Universidad Miguel Hernández de Elche

MÁSTER UNIVERSITARIO EN ROBÓTICA

"LiDAR-based Autonomous Power-line Perching with Multirotors"

Trabajo de Fin de Máster

2022/2023

Autor: Iván Gutiérrez Rodríguez
Tutores: Oscar Reinoso García,
José Ramiro Martínez de Dios

Agradecimientos

Agradecer en primer lugar a mi familia, pareja y amigos todo el apoyo recibido durante esta difícil e intensa etapa que llega a su fin. También, a todos los profesores del Master Universitario en Robótica, por la buena acogida y la gran formación que nos han proporcionado, en especial, a Oscar Reinoso por su dedicación. Agradecer también a Ramiro su constante motivación y apoyo, así como a todo el equipo del GRVC Robotics Lab de la Universidad de Sevilla, sin el que no habría sido posible este trabajo. En especial a Raúl, del que he tenido la suerte de poder aprender a diario y el que ha sido un gran apoyo durante esta etapa. Por último, agradecer a la fundación VALGRAI la ayuda para los estudios que me ha permitido llegar hasta aquí.

Iván Gutiérrez Rodríguez
Elche, 2023

Resumen

Este trabajo trata sobre el desarrollo de un método para permitir que vehículos aéreos de tipo multirotor puedan colgarse autónomamente de líneas eléctricas. Se ha desarrollado un sistema de percepción basado en dos LiDARs 2D rotacionales que se encarga de obtener con exactitud la posición y orientación relativa de la línea eléctrica con el robot. Estas estimaciones son recibidas por el sistema de control, que actúa sobre la velocidad del multirotor para alcanzar la posición que permite la actuación del mecanismo de enganche. El método se ha validado con un quadrotor real en el que se ejecutaba todo el software a bordo para controlar de forma precisa la posición relativa del robot con la línea.

Abstract

This work addresses the development of a method to enable autonomous multirotors to perch on power-lines. A perception system based on two rotatory 2D LiDARs has been designed and implemented to accurately estimate the pose of the power-line. These estimations are received by the control system of the aerial robot, which acts on the velocity of the platform to achieve the relative pose with the power-line that allows the actuation of a mechanism to grab the line. The method has been evaluated with a real multirotor that executed all the software on board to accurately control its relative pose with respect to a real power-line cable.

Contents

Resumen III
Abstract V

1 Introduction 1
1.1 Motivation 1
1.2 Objective 1
1.3 Structure 3

2 Related work 5
2.1 Autonomous power-line perching 5
2.2 2D LiDAR sensors 7

3 Perception system architecture 9
3.1 Introduction 9
3.2 Hypotheses 9
3.3 Detection stages 10
3.4 System modules 10
3.5 Conclusion 12

4 Implementation of the perception system 13
4.1 Introduction 13
4.2 Intersection detection module 13
4.3 Power-line filter module 19
4.4 Power-line detection module 24
4.5 Conclusion 26

5 Control system 27
5.1 Introduction 27
5.2 Autopilot 27
5.3 UAL 28
5.4 Velocity control 28
5.5 Conclusion 30

6 Experimental results 31
6.1 Introduction 31
6.2 Perception system performance 31

6.3 Simulated experiments 33
6.4 Flight experiments 37
6.5 Conclusion 41

7 Conclusion 43
7.1 Conclusion 43
7.2 Future work 43

Appendix A Elements of the H matrix 45

List of Figures 51
List of Tables 53
Bibliography 55

1 Introduction

1.1 Motivation

Performing autonomous perching with multirotors on power-lines is considered to be a key point in the development of aerial robots for inspection and maintenance of power-grids. Inspection and maintenance tasks on transmission lines and towers have traditionally been performed by means of extensive manpower. However, they involve high risks for workers and high costs for electric companies. As a consequence, several works have found Unmanned Aerial Vehicles (UAVs) to be a promising technology to automatize and reduce the costs of these tasks. In particular, multirotors offer a large number of possibilities due to their high maneuverability and relatively high payload compared to other types of aerial platforms. However, their flight endurance is not enough for many applications due to the large dimensions of power-grids.

Recent works propose multirotor designs that can recharge their batteries by perching on power-lines and harvesting energy from them [1] [2] [3]. The autonomous execution of the perching maneuver by a multirotor with one of these systems would allow its operation along long power-lines without the need of the intervention of a human to change its batteries. In addition, perching on the power-lines would facilitate the physical interaction with the environment, which is required for many tasks such as the installation of bird diverters, and the tightening of screws, among others.

1.2 Objective

The goal of this work is the development of a method to allow multirotors to autonomously perch on power-lines. The UAV must be able to estimate the pose of the power-line and to use these estimations to control its relative pose with the line. Once the robot is able to properly control its relative pose with respect to the power-line, the perching maneuver will consist in establishing the appropriate references for the control system to achieve those relative poses that allow the actuation of the mechanism which grabs the line.

The actuation of most perching mechanisms requires first aligning the robot with the power-line to then reach a certain distance from the power-line at which the line enters the mechanism that grab it. Therefore, the perception system must be able to detect the orientation of the power-line, and the horizontal and vertical distance with it. The robot must be able to perform the detection under difficult illumination conditions. On the one hand, power-lines are outdoor installations directly exposed to sunlight. On the other hand, the robot may be used at night, which implies being able to detect the power-line in dark conditions. Thus, visual cameras are not suitable for this application, since their performance is severely compromised under these conditions. In contrast, most 3D

LiDARs do not suffer from this problem and allow the estimation of the aforementioned variables if their scans are not exceedingly sparse. However, they are usually very heavy. Some current rotatory 2D LiDARs are lightweight, robust against sunlight, and able to work at night. As a consequence, the robot developed in this work relies on 2 rotatory 2D LiDARs to estimate the pose of the power-line to perch on. Note that it uses 2 sensors since the orientation of the power-line is required and they only measure distances within single planes. The angular resolution of the LiDARs and the width of the power-lines define the distance from the power-lines at which the maneuver can start. The robot developed in this work must be able to perch on 1 cm diameter power-lines from at least 2 meters from the power-line, a position that would be achieved by the manual control of the platform.

Both the control system and the perception system must be able to run in real time on the onboard computer. For the sake of simplicity, the LiDARs are assumed to be configured as shown in Figure 1.1. The detection planes of the LiDARs (XY planes) must be parallel, and their rotational axes (z axes) aligned with each other. This configuration allows the proper estimation of the orientation of the line and simplifies the equations of the probabilistic filter used to estimate the pose of the power-line. As a result, the computational performance of the filter is also improved, which is very important in software for aerial robots, whose limited payload restricts the power of the onboard computers.

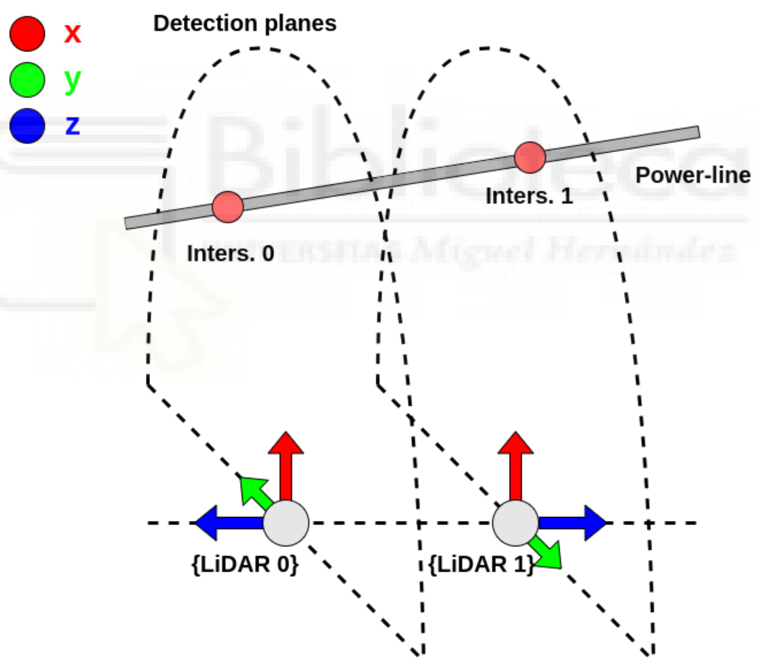


Figure 1.1 Configuration of the LiDARs on board the aerial platform.

The performance of the developed method must be validated with a real multirotor. The software must run on its onboard computer to accurately control the relative pose of the UAV with a power-line. Concurrently, in order to provide a quantitative evaluation of the capabilities of the robot, the errors of the perception system and of the control system must be measured.

1.3 Structure

This work is organized as follows. First, in Chapter 2, the most relevant prior works related to the performance of autonomous perching on cables are analyzed. In addition, a brief overview of the characteristics of current 2D LiDARs is addressed. Chapter 3, describes the architecture of the perception system of the developed robot. Once the different modules for the estimation of the power-line pose are presented, their implementation is explained in Chapter 4. Next, the control system that processes the output of the perception system to perform the maneuver is detailed in Chapter 5. Finally, the experiments to validate the correct functioning of the systems are described in Chapter 6 together with the obtained results.

2 Related work

2.1 Autonomous power-line perching

Power-lines represent a significant part of the infrastructure of many countries. They are composed of a large number of kilometric high voltage cables supported on large metallic towers which require periodic inspection and maintenance tasks. For instance, the detection of defects, the measurement of the distance to the vegetation to anticipate future collisions, and the installation of bird diverters, among others. These tasks have traditionally been performed by means of extensive manpower, involving significant risks due to the great height at which the power-lines hang and their high voltage. This dangerous work must be performed by highly qualified personnel, making it difficult to achieve a sufficient performance to properly maintain current power grids. In addition, these operations are also really expensive since companies not only have to pay the costs related to the operation itself, but also have to shut down the section of the power-grid for safety reasons. Together with the large dimensions of current power grids, these facts have led to a strong interest in the development of technologies for the automation and cost reduction of power-line inspection and maintenance.

Most research on the automation of these tasks has focused on robotic solutions. The ability of robots to operate without the intervention of humans avoids all the risks that workers still take today. Additionally, they can operate continuously without compromising the quality of the operations. There are numerous works that focus on mobile robots for these purposes [4]. However, the need of retrieving them on the power-lines and their difficulty to operate in cables that are separated by towers are common problems that restrict their capabilities. As discussed in the following, Unmanned Aerial Vehicles (UAVs) have attracted a significant attention from researchers to perform inspection and maintenance tasks on power-lines. These platforms can easily reach inaccessible areas where power-lines are usually located and have the ability to operate at any height, which makes them ideal platforms to perform these tasks.

A first step in the automation of power grid inspection with aerial vehicles was collecting data with sensors onboard manned or unmanned helicopters. In [5], the authors rely on the data obtained with a LiDAR sensor to obtain a 3D reconstruction of the electrical system. Other works such as [6] and [7] use visual and thermal cameras to take pictures and perform the inspection by their posterior analysis. However, helicopters are inefficient platforms that are not suitable for long flights following the kilometric power-lines.

Despite fixed-wing platforms being considered considerably efficient, there are not many works that suggest them as proper candidates to perform inspection and maintenance tasks on power-lines due to their lack of maneuverability. On the contrary, Vertical Take Off and Landing (VTOL) vehicles are efficient while being able to hover and move as multirotors. For this reason, authors as those in [8], [9], and [10] suggest them as proper options to perform inspection tasks. Nevertheless, their limited payload severely restricts the use of certain onboard sensors, the available onboard computation, and the possibility of carrying heavy mechanisms for manipulation.

Recently, multirotors have attracted considerable attention due to their high maneuverability and lower payload restrictions compared to VTOLs. On the one hand, works in [11], [12], and [13] use multicopters with onboard LiDARs and/or visual cameras to collect data and perform inspection tasks. On the other hand, works such as those in [14], [15], and [16] propose systems to physically interact with power-lines and perform maintenance tasks as the installation of bird diverters.

Similarly to helicopters, multicopters are not efficient platforms, and their flight endurance is not enough for many inspection applications. Nevertheless, this problem might be tackled by perching on the power-lines to recharge batteries. Several studies have found this as an interesting solution to increase the duration of multicopter flights. In [17] and [2] the authors propose multirotor systems with mechanisms that allow both perching on power-lines and recharging the battery by harvesting energy. Authors in [18] suggest a wireless drone recharging station that could allow the recharging of drones without sufficient payload to carry an energy harvester. Moreover, perching on the power-line might facilitate the physical manipulation with the environment of the robot.

Although the perching maneuver appears to be a key point in the development of aerial systems for autonomous power-line inspection and maintenance with UAVs, there is little agreement on the approach to perform it autonomously. The work [19] presents a method to generate trajectories for perching with multirotors. Although the trajectories are generated to improve the performance of the perception system, the work does not address the estimation of the power-line pose with onboard sensors to close the control loop. In [20], a camera and a LiDAR on board a UAV are used to assist a pilot and perform the perching maneuver semi-autonomously. The work [21] proposes a perception system based on electromagnetic fields near power-lines to allow a UAV to perch on them. There are many other works that rely on the electromagnetic fields to allow UAV navigation around power lines [22], or even grasping them [23]. Nevertheless, these approaches are only useful with active power-lines, usually assume very specific characteristics of the power-lines, and none of them have been validated with software executed onboard a UAV in a fully autonomous mission.

Authors in [24] propose a multirotor design that can estimate the pose of a cable and perch on it with two solid-state 8-segment LiDARs. However, they do not consider the case of multiple lines and use sensors with very low angular resolution, severely limiting the capabilities of the system. The accurate estimation of the power-line pose at several meters requires sensors with very high angular resolution since most power-lines are much thinner than the cables used in their work (38 mm). After a preliminary study of the most suitable sensor to detect and estimate the pose of power-lines [25], the same athors conclude that mmWare radars are remarkable sensors for detecting multiple power-lines. In [26], they used a mmWare radar together with an RGB camera on board a multicopter to autonomously approximate different lines. Nonetheless, the sparsity of the 3D point-clouds provided by mmWare sensors hampers the calculation of the power-line orientation. Thus, the system requires processing images of the RGB camera to calculate the orientation of the power-line, which involves certain computational power and is sensitive to the illumination conditions.

The system presented in this work relies on two rotatory 2D LiDARs to estimate the pose of the power-line and perform a perching maneuver. As noted below, certain models of these sensors have sufficient angular resolution to detect thin power-lines at several meters. Furthermore, they are robust to challenging illumination conditions. Whereas other works require a visual camera to estimate the orientation of the power-line, the estimation of the power-line pose in this work is completely based on LiDAR sensors. Cameras are susceptible to failure if the Sun appears in their field of view and are not able to work properly at night. On the contrary, LiDARs do not suffer from these problems and can be used to detect power-lines both at night and during the day, regardless of the position of the Sun. Although there are numerous cameras lighter than these sensors, their use results in a more robust perception system for those platforms that are able to carry them. In fact, most platforms used for inspection and maintenance tasks use heavy equipment such as 3D LiDARs, high-resolution cameras with stabilization systems, or robotic arms for manipulation. In this way, the weight of two rotatory 2D LiDARs involves a small increase of the payload compared to that required due to the task to be performed.

2.2 2D LiDAR sensors

2D LiDAR sensors provide distance measurements to objects at different angles within a plane, the detection plane. They are Time of Flight (ToF) sensors that emit infrared laser light and calculate the distance to the object the ray intersects from the time elapsed until the LiDAR receives the light reflected by the object. Since laser light its remarkably directional, they provide accurate geometrical information of the environment which is difficult to obtain with other sensors as sonars or visual cameras. Thus, they are extensively used in localization of mobile robots [27] and of UAVs when the payload of the platform is compatible [28] [29].

The purpose of this work is to enable an UAV to autonomously perch on power-lines of 1 cm diameter. The detection of such thin cables from a considerable distance with rotatory 2D LiDARs requires a demanding angular resolution for the sensors. The angular resolution defines the maximum distance at which the lines are always detected. As shown in Figure 2.1, at that distance, the separation of the laser rays is approximately equal to the line width. The maximum detection distance given a certain angular resolution can be calculated as $d = \frac{l_w}{2tan(\frac{\psi}{2})}$, where d, l_w, and ψ are defined in the figure. If the power-line is further than that distance, LiDAR rays might not collide with the 1 cm diameter cable, and might not be detected. Although the line could still be detected if a single ray intersects it, this situation is not taken into account for the calculation of the maximum detection distance.

Current LiDAR models of the company *SLAMTEC* [30] are lightweight, robust against difficult illumination conditions, and have a very high angular resolution. In this way, they are fully suitable for the proposed application. Table 2.1 shows the angular resolution of some of the last LiDAR models of this company together with their maximum detection distance with lines of 1 cm diameter, and other relevant properties such as their weight and blind rate (minimum detection distance). As shown, S2 and S3 models have almost the same angular resolution and are able to detect power-lines of 1 cm diameter from more than 4 meters. They also have a similar blind range, but, since RPLIDAR S3 is lighter than the S2, it would be the best sensor for this specific application. On the contrary, RPLIDAR A3 has the lowest angular resolution, the highest blind range, and is the heaviest. However, it is still able to detect power-lines of 1 cm of diameter at up to 2.54 m. The blind range is also important, but, however, in case it is necessary to detect at shorter distances, the problem can be solved by installing the LiDAR on the aerial robot such that a higher cable-LiDAR distance is obtained. As shown, the characteristics of current rotatory 2D LiDAR models allow the detection of power-lines from a considerable distance. The suitability of a specific model for a given application depends on the payload of the platform to be used and the required maximum detection distance. In

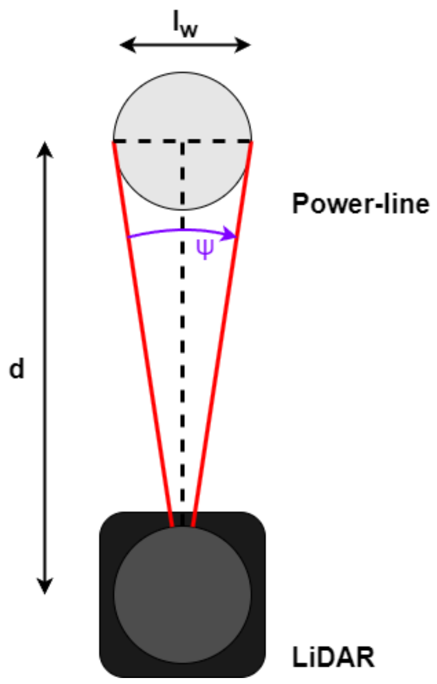


Figure 2.1 Variables to calculate the maximum detection distance given a certain angular resolution. l_w is the width of the power-line, d is the maximum detection distance, and ψ is the angular resolution.

the case of this work, all the models might be used since all provide a sufficient maximum detection distance for power-lines of 1 cm diameter, and the payload of the platform used for the experiments is much higher than the weight of the LiDARs.

Table 2.1 Properties of *SLAMTEC* LiDARs suggested for power-line detection on board a UAV. ψ is the angular resolution, d is the maximum detection distance with power-lines of 1cm diameter, BR is the blind range, and W is the weight.

LiDAR	ψ [º]	d [m]	BR [m]	W [g]
RPLIDAR A3	0.225	2.54	0.2	200
RPLIDAR S2	0.12	4.77	0.05	190
RPLIDAR S3	0.1125	5.09	0.05	115

3 Perception system architecture

3.1 Introduction

Despite 2D LiDARs appear to be suitable sensors for power-line pose estimation, robustly performing it requires the use of certain assumptions about the scenario and the maneuver to be performed. On the one hand, distinguishing power-line intersections from other objects in the scans implies previous knowledge about the geometry of the scenario. On the other hand, ensuring the correct match of the points detected with each LiDAR to get the pose of the power-line requires following a certain methodology for the detection. The following sections describe all these assumptions and how the perception system has been designed to robustly and accurately estimate the pose of a power-line.

This chapter is organized as follows. First, the hypotheses about the scenario and the usage of the system are listed. After that, it presents the stages followed by the perception system to estimate the pose of the power-line. Finally, it discusses the different modules implemented to perform the stages.

3.2 Hypotheses

In order to simplify the development of the perception system, some assumptions have been made regarding the maneuver performed, the geometry of the environment in which it will be used, and the available knowledge.

First, the maneuver consists of two steps that facilitate the accurate estimation of the power-line pose. Initially, the UAV is in a static position below or above the power-lines at a certain distance within the LiDAR detection interval. It must also be approximately aligned with the power-lines to ensure that the detection planes cut the power-lines. The first step consists in the alignment with one of the power-lines while maintaining the initial height. Then, it moves straight to the power-line until the perching mechanism is able to actuate. The power-line must be out of the blind range of the LiDARs in the perching position, involving an important design requirement for the configuration of the robot. Hence, the LiDARs are always able to see the power-lines, which is another hypothesis.

Secondly, in order to perform the detection of the intersections of the power-lines in the measured scans, the width and separation of the lines are supposed to be known. Furthermore, the separation of the lines from other objects of the environment is assumed to be greater than their separation from other lines, which is usually true for safety reasons.

Lastly, power-lines are assumed to be closer to the robotic platform than any object in the environment. Consequently, if the robot detects more possible lines than those at the electric asset, it rejects the farthest. The number of power-lines is also supposed to be known.

3.3 Detection stages

The objective of the perception system is to estimate the pose of a power-line. However, 2D LiDAR sensors can only measure the intersection point between the detection plane and the power-line. Therefore, the estimation is performed following a series of steps that will result in a robust and accurate perception system.

As shown in Figure 3.1, once both LiDAR drivers are initialized and provide scans, the detection of the intersection points of the power-lines starts. Although the perching maneuver is done on a single line, all line intersections must be detected due to the need of matching them to identify the different power-lines. This detection is supposed to be done in a static manner, allowing for an accurate estimation of the positions where the lines intersect.

After that, each of them has to be matched with an intersection detected in the opposite detection plane. Therefore, the different power-lines are distinguished and their accurate pose estimation is now possible, which is done during the last detection stage, the power-line tracking.

Since the maneuver is performed on a single line, the tracking of the power-line is performed for that specific line and not for all of them, which would imply an unnecessary computational cost. The tracked power-line would be the closest in order to minimize the time in which the maneuver is performed. Tracking is performed by detecting all intersection points and updating an Extended Kalman Filter (EKF) with the points most compatible with the previous estimation. The measurements are rejected and identified as outliers if the distance between the detected points and the intersection of the estimated line with the detection planes is greater than a threshold. If a significant number of outliers are received in a row, the estimation is assumed to be inconsistent, and the system returns to the stage for the detection of the intersections.

3.4 System modules

The system is composed of three different modules that collaborate to perform the previous stages. As shown in Figure 3.2, LiDAR scans are sent to the *Power-line Detection* module, which is the main module and uses the others to achieve the objective of the active stage.

If scans from both LiDARs have been received and, therefore, the stage to detect intersection points is active, the *Power-line Detection* module forwards them to the *Intersection Detection* module. This module uses an algorithm to detect possible power-line intersection points by taking advantage of their known fixed size and isolation, as will be detailed in Chapter 4. At first, before the matching stage, the system needs an accurate estimation of every intersection point position in each plane. However, intersection detections are noisy and might not be performed in all the scans so that some processing is required to reject outliers and filter them. Consequently, the *Intersection Detection* module uses EKFs to estimate the position of each line intersection in each detection plane. Since the position of each line intersection must be estimated independently, the module uses two EKFs for each of the power-lines to be detected.

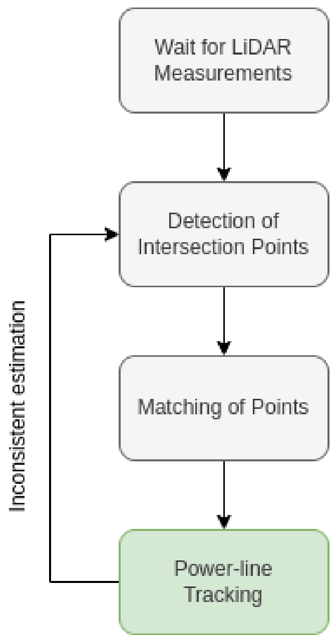


Figure 3.1 Diagram of the perception system stages.

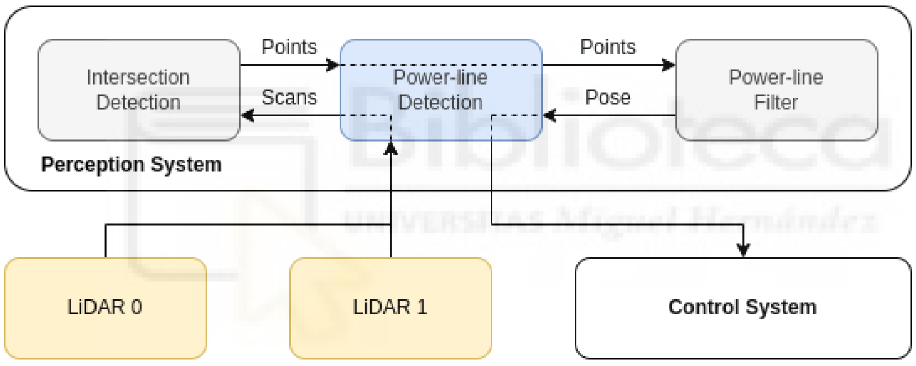


Figure 3.2 Diagram of the perception system modules.

Once every EKF has been initialized and there is a robust position estimation of each intersection, the matching stage starts. The *Power-line Detection* module is responsible for the matching operation, which is based on the geometry of the lines and the detection planes. This will be explained in Chapter 4.

After the identification of the different power-lines, the *Intersection Detection* module stops filtering the detected points and returns their location without any processing. Next, the *Power-line Detection* module gets the closest line and starts its tracking by estimating the pose and updating it with the closest points detected by the *Intersection Detection* module. The estimation of the line is performed by forwarding the detected points to the *Power-line Filter* module. This module implements the EKF for the pose estimation, whose measurement model takes into account the geometry of the problem to provide an accurate estimation of the power-line pose.

The *Power-line Detection* module forwards to the *Power-line Filter* module those detected intersection points which are closest to the intersections of the estimated line with the detection planes. As mentioned above, if the distance between them is larger than a threshold, they are rejected since they are possible outliers which the filter cannot manage. If a significant number of them are received in a row, the power-line is assumed to be lost and the system returns to the stage for detecting intersection points.

3.5 Conclusion

This chapter has presented the different modules of the perception system and the functionalities for which they are responsible. The characteristics of the problem impose the usage of different detection stages. Each of them relies on some hypotheses to achieve a certain objective. The achievement of the different objectives results in the identification and accurate estimation of the pose of the power-line, allowing the feedback of the control system.

The architecture is designed based on the different stages necessary for the detection and the functionalities required during each one, resulting in a simple system in which each module has a specific role. The following chapter deals with how each of these modules has been implemented.

4 Implementation of the perception system

4.1 Introduction

The implementation of the modules presented in the previous chapter requires the selection of the proper algorithms and techniques to process the data and obtain the desired results. This chapter presents a comprehensive overview of how the different modules have been implemented and the approaches selected to achieve their objectives. All the software has been developed in C++ using ROS.

Each section of this chapter describes the implementation of each module of the perception system, which are the *Intersection Detection* module, the *Power-line Filter* module, and the *Power-line Detection* module.

4.2 Intersection detection module

The *Intersection Detection* module has been implemented as a C++ class that provides the public member functions shown in Table 4.1, which represent the interface of the module with the others. The module is in charge of processing LiDAR scans and returning possible line intersection points. The computation of these points is performed with the computeScan() function, which receives a scan and returns a list of possible line intersection points defined in polar coordinates by their range (ρ) and angle (θ) in the LiDAR detection plane. Nevertheless, the processing of the scans is not always the same. As mentioned in the previous chapter, the detected points are initially filtered to allow a robust matching of the intersection points detected with the different LiDARs. The filtering is executed by this module, but the selection of the active stage is performed by the *Power-line Detection* module. That is why this module has to provide a mechanism to activate and deactivate the filtering. At first, this module always filters the points. It uses as many filters as lines in the environment, and they are initialized as new intersections are detected. The matching of the points cannot start until every filter has been initialized. Thus, the *Power-line Detection* module will call the isInitialized() function to check if the initialization has been completed. Once the filters are initialized and the matching has been done, the *Power-line Detection* module will call the stopFilter() function to stop the filtering. After that, the module returns the points directly detected and the filtering is performed by the *Power-line Filter* module. Finally, the reset() function allows recovering the module to the initial state, in which the detected points are filtered. This might might be necessary if the line tracking fails and the perception system needs to get back to a previous stage.

Table 4.1 Public member functions provided by the *Intersection Detection* module.

Name	Description
computeScan()	Receives a 2D LiDAR scan and applies an algorithm to detect possible line points. The function allows the tracking and filtering of the detected points. The rest of the public functions are related to this filtering functionality.
isInitialized()	Returns whether or not the module has detected all the lines in the scenario and, therefore, has started the separated filtering of their position. The number of lines is assumed to be known.
stopFilter()	After a call to this function, the module stops filtering the detected points and returns the set of points directly obtained with the detection algorithm.
reset()	This functions recovers the system to the initial state in which the points are filtered.

The main element of this module is the algorithm for differentiating line points within the laser scan. In most cases, power-lines are completely separated from other objects, resulting in a few isolated points associated with the intersection points. The system has been designed to perch on power lines above or below the robot. When the lines are above the robot, the LiDAR scan within the angle interval near the lines would only consist of the isolated lines, since the sky would not produce any detection. In this case, the detection might be easily performed by using a clustering algorithm such as K-means. However, when the lines are below the robot, other objects might appear in the scan, preventing the use of a simple clustering algorithm. This might be avoided by filtering those measured points farther than a certain threshold, but would result in a less adaptive and robust system. As a consequence, the algorithm used is based on the hypothesis of knowing the width and separation of the power-lines as well as that about the greater separation of random objects from the lines than between the lines themselves. In this way, it detects those points that meet these distance conditions, assuming that they can only be generated by power lines. It would fail if there are other lines in the environment apart from those related to the electrical system but, however, this problem is taken into account in the development so that it would not produce a failure of the detection system.

Although checking the distance condition for each point in the scan might seem an easy solution for the detection problem, it is not valid due to the extensive computational cost required for that. It would consist in iterating through the points measured by the LiDAR and, for each one, iterating through them again to check that there are no points closer than the line separation and farther than the line width at the same time. If any point meets the condition, it would be assumed to be related to the intersection of a line. Nevertheless, it would imply a number of operations in the order of the number of measured points up to two, which is usually a large number of operations that would prevent a real-time implementation of the algorithm. Therefore, the chosen approach takes advantage of the way in which the rotatory 2D LiDARs measure the scans to avoid extensive computation. Since scans consist of distance measurements at different angles, discarding a significant part of the scan is possible by studying the difference of distances at adjacent angles. Algorithm 1 shows how to perform that.

The points are studied in pairs, checking the distance between them. At first, the two points associated with the first two angles are selected and are assumed to be part of a cluster of points corresponding to an intersection point. The starting point is also saved in order to check the width

Algorithm 1 Obtains potential line intersection points within LiDAR scans. S represents the set of points of the scan and their elements are represented as S_{index}. The consecutive elements are assumed to be associated with measurements at adjacent angles. I is the set of intersection points the algorithm must return. Variables l_w and l_s represent the line width and separation, respectively.

```
Input: $S, l_w, l_s$
Output: $I$
  $p_{start} \leftarrow S_0$
  $ClusterStart \leftarrow TRUE$
  for $ix \leftarrow 0$ to getMaxIndex($S$) do
      $p_{first} \leftarrow S_{ix}$
      $p_{second} \leftarrow S_{ix+1}$
      if distance($p_{first}, p_{second}$) > $l_s$ and distance($p_{start}, p_{first}$) < $l_w$ then
          add $p_{first}$ to $CurrentCluster$
          add centroid($CurrentCluster$) to $I$
          clear $CurrentCluster$
      else if distance($p_{start}, p_{second}$) > $l_w$ then
          $ClusterStart \leftarrow FALSE$
          clear $CurrentCluster$
      else
          add $p_{first}$ to $CurrentCluster$
      end if
      if distance($p_{first}, p_{second}$) > $l_s$ then
          $ClusterStart \leftarrow TRUE$
          $p_{start} \leftarrow p_{second}$
      end if
  end for
```

of the cluster afterwards. This situation is shown in Figure 4.1 top. After that, the algorithm iterates through the following points (angles) checking if its distance with the previous point is further than the line separation or the line width. When the distance of the second point from the initial point exceeds the width of the line, that cluster cannot be related to an intersection, and the cluster is discarded as shown in Figure 4.1 bottom.

A possible intersection is assumed to start again if the separation of the second point from the first is greater than the separation of the lines as shown in Figure 4.2 top. After that the algorithm continues studying the following points. As already mentioned, the possible cluster would be discarded if the distance between the starting point and the second one is greater than the width of the lines. In contrast, if the distance between the first and the second point is greater than the distance between the lines and the distance between the starting and the first point is smaller than the line width, the cluster is assumed to be related to an intersection of a line and its centroid would be calculated as shown in Figure 4.2 bottom. After processing the complete scan, the centroids of all the detected clusters would be returned.

When the algorithm reaches the last point, the centroid of the last cluster must be included if and when its width is smaller than the width of the power-lines. In this way, the last cluster is also taken into account, but the fact that the following point to the last one is the initial one is not being considered. Therefore, an incorrect detection might be performed. Additionally, the algorithm only takes into account the previous and posterior points to the studied cluster so that there might be points at other angles which are closer than the distance between the power-lines but would not avoid the detection. As a consequence, although the algorithm is not valid for per-

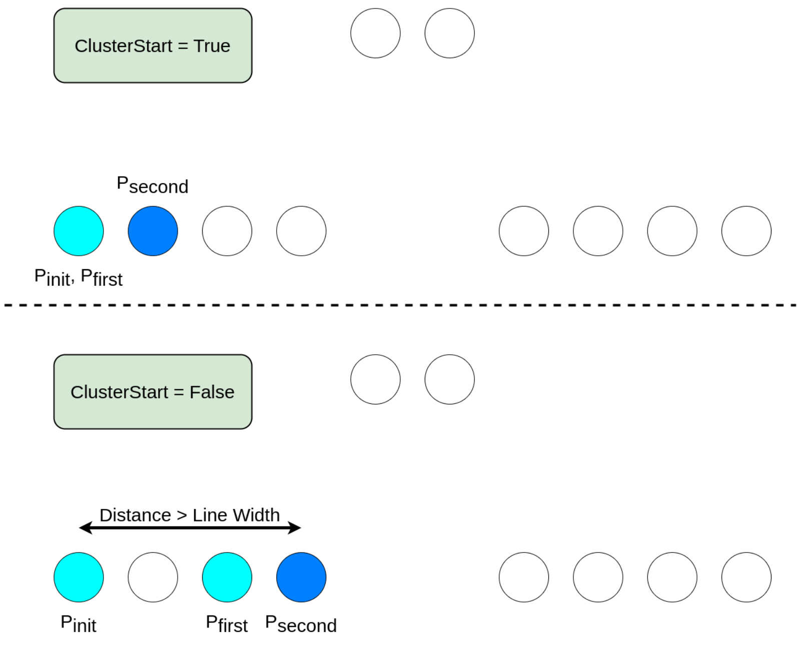


Figure 4.1 Example of rejection of a possible cluster of points related to the intersection of a line when its width is greater than the power-line width.

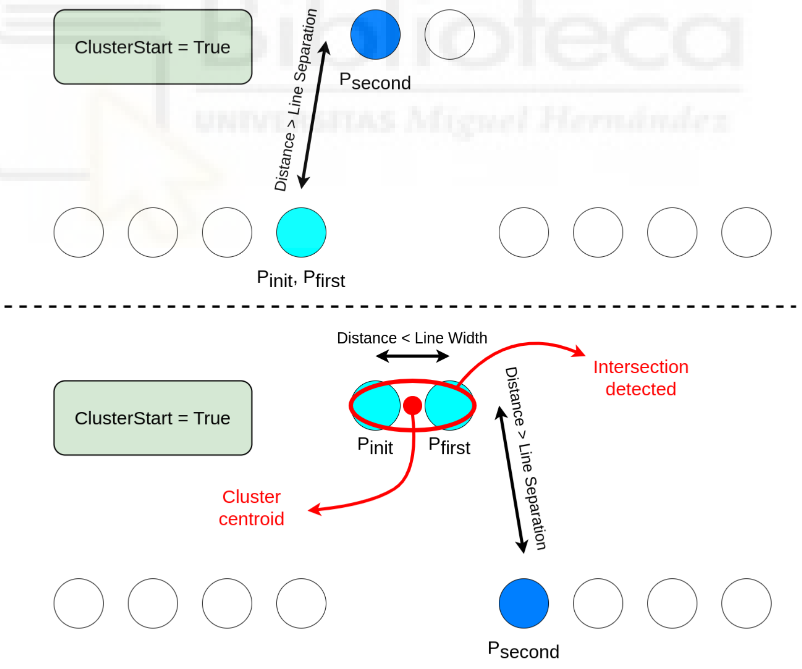


Figure 4.2 Example of detection of a possible cluster of points related to the intersection of a line when its width is smaller than the power-line width and the separation with the previous and posterior points is grater than the power-line separation.

forming the detection itself, it is a useful tool to drastically reduce the number of points to be studied.

To perform a valid and robust detection, after applying the previous algorithm, the module checks that each detected cluster meets the distance conditions with every point of the initial scan. Whereas performing the check for each point of the scan is not always a feasible operation for a real-time implementation, the output of the algorithm will consist of a small number of points whose check can be done in a short interval of time. This step allows discarding those clusters incorrectly detected due to the aforementioned problems.

Finally, the functioning of LiDAR sensors is not ideal, and there might be points related to random objects between a set of points related to the same power-line. If there is a point further than the power-line separation between points of the same power-line, the Algorithm 1 would return two possible intersections, one before the far point and another after it very close to each other. These points would meet the distance conditions with all the points of the scan. Thus, a last step is necessary to merge those clusters whose separation is smaller than the power-line width.

It is assumed that the resulting set of points would be associated with the intersections of the power-lines with the detection plane and, as mentioned, their positions must be filtered to allow their accurate estimation. The tool chosen to perform this filtering is the EKF algorithm described in [31], which is widely used in robotics due to its computational efficiency. This filter uses Gaussian representation for the estimation and the noises. Thus, it is assumed that range and angle measurements have only Gaussian noise and that the position of an intersection can be represented by a multivariable unimodal probability distribution, which are reasonable statements for this application. Furthermore, the EKF algorithm works properly in most situations even if these assumptions are not fully met.

Using the EKF algorithm requires the previous definition of the variables related to the state (x), which is represented by the mean (μ) and covariance (Σ) of a multivariable normal distribution. Then, the functions to obtain the state probability ($g(x,u)$) and the measurement probability ($h(x)$) must also be defined. Finally, the algorithm requires the definition of the jacobians G, and H in addition to the covariances related to the prediction and measurement noise, which are defined by the matrices R, and Q, respectively.

The matching to identify the different power-lines requires obtaining the position of each intersection. Thus, each of them is estimated with a different EKF. The state of each filter is defined as $x = [x, y]^T$ and represents the coordinates of the detected intersection referred to the LiDAR coordinate frame, which is assumed to be placed as shown in Figure 4.3. The points are assumed to be static as the filtering is only used during the matching stage, in which the UAV does not move. Therefore, the prediction stage of the filter operates on the assumption of static behavior ($g(x,0) = x$). Thus, the G matrix is equal to the identity matrix. The variance of the Gaussian noise related to the prediction is produced mainly by the movement of the UAV. However, this value is highly variable depending on different factors such as the wind, the type of aerial platform, or the control algorithm. Thus, this value is empirically selected to obtain a proper performance of the filter. It is assumed that the movement along both axes has a similar variance. Once the value of this variance ($\sigma_{prediction}$) is estimated, the matrix R can be calculated as $R = \sigma^2_{prediction} I$.

The measurement vector is defined as $z = [\rho, \theta]^T$. It is composed of the range (ρ) and the angle (θ) at which the intersection is detected. The EKF algorithm involves calculating the measurement associated to the prediction to then perform the update with the received measurement. Hence, the measurement probability $h(x)$ is needed. The function is just a conversion from Cartesian to polar coordinates as shown in Equation 4.1. Then, from this function, the jacobian $H = \frac{\partial h(x)}{\partial x}$ can be calculated as shown in Equation 4.2.

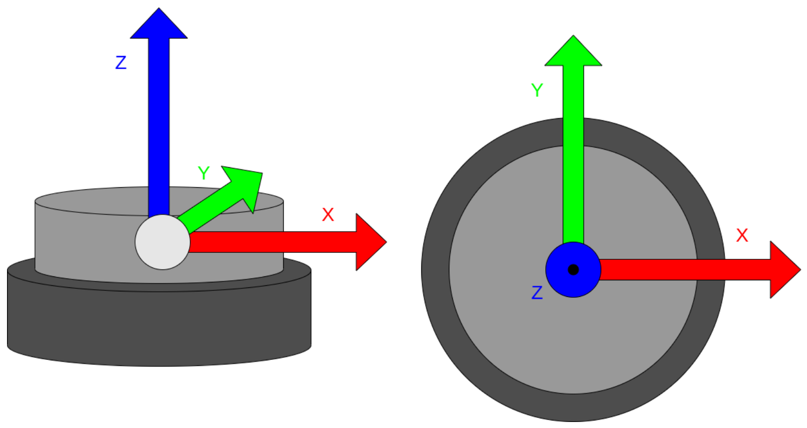


Figure 4.3 Visual representation of the position of the coordinate frame of the LiDARs. The LiDAR is represented as two cylinders. The smaller cylinder would contain the spinning laser which measures the distances. The center of the reference frame is therefore placed in the middle of this cylinder. The Z axis is perpendicular to the detection plane and its direction is defined by the spinning direction following the right hand rule. Lastly, the X axis is oriented in the direction at which the LiDAR measures an angle equal to 0. This direction depends on the LiDAR model and is usually indicated by an marker.

$$h(x) = [\sqrt{x^2+y^2}, atan2(y,x)]^T \tag{4.1}$$

$$H = \begin{bmatrix} \frac{x}{\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ \frac{-y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} \end{bmatrix} \tag{4.2}$$

Lastly, the variance of the noises related to the measurements is related mainly to the errors of the LiDAR. The values of the variances associated with the range (σ_ρ) and angle (σ_θ) noises can usually be consulted in the datasheet. Hence, the Q matrix is expressed as $Q = \begin{bmatrix} \sigma_\rho^2 & 0 \\ 0 & \sigma_\theta^2 \end{bmatrix}$.

The update of the filters requires associating the measured points with the filters for the different line intersections, which is performed by distance criteria. However, the finite resolution of LiDAR sensors often means that lines are not always detected. Furthermore, as mentioned previously, erroneous lines might be detected if there are random objects in the environment. As a result, there might be less or more detections than expected and some of them might not be related to the intersection of the lines of interest.

To tackle the problems described above, if there are more detected points than lines in the environment, the farthest exceeding points are discarded since the power-lines are assumed to be closer to the robot than any other object of the environment. Then, the remaining points update the filter with the closest estimation. At first, all the filters are initialized at the center of the LiDAR. As the points are detected, the filters are initialized. The algorithm to associate the detected points takes into account that each detected point must only update a filter if it is the closest point to the estimation of the filter and if that point is not closer to the estimation of another filter. In this way, when a filter is initialized, the other filters are never initialized in the same position, and there are no problems with the update if any of the intersections have not been detected. Finally, to deal with those possible far measurements related to other objects of the environment, the update is only performed if the detection is less than a distance threshold apart from the estimation. If not, it is assumed to be an outlier. When a series of outliers is received in a row, the filter is initialized again with the following detections as it could have been initialized with an erroneous point.

As outlined above, once the matching stage has finished and there is an initial estimation of the pose of the line, the tracking stage starts. This stage consists in providing measurements of the intersections to an EKF which is in charge of the filtering and, therfore, which must receive raw measurements. That is why, after a call to the stopFilter() function, the points detected with the algorithm are not filtered and are directly returned.

As already mentioned, if the estimation is inconsistent with the received measurements, the line would be assumed lost and the system would get back to the previous stages. Consequently, the points must be filtered again. For this reason, the module provides the reset() function, whose call returns the module to the initial state in which the points are filtered.

4.3 Power-line filter module

The *Power-line Filter* module has been implemented as another C++ class with the public member functions shown in Table 4.2. As well as those related to the prior module, these functions constitute its interface with the others. As previously mentioned, this module implements the EKF to estimate the pose of the power-line on the basis of the received measurements, that are the intersections detected with the *Intersection Detection* module. It is worth mentioning that a single measurement is assumed to be composed of the two intersection points of the power-line of interest, one with each detection plane. The problem of not detecting the intersection in one or any of the scans will be discussed in the next section, since it is the *Power-line Detection* module that is responsible for obtaining the measurements and calling the function. As for the implementation of the EKF described in the previous section, the algorithm used for the implementation is in [31]. The module provides a function to initialize the filter and functions to perform its prediction and update, which must be performed recursively after the initialization. The module also allows obtaining the intersection of its estimation with the detection planes since the *Power-line Detection* module requires this information to select the detected points more suitable to perform the update.

Table 4.2 Public member functions provided by the *Power-line Filter* module.

Name	**Description**
initialization(p0, p1)	Initializes the state of the filter with a line that joins the spatial points p0 and p1.
prediction()	Perform the prediction of the filter.
update(p0, p1)	Perform the update of the filter with the measured points p0 and p1.
getPoint0()	Returns the intersection of the estimated line with the detection plane of the LiDAR 0.
getPoint1()	Returns the intersection of the estimated line with the detection plane of the LiDAR 1.
getLine()	Returns the current estimation of the power-line pose.

This module has to provide an estimation of the relative pose of the power-line with respect to the UAV. The coordinate frame fixed to the power-line is placed as shown in Figure 4.4, with its x axis aligned with the power-line. The orientation of this x axis is the only orientation that the designed perception system can estimate since a rotation of the power-line around this x axis would not have any effect in the scans. As a result, only 2 of 3 Degrees Of Freedom (DOG) related to the orientation can be obtained. In a similar way, the perception system cannot estimate the position of the power-line along its direction, restricting the estimation of the position to an estimation with 2 of 3 DOF. Thus, what the filter will actually provide is the intersection of the middle plane between the LiDARs and the orientation of the x axis of the reference frame of the power-line. However, this does not limit the capabilities of the system since, as noted in Chapter 5, the missing DOFs are not required to feed back the control system.

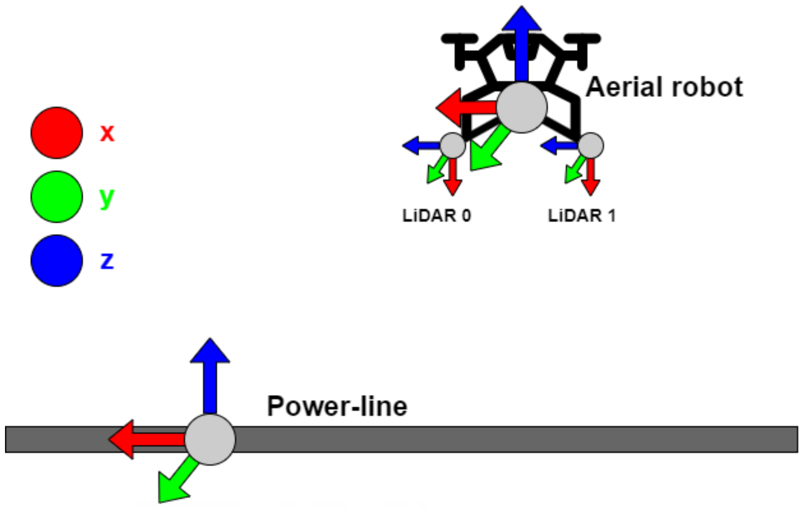


Figure 4.4 Representation of the coordinates frames related to the perception problem.

To use an EKF to estimate this relative pose, it is necessary to define the state vector of the filter. For the sake of simplicity, since the extrinsic calibration of the LiDARs with the aerial platform is known, the filter works referring everything to the coordinate frame related to the LiDAR 0. Theoretically, the variables included in the state vector must represent the system as completely as possible, but, however, using simpler representations is a common practice in robotics due to the reduction of the computational complexity of the implementation and the minor effect on the performance of the filter. Therefore, the filter is just composed of 4 variables for the aforementioned 4 DOF and 2 more for the linear velocity. The accelerations has not been included in the vector because, to increase the safety of the operation, the maneuver will be performed with low and constant velocities. Additionally, since the UAV is assumed to keep aligned with the power-line during the performance of the maneuver, the angular velocities have not been taken into account. The resulting state vector is defined as $x = \begin{bmatrix} x & y & v^x & v^y & \alpha & \beta \end{bmatrix}^T$, where the first two elements are the x and y coordinates of the intersection of the power-line with the middle plane between the LiDARs (the z coordinate is therefore constant), the next two represent the velocity of this point, and the last ones represent the orientation of the x axis.

The points used to define the filter equations are shown in Figure 4.5. The intersection of the power-line with the middle plane between the LiDARs is defined as p, and the intersections with the detection planes of the LiDARs 0 and 1 are defined as p_0 and p_1, respectively. In the following, all the points have 3 spatial coordinates and are referred to the LiDAR 0.

The output of the module, which is accessible by a call to the getLine() function, is the initial position and the orientation of the $\overrightarrow{pp_0}$ vector, which are calculable from the state vector. The first two coordinates of the initial position are explicitly included in the state vector, whereas the third one is constant and equal to half the separation of the LiDARs. Regarding the orientation, it is fully represented by the angles α and β in the state vector. These angles are related to the variables Δx,

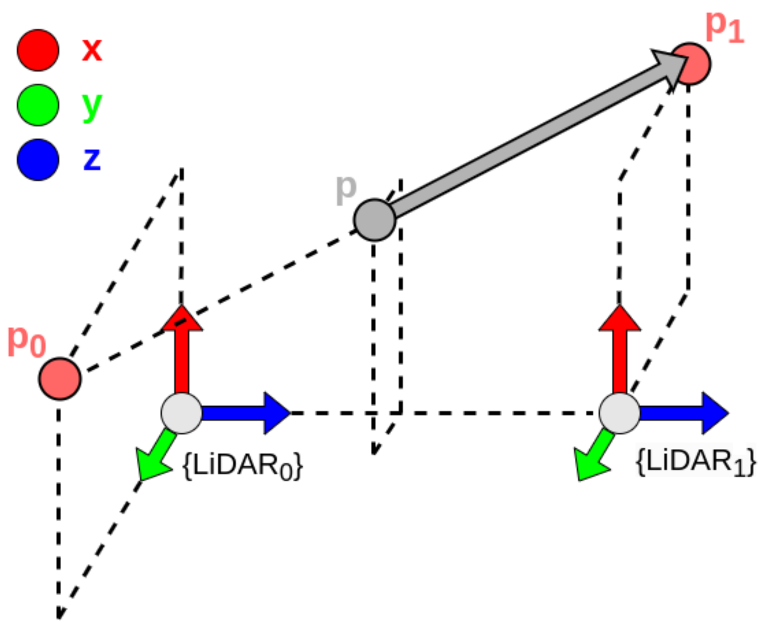


Figure 4.5 Points to define the equations of the filter.

Δy, and Δz, which are the coordinates of the vector $\overrightarrow{pp_1} = \begin{bmatrix}\Delta x & \Delta y & \Delta z\end{bmatrix}^T$. In this way, the angles α and β can be calculated as $\alpha = atan(\frac{\Delta y}{\Delta z})$ and $\beta = atan(\frac{\Delta x}{\sqrt{\Delta z^2 + \Delta y^2}})$, respectively. A visual representation of the angles is shown in Figure 4.6.

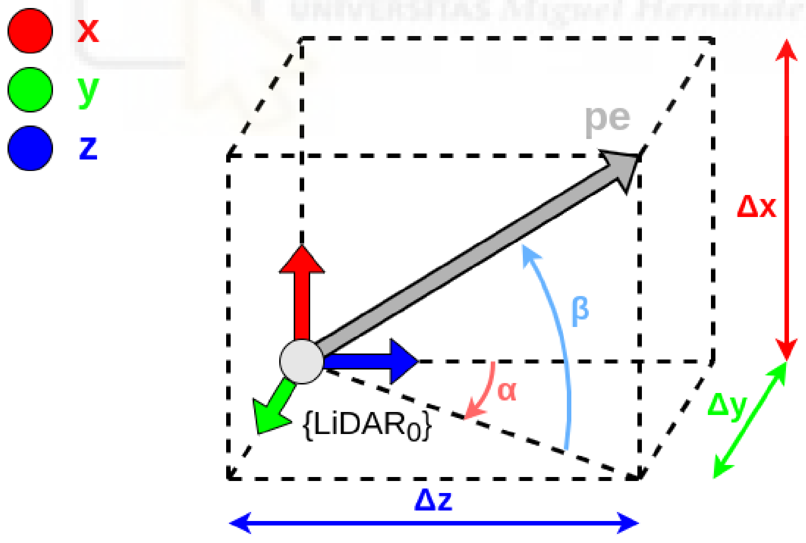


Figure 4.6 Angles defined to represent the orientation of the power-line (α and β).

As for the EKF used in the previous section, the functions for the prediction ($g(x,u)$) and the update ($h(x)$), and the matrices G, H, R, and Q must be defined to use the filter. The prediction is made assuming a constant velocity model for the position and a static model for the orientation as shown in Equation 4.3. Once this function is defined, the jacobian G can be calculated as shown in Equation 4.4.

$$g(x_t,u_t) = \begin{bmatrix} x_t \\ y_t \\ v_{xt} \\ v_{yt} \\ \alpha_t \\ \beta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + v_{xt-1}T \\ y_{t-1} + v_{yt-1}T \\ v_{xt-1} \\ v_{yt-1} \\ \alpha_{t-1} \\ \beta_{t-1} \end{bmatrix} \tag{4.3}$$

$$G = \frac{\partial g(x)}{\partial x} = \begin{bmatrix} 1 & 0 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & T & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

where T is the elapsed time between the iterations of the filter.

The measurement vector must be defined before the definition of the function $h(x)$ and the matrix H. It is composed of the ranges and angles of the detected points with each LiDAR. Additionally, the range and angle of the previous measurement are also included to introduce temporal information and allow estimation of the velocity. In this way, the measurement vector is defined as $z = \begin{bmatrix} \rho_0 & \theta_0 & \rho_{b0} & \theta_{b0} & \rho_1 & \theta_1 & \rho_{b1} & \theta_{b1} \end{bmatrix}^T$, where ρ and θ represent ranges and angles, respectively. The variables with the subscripts 0 and 1 are related to LiDAR 0 and 1, respectively. And lastly, the subscript b indicates that the measurement is related to the detected intersection before the current one.

The aforementioned points p, p_0, and p_1 are defined as shown from Equation 4.5 to Equation 4.7 and are associated with the variables in the measurement vector as shown from Equation 4.8 to Equation 4.11. Additionally, they can be obtained from the state vector as detailed below, which allows calculation of the measurement model $h(x)$.

$$p = \begin{bmatrix} x & y & z \end{bmatrix}^T \tag{4.5}$$

$$p_0 = \begin{bmatrix} x_0 & y_0 & z_0 \end{bmatrix}^T \tag{4.6}$$

$$p_1 = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}^T \tag{4.7}$$

$$x_0 = \rho_0 cos(\theta_0) \tag{4.8}$$

$$y_0 = \rho_1 sin(\theta_1) \tag{4.9}$$

$$x_1 = \rho_0 cos(\theta_0) \tag{4.10}$$

$$y_1 = \rho_1 sin(\theta_1) \tag{4.11}$$

To obtain the function $h(x)$ from the state vector, as shown from Equation 4.12 to Equation 4.15, the values of the variables Δx and Δy are calculated from the angles α and β, and the variable Δz, which is equal to half the distance between the LiDARs. Then, as shown in Equation 4.16 and Equation 4.17, the points p_0 and p_1 can be easily calculated from the point p. Lastly, with the conversion from Cartesian to polar coordinates shown in Equation 4.18 to Equation 4.21 and the assumption that the Equation 4.22 and Equation 4.23 are satisfied, the function $h(x)$ is calculated as shown in Equation 4.24.

$$tan(\alpha) = \frac{\Delta y}{\Delta z} \tag{4.12}$$

$$\Delta y = tan(\alpha)\Delta z \tag{4.13}$$

$$tan(\beta) = \frac{\Delta x}{\sqrt{\Delta z^2 + \Delta y^2}} \tag{4.14}$$

$$\Delta x = tan(\beta)\sqrt{\Delta z^2 + \Delta y^2} \tag{4.15}$$

$$p + \overrightarrow{pp_1} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = p_1 \tag{4.16}$$

$$p - \overrightarrow{pp_1} = p_0 \tag{4.17}$$

$$\rho_0 = \sqrt{x_0^2 + y_0^2} \tag{4.18}$$

$$\theta_0 = atan2(y_0, x_0) \tag{4.19}$$

$$\rho_1 = \sqrt{x_1^2 + y_1^2} \tag{4.20}$$

$$\theta_1 = atan2(y_1, x_1) \tag{4.21}$$

$$x_{t-1} = x_t - v_t^x T \tag{4.22}$$

$$y_{t-1} = y_t - v_t^y T \tag{4.23}$$

$$h(x_t) = \begin{bmatrix} \sqrt{(x_t - \Delta x_t)^2 + (y_t - \Delta y_t)^2} \\ atan2(y_t - \Delta y_t, x_t - \Delta x_t) \\ \sqrt{(x_t - Tv_t^x - \Delta x_t)^2 + (y_t - Tv_t^y - \Delta y_t)^2} \\ atan2(y_t - Tv_t^y - \Delta y_t, x_t - Tv_t^x - \Delta x_t) \\ \sqrt{(x_t + \Delta x_t)^2 + (y_t + \Delta y_t)^2} \\ atan2(y_t + \Delta y_t, x_t + \Delta x_t) \\ \sqrt{(x_t - Tv_t^x + \Delta x_t)^2 + (y_t - Tv_t^y + \Delta y_t)^2} \\ atan2(y_t - Tv_t^y + \Delta y_t, x_t - Tv_t^x + \Delta x_t) \end{bmatrix} \tag{4.24}$$

Once the function $h(x)$ is obtained, the jacobian H can be calculated by deriving the function with respect to the variables in the state vector x. However, these derivatives are very complex to be calculated by hand. That is why they have been calculated with the software *wxMaxima*. Due to the long length of the resulting equations, the expressions to obtain the different elements of the matrix are attached in the Appendix A.

Lastly, the matrices R and Q related to the covariance of the noise of the prediction model and the measurements must also be defined. In the same way as for the EKF implemented for the *Intersection Detection* module, the covariances of the noises related to the prediction model are manually selected to achieve a good performance of the filter. The resulting R matrix is shown in the Equation 4.25, in which the σ_p, σ_v, and σ_a are the variances of the noise related to the position, the velocity, and the angles, respectively. In contrast, similarly as for the EKF in the previous section, the Q matrix can be calculated with the specifications of the sensors which usually provide the variances of the noises related to measurements of range (ρ) and angles (θ). The resulting Q matrix is shown in Equation 4.26.

$$R = \begin{bmatrix} \sigma_p^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_p^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_v^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_a^2 \end{bmatrix} \tag{4.25}$$

$$Q = \begin{bmatrix} \sigma_\rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_\theta^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_\rho^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_\theta^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_\rho^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_\theta^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\rho^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_\theta^2 \end{bmatrix} \tag{4.26}$$

4.4 Power-line detection module

The module presented in this section is the core of the perception system. It is responsible for receiving the LiDAR scans, making use of the other modules to obtain an accurate estimation of the power-line pose, and feedbacking the control system with it. The communications of this module with the control system and with the LiDARs are based on ROS topics. Thus, it is subscribed to the topics in which each LiDAR driver publishes the scans and publishes the resulting estimation in a topic to which the control system would be subscribed.

First, the module creates a class object of the *Power-line Filter* module and two of the *Intersection Detection* module, one to process the scans of each LiDAR. The scans are processed with separated class objects to filter the detections related to each LiDAR separately during the matching stage. Once the functionalities of the other modules are available, the module starts the execution of the state machine shown in Figure 4.7.

The first two states of the module are a wait until both LiDAR drivers are initialized and provide measurements. The following is the *No Points Detected* state, which will leave once the *Intersection Detection* class object for processing the scans of the LiDAR 0 returns *true* to a call to its isInitialized() member function. As mentioned, this function returns whether or not the module has detected all the intersections of the lines, and, therefore, there is an available set of points to match. Then it would enter the *Points 0 Initialized* and wait until the *Intersection Detection* class object for the LiDAR 1 has also detected all the intersections. Then, the module would go to the *Points 1 Initialized* state and the matching operation would start. After performing the matching, the filter of the *Power-line Filter* module is initialized with a call to its initialization(p0, p1) member function with the matched points as arguments. At this moment, the pose estimation of the closest line would start, which requires switching between the states *Wait Detection 0* and *Wait Detection 1*. The module leaves the first one after receiving a scan from the LiDAR 0 and obtaining a set of possible intersections with the *Intersection Detection* module. Then it would pass to the *Wait Detection 1* until a scan from the LiDAR 1 is received. Once obtained the set of possible intersection points in the scans of both LiDARs, the module would take the closest ones to the intersection of the estimated line with the detection planes and would perform the prediction and update of the filter with a call to the pertinent member functions of the *Power-line Filter* module. After that, it would return to the *Wait Detection 0* and would continue switching between them if and when the estimation

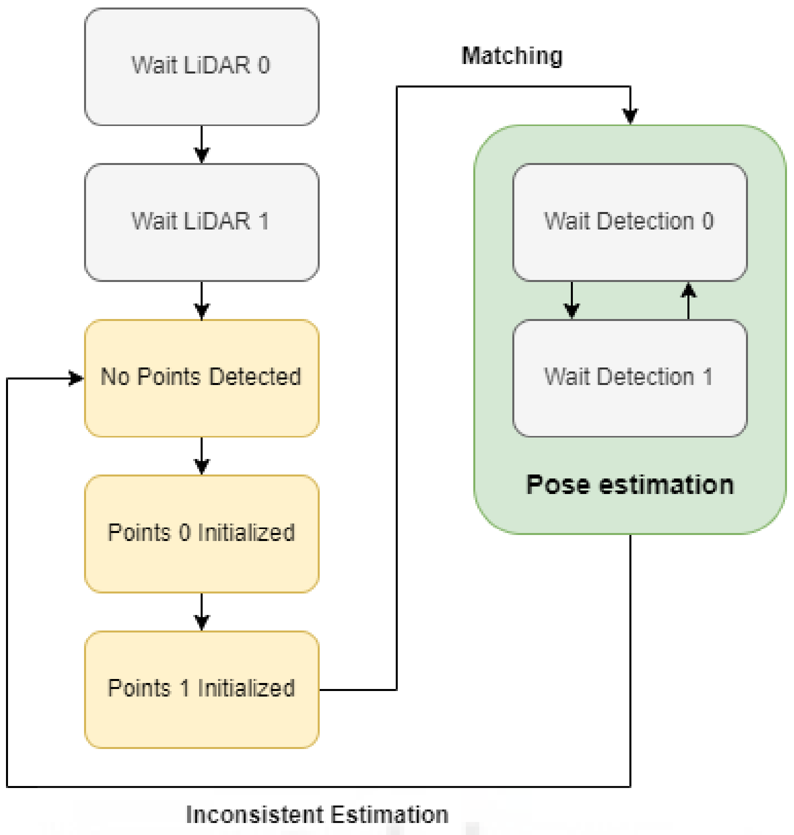


Figure 4.7 State machine executed by the *Power-line Detection* module.

is consistent with the measurements and the module does not enter the *No Points Detected* state again.

As noted earlier, before starting the estimation of the power-line pose, the points associated with the same power-lines must be identified to initialize the *Power-line Filter* module. This matching operation is performed during the *Points 1 Initialized* state, in which the stopFilter() function of the *Intersection Detection* module would not have been called yet, and consequently, the returned sets of points are filtered to provide a more accurate estimation of their position. To match them, the module first takes the closest intersection detected with the LiDAR 0 (point 0) and associates it with one of the intersections detected with the LiDAR 1. Then, it calculates the intersections with the detection plane of the LiDAR 1 of the lines that start at each point detected with the LiDAR 0 and that have the same direction of the line that joins the point 0 and the associated point. After that, the sum of the distances from each calculated intersection to the closest detected intersection in the LiDAR 1 is calculated. As shown in Figure 4.8 left, when the point 0 is associated with the intersection of its line, the sum of these distances would ideally be 0. However, as shown in Figure 4.8 right, when the association is performed with the intersection of other power-line, the sum of distances would be a higher value in the order of the distance between the lines. Therefore, the module calculates the sum of distances for each point detected with the LiDAR 1 and uses them as a quality metric for the points to be matched. The module matches the point 0 with that point detected with the LiDAR 1 which minimizes this sum of distances.

Once the points are matched, the module calls the method stopFilter() for each class object of the *Intersection Detection* module, initializes the filter of the *Power-line Filter* module with the matched points, and goes to the *Wait Detection 0* state to start the estimation of the power-line pose. After receiving scans from both LiDARs, the module has to provide the filter with a measurement

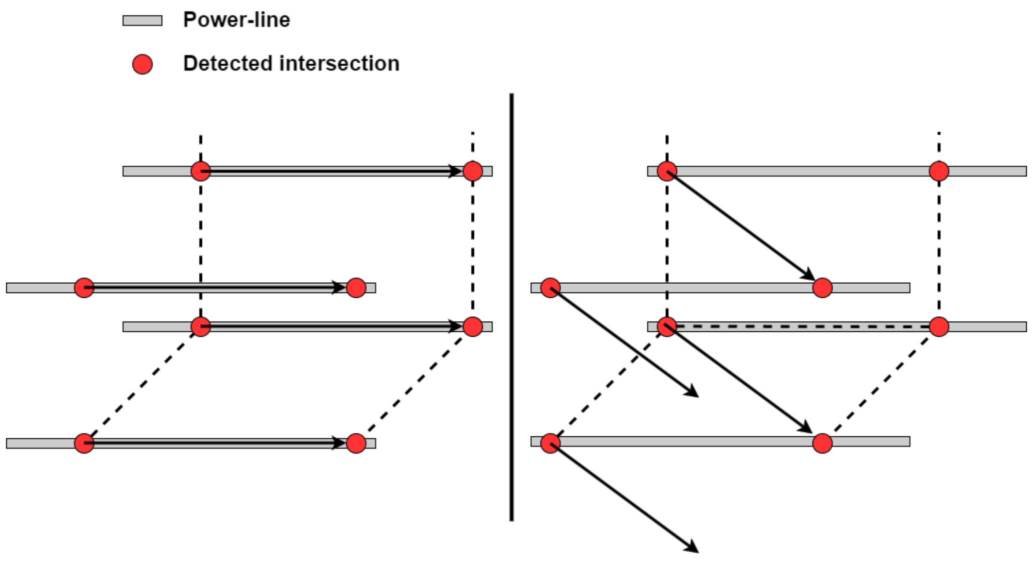


Figure 4.8 Visual representation of the matching operation. A correct matching is shown at the left example whereas an erroneous matching is shown at the right.

to update it. This measurement must consist of 2 detected intersections, one with each LiDAR. However, as there are usually several power-lines, the number of intersections detected with each LiDAR will rarely be 1, and, consequently, the module takes the intersections that are closer to the intersections of the estimated line with the detection planes of the LiDARs, which are provided by the *Power-line Filter* module. The module uses a distance threshold in the distance between these points to detect inconsistent measurements. When the distance is greater than the threshold, the detection is not taken into account, and the update is performed with the previous detection assuming a static model. If a certain number of inconsistent detections is received in a row, the module gets back to the *No Points Detected*, resetting the class objects of the *Intersection Detection* module to detect all the intersections again. In contrast, if the line is properly estimated, the module predicts and updates the filter with the pertinent functions of the *Power-line Filter* class object. This last operations are repeated over the time, switching between the states *Wait Detection 0* and *Wait Detection 1* to ensure that the filter is updated with a detection of each LiDAR and publishing the estimated line every time the filter is updated.

4.5 Conclusion

This chapter has described the implementation of the perception system, addressing the functioning of each module independently. Each of them has been carefully designed to achieve the required performance in terms of robustness, accuracy, and efficiency. These properties are essential for software executed on board aerial robots because of the danger they imply and their limited payload, which restricts the power of the onboard computers.

As proved in Chapter 6, the resulting software allows its real-time execution on board an aerial platform and is accurate enough to properly feedback the control system described in the following chapter.

5 Control system

5.1 Introduction

Performing the perching maneuver on a power-line requires both being able to detect and estimate its pose and moving the platform to achieve certain relative poses with the line to allow the actuation of the perching mechanism. The previous chapter has presented a perception system that is able to obtain an estimation of this relative pose. The following describes how the robot uses these measurements to set its velocity and reach the required positions to perform the maneuver. In the same way as the perception system, all the software has been developed in C++ using ROS.

The structure of this chapter is as follows. First, a brief explanation is given of how quadrotors can perform autonomous flies. Then, the tool used to simplify the implementation of the control system is presented. Finally, the algorithm developed to process the estimations of the power-line pose and close the loop is described.

5.2 Autopilot

The control of a quadrotor to navigate autonomously is usually based on a cascade controller with two loops, one related to its position and the other to the velocity of the motors [32]. To feedback the control with the position and orientation of the platform, they must first be estimated by means of the measurements from the onboard sensors, which are usually an Inertial Measurement Unit (IMU) and a Global Positioning System (GPS). Over the last few years, a multitude of commercial devices have emerged to perform this control [33]. They are known as *Autopilots* and include all the hardware and software required to estimate the pose of the UAV, calculate the proper control signals, and provide the Pulse Width Modulation (PWM) references to the motors. These off-the-shelf devices lead to results that are hard to beat. Moreover, they include all the safety features required due to the danger associated with aerial platforms. Hence, the control of most current UAVs is based on these devices.

The low-level control of the platform to perform the perching maneuver is assumed to rely on an *Autopilot*. Therefore, it is assumed that the UAV has a reliable localization system, that it is properly controlled by the *Autopilot*, and that it can receive and handle velocity commands.

5.3 UAL

The *UAV Abstraction Layer* (UAL) [34] is a software that allows the easy development of high-level algorithms for aerial platforms regardless of the model or the version of the *Autopilot*. It is able to connect with the *Autopilot* and provide common functionalities related to aerial platforms which can be accessed by calls to the public member functions of an object of the UAL C++ class. For instance, it allows taking-off and landing the platform, going to waypoints, or controlling its linear velocity and yaw rate, among others. Additionally, the UAL also allows the easy simulation of quadrotors, which will be an interesting tool to test the system before its test with a real platform.

The algorithm developed to control the UAV receives the estimation about the pose of the power-line and calculates the required velocities to achieve certain poses. Since these velocities can be easily commanded to the platform via the UAL, the velocity control of the UAV is assumed to be solved and what follows will focus on how to calculate its references.

5.4 Velocity control

In most cases perching on a power-line consists in reaching a certain relative pose with respect to the line to allow the actuation of a mechanism which grabs the line. Consequently, it is this mechanism that establishes the final relative pose. Both the final pose and the configuration of the LiDARs in the platform can be any, as long as the detection planes cut the power-line during the maneuver. Nevertheless, despite not being required, a set of hypotheses about them are used to simplify the description of the control approach in this work.

First, the following will focus on the control of an UAV to perform perching with a mechanism similar to that in [1]. Since multicopters always fly horizontally, the mechanism requires that the power-lines are also horizontal or almost horizontal. The UAV must start the maneuver under the power-line, align the mechanism with the direction of the power-lines while maintainign its center under one of them, and then go up until the power-line enters the grasper. Regarding the configuration of the LiDARs, it is assumed that their z axes are aligned with the orientation of the mechanism and that their x axes are vertical and point upwards. As mentioned, these assumptions are not really necessary, and the control approach might be extended to many other applications with minor changes. However, they are really useful to simplify the description of the developed algorithm.

For the sake of simplicity, all the variables are referred to the LiDAR 0, the same coordinate frame to which the output of the perception algorithm is referred. The variables that must be controlled are shown in Figure 5.1. First, since the z axis of the LiDAR is aligned with the orientation of the mechanism and the y axis is horizontal, the orientation of the projection of the detected line in the yz plane (α) must be similar to that of the z axis. Then, the y coordinate of the power-line (Δy) must be controlled to align the aperture of the mechanism with the power-line. Lastly, the relative height of the power-line (Δx) must also be controlled to enter the power-line in the mechanism.

The UAV is assumed to be already close to the power-line and approximately aligned with it before starting the maneuver, a position that might be reached by following a georeferenced path or by the manual control of the platform. Once the robot is there, and the perception system starts the pose estimation of the power-line, a velocity control of the robot would start to reach the desired positions. The α angle can be controlled by changing the yaw rate of the platform, which would have a direct effect on the variable if the prior assumptions are met. Similarly, Δx can be controlled

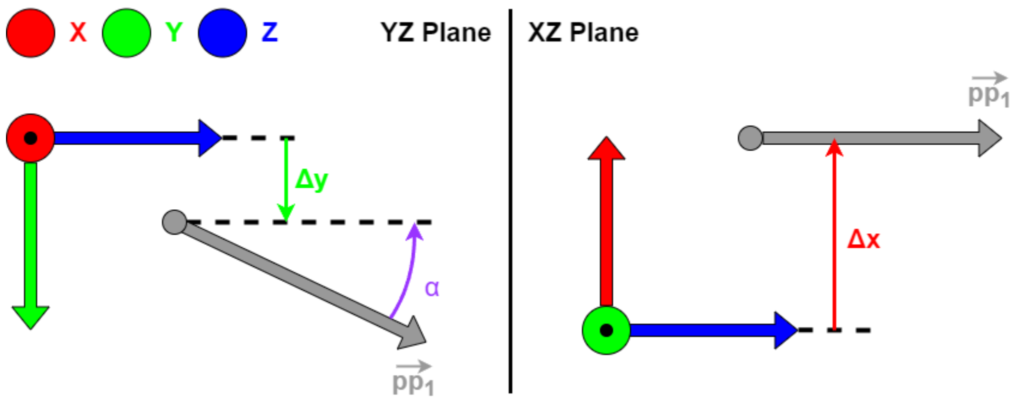


Figure 5.1 Variables related to the velocity control.

by setting the velocity of the UAV on the x axis of the LiDAR. In contrast, a movement along the y axis of the LiDAR is only fully reflected in Δy if the z axis is completely aligned with the power-line ($\alpha = 0$). But taking into account that the control on the α angle would force this condition, the velocity component in the direction of the power-line can be neglected.

Therefore, the variables to be controlled are α, Δy, and Δx, which are visually represented in Figure 5.1. These variables are controlled by setting the yaw rate and the linear velocity on the y and x axes of the LiDAR 0, respectively. Each variable is controlled with a different control signal, and it has been assumed that they are independent so that acting on a single control signal only causes effect in its associated control variable.

To track references and reject perturbations, simple proportional controllers have been used. In addition, all the applied control signals are saturated to improve the stability of the systems. The saturations ensure that the commanded velocities are low, which avoids the failure of the controller and the safety of the operation. As a consequence, the systems will evolve slowly, and a derivative part in the controller is not necessary. Furthermore, as the systems to be controlled are of type 0, applying a constant control signal produces a constant change of the output. Thus, an integral controller is also not required to achieve small steady-state errors.

Lastly, the references for the final pose can not be given to the controllers directly because it would probably result in aggressive movements that might worsen the estimation of the power-line pose and of the control. Thus, the state machine shown in Figure 5.2 is used to change the target pose sequentially. Initially, the UAV would maintain its Δx distance while setting the α angle and the Δy distance to 0 to align the mechanism with the power-line. The change of state is produced when the error with the reference of all the variables is lower than a threshold during a certain elapsed time. Once the perching mechanism is aligned with the line, the UAV would set the reference on the x axis so that the power-line enters the mechanism. Lastly, once all variables are within the threshold during the defined elapsed time, the mechanism would act.

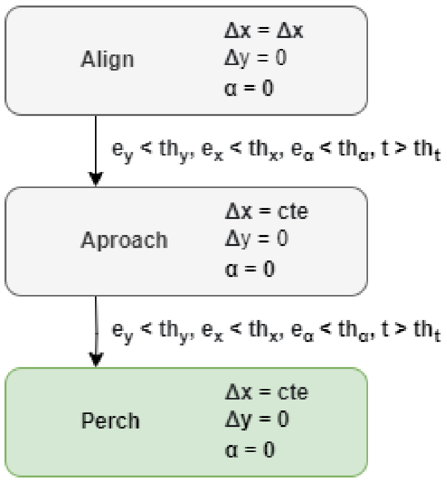

Align
Δx = Δx
Δy = 0
α = 0
e_y < th_y, e_x < th_x, e_α < th_α, t > th_t
Aproach
Δx = cte
Δy = 0
α = 0
e_y < th_y, e_x < th_x, e_α < th_α, t > th_t
Perch
Δx = cte
Δy = 0
α = 0

Figure 5.2 State machine that establishes the reference of the controllers. e_x, e_y, and e_α are the errors with the reference of the variables Δx, Δy, and α, respectively. th_x, th_y, and th_α are their thresholds to assume that a variable has successfully reached its reference. Lastly, t is the elapsed time since the value of all the errors is below the thresholds, and th_t the threshold on this time to switch to the following state..

5.5 Conclusion

This chapter has addressed how the estimation of the power-line pose obtained with the prior perception system can be used to allow the actuation of a perching mechanism onboard an aerial robot. Moreover, it could be used to perform the inverse operation and takeoff the platform from the power-line by just executing the state machine in Figure 5.2 inversely.

The following chapter will address the experimental results that validate the correct functioning of both the perception system and the control system. Although their design required the assumption of multiple hypotheses, it will be proved that they are well-founded and that the system is still able to perform the autonomous perching maneuver safely and robustly.

6 Experimental results

6.1 Introduction

This chapter addresses a quantitative evaluation of the perception and control systems developed in this work. The evaluation consisted of different experiments designed to test the different parts of the systems in isolation and flight experiments with a real quadrotor in which both the control and the perception system collaborated to achieve given relative poses with a power-line. The experiments provide measurements of the errors committed by the perception system to estimate the pose of the power-line and by the control system to achieve a given reference. These errors indicate how valid the systems are and whether or not they can be used to perform perching with a multirotor.

The organization of this chapter is as follows. First, the perception system is tested with the pose estimation of a moving power-line from a static pose. Then, the correct functioning of the control system is tested with a simulation. Finally, the results of the flight experiments are described.

6.2 Perception system performance

The correct functioning of the perception system is essential to properly feedback the control system and execute the desired maneuver. Therefore, the performance of the perception system was first evaluated in isolation. First, the robustness of the LiDARs under hard illumination conditions was checked by comparing scans with metallic objects obtained indoors, indoors in dark conditions, and outdoors with direct sunlight on the LiDAR. Since no differences in the performance were noted, the sensor is assumed to be suitable for the proposed application. Then, the accuracy of the estimations was measured. The LiDARs were placed in a fixed pose and the metallic stick shown in Figure 6.1 of 1 cm diameter was moved in front of them. Although the system is designed to detect the pose of static power-lines from a moving aerial robot, the line was moved instead to facilitate the execution of more abrupt relative movements that could compromise the performance of the system. The estimations of the perception systems were compared with a ground truth generated with a motion capture system, which is a set of infrared cameras that detect the position of small markers attached to rigid bodies to then calculate a remarkably accurate estimation of their pose. Specifically, a set of 24 *OptiTrack Primex 13* cameras was used to obtain millimeter accuracy robot pose estimations. The ground truth was obtained by recording the pose of the LiDARs and of the line to be detected. After that, the data were processed to calculate the position and orientation theoretically estimated by the perception system at all times.

The stick was moved toward and away from the sensors while making random oscillations and changes of orientation. In this way, the relative movement was similar to performing consecutive

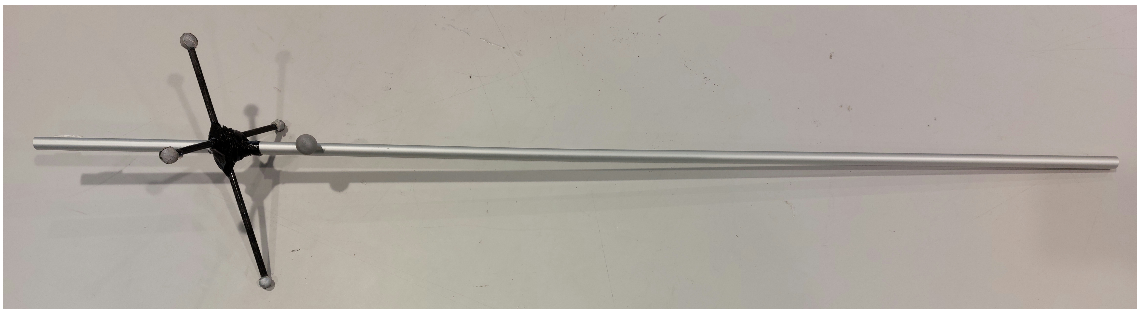


Figure 6.1 Stick with markers for the motion capture system to simulate the relative movement between the LiDARs and a power-line.

perching maneuvers with severe perturbations that might occur during the execution of the maneuver. The results obtained in one of the experiments are shown in Figure 6.2. All variables are correctly estimated with a small delay related to filtering. However, this delay is only notable when the movements are fast, which will not occur during the execution of the maneuver. From second 35 to 40 the error increases momentarily due to an abrupt movement in the x axis when the distance from the LiDAR to the stick is relatively large (note the y axis coordinate). The error is probably higher because the functioning of the LiDARs is not ideal when the power-line is close to the maximum detection distance. Thus, the line might not appear in certain scans, resulting in a worse performance of the detection system. Nevertheless, the error is not very significant and is quickly reduced once the movement stops. Moreover, the grab of the power-line is supposed to be performed at close distances, and, therefore, a momentary increase of the error when the UAV is far from the power-line would not produce a failure in the execution of the maneuver.

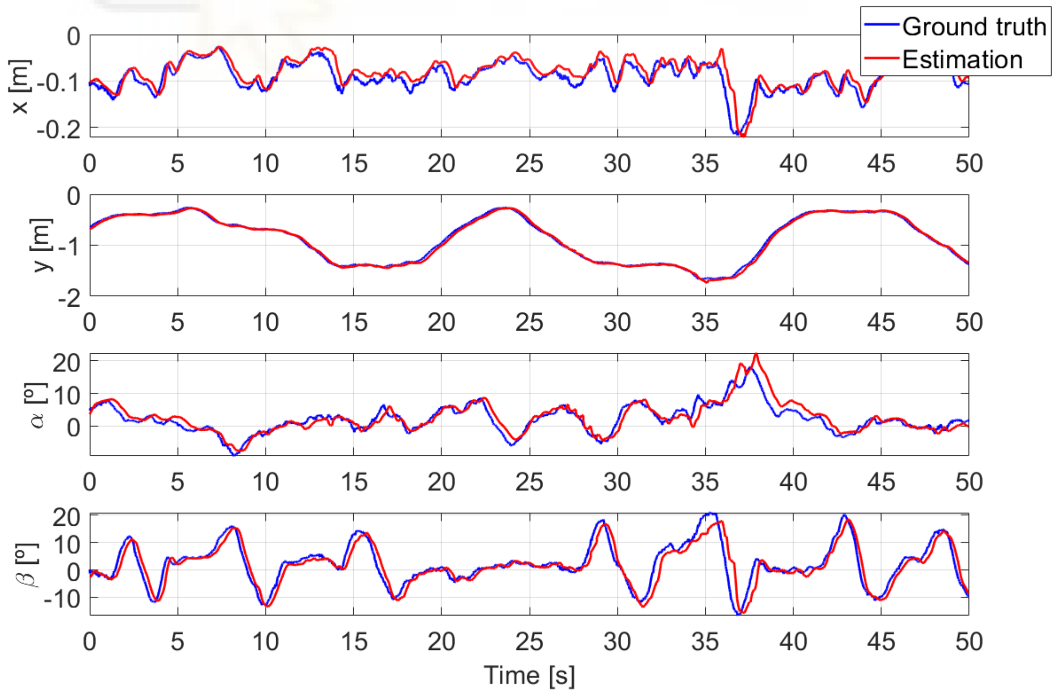


Figure 6.2 Line pose estimated by the perception system and ground truth obtained with a motion capture system.

The absolute error of each estimated variable with the ground truth during the experiment is represented in Figure 6.3. The mean and standard deviation of the absolute error for each variable are shown in Table 6.1. Neither the mean nor the standard deviation of the absolute error related to the variable *x* are greater than twice the width of the simulated power-line (1 cm). The mean and standard deviation of the error related to the variable *y* are slightly higher than those of the *x* error. However, this error is only considerably higher from second 18 to 22, and from second 35 to 40, when power-line is far from the power-lines (see *y* graph in Figure 6.2). As mentioned, being far from the power-line might result in a worse performance of the perception system but is not a problem since the critical part of the maneuver is performed at a close distance from the power-line. Nevertheless, they are always smaller than 10 cm. Regarding orientation, the mean and standard deviation of the absolute error of both variables are less than 3 degrees. The errors also reach higher values when the power-line moves away, which is neither a problem since the errors are moderately low (10°approximately) and the entrance of the line in the mechanism is performed when the distance is short.

The experiments prove the ability of the perception system to estimate the pose of a power-line with considerably low errors. Together with the errors of the control system to track references, the errors of the perception system are important to determine the characteristics of the perching mechanisms that can be used with the proposed approach. Although the observed performance would probably allow the use of a large number of perching mechanisms, the perception system has been evaluated under challenging conditions that were not taken into account during its development. The experiments involved quick random movements that would not occur during the use of the system and that severely compromise the performance of the filter, which assume constant linear velocities and a static orientation. Thus, these results should not be used to draw conclusions about the suitability of the approach for a given application, but about its robustness and proper functioning. In Section 6.4, the accuracy of the perception system is again evaluated with the concurrent performance of the control system, providing a fairer measure of the accuracy the system can achieve.

Table 6.1 Mean Absolute Error (MAE) and STandard Deviation of the Absolute Error (STDAE) of the perception system to detect a moving line from a static pose.

Variable	MAE	STDAE	Unit
x	1.36	1.22	cm
y	2.47	1.98	cm
α	1.43	1.34	°
β	2.36	2.31	°

6.3 Simulated experiments

The control system has been initially tested in a simulation, to ensure its proper functioning before its use on board the real platform and to obtain an initial estimation of the control parameters. To use the robot models provided by the UAL (which is described in Chapter 5), the framework selected for the simulation is *Gazebo*. As shown in Figure 6.4 the robot was provided with 2 simulated LiDARs whose properties are similar to those of RPLIDAR A3, which is the sensor used in the flight experiment described in the next section. Besides, a set of 4 power-lines was placed in the world model to approximate one of them autonomously.

Due to the availability of a set of power-lines, the matching algorithm have been tested with the simulation. First, the UAV received waypoints above the power-lines with different yaw references.

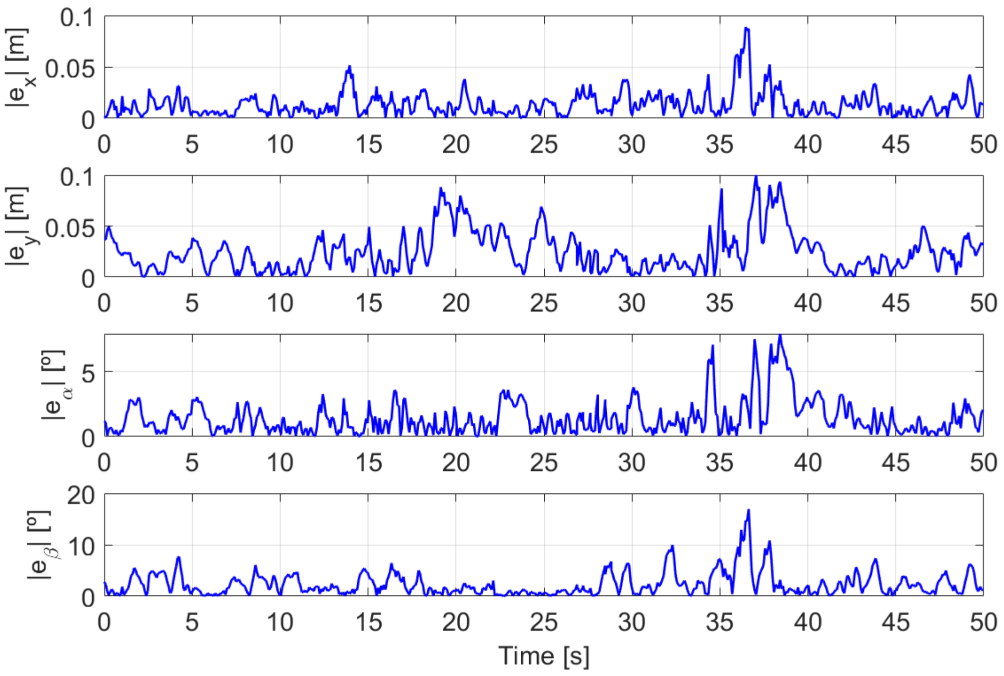


Figure 6.3 Absolute errors of the perception system to detect a moving line from a static pose.

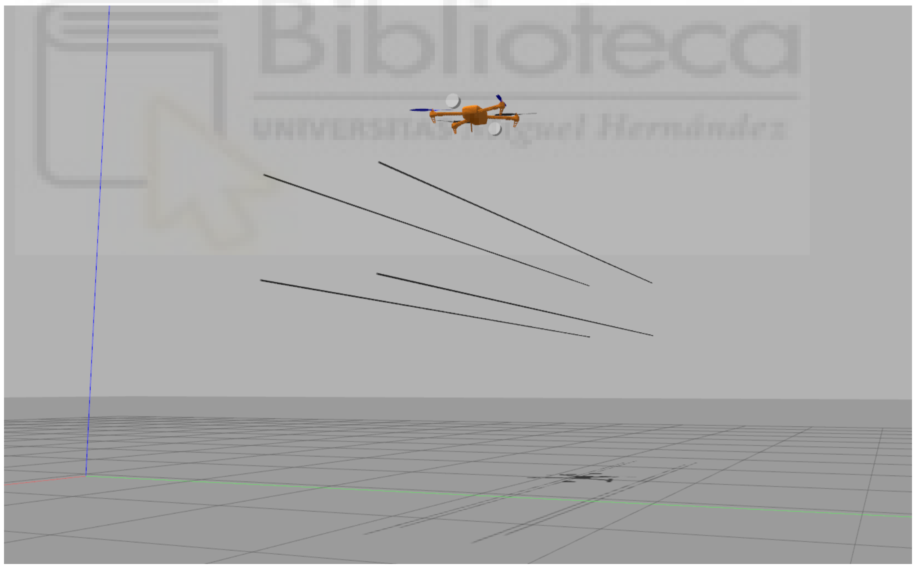
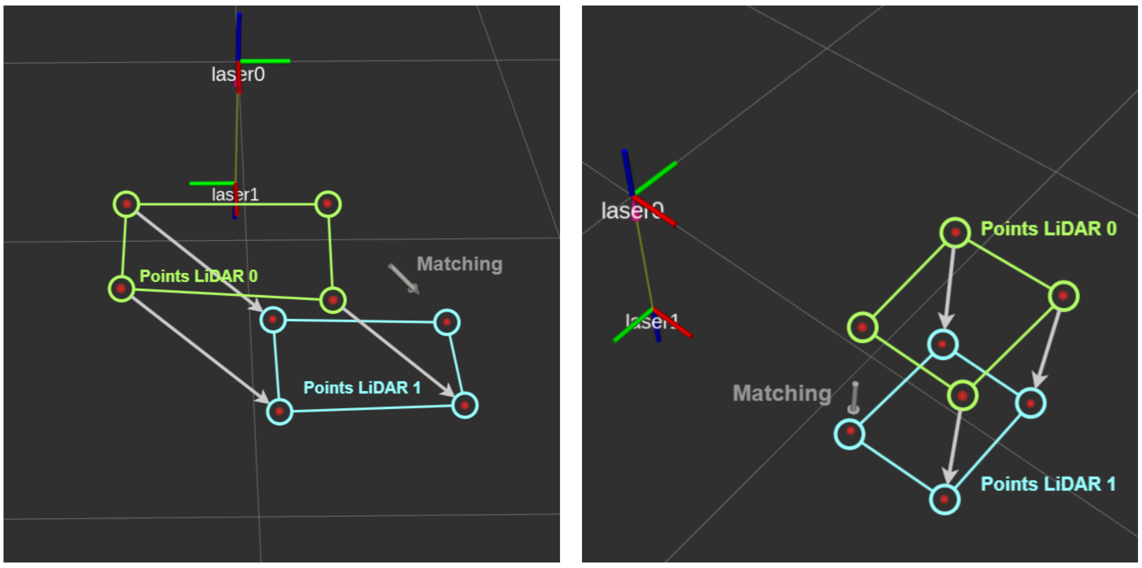


Figure 6.4 Simulation setup in *Gazebo*. The robot has 2 LiDARs (white cylinders) and the world includes 4 power-lines that the robot must detect to then simulate the perching maneuver on one of them.

Once the robot reached the desired pose, the perception system was initialized and the success of the matching algorithm was visually checked by the representation of the detected points and the vector that joins the matched points. The algorithm was tested with a large number of different initial yaw references and no failures were detected. Figure 6.5 shows 2 examples of the matching operation, the first was performed when the robot had a relative yaw of 45°with the power-line and the second when it was aligned with the lines. The first is probably the most failure-prone case, since there are points related to different lines that are detected with different LiDARs and are closer to each other than to the points of the same power-line. The second is the most likely since the UAV will usually start almost aligned with the power-line. As shown, in both cases the matching algorithm is able to identify the power-line correctly.



(a) 45°with the lines. **(b)** 0°with the lines.

Figure 6.5 Examples of performance of the matching algorithm. The red dots are the points detected with the LiDARs. The shortest grey arrow joins the points matched by the perception system.

Once the functioning of the matching algorithm was tested, the control system was used to align the robot with the closest power-line and then reach a pose at 20 cm from the power-line. Figure 6.6 shows the estimations of the perception system during the execution of one of the maneuvers performed together with the ground truth, which is known because it is a simulation. It should be noted that the perception system obtains estimations even more accurate than those shown in the previous section, which is due to the smoother movements compared to the movements in the prior experiments and to the ideal functioning of the simulated LiDARs. In this way, the control system can be properly tested, ensuring that it receives a reliable estimation of the power-line pose.

In order to align the robot with one of the lines and then control its relative height, the control system acts on the horizontal distance from the UAV to the power-line (H), the vertical distance (V), and the relative yaw. The mock power-line with which the flight experiments are performed has a support that prevents the perching from below. Thus, although Chapter 5 described a control system to perch on power-lines above the robot, the maneuver is performed on lines below the robot. In this way, the software tested in simulation is the same as that executed on board the real aerial platform. It should be noted that this change does not compromise the test of the control system as the only differences are the initial height and the final reference.

The evolution of the control variables together with the references of the control system during the performance of the maneuver are shown in Figure 6.7. The mean and standard deviation of the steady-state errors are shown in Table 6.2. Since the control performed as expected in simulation, without high steady-state errors or over-oscillations, the experiments were repeated with a real quadrotor. The following section describes the obtained results and provides a further analysis of

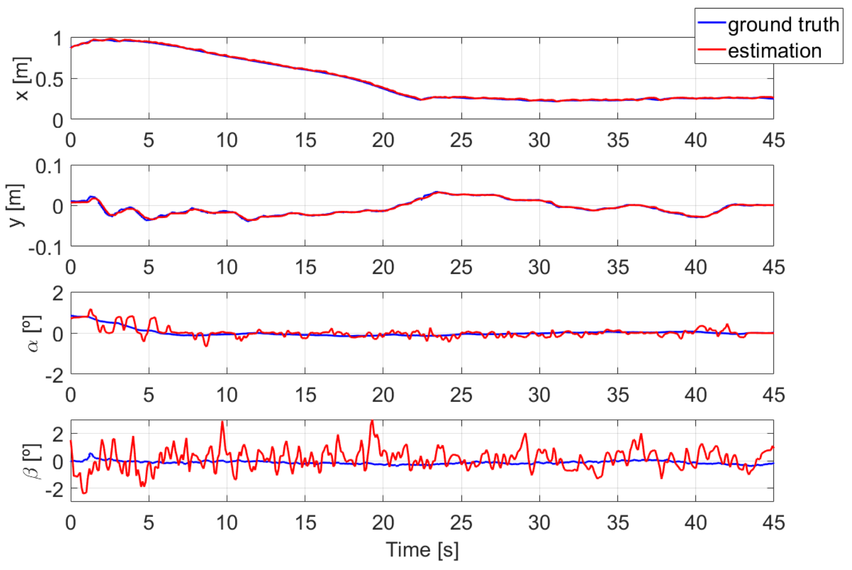


Figure 6.6 Line pose estimated by the perception system and ground truth during a simulated perching maneuver.

the errors of the system while controlling its relative position with a power-line.

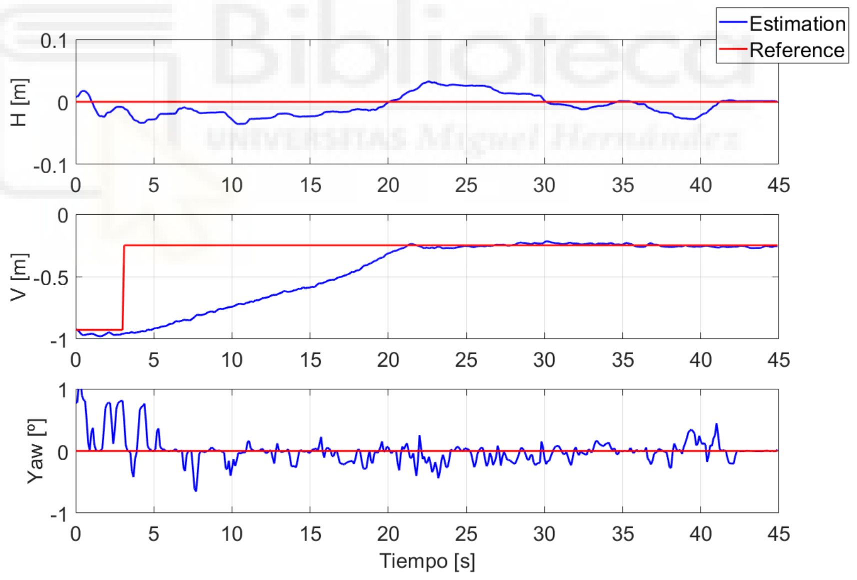


Figure 6.7 Control variables measured with the perception system and references during a simulated experiment.

Table 6.2 Mean Absolute Error (MAE) and STandard Deviation of the Absolute Error (STDAE) of the control system during the steady-state in a simulated experiment.

Variable	MAE	STDAE	Unit
H	1.04	0.88	cm
V	1.13	0.61	cm
Yaw	0.09	0.09	$^\circ$

6.4 Flight experiments

After the successful tests of the perception and control systems, the software was executed on board a real quadrotor. The platform used for the experiments is shown in Figure 6.8. Its onboard computer is an Intel NUC i7 that receives scans from two LiDARs RPLIDAR A3. The experiments were performed inside the motion capture system, which was used as a ground truth of the relative pose of the UAV with respect to the power-line. Since they were not available, the GPS measurements required by the *Autopilot* to fly autonomously were replaced by those of the motion capture system. Figure 6.9 shows the mock powe-line used for the experiments, which is a real power-line cable with a diameter of 1 cm. As previously mentioned, the mock power-line prevents perching from below. Thus, the robot controls its pose with a line below it, which does not compromise the testing of the control system as the only differences are the initial height and the final reference.

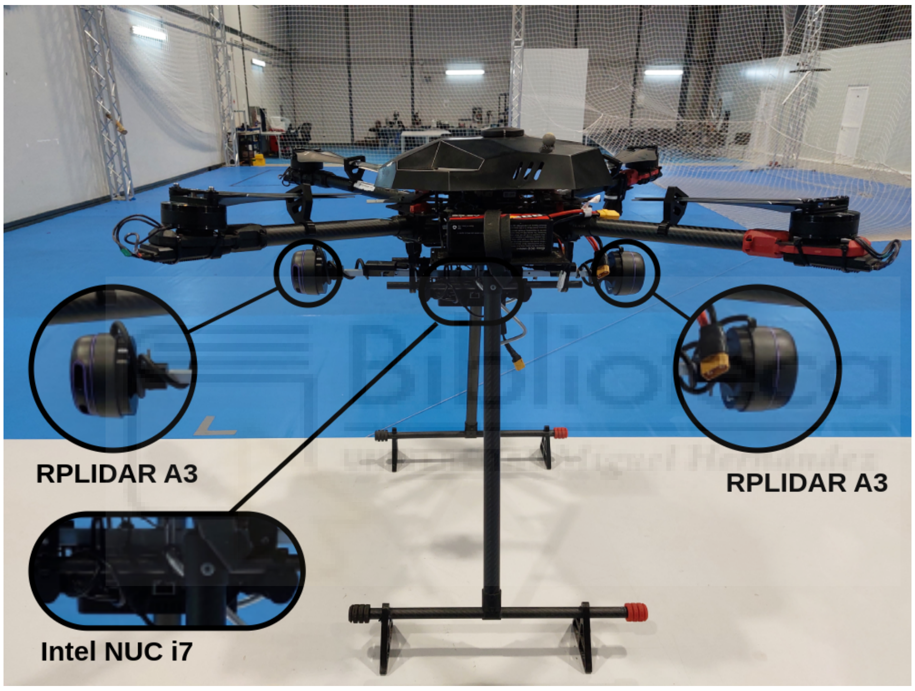


Figure 6.8 Aerial platform used for the flight experiments.

The UAV performs a maneuver similar to that of the simulated platform. First, a pilot manually places the UAV above the lines approximately aligned with them. Secondly, the automatic control starts and the robot aligns itself with the power-line to then approach it at a distance of 35 cm. Finally, the UAV tries to maintain this position to measure the errors of the entire system to reach the desired pose, which will indicate how valid the system is and whether or not a perching mechanism can be used. Figure 6.10 shows a time lapse of the flight with the final pose of the UAV above the power-line completely aligned with it.

Figure 6.11 shows a comparison between the estimations of the perception system and the ground truth obtained with the motion capture system during one of the experiments. The absolute errors between the estimations and the ground truth are represented in Figure 6.12. The mean and standard deviation of the absolute errors of each variable are shown in Table 6.3. Since the relative movement between the LiDARs and the power-line is now the expected one, the errors are much lower than those presented in section 6.2, being in the order of 1 cm for the position and 1°for the orientation. To draw conclusions about the suitability of the system, the performance of the control system should

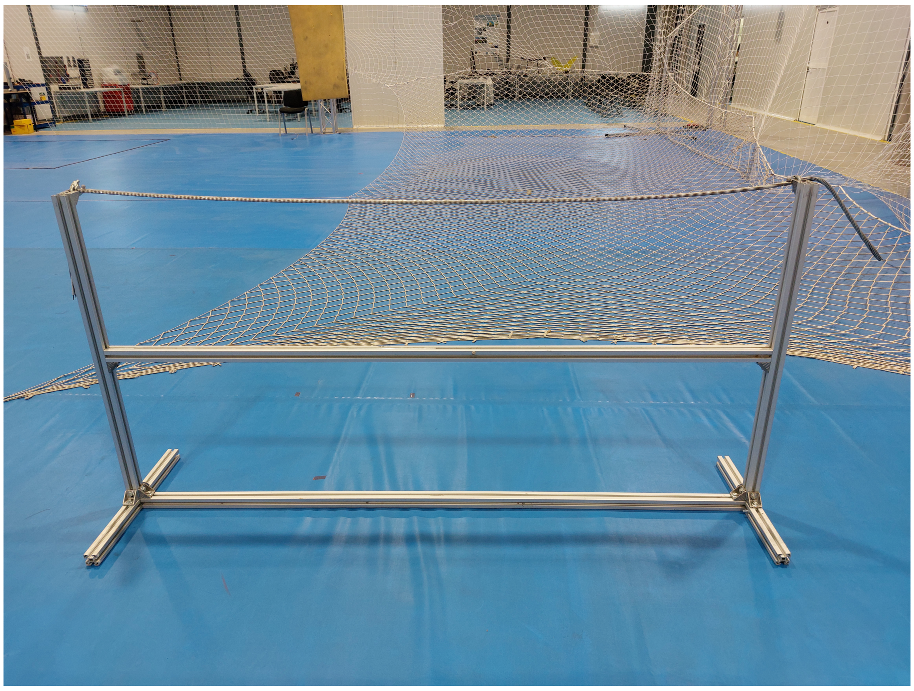


Figure 6.9 Mock power-line used for the flight experiments.

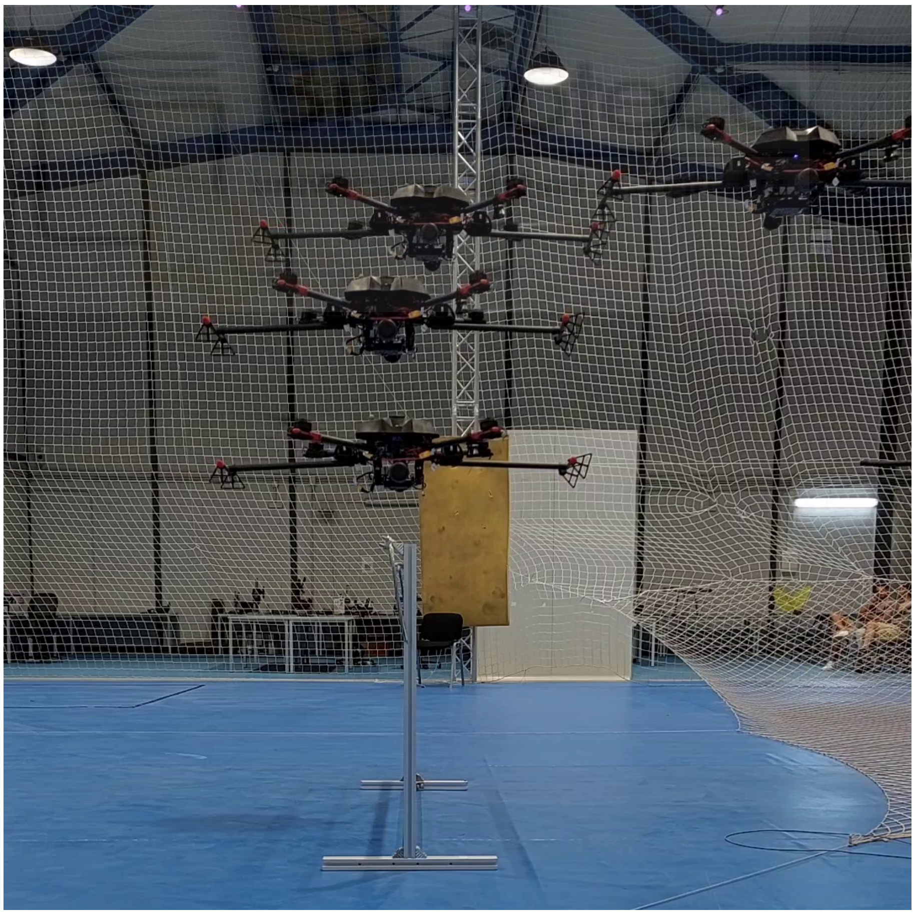


Figure 6.10 Time lapse of the performed flight.

also be taken into account. The control variables have been represented together with the references in Figure 6.13. The mean and standard deviation of the absolute steady-state errors are shown in Table 6.4. As in simulation, the control system performs properly, without high steady-state errors or over-oscillations.

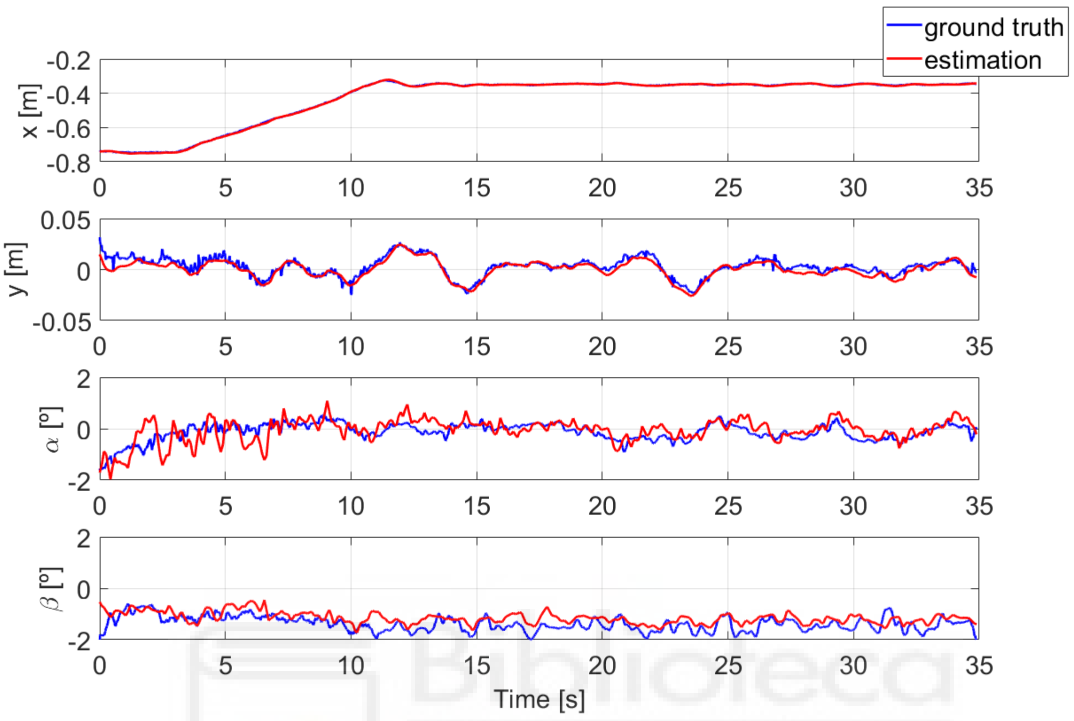


Figure 6.11 Line pose estimated by the perception system and ground truth obtained with the motion capture system during a flight experiment.

Table 6.3 Mean Absolute Error (MAE) and STandard Deviation of the Absolute Error (STDAE) of the perception system to estimate the pose of a power-line during a real flight.

Variable	MAE	STDAE	Unit
x	0.22	0.19	cm
y	0.34	0.31	cm
α	0.31	0.27	°
β	0.37	0.26	°

Table 6.4 Mean Absolute Error (MAE) and STandard Deviation of the Absolute Error (STDAE) of the control system during the steady-state in a flight experiment.

Variable	MAE	STDAE	Unit
H	0.67	0.57	cm
V	0.38	0.30	cm
Yaw	0.22	0.17	°

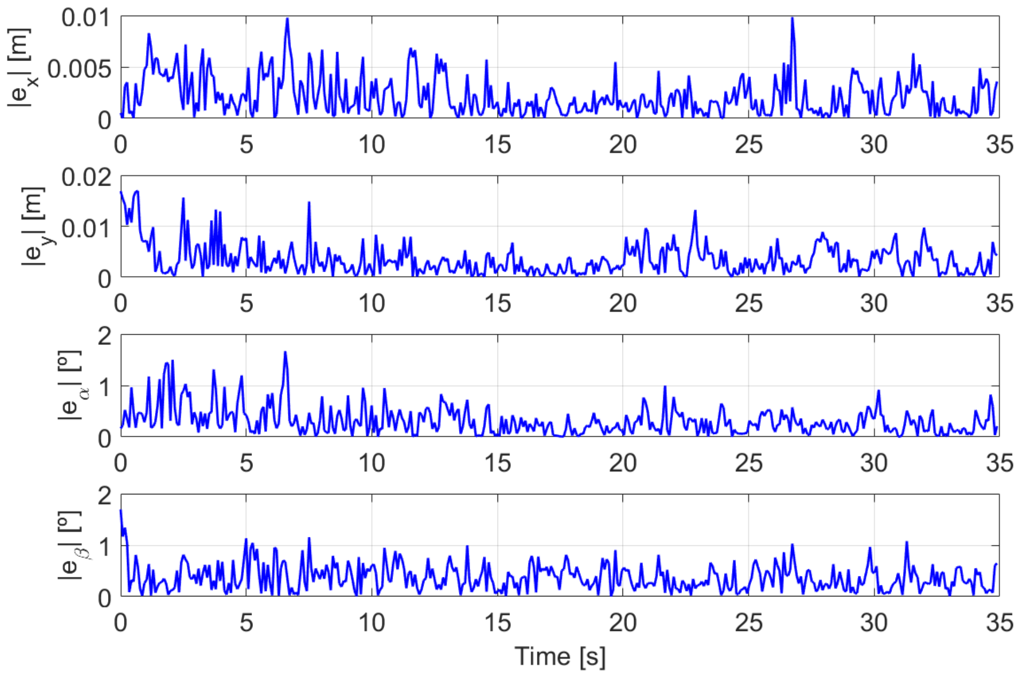


Figure 6.12 Absolute errors of the perception system during a flight experiment.

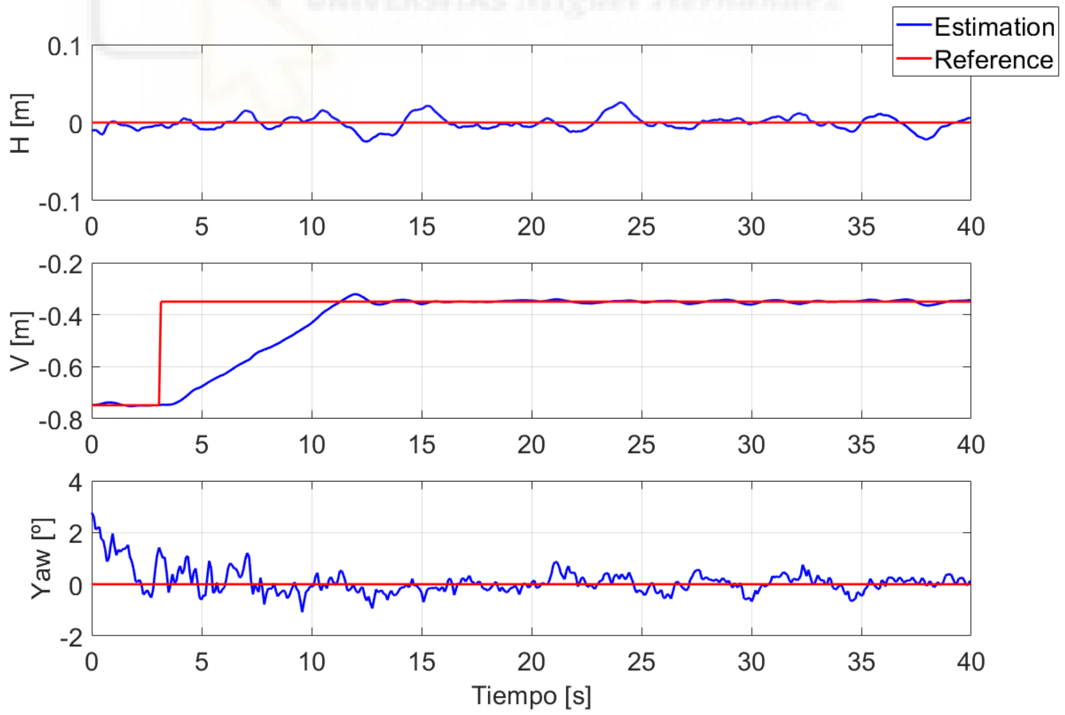


Figure 6.13 Control variables measured with the perception system and references during a flight experiment.

6.5 Conclusion

The experiments described in this chapter prove the correct functioning of the perception and control systems and provide a quantitative evaluation of their capabilities. The perception system was first evaluated with a challenging test that proved its robustness and its ability to track and estimate the pose of a power-line performing random movements. Despite being considerably low, the errors measured in this experiments are not representative of the capabilities of the perception system, which was designed to estimate the pose of the power-line during the performance of a slow perching maneuver. The control system performed properly in simulation as well as the matching algorithm, which was tested in simulation due to the available set of power-lines and the fact that there is no difference between the test with measurements from simulated or real LiDARs. Finally, both systems were tested together in flight experiments with a real aerial platform. During the experiment, all the software was executed on board, and the robot was able to properly align itself with the power-line and approximate it at the desired distance.

All the experiments had successful results with remarkably low errors of the perception system while estimating the pose of a power-line, and of the control system while tracking references. In the case of the flight experiment, which is the most representative, the absolute errors of the estimations of the perception system had a mean and a standard deviation lower than 1 cm for the position and 1°for the orientation. Similarly, the mean and standard deviation of the absolute steady-state errors of the control system were lower than 1 cm for the position and 1°for the orientation.

7 Conclusion

7.1 Conclusion

This work has addressed the development of a method to enable multirotors to autonomously perch on power-lines. Since aerial platforms are dangerous and have limited computational power, the method has been designed to ensure its robustness, efficiency, and accuracy. Two systems have been developed, one in charge of the perception and the other for the control of the platform. The perception system is in turn composed of 3 modules: the *Intersection Detection*, *Power-line Detection*, and *Power-line Filter* modules. This modular architecture has facilitated the development and might allow the extension to other applications, for instance, to perch on pipes if the *Intersection Detection* module is modified to detect pipe intersections instead of power-line intersections.

The suitability of the developed method has been validated in simulation and using a real quadrotor, providing a quantitative evaluation of its capabilities. The goal of the method is to perform perching on a power-line, which involves the entrance of the line in a certain mechanism to grab it. Most perching mechanisms have an aperture of a certain width and an interval of angles within which the power-line can enter the mechanism. To use the autonomous perching method presented in this work, the width and interval of allowed angles for the entrance must be compatible with the errors of the control and perception systems, which, as shown, are in the order of 1 cm for the position and $1^{\circ}$for the orientation (see Chapter 6).

Taking into account that an error of 1°in the orientation is remarkably low and that the position error is in the order of the width of the power-line, it has been assumed that the presented method is compatible with a large number of perching mechanisms and that the goal of the work has been achieved.

7.2 Future work

The achieved errors in the robot relative pose with respect to the power-line are considered low enough to allow the use of the method in a large number of applications. Nevertheless, there is still some future work that might improve its capabilities.

First, the perception system can only track single lines. However, there are usually several lines between electrical towers whose cross-section describes a specific pattern. The accuracy and robustness of the system might be improved by the simultaneous estimation of all the lines and the comparison of the estimations with the pattern.

Second, this work has been completely focused on the perching maneuver. However, achieving relative poses with power-lines might be required by other applications, as the physical interaction with the power-line. Despite the good performance of the control system during the validation experiments, it is based on simple proportional controllers and might not be suitable in the presence of hard perturbations, such as the aforementioned physical interaction or the wind. Therefore, the use of more complex controllers is an interesting topic to research.

Lastly, the low-level control of the platform relies on the GPS control of the *Autopilot*. Although power-lines are usually located in clear outdoor spaces and GPS is almost always available, this requirement is still a limitation of the method. The estimation of the pose of the power-line might be used to localize the robot and avoid the use of GPS. Nevertheless, the perception system is not able to estimate the pose of the robot along the direction of the power-line. There are multiple options to estimate the missing DOF. For example, the odometry estimated by the *Autopilot* or the addition of a sensor. Since multirotors fly horizontal and require tilting to move, the orientation of the power-line might also be used to avoid movements in the direction of the line. However, more research is required to confirm the suitability of these approaches.

Appendix A

Elements of the H matrix

In this appendix the elements of the *H* matrix for the EKF in the *Power-line Filter* module are listed.

$$\frac{\partial \rho_0}{\partial x} = \frac{x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)}{\sqrt{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)^2 + (y - \Delta z \tan(\alpha))^2}} \tag{A.1}$$

$$\frac{\partial \rho_0}{\partial y} = \frac{y - \Delta z \tan(\alpha)}{\sqrt{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)^2 + (y - \Delta z \tan(\alpha))^2}} \tag{A.2}$$

$$\frac{\partial \rho_0}{\partial v^x} = 0 \tag{A.3}$$

$$\frac{\partial \rho_0}{\partial v^y} = 0 \tag{A.4}$$

$$\frac{\partial \rho_0}{\partial \alpha} = \frac{-\frac{2\Delta z^2 \sec(\alpha)^2 \tan(\alpha)\tan(\beta)\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)}{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}} - 2\Delta z \sec(\alpha)^2\,(y - \Delta z \tan(\alpha))}{2\sqrt{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)^2 + (y - \Delta z \tan(\alpha))^2}} \tag{A.5}$$

$$\frac{\partial \rho_0}{\partial \beta} = -\frac{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\sec(\beta)^2\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)}{\sqrt{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\,\tan(\beta)\right)^2 + (y - \Delta z \tan(\alpha))^2}} \tag{A.6}$$

$$\frac{\partial \rho_{b0}}{\partial x} = \frac{-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x}{\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.7}$$

$$\frac{\partial \rho_{b0}}{\partial y} = \frac{-\Delta z \tan(\alpha) + y - T\,v^y}{\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.8}$$

$$\frac{\partial \rho_{b0}}{\partial v^x} = -\frac{T\,\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)}{\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.9}$$

$$\frac{\partial \rho_{b0}}{\partial v^y} = -\frac{T\,(-\Delta z \tan(\alpha) + y - T\,v^y)}{\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.10}$$

$$\frac{\partial \rho_{b0}}{\partial \alpha} = \frac{-\frac{2\Delta z^2 \sec(\alpha)^2 \tan(\alpha)\tan(\beta)\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)}{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}} - 2\Delta z \sec(\alpha)^2\,(-\Delta z \tan(\alpha) + y - T\,v^y)}{2\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.11}$$

$$\frac{\partial \rho_{b0}}{\partial \beta} = -\frac{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\sec(\beta)^2\,\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)}{\sqrt{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\,v^x\right)^2 + (-\Delta z \tan(\alpha) + y - T\,v^y)^2}} \tag{A.12}$$

$$\frac{\partial \theta_0}{\partial x} = -\frac{y - \Delta z \tan(\alpha)}{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta)\right)^2 \left(\frac{(y - \Delta z \tan(\alpha))^2}{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta)\right)^2} + 1\right)} \tag{A.13}$$

$$\frac{\partial \theta_0}{\partial y} = \frac{1}{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta)\right)\left(\frac{(y - \Delta z \tan(\alpha))^2}{\left(x - \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta)\right)^2} + 1\right)} \tag{A.14}$$

$$\frac{\partial \theta_0}{\partial v^x} = 0 \tag{A.15}$$

$$\frac{\partial \theta_0}{\partial v^y} = 0 \tag{A.16}$$

$$\frac{\partial \theta_0}{\partial \alpha} = \frac{\frac{\Delta z^2 \sec(\alpha)^2 \tan(\alpha)(y-\Delta z \tan(\alpha))\tan(\beta)}{\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\left(x-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)\right)^2} - \frac{\Delta z \sec(\alpha)^2}{x-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)}}{\frac{(y-\Delta z \tan(\alpha))^2}{\left(x-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)\right)^2}+1} \tag{A.17}$$

$$\frac{\partial \theta_0}{\partial \beta} = \frac{(y-\Delta z \tan(\alpha))\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\sec(\beta)^2}{\left(x-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)\right)^2\left(\frac{(y-\Delta z \tan(\alpha))^2}{\left(x-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)\right)^2}+1\right)} \tag{A.18}$$

$$\frac{\partial \theta_{b0}}{\partial x} = -\frac{-\Delta z \tan(\alpha)+y-T\,v^y}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2\left(\frac{(-\Delta z \tan(\alpha)+y-T\,v^y)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2}+1\right)} \tag{A.19}$$

$$\frac{\partial \theta_{b0}}{\partial y} = \frac{1}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)\left(\frac{(-\Delta z \tan(\alpha)+y-T\,v^y)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2}+1\right)} \tag{A.20}$$

$$\frac{\partial \theta_{b0}}{\partial v^x} = \frac{T\,(-\Delta z \tan(\alpha)+y-T\,v^y)}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2\left(\frac{(-\Delta z \tan(\alpha)+y-T\,v^y)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2}+1\right)} \tag{A.21}$$

$$\frac{\partial \theta_{b0}}{\partial v^y} = -\frac{T}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)\left(\frac{(-\Delta z \tan(\alpha)+y-T\,v^y)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2}+1\right)} \tag{A.22}$$

$$\frac{\partial \theta_{b0}}{\partial \alpha} = \frac{\frac{\Delta z^2 \sec(\alpha)^2 \tan(\alpha)(-\Delta z \tan(\alpha)+y-T\,v^y)\tan(\beta)}{\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2} - \frac{\Delta z \sec(\alpha)^2}{-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x}}{\frac{(-\Delta z \tan(\alpha)+y-T\,v^y)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2}+1} \tag{A.23}$$

$$\frac{\partial \theta_{b0}}{\partial \beta} = \frac{\left(-\Delta z \tan(\alpha) + y - T\, v^y\right) \sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\, \sec(\beta)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x - T\, v^x\right)^2 \left(\frac{\left(-\Delta z \tan(\alpha) + y - T\, v^y\right)^2}{\left(-\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x - T\, v^x\right)^2} + 1\right)} \tag{A.24}$$

$$\frac{\partial \rho_1}{\partial x} = \frac{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x}{\sqrt{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)^2 + \left(\Delta z \tan(\alpha) + y\right)^2}} \tag{A.25}$$

$$\frac{\partial \rho_1}{\partial y} = \frac{\Delta z \tan(\alpha) + y}{\sqrt{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)^2 + \left(\Delta z \tan(\alpha) + y\right)^2}} \tag{A.26}$$

$$\frac{\partial \rho_1}{\partial v^x} = 0 \tag{A.27}$$

$$\frac{\partial \rho_1}{\partial v^y} = 0 \tag{A.28}$$

$$\frac{\partial \rho_1}{\partial \alpha} = \frac{\frac{2\Delta z^2 \sec(\alpha)^2 \tan(\alpha) \tan(\beta) \left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)}{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}} + 2\Delta z \sec(\alpha)^2 \left(\Delta z \tan(\alpha) + y\right)}{2\sqrt{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)^2 + \left(\Delta z \tan(\alpha) + y\right)^2}} \tag{A.29}$$

$$\frac{\partial \rho_1}{\partial \beta} = \frac{\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\, \sec(\beta)^2 \left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)}{\sqrt{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x\right)^2 + \left(\Delta z \tan(\alpha) + y\right)^2}} \tag{A.30}$$

(A.31)

$$\frac{\partial \rho_{b1}}{\partial x} = \frac{1}{2\sqrt{\left(\Delta z^2 \tan(\alpha)^2 + \Delta z^2\right) \tan(\beta)^2 + \left(\Delta z \tan(\alpha) + y - T\, v^y\right)^2 + x - T\, v^x}} \tag{A.32}$$

$$\frac{\partial \rho_{b1}}{\partial y} = \frac{\Delta z \tan(\alpha) + y - T\, v^y}{\sqrt{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2} \tan(\beta) + x - T\, v^x\right)^2 + \left(\Delta z \tan(\alpha) + y - T\, v^y\right)^2}} \tag{A.33}$$

$$\frac{\partial \rho_{b1}}{\partial v^x} = -\frac{T\,\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)}{\sqrt{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2+(\Delta z\tan(\alpha)+y-T\,v^y)^2}} \tag{A.34}$$

$$\frac{\partial \rho_{b1}}{\partial v^y} = -\frac{T\,(\Delta z\tan(\alpha)+y-T\,v^y)}{\sqrt{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2+(\Delta z\tan(\alpha)+y-T\,v^y)^2}} \tag{A.35}$$

$$\frac{\partial \rho_{b1}}{\partial \alpha} = \frac{\frac{2\Delta z^2\sec(\alpha)^2\tan(\alpha)\tan(\beta)\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)}{\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}}+2\Delta z\sec(\alpha)^2\,(\Delta z\tan(\alpha)+y-T\,v^y)}{2\sqrt{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2+(\Delta z\tan(\alpha)+y-T\,v^y)^2}} \tag{A.36}$$

$$\frac{\partial \rho_{b1}}{\partial \beta} = \frac{\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\sec(\beta)^2\,\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)}{\sqrt{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\,v^x\right)^2+(\Delta z\tan(\alpha)+y-T\,v^y)^2}} \tag{A.37}$$

$$\frac{\partial \theta_1}{\partial x} = -\frac{\Delta z\tan(\alpha)+y}{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2\left(\frac{(\Delta z\tan(\alpha)+y)^2}{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2}+1\right)} \tag{A.38}$$

$$\frac{\partial \theta_1}{\partial y} = \frac{1}{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)\left(\frac{(\Delta z\tan(\alpha)+y)^2}{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2}+1\right)} \tag{A.39}$$

$$\frac{\partial \theta_1}{\partial v^x} = 0 \tag{A.40}$$

$$\frac{\partial \theta_1}{\partial v^y} = 0 \tag{A.41}$$

$$\frac{\partial \theta_1}{\partial \alpha} = \frac{\frac{\Delta z\sec(\alpha)^2}{\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x}-\frac{\Delta z^2\sec(\alpha)^2\tan(\alpha)(\Delta z\tan(\alpha)+y)\tan(\beta)}{\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2}}{\frac{(\Delta z\tan(\alpha)+y)^2}{\left(\sqrt{\Delta z^2\tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2}+1} \tag{A.42}$$

$$\frac{\partial \theta_1}{\partial \beta} = -\frac{(\Delta z \tan(\alpha) + y)\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\sec(\beta)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x\right)^2 \left(\frac{(\Delta z \tan(\alpha)+y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x\right)^2} + 1\right)} \tag{A.43}$$

$$\frac{\partial \theta_{b1}}{\partial x} = -\frac{\Delta z \tan(\alpha) + y - T\, v^y}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\, v^x\right)^2 \left(\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1\right)} \tag{A.44}$$

$$\frac{\partial \theta_{b1}}{\partial y} = \frac{1}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\, v^x\right) \left(\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1\right)} \tag{A.45}$$

$$\frac{\partial \theta_{b1}}{\partial v^x} = \frac{T\,(\Delta z \tan(\alpha) + y - T\, v^y)}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\, v^x\right)^2 \left(\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1\right)} \tag{A.46}$$

$$\frac{\partial \theta_{b1}}{\partial v^y} = -\frac{T}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\, v^x\right) \left(\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1\right)} \tag{A.47}$$

$$\frac{\partial \theta_{b1}}{\partial \alpha} = \frac{\frac{\Delta z \sec(\alpha)^2}{\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x} - \frac{\Delta z^2 \sec(\alpha)^2 \tan(\alpha)(\Delta z \tan(\alpha)+y-T\, v^y)\tan(\beta)}{\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2}}{\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1} \tag{A.48}$$

$$\frac{\partial \theta_{b1}}{\partial \beta} = -\frac{(\Delta z \tan(\alpha) + y - T\, v^y)\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\sec(\beta)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2 + \Delta z^2}\tan(\beta) + x - T\, v^x\right)^2 \left(\frac{(\Delta z \tan(\alpha)+y-T\, v^y)^2}{\left(\sqrt{\Delta z^2 \tan(\alpha)^2+\Delta z^2}\tan(\beta)+x-T\, v^x\right)^2} + 1\right)} \tag{A.49}$$

List of Figures

1.1 Configuration of the LiDARs on board the aerial platform 2

2.1 Variables to calculate the maximum detection distance given a certain angular resolution 8

3.1 Diagram of the perception system stages 11
3.2 Diagram of the perception system modules 11

4.1 Example of rejection of a possible cluster 16
4.2 Example of detection of a possible cluster 16
4.3 Visual representation of the position of the coordinate frame of the LiDARs 18
4.4 Representation of the coordinates frames related to the perception problem 20
4.5 Points to define the equations of the filter 21
4.6 Angles defined to represent the orientation of the power-line 21
4.7 State machine executed by the *Power-line Detection* module 25
4.8 Visual representation of the matching operation 26

5.1 Variables related to the velocity control 29
5.2 State machine that establishes the reference of the controllers 30

6.1 Stick with markers for the motion capture system to simulate the relative movement between the LiDARs and a power-line 32
6.2 Line pose estimated by the perception system and ground truth obtained with a motion capture system 32
6.3 Absolute errors of the perception system to detect a moving line from a static pose 34
6.4 Simulation setup in *Gazebo* 34
6.5 Examples of the performance of the matching algorithm 35
6.6 Line pose estimated by the perception system and ground truth during a simulated perching maneuver 36
6.7 Control variables measured with the perception system and references during a simulated experiment 36
6.8 Aerial platform used for the flight experiments 37
6.9 Mock power-line used for the flight experiments 38
6.10 Time lapse of the performed flight 38
6.11 Line pose estimated by the perception system and ground truth obtained with the motion capture system during a flight experiment 39
6.12 Absolute errors of the perception system during a flight experiment 40
6.13 Control variables measured with the perception system and references during a flight experiment 40

List of Tables

2.1 Properties of *SLAMTEC* LiDARs suggested for power-line detection on board a UAV 8

4.1 Public member functions provided by the *Intersection Detection* module 14
4.2 Public member functions provided by the *Power-line Filter* module 19

6.1 MAE and STDAE of the perception system to detect a moving line from a static pose 33
6.2 MAE and STDAE of the control system during the steady-state in a simulated experiment 36
6.3 MAE and STDAE of the perception system to estimate the pose of a power-line during a real flight 39
6.4 MAE and STDAE of the control system during the steady-state in a flight experiment 39

Bibliography

[1] Nicolai Iversen, Aljaž Kramberger, Oscar Bowen Schofield, and Emad Ebeid. Novel power line grasping mechanism with integrated energy harvester for uav applications. In *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 34–39, 2021.

[2] Ryan Kitchen, Nick Bierwolf, Sean Harbertson, Brage Platt, Dean Owen, Klaus Griessmann, and Mark A. Minor. Design and evaluation of a perching hexacopter drone for energy harvesting from power lines. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1192–1198, 2020.

[3] Gerd vom Bögel, Linda Cousin, Nicolai Iversen, Emad Samuel Malki Ebeid, and Andreas Hennig. Drones for inspection of overhead power lines with recharge function. In *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pages 497–502, 2020.

[4] Jaka Katrasnik, Franjo Pernus, and Bostjan Likar. A survey of mobile robots for distribution power line inspection. *IEEE Transactions on Power Delivery*, 25(1):485–493, 2010.

[5] Yoonseok Jwa, Gunho Sohn, and HB Kim. Automatic 3d powerline reconstruction using airborne lidar data. *Int. Arch. Photogramm. Remote Sens*, 38(Part 3):W8, 2009.

[6] Binhai Wang, Lei Han, Hailong Zhang, Qian Wang, and Bingqiang Li. A flying robotic system for power line corridor inspection. In *2009 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2468–2473. IEEE, 2009.

[7] Hongmei Li, Binhai Wang, Liang Liu, Gangyin Tian, Tianru Zheng, and Jingjing Zhang. The design and application of smartcopter: An unmanned helicopter based robot for transmission line inspection. In *2013 Chinese automation congress*, pages 697–702. IEEE, 2013.

[8] Walter de Britto Vidal Filho and André Murilo de Almeida Pinto. Vtol aerial robot for inspection of transmission line. In *Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry*, pages 1–4. IEEE, 2014.

[9] Mingxin Wang. Inspection technology of remote transmission towers based on a vertical take-off and landing fixed-wing uav. In *Journal of Physics: Conference Series*, volume 1865, page 022076. IOP Publishing, 2021.

[10] Rishav Bhola, Nandigam Hari Krishna, KN Ramesh, J Senthilnath, and Gautham Anand. Detection of the power lines in uav remote sensed images using spectral-spatial methods. *Journal of environmental management*, 206:1233–1242, 2018.

[11] J Paneque, V Valseca, JR Martínez-de Dios, and A Ollero. Autonomous reactive lidar-based mapping for powerline inspection. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 962–971. IEEE, 2022.

[12] Kenta Takaya, Hiroshi Ohta, Valeri Kroumov, Keishi Shibayama, and Masanao Nakamura. Development of uav system for autonomous power line inspection. In *2019 23rd International Conference on System Theory, Control and Computing (ICSTCC)*, pages 762–767. IEEE, 2019.

[13] Jiang Bian, Xiaolong Hui, Xiaoguang Zhao, and Min Tan. A novel monocular-based navigation approach for uav autonomous transmission-line inspection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–7. IEEE, 2018.

[14] Alejandro Suarez, Rafael Salmoral, Pedro J. Zarco-Periñan, and Anibal Ollero. Experimental evaluation of aerial manipulation robot in contact with 15 kv power line: Shielded and long reach configurations. *IEEE Access*, 9:94573–94585, 2021.

[15] Jonathan Cacace, Santos M. Orozco-Soto, Alejandro Suarez, Alvaro Caballero, Matko Orsag, Stjepan Bogdan, Goran Vasiljevic, Emad Ebeid, Jose Alberto Acosta Rodriguez, and Anibal Ollero. Safe local aerial manipulation for the installation of devices on power lines: Aerial-core first year results and designs. *Applied Sciences*, 11(13), 2021.

[16] Alejandro Suarez, Saeed Rafee Nekoo, and Anibal Ollero. Ultra-lightweight anthropomorphic dual-arm rolling robot for dexterous manipulation tasks on linear infrastructures: A self-stabilizing system. *Mechatronics*, 94:103021, 2023.

[17] Nicolai Iversen, Oscar Bowen Schofield, Linda Cousin, Naeem Ayoub, Gerd Vom Bögel, and Emad Ebeid. Design, integration and implementation of an intelligent and self-recharging drone system for autonomous power line inspection. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4168–4175. IEEE, 2021.

[18] Dario Stuhne, Viet Duong Hoang, Goran Vasiljevic, Stjepan Bogdan, Zdenko Kovacic, Anibal Ollero, and Emad Samuel Malki Ebeid. Design of a wireless drone recharging station and a special robot end effector for installation on a power line. *IEEE Access*, 10:88719–88737, 2022.

[19] Julio L. Paneque, Jose Ramiro Martínez-de Dios, Anibal Ollero, Drew Hanover, Sihao Sun, Angel Romero, and Davide Scaramuzza. Perception-aware perching on powerlines with multirotors. *IEEE Robotics and Automation Letters*, 7(2):3077–3084, 2022.

[20] François Mirallès, Philippe Hamelin, Ghislain Lambert, Samuel Lavoie, Nicolas Pouliot, Matthieu Montfrond, and Serge Montambault. Linedrone technology: Landing an unmanned aerial vehicle on a power line. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6545–6552. IEEE, 2018.

[21] Joseph Moore and Russ Tedrake. Magnetic localization for perching uavs on powerlines. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2700–2707, 2011.

[22] Dean Martinović, Stjepan Bogdan, and Zdenko Kovačić. Mathematical considerations for unmanned aerial vehicle navigation in the magnetic field of two parallel transmission lines. *Applied Sciences*, 11(8):3323, 2021.

[23] Goran Vasiljević, Dean Martinović, Matko Orsag, and Stjepan Bogdan. Grabbing power line conductors based on the measurements of the magnetic field strength. In *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, pages 1–7, 2021.

[24] Nicolai Iversen, Oscar Bowen Schofield, and Emad Ebeid. Locator-lightweight and low-cost autonomous drone system for overhead cable detection and soft grasping. In *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 205–212. IEEE, 2020.

[25] Nicolaj Haarhøj Malle, Frederik Falk Nyboe, and Emad Ebeid. Survey and evaluation of sensors for overhead cable detection using uavs. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 361–370. IEEE, 2021.

[26] Nicolaj Haarhøj Malle, Frederik Falk Nyboe, and Emad Samuel Malki Ebeid. Onboard powerline perception system for uavs using mmwave radar and fpga-accelerated vision. *IEEE Access*, 10:113543–113559, 2022.

[27] Yi Kiat Tee and Yi Chiew Han. Lidar-based 2d slam for mobile robot in an indoor environment: A review. In *2021 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)*, pages 1–7. IEEE, 2021.

[28] Matěj Petrlík, Tomáš Báča, Daniel Heřt, Matouš Vrba, Tomáš Krajník, and Martin Saska. A robust uav system for operations in a constrained environment. *IEEE Robotics and Automation Letters*, 5(2):2169–2176, 2020.

[29] Matěj Petrlík, Tomáš Krajník, and Martin Saska. Lidar-based stabilization, navigation and localization for uavs operating in dark indoor environments. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 243–251, 2021.

[30] SLAMTEC. Slamtec, 2023.

[31] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[32] Roohul Amin, Li Aijun, and Shahaboddin Shamshirband. A review of quadrotor uav: control methodologies and performance evaluation. *International Journal of Automation and Control*, 10(2):87–103, 2016.

[33] VI Kortunov, OV Mazurenko, AV Gorbenko, Watheq Mohammed, and Ali Hussein. Review and comparative analysis of mini-and micro-uav autopilots. In *2015 IEEE international conference actual problems of unmanned aerial vehicles developments (APUAVD)*, pages 284–289. IEEE, 2015.

[34] Fran Real, Arturo Torres-Gonzalez, Pablo Ramón-Soria, Jesús Capitán, and Anıbal Ollero. Ual: An abstraction layer for unmanned aerial vehicles. In *2nd International Symposium on Aerial Robotics*, 2018.