimensional; for higher dimensions, the method is too computer-intensive. These experiments also demonstrate that collision constraints can drastically alter the distribution of interior barriers inside the workspace.

- A variant of the previous method, which analyzes lower-dimensional projections of such self-motion manifolds, has also been sketched for dealing with robots with higher-dimensional manifolds, like the HyReCRo robot (whose manifolds are four- and five-dimensional). This variant still needs to be improved in order to avoid losing relevant information when projecting the manifolds on lower-dimensional subspaces, since some barriers may be missed due to this loss of information.

Chapter 8

- A completely functional prototype of the HyReCRo robot has been developed. This prototype weighs 2.19 kg (including two magnetic grippers) and is manually teleoperated.
- Novel magnetic grippers for this prototype have been developed. These grippers offer an adhesion force of 33 kg on 3mm-thick steel plates, providing a stable adhesion on real steel structures. They are based on the technology of switchable magnets, and include rubber pads for increasing friction and preventing slippage. The housings of these switchable magnets include notches that reduce magnetic short-circuits and increase adhesion force by a factor of 1.6.
- A study of the worst design scenarios for preventing the detaching of the magnetic grippers while climbing has been presented, based on the notion of Zero Moment Point. The worst situation turns out to be when the robot is performing a concave (or interior) transition between two orthogonal planes, with gravity acting along the direction perpendicular to the new plane to which the free gripper must be attached, e.g., when performing transitions between floor and wall, or between wall and ceiling.
- An algebraic-graphical procedure for determining the coordinates of the Instantaneous Center of Rotation about which the gripper will rotate when starting to slip due to insufficient friction has been presented. This procedure also yields the minimum static friction coefficient necessary for preventing slippage.

9.2 Future Work

The next list suggests some future research works that may be derived from some of the research lines and results presented in the previous chapters of this thesis.

- **Mixed binary-continuous actuation of the HyReCRo robot.** The original purpose of the HyReCRo robot was to be a purely binary robot, in order to greatly simplify the control and motion planning of this robot when climbing and exploring structures. However, as argued at the beginning of this thesis, a purely binary climbing robot would be unable to completely explore a three dimensional structure, since it can only attain a finite set of discrete poses, none of which may finely place the grippers of the robot at the surface to be climbed.

After solving the kinematic problems and workspace of this robot considering continuous actuation in this thesis, a mixed binary-continuous actuation strategy may be adopted, which may be a way of returning to the original proposal of the HyReCRo robot. According to this strategy, the motion of the robot along the structure may follow steps similar to those sketched next:

– **First step: coarse binary motion.** Firstly, the robot would try to reach the desired position using purely binary actuation. Since the robot has so many degrees of freedom, it can attain a large number of discrete postures even if all actuators are binary, so one of these discrete postures may be close enough to the required pose for performing a plane transition in the structure. In that case, the motion planning of the robot would not be necessary, since a binary posture would suffice.

– **Second step: fine continuous motion.** Secondly, in case a purely binary posture does not place the gripper of the robot at the required pose for performing a plane transition, then some actuators of the robot would be continuously actuated progressively, until the desired pose was attainable. For example: imagine that a binary posture of the robot places its free gripper close to the desired pose, but not close enough to perform a plane transition. Then, one would "relax" the binary actuation condition of only one of the two linear actuators directly connected to the free gripper, actuating it continuously, in order to try to attain the desired pose in this way. If relaxing only one actuator was not enough, then the adjacent linear actuator would also be relaxed, in order to try to finely place the free gripper at the desired pose. If relaxing two linear actuators was still not enough, then one would continue this procedure, releasing linear actuators one at a time, from the free gripper to the fixed gripper, until the desired pose could be reached. Note that it would be reasonable to relax and actuate continuously first the linear actuators that are closest to the free gripper, since they are easier to control because they support smaller payloads than those near the fixed gripper of the robot. However, one should explore all possibilities, since it may occur that, in order to finely reach a desired pose, one only needed to relax the binary actuation of one of the linear actuators placed near the fixed gripper instead of many of the actuators near the free gripper (usually, in serial robots the actuators near the fixed base of the robot have a greater influence on the motion of the end-effector due to the higher distance between the rotation axis of the actuator and the end-effector).

In order to perform fine continuous motions during the second step of the previous strategy, the kinematic analyses performed in the present thesis can be used at all required levels. For example, if one only needs to continuously actuate one of the linear actuators or parallel modules of the robot, then one would follow the kinematic analysis presented in Chapter 3. If a whole leg of the robot should be continuously actuated, then one would follow the analysis presented in Section

5.2.2. Finally, if one needs to continuously actuate the whole robot, then the analyses presented in all previous chapters can be used.

- **Trajectory planning.** After the kinematics of the HyReCRo robot has been thoroughly analyzed and a functional prototype of this robot has been built and teleoperated on a real steel structure, one of the next steps will be to solve its trajectory planning problem. This problem will consist in determining the sequence of movements to be performed to move the robot from one starting point of the structure to another goal point. This problem encompasses two sub-problems: trajectory planning of the manipulator as a conventional manipulator (i.e., when one of the grippers is fixed to the structure and the free gripper must be moved to the next attachment point of the structure) and trajectory planning of the whole climbing robot as a mobile robot moving along the structure, in order to visit all the parts that need to be inspected. The first sub-problem can be certainly simplified if the mixed binary-continuous strategy sketched above is used, since in that case one would only need to plan trajectories in the continuous subspace of the actuators whose binary actuation has been relaxed, which will typically be a lower-dimensional subspace. In that case, instead of searching a ten-dimensional space of ten continuously-actuated joint coordinates, one would only need to search the continuous space of the two or three actuated joint coordinates that are continuously actuated; the remaining coordinates would still be binarily actuated.

- **Interior barriers of the workspace of highly-redundant robots.** The method presented in Section 7.2 for obtaining the interior barriers of redundant manipulators under collision constraints is very computer-intensive, and is therefore feasible only for robots whose self-motion manifolds are one- or two-dimensional, as the experiments in Section 7.3 have demonstrated. Although an attempt was made in Section 7.4 to extend this method to the HyReCRo robot, whose self-motion manifolds are four- and five-dimensional, the approximate method presented in Section 7.4 omitted information since it obtained barriers by analyzing only lower-dimensional projections of such self-motion manifolds. As a result of this loss of information during the projection, the approximate method presented in Section 7.4 will certainly miss some barriers. In order to effectively extend the method presented in Section 7.2 to robots with higher degree of redundancy, it will be necessary to find some lower-dimensional projection of self-motion manifolds that does not lose information relevant to the computation of interior barriers. For example, for robots with a single working mode in the non-redundant case, like the Stewart platform, the complete configuration of the robot is uniquely defined by the pose of its mobile platform. This means that, for this robot, it is not necessary to analyze the self-motion manifolds in its six-dimensional actuated joint space. Instead, it is sufficient to analyze the projections of these manifolds on the lower-dimensional subspace of redundant pose variables [space of vector ψ defined in the paragraph above Equation (7.2)], since any configuration in this subspace results in a unique configuration in its actuated joint space [space of vector $\boldsymbol{\theta}$ defined in the paragraph above Equation (7.2)]. However, this is not true for robots with several working modes (like the

3$\underline{\text{R}}$RR robot studied in Section 7.3.3), since in that case any configuration in the subspace of redundant pose variables yields several possible configurations in the actuated joint space.

- **Autonomous inspection of structures**. Currently, the HyReCRo robot is teleoperated. However, in the future, the HyReCRo robot will be equipped with an omnidirectional camera in order to autonomously explore and inspect three-dimensional structures. By means of this camera, the robot will perform Simultaneous Localization and Mapping (SLAM) on the structure.

A Appendix: Set of Publications

The major contributions made in the present thesis are supported by two papers published in journals ranked in JCR (Science Edition). The metadata of these journal papers are presented next:

Journal Paper 1

An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots. [145]
A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
Eng. Appl. of Artificial Intelligence. Vol 64, pp. 197-207 (2017)
ISSN: 0952-1976. Ed. Elsevier.
JCR-SCI Impact Factor: 2.819, Quartile **Q1**
Web: https://doi.org/10.1016/j.engappai.2017.06.009
DOI: 10.1016/j.engappai.2017.06.009

Journal Paper 2

A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. [146]
A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá
Mechanism and Machine Theory. Vol 128, pp. 84-109 (2018)
ISSN: 0094-114X. Ed. Elsevier.
JCR-SCI Impact Factor: 2.796, Quartile **Q1**
Web: https://doi.org/10.1016/j.mechmachtheory.2018.05.013
DOI: 10.1016/j.mechmachtheory.2018.05.013

Preprints of these publications are appended next.

Bibliography

[1] K. Abdel-Malek and J. Yang. Workspace boundaries of serial manipulators using manifold stratification. *The International Journal of Advanced Manufacturing Technology*, 28(11):1211–1229, 2006. 160, 192

[2] K. Abdel-Malek, H.-J. Yeh, and S. Othman. Interior and exterior boundaries to the workspace of mechanical manipulators. *Robotics and Computer-Integrated Manufacturing*, 16(5):365 – 376, 2000. 160

[3] Actuonix. Actuonix inc. https://www.actuonix.com/. Accessed: 2018-07-24. 242

[4] M. Agheli and S.S. Nestinger. Comprehensive closed-form solution for the reachable workspace of 2-RPR planar parallel mechanisms. *Mechanism and Machine Theory*, 74:102 – 116, 2014. 160

[5] T. Akinfiev, M. Armada, M. Prieto, and M. Uquillas. Concerning a technique for increasing stability of climbing robots. *Journal of Intelligent and Robotic Systems*, 27(1):195–209, 2000. 244

[6] D. Alciatore and C. Ng. Determining manipulator workspace boundaries using the Monte Carlo method and least squares segmentation. In *ASME Robotics: Kinematics, Dynamics and Controls*, volume 72, pages 141–146, 1994. 132, 163, 164

[7] R. Aracil, R. Saltarén, and O. Reinoso. Parallel robots for autonomous climbing along tubular structures. *Robotics and Autonomous Systems*, 42(2):125 – 134, 2003. 28

[8] M. Armada, M. Prieto, T. Akinfiev, R. Fernández, P. González, E. García, H. Montes, S. Nabulsi, R. Ponticelli, J. Sarria, J. Estremera, S. Ros, J. Grieco, and G. Fernandez. On the design and development of climbing and walking robots for the maritime industries. *Journal of Maritime Research*, 2(1):9–32, 2005. 24, 244

[9] K.A. Arrouk, B.C. Bouzgarrou, and G. Gogu. CAD based techniques for workspace analysis and representation of the 3$\underline{\text{C}}$RS parallel manipulator. In *19th International Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2010)*, pages 155–160, 2010. 160

[10] M. Badescu and C. Mavroidis. New performance indices and workspace analysis of reconfigurable hyper-redundant robotic arms. *The International Journal of Robotics Research*, 23(6):643–659, 2004. 160, 163, 204

[11] A. Baghani, M.N. Ahmadabadi, and A. Harati. Kinematics Modeling of a Wheel-Based Pole Climbing Robot (UT-PCR). In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2099–2104, 2005. 26

[12] T. Bajd, M. Mihelj, and M. Munih. *Introduction to Robotics*. Springer Netherlands, 2013. 125

[13] C. Balaguer, A. Gimenez, and A. Jardon. Climbing robots' mobility for inspection and maintenance of 3D complex environments. *Autonomous Robots*, 18(2):157–169, 2005. 136

[14] C. Balaguer, A. Giménez, J.M. Pastor, V.M. Padrón, and M. Abderrahim. A climbing autonomous robot for inspection applications in 3D complex environments. *Robotica*, 18(3):287–297, 2000. 1, 25, 27, 28, 104, 123, 247

[15] H. Bamberger, A. Wolf, and M. Shoham. Assembly Mode Changing in Parallel Mechanisms. *IEEE Transactions on Robotics*, 24(4):765–772, 2008. 48

[16] S. Bartsch, T. Birnschein, M. Römmermann, J. Hilljegerdes, D. Kühn, and F. Kirchner. Development of the six-legged walking and climbing robot Space-Climber. *Journal of Field Robotics*, 29(3):506–532. 34

[17] D.J. Bates, J.D. Hauenstein, A.J. Sommese, and C.W. Wampler. *Numerically Solving Polynomial Systems with Bertini*. SIAM, 2013. 204

[18] B. Bihari, D. Kumar, C. Jha, V.S. Rathore, and A.K. Dash. A geometric approach for the workspace analysis of two symmetric planar parallel manipulators. *Robotica*, 34(4):738–763, 2016. 160

[19] O. Bohigas, M. Manubens, and L. Ros. A complete method for workspace boundary determination on general structure manipulators. *IEEE Transactions on Robotics*, 28(5):993–1006, 2012. j, 160, 192, 196, 217, 221

[20] M. Bonani, S. Magnenat, P. Rétornaz, and F. Mondada. The Hand-Bot, a robot design for simultaneous climbing and manipulation. In M. Xie, Y. Xiong, C. Xiong, H. Liu, and Z. Hu, editors, *Intelligent Robotics and Applications*, pages 11–22. Springer Berlin Heidelberg, 2009. 245

[21] J.W. Brown and R.V. Churchill. *Complex variables and applications; Seventh Edition*. McGraw-Hill, 2004. 50

[22] J.W. Burdick. On the inverse kinematics of redundant manipulators: characterization of the self-motion manifolds. In *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, pages 264–270, 1989. 144, 195, 198

[23] J. Burgner-Kahrs, H.B. Gilbert, J. Granna, P.J. Swaney, and R.J. Webster III. Workspace characterization for concentric tube continuum robots. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 1269–1275, 2014. 163, 171

[24] Y. Cao, K. Lu, X. Li, and Y. Zang. Accurate numerical methods for computing 2D and 3D robot workspace. *International Journal of Advanced Robotic Systems*, 8:1–13, 2011. 160, 163, 165, 170, 171, 180, 185

[25] S. Caro, D. Chablat, A. Goldsztejn, D. Ishii, and C. Jermann. A branch and prune algorithm for the computation of generalized aspects of parallel robots. *Artificial Intelligence*, 211:34 – 50, 2014. 193

[26] S. Caro, P. Wenger, and D. Chablat. Non-Singular Assembly Mode Changing Trajectories of a 6-DOF Parallel Robot. In *The ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. 2012. 48

[27] J.J. Cervantes-Sánchez, J.C. Hernández-Rodríguez, and J.G. Rendón-Sánchez. On the workspace, assembly configurations and singularity curves of the RRRRR-type planar manipulator. *Mechanism and Machine Theory*, 35(8):1117 – 1139, 2000. 91

[28] D. Chablat, G. Moroz, and P. Wenger. Uniqueness Domains and Non Singular Assembly Mode Changing Trajectories. In *IEEE International Conference on Robotics and Automation*, pages 3946–3951. 2011. 49

[29] Y. Chen, C. Wang, and J. Bao. Design optimization of a novel magnetic switchable device based on Halbach array. *Progress In Electromagnetics Research M*, 31:143–158, 2013. 246

[30] Y. Chen and C.M. Wang. The design of permanent magnetic adhesion system for wall-climbing robot. In *Mechatronics and Applied Mechanics II*, volume 300 of *Applied Mechanics and Materials*, pages 531–536. Trans Tech Publications, 2013. 246

[31] Y. Chen, C.M. Wang, and J.D. Bao. Analytical optimization of a new adjustable magnetic field adhesion mechanism. In *Industrial Instrumentation and Control Systems II*, volume 336 of *Applied Mechanics and Materials*, pages 1129–1133. Trans Tech Publications, 2013. 246

[32] S. Chiaverini, G. Oriolo, and I.D. Walker. Kinematically Redundant Manipulators. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*. Springer, 2008. 123

[33] K.H. Cho, Y.H. Jin, H.M. Kim, H. Moon, J.C. Koo, and H.R. Choi. Multifunctional robotic crawler for inspection of suspension bridge hanger cables: Mechanism design and performance validation. *IEEE/ASME Transactions on Mechatronics*, 22(1):236–246, 2017. 31

[34] K.H. Cho, H.M. Kim, Y.H. Jin, F. Liu, H. Moon, J.C. Koo, and H.R. Choi. Inspection robot for hanger cable of suspension bridge: Mechanism design and analysis. *IEEE/ASME Transactions on Mechatronics*, 18(6):1665–1674, 2013. 31

[35] W.K. Chung and Y. Xu. A novel frame climbing robot: Frambot. In *2011 IEEE International Conference on Robotics and Biomimetics*, pages 2559–2566, 2011. 27

[36] M. Coste. Asymptotic Singularities of Planar Parallel 3-R$\underline{\text{P}}$R Manipulators. In Jadran Lenarcic and Manfred Husty, editors, *Latest Advances in Robot Kinematics*, pages 35–42. Springer Netherlands, 2012. 48

[37] M. Coste. A Simple Proof that Generic 3-RPR Manipulators Have Two Aspects. *ASME J. Mech. Robot.*, 4(1), 2012. 48

[38] M. Coste, D. Chablat, and P. Wenger. *New Trends in Mechanism and Machine Science: Theory and Applications in Engineering*, chapter Perturbation of Symmetric 3-RPR Manipulators and Asymptotic Singularities, pages 23–31. Springer Netherlands, Dordrecht, 2013. 76, 90

[39] M. Coste, D. Chablat, and P. Wenger. Nonsingular Change of Assembly Mode Without any Cusp. In Jadran Lenarcic and Oussama Khatib, editors, *Advances in Robot Kinematics*, pages 105–112. Springer International Publishing, 2014. 48

[40] M. Coste, P. Wenger, and D. Chablat. Hidden cusps. In Jadran Lenarčič and Jean-Pierre Merlet, editors, *Advances in Robot Kinematics 2016*, pages 129–138. Springer International Publishing, 2018. 71, 92

[41] S.R. Dahmen, Z. Farkas, H. Hinrichsen, and D.E. Wolf. Macroscopic diagnostics of microscopic friction phenomena. *Phys Rev E Stat Nonlin Soft Matter Phys*, 71(6-2):066602, 2005. 259

[42] S.R. Dahmen, H. Hinrichsen, A. Lysov, and D.E. Wolf. Coupling between static friction force and torque for a tripod. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(03):P03004, 2005. 260, 261, 263, 264, 271

[43] F. DallaLibera and H. Ishiguro. Non-singular Transitions Between Assembly Modes of 2-DOF Planar Parallel Manipulators With a Passive Leg. *Mechanism and Machine Theory*, 77:182–197, 2014. 48

[44] K.A. Daltorio, T.C. Witushynsky, G.D. Wile, L.R. Palmer, A. Ab Malek, M.R. Ahmad, L. Southard, S.N. Gorb, R.E. Ritzmann, and R.D. Quinn. A body joint improves vertical to horizontal transitions of a wall-climbing robot. In *2008 IEEE International Conference on Robotics and Automation*, pages 3046–3051, 2008. 246

[45] B. Danaei, N. Karbasizadeh, and M.T. Masouleh. A general approach on collision-free workspace determination via triangle-to-triangle intersection test. *Robotics and Computer-Integrated Manufacturing*, 44:230 – 241, 2017. 161, 229

[46] D. DeMers and K. Kreutz-Delgado. Issues in learning global properties of the robot kinematic mapping. In *Proceedings of the 1993 IEEE International Conference on Robotics and Automation*, pages 205–212, 1993. 198

[47] D. DeMers and K. Kreutz-Delgado. Canonically parameterized families of inverse kinematic functions for redundant manipulators. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1881–1886, 1994. 198, 199

[48] D.E. Demers. *Learning to Invert Many-to-one Mappings*. PhD thesis, University of California at San Diego, La Jolla, CA, USA, 1993. 198

[49] L. Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986. 16, 178

[50] P. Dong, X. Wang, H. Xing, Y. Liu, and M. Zhang. Design and control of a tracked robot for search and rescue in nuclear power plant. In *2016 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 330–335, 2016. 34

[51] A.P. Dubey, S.M. Pattnaik, A. Banerjee, R. Sarkar, and S. Kumar. Autonomous control and implementation of coconut tree climbing and harvesting robot. *Procedia Computer Science*, 85:755 – 766, 2016. International Conference on Computational Modelling and Security (CMS 2016). 30

[52] C. Ericson. *Real-Time Collision Detection*. CRC Press, 2004. 178, 235

[53] F. Esquembre. *Creación de simulaciones interactivas en Java: aplicación a la enseñanza de la física*. Pearson Educación, 2004. 98

[54] F. Esquembre. Easy Java Simulations: a software tool to create scientific simulations in java. *Computer Physics Communications*, 156(2):199–204, 2004. 95

[55] B. Esser and D.R. Huston. Versatile robotic platform for structural health monitoring and surveillance. *Smart Structures and Systems*, 1(4):325–338, 2005. 245

[56] T. Estier, Y. Crausaz, B. Merminod, M. Lauria, R. Piguet, and R. Siegwart. An innovative space rover with extended climbing abilities. In *Proc. Space and Robotics 2000*. 34

[57] M.I.N. Faizal, W.A.F.W. Othman, and S.S.N.A.S. Hassan. Development of pole-like tree climbing robot. In *2015 IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, pages 224–229, 2015. 30

[58] G. Figliolini, P. Rea, and M. Conte. Mechanical design of a novel biped climbing and walking robot. In Vincenzo Parenti-Castelli and Werner Schiehlen, editors, *ROMANSY 18 Robot Design, Dynamics and Control*, pages 199–206. Springer Vienna, 2010. 28

[59] A. Gil, A. Peidró, J.M. Marín, O. Reinoso, D. Valiente, L.M. Jiménez, and M. Juliá. Laboratorio virtual y remoto de robots paralelos. In *Actas de las XXXIV Jornadas de Automática*, pages 235–241. CEA, 2013. 12

[60] A. Gil, A. Peidró, Ó. Reinoso, and J.M. Marín. Implementation and assessment of a virtual laboratory of parallel robots developed for engineering students. *IEEE Transactions on Education*, 57(2):92–98, 2014. 10

[61] K. Gilpin, K. Kotay, D. Rus, and I. Vasilescu. Miche: Modular shape formation by self-disassembly. *The International Journal of Robotics Research*, 27(3-4):345–372, 2008. 245

[62] A. Gimenez, M. Abderrahim, V.M. Padron, and C. Balaguer. Adaptive control strategy of climbing robot for inspection applications in construction industry. *IFAC Proceedings Volumes*, 35(1):19–24, 2002. 27, 29, 123

[63] P. Gonzalez de Santos, M.A. Jiménez, and M.A. Armada. Improving the Motion of Walking Machines by Autonomous Kinematic Calibration. *Autonomous Robots*, 12(2):187–199, 2002. 123

[64] C. Gosselin. Determination of the workspace of 6-DOF parallel manipulators. *Journal of Mechanical Design*, 112(3):331–336, 1990. 160

[65] C. Gosselin and J. Angeles. Singularity analysis of closed-loop kinematic chains. *IEEE Transactions on Robotics and Automation*, 6(3):281–290, 1990. 45

[66] C.M. Gosselin and J.-P. Merlet. The Direct Kinematics of Planar Parallel Manipulators: Special Architectures and Number of Solutions. *Mechanism and Machine Theory*, 29(8):1083–1097, 1994. 49, 59, 62

[67] Y. Guan, L. Jiang, H. Zhu, X. Zhou, C. Cai, W. Wu, Z. Li, H. Zhang, and X. Zhang. Climbot: A modular bio-inspired biped climbing robot. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1473–1478, 2011. 26, 28, 247

[68] Y. Guan, K. Yokoi, and X. Zhang. Numerical methods for reachable space generation of humanoid robots. *The International Journal of Robotics Research*, 27(8):935–950, 2008. 160, 162, 163, 182

[69] P. Gui, L. Tang, and S. Mukhopadhyay. A novel design of anti-falling mechanism for tree pruning robot. In *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, pages 812–816, 2015. 30

[70] S.-C. Han, J. Kim, and H.-C. Yi. A novel design of permanent magnet wheel with induction pin for mobile robot. *International Journal of Precision Engineering and Manufacturing*, 10(4):143–146, 2009. 245

[71] E.J. Haug, C.-M. Luh, F.A. Adkins, and J.-Y. Wang. Numerical algorithms for mapping boundaries of manipulator workspaces. *Journal of Mechanical Design*, 118(2):228–234, 1996. 160, 221

[72] A. Hernández, O. Altuzarra, V. Petuya, and E. Macho. Defining Conditions for Nonsingular Transitions Between Assembly Modes. *IEEE Transactions on Robotics*, 25(6):1438–1447, 2009. 49

[73] N. Hewapathirana, L. Uddawatta, J. Karunadasa, and T. Nanayakkara. Analysis on four legged multipurpose rope climbing robot. In *2009 International Conference on Industrial and Information Systems (ICIIS)*, pages 505–510, 2009. 33

[74] S. Hirose, Y. Fukuda, K. Yoneda, A. Nagakubo, H. Tsukagoshi, K. Arikawa, G. Endo, T. Doi, and R. Hodoshima. Quadruped walking robots at Tokyo Institute of Technology. *IEEE Robotics Automation Magazine*, 16(2):104–114, 2009. 24

[75] HKCM. HKCM Engineering. `https://www.hkcm.de/`. Accessed: 2018-07-01. 255

[76] H. Huang, D. Li, Z. Xue, X. Chen, S. Liu, J. Leng, and Y. Wei. Design and performance analysis of a tracked wall-climbing robot for ship inspection in shipbuilding. *Ocean Engineering*, 131:224 – 230, 2017. 24, 25

[77] M. Husty, J. Schadlbauer, S. Caro, and P. Wenger. The 3-RPS Manipulator Can Have Non-Singular Assembly-Mode Changes. In Federico Thomas and Alba Pérez Gracia, editors, *Computational Kinematics*, volume 15 of *Mechanisms and Machine Science*, pages 339–348. Springer Netherlands, 2014. 48

[78] C. Innocenti and V. Parenti-Castelli. Singularity-Free Evolution From One Configuration to Another in Serial and Fully-Parallel Manipulators. *ASME J. Mech. Design*, 120(1):73–79, 1998. 47

[79] M.F. Kaloorazi, M.T. Masouleh, and S. Caro. Determination of the maximal singularity-free workspace of 3-DOF parallel mechanisms with a constructive geometric approach. *Mechanism and Machine Theory*, 84:25 – 36, 2015. 160

[80] M.F. Kaloorazi, M.T. Masouleh, and S. Caro. Collision-free workspace of parallel mechanisms based on an interval analysis approach. *Robotica*, 35(8):1747–1760, 2017. 193

[81] M. Kamezaki, H. Ishii, T. Ishida, M. Seki, K. Ichiryu, Y. Kobayashi, K. Hashimoto, S. Sugano, A. Takanishi, M.G. Fujie, S. Hashimoto, and H. Yamakawa. Design of four-arm four-crawler disaster response robot octopus. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2840–2845, 2016. 35

[82] T. Kang, H. Kim, T. Son, and H. Choi. Design of quadruped walking and climbing robot. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 619–624, 2003. 24

[83] G.R. Kanna and M. Ashik. Design and development of a rope climbing robot using four bar mechanism with wireless control using TX2/RX2 RF module. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–6, 2015. 33

[84] J. Katrasnik, F. Pernus, and B. Likar. New robot for power line inspection. In *2008 IEEE Conference on Robotics, Automation and Mechatronics*, pages 1195–1200, 2008. 33

[85] J. Katrasnik, F. Pernus, and B. Likar. A survey of mobile robots for distribution power line inspection. *IEEE Transactions on Power Delivery*, 25(1):485–493, 2010. 31, 33

[86] X. Kong and C.M. Gosselin. Forward Displacement Analysis of Third-class Analytic 3-RPR Planar Parallel Manipulators. *Mechanism and Machine Theory*, 36(9):1009–1018, 2001. 49, 58

[87] X. Kong and C.M. Gosselin. Generation and forward displacement analysis of R$\underline{P}$R-PR-R$\underline{P}$R analytic planar parallel manipulators. *J. Mech. Design*, 124(2):294–300, 2002. 40, 41, 47, 48, 125

[88] F.M. Köse, M. Kuncan, and H.M. Ertunc. Development of rope climbing robot with caterpillar. In *Proceedings of 18th Mechanika International Conference*, 2013. 33

[89] K.D. Kotay and D.L. Rus. Navigating 3D steel web structures with an inchworm robot. In *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 96)*, volume 1, pages 368–375, 1996. 244

[90] G. Lee, G. Wu, S.H. Kim, J. Kim, and T. Seo. Combot: Compliant climbing robotic platform with transitioning capability and payload capacity. In *2012 IEEE International Conference on Robotics and Automation*, pages 2737–2742, 2012. 24

[91] J. Li, X. Liu, S. Jiang, R. Li, and L. Ren. Design of continuous climbing pneumatic cable maintenance robot. In *2009 International Conference on Mechatronics and Automation*, pages 4633–4637, 2009. 31

[92] Y. Li, M.Z.Q. Chen, Y.H. Chen, and J. Lam. Design of a one-motor tree-climbing robot. In *2015 IEEE International Conference on Information and Automation*, pages 26–31, 2015. 30

[93] Z. Li, H. Wang, and Y. Wang. Path planning for power transmission line inspection robot based on visual obstacle detection. In *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 158–163, 2013. 32

[94] R. Liang, M. Altaf, E. Ahmad, R. Liu, and K. Wang. A low-cost, light-weight climbing robot for inspection of class curtains. *International Journal of Advanced Robotic Systems*, 11(7):106, 2014. 24

[95] S.K. Lim, D.I. Park, Y.K. Kwak, B.S. Kim, and S.W. Jeon. Variable geometry single-tracked mechanism for a rescue robot. In *IEEE 2005 International Safety, Security and Rescue Rototics Workshop*, pages 111–115, 2005. 34

[96] X.-J. Liu and J. Wang. *Parallel Kinematics: Type, Kinematics, and Optimal Design*. Springer Tracts in Mechanical Engineering. Springer Berlin Heidelberg, 2013. 160

[97] D. Longo and G. Muscato. SCID - a non-actuated robot for walls exploration. In *Proceedings of the 2001 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 874–879, 2001. 244

[98] D. Longo and G. Muscato. The Alicia3 climbing robot: a three-module robot for automatic wall inspection. *IEEE Robotics Automation Magazine*, 13(1):42–50, 2006. 24, 25

[99] K.H. Low, W.K. Loh, H. Wang, and J. Angeles. Motion study of an omni-directional rover for step climbing. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 1585–1590, 2005. 34

[100] C.L. Lück and S. Lee. Redundant manipulator self-motion topology under joint limits with an 8-DOF case study. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 848–855, 1993. 197, 225

[101] C.L. Lück and S. Lee. Redundant manipulators under kinematic constraints: a topology-based kinematic map generation and discretization. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 3, pages 2496–2501, 1995. 198

[102] C.L. Lück and S. Lee. Topology-based analysis for redundant manipulators under kinematic constraints. In *Proceedings of the 34th IEEE Conference on Decision and Control*, volume 2, pages 1603–1608, 1995. 197

[103] B.L. Luk, D.S. Cooke, S. Galt, A.A. Collie, and S. Chen. Intelligent legged climbing service robot for remote maintenance applications in hazardous environments. *Robotics and Autonomous Systems*, 53(2):142 – 152, 2005. 24, 25

[104] L.T. Lun and X. Yangsheng. Biologically inspired tree-climbing robot with continuum maneuvering mechanism. *Journal of Field Robotics*, 29(6):843–860, 2012. 29

[105] J. Luo, S. Xie, and Z. Gong. Cable maintenance robot and its dynamic response moving on the horizontal cable. In *Proceedings of the 12th International Conference on Advanced Robotics (ICAR '05)*, pages 514–517, 2005. 31

[106] E. Macho, O. Altuzarra, E. Amezua, and A. Hernandez. Obtaining configuration space and singularity maps for parallel manipulators. *Mechanism and Machine Theory*, 44(11):2110 – 2125, 2009. 161

[107] E. Macho, O. Altuzarra, V. Petuya, and A. Hernández. Workspace Enlargement Merging Assembly Modes. Application to the 3-$\underline{R}$RR Planar Platform. *Int. J. Mechanics and Control*, 10(1):13–20, 2009. 47

[108] E. Macho, O. Altuzarra, C. Pinto, and A. Hernández. Transitions Between Multiple Solutions of the Direct Kinematic Problem. In Jadran Lenarcic and Philippe Wenger, editors, *Advances in Robot Kinematics: Analysis and Design*, pages 301–310. Springer Netherlands, 2008. 48

[109] S. Magnenat, R. Philippsen, and F. Mondada. Autonomous construction using scarce resources in unknown environments. *Autonomous Robots*, 33(4):467–485, 2012. 246

[110] J. Mampel, K. Gerlach, C. Schilling, and H. Witte. A modular robot climbing on pipe-like structures. In *Proceedings of the 4th International Conference on Autonomous Robots and Agents*, pages 87–91, 2009. 28

[111] T. Matsuzawa, T. Matsubara, K. Hashimoto, T. Teramachi, X. Sun, S. Kimura, N. Sakai, Y. Yoshida, A. Imai, K. Kumagai, K. Yamaguchi, K. Namura, and A. Takanishi. Body mechanism with linear spikes for slippage reduction of four-limbed robot crawling on uneven terrain. In Vigen Arakelian and Philippe Wenger, editors, *ROMANSY 22 – Robot Design, Dynamics and Control*, pages 280–287. Springer International Publishing, 2019. 34

[112] Maxon. Maxon Motor. `https://www.maxonmotor.com`. Accessed: 2018-07-24. 242

[113] P. McAree and R. Daniel. An Explanation of Never-Special Assembly Changing Motions for 3-3 Parallel Manipulators. *Int. J. Robotics Research*, 18(6):556–574, 1999. 47

[114] C. Meng, T. Wang, S. Guan, L. Zhang, J. Wang, and X. Li. Design and analysis of gecko-like robot. *Chinese Journal of Mechanical Engineering-English Edition*, 24(2):224, 2011. 24

[115] J.-P. Merlet. Determination of the orientation workspace of parallel manipulators. *Journal of Intelligent and Robotic Systems*, 13(2):143–160, 1995. 160, 219

[116] J.-P. Merlet. *Parallel Robots*. Solid Mechanics and Its Applications. Springer Netherlands, 2006. 160

[117] J.-P. Merlet and D. Daney. Legs interference checking of parallel robots over a given workspace or trajectory. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA 2006)*, pages 757–762, 2006. 193

[118] J.-P. Merlet, C.M. Gosselin, and N. Mouly. Workspaces of planar parallel manipulators. *Mechanism and Machine Theory*, 33(1-2):7 – 20, 1998. 160, 193, 197

[119] R. Mittal, V.R. Varada, S. Dave, A. Khanna, R. Korpu, and S. Tilak. Semi-autonomous arecanut tree-climbing robot. In *2014 9th International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5, 2014. 30

[120] G. Moroz, D. Chablat, P. Wenger, and F. Rouiller. Cusp Points in the Parameter Space of R$\underline{P}$R-2P$\underline{R}$R Parallel Manipulators. In Doina Pisla, Marco Ceccarelli, Manfred Husty, and Burkhard Corves, editors, *New Trends in Mechanism Science*, volume 5 of *Mechanisms and Machine Science*, pages 29–37. Springer Netherlands, 2010. 49

[121] G. Moroz, F. Rouiller, D. Chablat, and P. Wenger. On the Determination of Cusp Points of 3-R$\underline{P}$R Parallel Manipulators. *Mechanism and Machine Theory*, 45(11):1555–1567, 2010. 48

[122] A. Mostashfi, A. Fakhari, and M.A. Badri. A novel design of inspection robot for high-voltage power lines. *Industrial Robot: An International Journal*, 41(2):166–175, 2014. 31, 32

[123] A. Müller. On the terminology and geometric aspects of redundant parallel manipulators. *Robotica*, 31(1):137–147, 2013. 195

[124] K.E. Neumann. Robot, US4732525A, 1988. 102

[125] F. Nigl, S. Li, J.E. Blum, and H. Lipson. Structure-reconfiguring robots: Autonomous truss reconfiguration and manipulation. *IEEE Robotics Automation Magazine*, 20(3):60–71, 2013. 26

[126] D. Pagano, D. Liu, and K. Waldron. A method for optimal design of an inchworm climbing robot. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1293–1298, 2012. 25, 28

[127] G. Pagis, N. Bouton, S. Briot, and P. Martinet. Enlarging parallel robot workspace through type-2 singularity crossing. *Control Engineering Practice*, 39:1 – 11, 2015. 117

[128] J. Paskarbeit, S. Beyer, A. Gucze, J. Schröder, M. Wiltzok, M. Fingberg, and A. Schneider. Ourobot - a self-propelled continuous-track-robot for rugged terrain. In *2016 IEEE International Conference on Robotics and Automation*, pages 4708–4713, 2016. 34

[129] N.M. Patrikalakis and T. Maekawa. *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer-Verlag Berlin Heidelberg, 2002. 148

[130] A. Peidró, A. Belando, D. Valiente, O. Reinoso, and L. Payá. A multi-perspective simulator for visualizing and analyzing the kinematics and singularities of 2UPS/U parallel mechanisms. In *INTED2018 Proceedings*, 12th International Technology, Education and Development Conference, pages 3785–3793. IATED, 2018. g, 10, 91, 109, 110, 119

[131] A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, L. Payá, and O. Reinoso. Montecarlo workspace calculation of a serial-parallel biped robot. In Luís Paulo Reis, António Paulo Moreira, Pedro U. Lima, Luis Montano, and Victor Muñoz-Martinez, editors, *Robot 2015: Second Iberian Robotics Conference*, pages 157–169. Springer International Publishing, 2016. 11, 190

[132] A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematic analysis and simulation of a hybrid biped climbing robot. In *2015 12th International Conference on Informatics in Control, Automation and Robotics (ICINCO)*, volume 2, pages 24–34, 2015. 11, 157, 247

[133] A. Peidró, A. Gil, J.M. Marín, Y. Berenguer, and Ó. Reinoso. Kinematics, simulation, and analysis of the planar and symmetric postures of a serial-parallel climbing robot. In Joaquim Filipe, Kurosh Madani, Oleg Gusikhin, and Jurek Sasiadek, editors, *Informatics in Control, Automation and Robotics 12th International Conference, ICINCO 2015 Colmar, France, July 21-23, 2015 Revised Selected Papers*, pages 115–135. Springer International Publishing, 2016. 10, 131, 154, 157

[134] A. Peidró, A. Gil, J.M. Marín, L. Payá, and O. Reinoso. Control de un ascensor como caso práctico para la docencia de control avanzado. In *Actas de las XXXV Jornadas de Automática*, page Capítulo 2. CEA, 2014. 12

[135] A. Peidró, A. Gil, J.M. Marín, L. Payá, and Ó. Reinoso. On the stability of the quadruple solutions of the forward kinematic problem in analytic parallel robots. *Journal of Intelligent & Robotic Systems*, 86(3):381–396, 2017. 10, 92

[136] A. Peidro, A. Gil, J.M. Marin, and O. Reinoso. Inverse kinematic analysis of a redundant hybrid climbing robot. *International Journal of Advanced Robotic Systems*, 12(11):163, 2015. 10, 141, 142, 157

[137] A. Peidró, A. Gil, J.M. Marín, and Ó. Reinoso. A web-based tool to analyze the kinematics and singularities of parallel robots. *Journal of Intelligent & Robotic Systems*, 81(1):145–163, 2016. 10, 118

[138] A. Peidró, A. Hortal, A. Gil, J.M. Marín, D. Úbeda, and Ó. Reinoso. Modelado dinámico y simulación de un robot trepador tipo serie con 4 grados de libertad. In *Actas de las XXXVII Jornadas de Automática*, pages 1067–1074, 2016. 12, 104

[139] A. Peidró, J.M. Marín, A. Gil, Y. Berenguer, and Ó. Reinoso. Analysis and design of a serial-parallel redundant biped climbing robot. In *Libro de Actas de las Jornadas Nacionales de Robótica 2018*, 2018. 8, 11

[140] A. Peidró, J.M. Marín, A. Gil, and O. Reinoso. Performing nonsingular transitions between assembly modes in analytic parallel manipulators by enclosing quadruple solutions. *ASME Journal of Mechanical Design*, 137(12):122302, 2015. 10, 92

[141] A. Peidró, J.M. Marín, Ó. Reinoso, L. Payá, and A. Gil. Parallelisms between planar and spatial tricept-like parallel robots. In Vigen Arakelian and Philippe Wenger, editors, *ROMANSY 22 – Robot Design, Dynamics and Control*, pages 155–162. Springer International Publishing, 2019. 10, 102

[142] A. Peidró, L. Payá, V. Román, J.M. Marín, A. Gil, and O. Reinoso. Laboratorio virtual móvil de robots paralelos. In *Aceptado, a ser publicado en las Actas de las XXXIX Jornadas de Automática*. CEA, 2018. 11

[143] A. Peidró, O. Reinoso, A. Gil, J.M. Marín, and L. Payá. A virtual laboratory to simulate the control of parallel robots. *IFAC-PapersOnLine*, 48(29):19 – 24, 2015. 11, 115, 118

[144] A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. Análisis de estabilidad de singularidades aisladas en robots paralelos mediante desarrollos de Taylor de segundo orden. In *Actas de las XXXVIII Jornadas de Automática*, pages 821–828. CEA, 2017. 12

[145] A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. An improved Monte Carlo method based on Gaussian growth to calculate the workspace of robots. *Engineering Applications of Artificial Intelligence*, 64:197 – 207, 2017. iii, 9, 190, 281

[146] A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, and L. Payá. A method based on the vanishing of self-motion manifolds to determine the collision-free workspace of redundant robots. *Mechanism and Machine Theory*, 128:84 – 109, 2018. iii, 9, 240, 281

[147] A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Calculation of the boundaries and barriers of the workspace of a redundant serial-parallel robot using the inverse kinematics. In *Proceedings of the 13th International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 412–420, 2016. 11, 161, 240

[148] A. Peidró, Ó. Reinoso, A. Gil, J.M. Marín, L. Payá, and Y. Berenguer. Second-order Taylor stability analysis of isolated kinematic singularities of closed-chain mechanisms. In *Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics - Volume 2*, pages 351–358. INSTICC, SciTePress, 2017. 11, 92

[149] A. Peidró, Ó. Reinoso, A. Gil J.M. Marín, and L. Payá. A simulation tool to study the kinematics and control of 2RPR-PR parallel robots. *IFAC-PapersOnLine*, 49(6):268–273, 2016. 11th IFAC Symposium on Advances in Control Education (ACE 2016). 11, 118

[150] A. Peidró, Ó. Reinoso, J.M. Marín, A. Gil, L. Payá, and Y. Berenguer. A simulation tool for visualizing the assembly modes and singularity locus of 3RPR planar parallel robots. In Anibal Ollero, Alberto Sanfeliu, Luis Montano, Nuno

Lau, and Carlos Cardeira, editors, *ROBOT 2017: Third Iberian Robotics Conference: Volume 1*, pages 516–528. Springer International Publishing, 2018. 11, 119

[151] A. Peidró, O. Reinoso, L. Payá, Y. Berenguer, A. Gil, and J.M. Marín. Análisis cinemático y simulación de un robot trepador con arquitectura serie-paralela. In *Actas de las XXXVI Jornadas de Automática*, pages 400–407. CEA, 2015. 12

[152] A. Peidró, J.J. Rodríguez, J.M. Azorín, and O. Reinoso. Implementación de una maqueta de control bilateral de 1 GDL con Arduino para telerrobótica. In *Actas de las XXXV Jornadas de Automática*, page Capítulo 68. CEA, 2014. 12

[153] A. Peidró, C. Tendero, J.M. Marín, A. Gil, L. Payá, and Ó. Reinoso. m-PaRoLa: a mobile virtual laboratory for studying the kinematics of five-bar and 3RRR planar parallel robots. *IFAC-PapersOnLine*, 51(4):178 – 183, 2018. 3rd IFAC Conference on Advances in Proportional-Integral-Derivative Control (PID 2018). 10, 99, 118

[154] J.M. Porta, L. Ros, and F. Thomas. A linear relaxation technique for the position analysis of multiloop linkages. *IEEE Transactions on Robotics*, 25(2):225–239, 2009. 204

[155] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes: The Art of Scientific Computing (3rd Edition)*. Cambridge University Press, 2007. 205, 225

[156] J. Rastegar and D. Perel. Generation of manipulator workspace boundary geometry using the Monte Carlo method and interactive computer graphics. *ASME Journal of Mechanical Design*, 112:452–454, 1990. 160, 163, 193

[157] G. Reina and M. Foglia. On the mobility of all-terrain rovers. *Industrial Robot: An International Journal*, 40(2):121–131, 2013. 34

[158] D. Ren, S. Yang, G. Yan, and Y. Zhang. Study on a novel wheel type tree-climbing robot. In *2014 Seventh International Symposium on Computational Intelligence and Design*, volume 1, pages 150–153, 2014. 30

[159] S.C. Ridgeway, C.D. Crane, and J. Duffy. A forward analysis of a two degree of freedom parallel manipulator. In Jadran Lenarcic and Vincenzo Parenti-Castelli, editors, *Recent Advances in Robot Kinematics*, pages 431–440. Springer Netherlands, 1996. 40, 41

[160] F. Rochat, R. Beira, H. Bleuler, and F. Mondada. *Tremo: An Inspection Climbing Inchworm Based on Magnetic Switchable Device*, pages 421–428. World Scientific, 2012. 28, 246, 249

[161] F. Rochat, P. Schoeneich, M. Bonani, S. Magnenat, F. Mondada, H. Bleuler, and C. Hürzeler. *Design of magnetic switchable device (MSD) and applications in climbing robot*, pages 375–382. World Scientific, 2010. 244

[162] F. Rochat, P. Schoeneich, O.T.-D. Nguyen, H. Bleuler, and F. Mondada. *Tubulo II: Magnetic Climbing Inchworm for Inspection of Boiler Tubes*, pages 949–955. World Scientific, 2012. 246

[163] J.C. Romão, M. Tavakoli, C. Viegas, P. Neto, and A.T. de Almeida. Inchworm-climber: A light-weight biped climbing robot with a switchable magnet adhesion unit. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3320–3325, 2015. 246

[164] J. Saenz, F. Esquembre, F. García, L. de la Torre, and S. Dormido. A new model for a remote connection with hardware devices using javascript. *IFAC-PapersOnLine*, 49:133–137, 2016. 100

[165] R. Saltaren, R. Aracil, O. Reinoso, and M.A. Scarano. Climbing parallel robot: a computational and experimental study of its performance around structural nodes. *IEEE Transactions on Robotics*, 21(6):1056–1066, 2005. 28, 123

[166] R. Saltaren, R. Aracil, J.M. Sabater, Ó. Reinoso, and L.M. Jimenez. Modeling, simulation and conception of parallel climbing robots for construction and service. In *Proceedings of the 2nd International Workshop and Conference on Climbing and Walking Robots (CLAWAR)*, pages 253–265, 1999. 28, 29

[167] H. Schempf, B. Chemel, and N. Everett. Neptune: above-ground storage tank inspection robot system. *IEEE Robotics Automation Magazine*, 2(2):9–15, 1995. 245

[168] D. Schmidt and K. Berns. Climbing robots for maintenance and inspections of vertical structures - A survey of design aspects and technologies. *Robotics and Autonomous Systems*, 61(12):1288–1305, 2013. 22, 23, 24

[169] T. Schober. Clibot - ein seilkletternder roboter zur bauwerksinspektion. *Bautechnik*, 87(2):81–85. 33

[170] A. Shafi, B. Carpenter, and M. Baker. Nested parallelism for multi-core HPC systems using Java. *Journal of Parallel and Distributed Computing*, 69(6):532 – 545, 2009. 15, 216

[171] J. Shang, B. Bridge, T. Sattar, S. Mondal, and A. Brenner. Development of a climbing robot for inspection of long weld lines. *The Industrial Robot*, 35(3):217–223, 2008. 259

[172] H. Shokripour, W.I. Wan Ismail, and Z.M. Karimi. Development of an automatic self balancing control system for a tree climbing robot. *African Journal of Agricultural Research*, 5(21):2964–2971, 2011. 30

[173] N. Shvalb, B. Ben Moshe, and O. Medina. A real-time motion planning algorithm for a hyper-redundant set of mechanisms. *Robotica*, 31(8):1327–1335, 2013. 28

[174] V.G. Silva, M. Tavakoli, and L. Marques. Optimization of a Three Degrees of Freedom DELTA Manipulator for Well-Conditioned Workspace with a Floating Point Genetic Algorithm. *Int. J. Natural Computing Research*, 4(4):1–14, 2014. 161

[175] V. Srinivasan. *Theory of Dimensioning: An Introduction to Parameterizing Geometric Models*. CRC Press, 2003. 67, 81, 84

[176] F. Tâche, W. Fischer, G. Caprari, R. Siegwart, R. Moser, and F. Mondada. Magnebike: A magnetic wheeled robot with high mobility for inspecting complex-shaped structures. *Journal of Field Robotics*, 26(5):453–476, 2009. 26

[177] M. Tavakoli, J. Lourenço, C. Viegas, P. Neto, and A.T. de Almeida. The hybrid OmniClimber robot: Wheel based climbing, arm based plane transition, and switchable magnet adhesion. *Mechatronics*, 36:136 – 146, 2016. 2, 35, 246

[178] M. Tavakoli, L. Marques, and A.T. De Almeida. 3DCLIMBER: Climbing and manipulation over 3D structures. *Mechatronics*, 21(1):48–62, 2011. 1, 2, 26, 27, 28, 100, 104, 123

[179] M. Tavakoli, L. Marques, and A.T. De Almeida. A low-cost approach for self-calibration of climbing robots. *Robotica*, 29(1):23–34, 2011. 123

[180] M. Tavakoli, C. Viegas, J.C. Romao, P. Neto, and A.T. de Almeida. Switchable magnets for robotics applications. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4325–4330, 2015. 245, 246, 270

[181] M. Tavakoli, M.R. Zakerzadeh, G.R. Vossoughi, and S. Bagheri. A hybrid pole climbing and manipulating robot with minimum DOFs for construction and service applications. *Industrial Robot: An International Journal*, 32(2):171–178, 2005. 26, 28, 123

[182] F. Thomas and P. Wenger. On the topological characterization of robot singularity loci. A catastrophe-theoretic approach. In *2011 IEEE International Conference on Robotics and Automation*, pages 3940–3945, 2011. 69

[183] D. Ubeda, J.M. Marin, A. Gil, and O. Reinoso. Design and postures of a serial robot composed by closed-loop kinematics chains. In Serdar Kucuk, editor, *Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization*, pages 125–142. InTech, 2012. e, 2, 35, 36

[184] O. Unver, M. Murphy, and M. Sitti. Geckobot and waalbot: Small-scale wall climbing robots. In *Infotech@ Aerospace*, page 6940. 2005. 24

[185] M. Urízar, V. Petuya, O. Altuzarra, M. Diez, and A. Hernandez. Non-singular transitions based design methodology for parallel manipulators. *Mech. Mach. Theory*, 91:168–186, 2015. 41

[186] M. Urízar, V. Petuya, O. Altuzarra, and A. Hernández. Researching into Non-Singular Transitions in the Joint Space. In Jadran Lenarcic and Michael M. Stanisic, editors, *Advances in Robot Kinematics: Motion in Man and Machine*, pages 45–52. Springer Netherlands, 2010. 48

[187] M. Urízar, V. Petuya, O. Altuzarra, and A. Hernández. Assembly Mode Changing in the Cuspidal Analytic 3-R$\underline{P}$R. *IEEE Transactions on Robotics*, 28(2):506–513, 2012. 49

[188] G. van den Bergen. *Collision Detection in Interactive 3D Environments*. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann Publishers, 2004. 15, 219

[189] C. Viegas and M. Tavakoli. A single DOF arm for transition of climbing robots between perpendicular planes. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2867–2872, 2014. 27

[190] M. Vukobratovic and B. Borovac. Zero-moment point - thirty five years of its life. *International Journal of Humanoid Robotics*, 1(1):157–173, 2004. 250

[191] L. Wang, J. Wu, and D. Tang. Research on workspace of manipulator with complicated constraints. In *Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA 2008)*, pages 995–999, 2008. 163, 184

[192] W. Wang, Y.-C. Bai, G.-P. Wu, S.-X. Li, and Q. Chen. The mechanism of a snake-like robot's clamping obstacle navigation on high voltage transmission lines. *International Journal of Advanced Robotic Systems*, 10(9):330, 2013. 33

[193] P.K. Ward, Faculty of Engineering University of Technology (Sydney), and Information Technology at. *Design of a Biologically Inspired Climbing Robot and an Adhesion Mechanism for Reliable and Versatile Climbing in Complex Steel Structures*. 2016. 28

[194] W. Wei, J.X. Yu, W.H. Lin, R.S. Mai, C.X. Chen, Z.Y. He, and Z. Li. Structure design of climbing snake-like robot for detection of cable-stayed bridge. In *Advanced Materials, Mechanics and Industrial Engineering*, volume 598 of *Applied Mechanics and Materials*, pages 610–618. Trans Tech Publications, 2014. 31

[195] P. Wenger and D. Chablat. Workspace and assembly modes in fully-parallel manipulators: A descriptive study. In Jadran Lenarčič and Manfred L. Husty, editors, *Advances in Robot Kinematics: Analysis and Control*, pages 117–126, Dordrecht, 1998. Springer Netherlands. 115

[196] P. Wenger, D. Chablat, and M. Zein. Degeneracy Study of the Forward Kinematics of Planar 3-R$\underline{P}$R Parallel Manipulators. *ASME J. Mech. Design*, 129(12):1265–1268, 2006. 49

[197] B. Xu, X. Wang, Y. Zhu, and H. Chen. Design of obstacle crossing mechanism of high-voltage transmission line inspection robot. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2539–2544, 2015. 32

[198] F. Xu, J. Shen, and G. Jiang. Kinematic and dynamic analysis of a cable-climbing robot. *International Journal of Advanced Robotic Systems*, 12(7):99, 2015. 30, 31

[199] F. Xu, X. Wang, and L. Wang. Cable inspection robot for cable-stayed bridges: Design, analysis, and application. *Journal of Field Robotics*, 28(3):441–459, 2011. 31

[200] D. Yang, Z. Feng, and X. Zhang. A novel tribrachiation robot for power line inspection and its inverse kinematics analysis. In *2012 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 502–507, 2012. 32

[201] P. Yao and D. Li. The magnetic field analysis and optimization of permanent-magnetic adhesion device for a novel wall-climbing robot. In *International Technology and Innovation Conference 2009 (ITIC 2009)*, pages 1–5, 2009. 246

[202] S. Yifeng, W. Hongguang, and L. Lie. Research on the influence of the driving wheel and robot posture on climbing capability of a transmission line inspection robot. In *2011 6th IEEE Conference on Industrial Electronics and Applications*, pages 1632–1639, 2011. 32

[203] S. Yokota, Y. Ohyama, H. Hashimoto, J.-H. She, H. Kobayashi, and P. Blazevic. Development of rough terrain mobile robot using connected crawler - Derivation of sub-optimal number of crawler stages. In Alexander Zemliak, editor, *Frontiers in Robotics, Automation and Control*, chapter 20. IntechOpen, 2008. 34

[204] Y. Yoon and D. Rus. Shady3D: A Robot that Climbs 3D Trusses. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 4071–4076, 2007. 28

[205] X. Yue, H. Wang, and Y. Jiang. A novel 110 kV power line inspection robot and its climbing ability analysis. *International Journal of Advanced Robotic Systems*, 14(3):1–10, 2017. 32

[206] M. Zein, P. Wenger, and D. Chablat. Singular Curves in the Joint Space and Cusp Points of 3-R$\underline{P}$R Parallel Manipulators. *Robotica*, 25(6):717–724, 2007. 48, 82

[207] M. Zein, P. Wenger, and D. Chablat. Non-singular Assembly-Mode Changing Motions for 3-R$\underline{P}$R Parallel Manipulators. *Mechanism and Machine Theory*, 43(4):480–490, 2008. 48

[208] X. Zhou, Y. Guan, L. Jiang, H. Zhu, C. Cai, W. Wu, and H. Zhang. Stability of biped robotic walking with frictional constraints. *Robotica*, 31(4):573–588, 2013. 250