UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

"Aplicación del Flujo Óptico de Lucas-Kanade en la Correlación de Imágenes Digitales (DIC) para la medición de Deformaciones en Ensayos de Tracción"

TRABAJO FIN DE GRADO

Junio 2025

AUTOR: Javier Guilabert Martínez

DIRECTOR/ES: Carlos Pérez Vidal

Gonzalo Cerezo Calle

AGRADECIMIENTOS

En primer lugar, me gustaría dedicar unas palabras en honor a mi padre:

"*Papá,*

Gracias por ayudar, escuchar y aconsejar. Durante los últimos cuatro años y medio has sido un ejemplo de lucha, valentía y amor. Me siento afortunado de haberte tenido como referente, como padre y como mejor amigo. No me llegan los brazos al cielo para darte un abrazo, pero quiero recordarte que te quiero con todo mi corazón y siempre serás la mejor persona que he conocido.

Xavi"

A mi madre y mi hermano, sois un pilar fundamental en mi vida. Gracias por haberme ayudado siempre que lo he necesitado.

Por supuesto, gracias Carlos por haberme acogido y ayudado desde el primer día. También agradecer la implicación de Gonzalo y Miguel Ángel en el proyecto, por trabajar y sacar adelante el trabajo a pesar de las dificultades encontradas.

A todas las personas que me acompañaron en silencio durante los momentos difíciles, y muy especialmente a quien me dio fuerzas incluso en su ausencia. Este trabajo late con tu memoria.

En honor a José Antonio Guilabert Martínez

RESUMEN

El presente trabajo se centra en el procesamiento y análisis de videos de ensayos de tracción, con especial énfasis en la detección y seguimiento de puntos de referencia. Para ello, se ha desarrollado un programa en Python utilizando OpenCV, empleando el algoritmo de Lucas-Kanade para el seguimiento óptico de los puntos. La disposición de estos puntos puede realizarse de forma manual o automática mediante una matriz generada a lo largo de la superficie visible de la probeta. Además, se han definido dos ROIs (Regions of Interest) para analizar la diferencia entre las velocidades medias de los puntos dentro de cada una, permitiendo así estudiar la deformación de la probeta y compararla con los valores reales proporcionados por los sensores lineales de posición de cada cilindro. La creación de estas ROIs puede efectuarse manualmente o de manera automatizada, basándose en el ancho de la probeta y la posición del pin, garantizando su correcta ubicación en la pantalla de estudio y ajustándolas cuando sea necesario para evitar pérdidas de información. Asimismo, se ha estandarizado la resolución de los videos para eliminar distorsiones y asegurar la coherencia en el seguimiento de los puntos, dado que los videos originales presentan resoluciones variables. Se ha optimizado el procesamiento de datos en Excel, organizando los ensayos de forma estructurada sin pérdida de información y facilitando su análisis.

Además, se han implementado mejoras en la precisión del cálculo de desplazamientos, teniendo en cuenta la conversión de coordenadas, la interpolación de píxeles y la corrección de la curvatura que se pierde en la imagen al realizar un análisis DIC 2D con las características particulares del ensayo. Por último, se ha automatizado la detección del inicio del movimiento y la rotura de la probeta, contribuyendo a un análisis más preciso y eficiente de los ensayos. Todo ello permite una captura de datos más fiable y un procesamiento optimizado, mejorando la precisión y automatización del estudio de la tracción.

Palabras clave: DIC, Flujo óptico, Python, Región de Interés (ROIs), OpenCV

ABSTRACT

This project focuses on the processing and analysis of tensile test videos, with a particular emphasis on detecting and tracking reference points. To achieve this, a Python-based program has been developed using OpenCV, utilizing the Lucas-Kanade algorithm for optical flow tracking. The placement of these points can be done either manually or automatically by generating a grid across the visible surface of the specimen. Additionally, two Regions of Interest (ROIs) have been defined to analyze the difference in average velocities of the points within each region, enabling the study of the specimen's deformation and its comparison with the actual values obtained from the linear position sensors of each cylinder. These ROIs can be created manually or automatically, based on the specimen's width and the pin's position, ensuring their accurate placement on the study screen and adjusting them as needed to prevent data loss.

Furthermore, the video resolution has been standardized to eliminate distortions and ensure consistent point tracking, as the original videos have varying resolutions. Data processing in Excel has been optimized, organizing the tests in a structured format without losing information and simplifying their analysis. Additionally, improvements have been made to enhance the accuracy of displacement calculations, incorporating coordinate conversion, pixel interpolation, and curvature correction to account for the limitations of 2D Digital Image Correlation (DIC) analysis in the context of this specific test. Finally, the detection of the initial movement and the specimen's fracture has been automated, contributing to a more precise and efficient analysis of the tests. Overall, these advancements enable more reliable data capture and streamlined processing, improving the accuracy and automation of tensile testing studies.

Keywords: DIC, Optic Flow, Python, Region Of Interest, OpenCV

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1
INTRODUCCIÓN

1.1. Contexto y Motivación

En la actualidad, el análisis de experimental de tensiones y deformaciones es fundamental en la ingeniería y la ciencia de materiales para comprender el comportamiento mecánico de los materiales bajo diferentes condiciones de carga. Sin embargo, la mayoría de las técnicas tradicionales de medición de tensiones (deformaciones) experimentales presentan limitaciones significativas. En algunos casos (como en los extensómetros), es posible realizar mediciones precisas, pero solo en ubicaciones seleccionadas, mientras que en las áreas restantes (entre los extensómetros) no se investigan. Por otro lado, métodos de campo completo como la fotoelasticidad o las técnicas interferométricas, requieren un esfuerzo considerable, están asociadas con una resolución de deformación limitada (fotoelasticidad), dependen de modelos en lugar del espécimen real o están asociadas con una perturbación significativa del sistema real. En este contexto, la Correlación de Imágenes Digitales (DIC) surge como una técnica revolucionaria que supera estas limitaciones.

Este trabajo se basa en el uso de esta técnica mediante la creación de un programa en Python utilizando la librería de OpenCV y la función de Lucas-Kanade para hacer un estudio exhaustivo de diferentes materiales (BAO QP 980 y TRIP 590) bajo diversas situaciones (lubricante, ángulo de tracción, velocidad de ensayo, diámetro de pin, temperatura y rotación de pin).

La motivación para utilizar una adaptación de DIC mediante Lucas-Kanade radica en su capacidad para proporcionar mediciones precisas y completas sin alterar el sistema bajo estudio, lo que la convierte en una herramienta invaluable para la validación de modelos computacionales, el diseño de componentes y la investigación avanzada de materiales.

1.2. Objetivos

El principal objetivo de este trabajo es desarrollar un sistema para estudiar deformaciones en ensayos de tracción. Este objetivo global se descompone en los siguientes objetivos:

- Permitir el seguimiento de puntos característicos en vídeos de ensayos reales.
- Detectar el inicio de movimiento (t_0) y la rotura de la probeta sin intervención manual.
- Definir regiones de interés y distribuir puntos de seguimiento de forma flexible.
- Calcular desplazamientos y velocidades en función del ensayo.
- Exportar datos y vídeos procesados de forma clara y organizada.

1.3. Planteamiento del problema

En los ensayos de tracción con probetas, obtener información precisa y representativa del comportamiento del material bajo distintas condiciones supone un reto, especialmente cuando se pretende analizar zonas amplias de la muestra o procesar numerosos vídeos de forma automatizada. Las técnicas tradicionales, al estar limitadas a puntos concretos, no permiten observar de forma completa la distribución del movimiento.

Además, problemas como diferencias en calidad de los vídeos o la pérdida de puntos por ruido complican el análisis fiable de los resultados.

Por tanto, surge la necesidad de un sistema automatizado que permita cumplir con los objetivos definidos en el punto anterior. Todo ello con el fin de agilizar el análisis, mejorar la repetitividad y asegurar la fiabilidad de los datos obtenidos.

1.4. Estructura de la memoria

El presenta trabajo se divide en los siguientes siete capítulos:

- **Capítulo 1 - Introducción:** se introduce la temática de la memoria, se expone el problema global y se describen las motivaciones que han llevado a realizar el trabajo, así como los objetivos que se pretenden alcanzar.
- **Capítulo 2 – Contexto Tecnológico:** se explican algunos de los primeros avances de la Correlación de Imágenes Digitales DIC, los principales enfoques de este campo

que han surgido a lo largo de la historia y las tecnologías actuales, para dar una visión general de la evolución de esta tecnología. También se tratan avances en el flujo óptico y los métodos empleados para el análisis de desplazamientos.

- **Capítulo 3 – Procesamiento en Python:** se describe el entorno de desarrollo utilizado para la implementación del sistema, detallando las librerías empleadas y explicando su importancia en el tratamiento de imágenes y vídeos, el seguimiento de puntos y la automatización del análisis de datos. También se profundiza en la aplicación del algoritmo de flujo óptico sobre vídeos reales de ensayos.
- **Capítulo 4 – Desarrollo del Sistema:** es el núcleo del trabajo, donde se expone detalladamente cada etapa del sistema: desde la lectura de datos y el preprocesamiento de vídeos, hasta la definición de regiones de interés, el seguimiento punto a punto, la detección de eventos clave, y la exportación estructurada de resultados.
- **Capítulo 5 – Resultados y Validación:** se presentan los resultados obtenidos aplicando el sistema sobre los materiales y configuraciones de ensayo. Además, se analiza la validez del método, la calidad de los seguimientos y se muestran comparativas gráficas entre diversos ensayos.
- **Capítulo 6 – Conclusiones y Líneas Futuras:** se resumen los objetivos alcanzados, las principales aportaciones del sistema y se proponen posibles mejoras para trabajos futuros.

CAPÍTULO 2
CONTEXTO TECNOLÓGICO

Este capítulo presenta los avances más relevantes en el campo de la correlación de imágenes digitales y del flujo óptico. Se explica ampliamente qué es la tecnología DIC, sus inicios y su historia reciente, así como la diferencia que existe entre el uso de una (2D) o varias cámaras (3D). Por otro lado, en cuanto al flujo óptico, se detallan sus fundamentos y su importancia en la visión por computadora, analizando matemáticamente dos métodos (Lucas-Kanade, Horn-Shunck) sus diferencias.

2.1. Introducción

Durante los años 1960 y 1970, investigadores en el área de la inteligencia artificial y robótica comenzaron a desarrollar algoritmos en relación con la visión artificial, y metodologías de estéreo visión en paralelo con aplicaciones de fotogrametría. Las distintas áreas de interés en las que se centraron las primeras investigaciones del procesamiento de imágenes digitales fueron: medicina y radiología, microscopía, reconocimiento de caracteres y fotogrametría; sin embargo, no existían aplicaciones ingenieriles para medir deformaciones.

Uno de los primeros documentos en los que se propone la utilización de imágenes digitales para obtener las deformaciones fue publicado por Peter y Ranson en 1982 en la Universidad de Carolina del Sur **[1**,**2]**. El objetivo era medir deformaciones en el plano, gradientes de deformación e iniciación y propagación de grietas en materiales frágiles. En este estudio en concreto se propuso una técnica puramente planar, en la que una sola cámara permitía medir desplazamientos y deformaciones en el plano, pero no fuera de él.

Posteriormente, en 1983, Sutton et al. **[17]** mejoraron la técnica DIC para obtener las deformaciones en el plano de campo completo de una viga en voladizo. En el experimento, se colocó una muestra con un patrón aleatorio de moteado blanco sobre la superficie, perpendicular al eje óptico de una cámara de video. Durante el evento de inducción de deformación, se capturó una imagen en su estado no deformado, seguida de la captura continua de las imágenes deformadas posteriores. Con base en estas imágenes, los autores

sugirieron que la distribución de intensidad de la luz reflejada por la muestra puede almacenarse como un conjunto de niveles de gris en una computadora. Estos valores oscilaban entre 0 y 255, donde 0 representa la intensidad de luz cero y 255 representa la intensidad de luz máxima. Dado que los sensores registraban el patrón de intensidad continuamente variable en forma discreta, se aplicó un método de ajuste de superficie conocido como **interpolación bilineal** para representar los datos en forma continua. Para la determinación de los valores de desplazamiento utilizando la técnica DIC, introdujeron un coeficiente de correlación, C, expresado en la forma de:

$$C\left(u, v, \frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial x}\right) = \iint [f(x) - g(x')]^2 dx, \qquad (1)$$

Con posterioridad a este evento, se comenzaron a realizar numerosas investigaciones en relación con el algoritmo de optimización del método. A esta técnica del cálculo de deformaciones se le denomina hoy en día correlación de imágenes digitales (DIC) **[3**,**4]**.

Las primeras aplicaciones tridimensionales surgieron a principios de la década de 1990, cuando los principios de la visión estereoscópica se combinaron con los de la correlación de imágenes digitales en el plano **[5]**. Este sistema, que incorporaba dos cámaras digitales, permitía medir el estado tridimensional de deformación alrededor de una muesca.

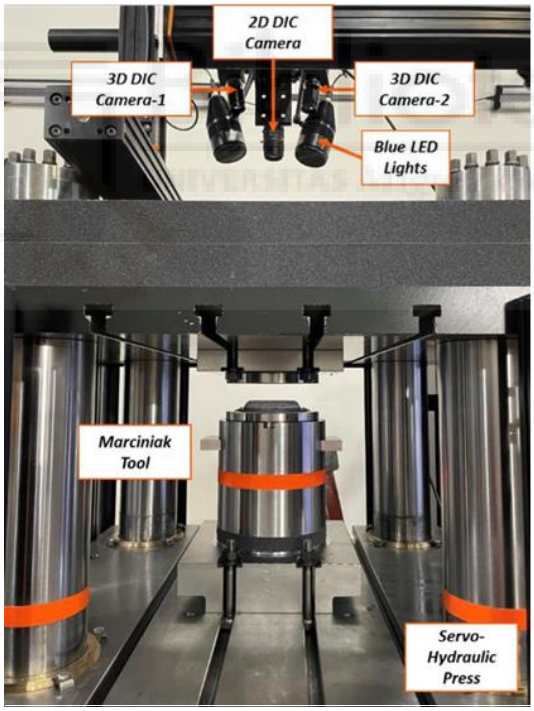
Las mejoras adicionales incluyeron el efecto de la perspectiva en la forma detectada de los subconjuntos de píxeles (facetas). También se incorporaron restricciones epipolares en el análisis para mejorar la precisión al considerar la correspondencia de elementos geométricos entre dos imágenes conjugadas. Un mejor proceso de reconstrucción de imágenes **[6]** mejoró aún más el cálculo de desplazamientos: se introdujeron splines de alto orden para interpolar los patrones de las imágenes. Esto permitió mejorar la precisión de las mediciones de desplazamiento hasta el orden de 0.01 píxeles.

La viabilidad práctica y la resolución espacial de la técnica mejoraron con el tiempo debido a: (i) el avance de las cámaras digitales; (ii) el aumento del poder de cómputo (los algoritmos de correlación de imágenes pueden ser bastante complejos cuando se procesan imágenes de alta resolución); y (iii) el desarrollo de algoritmos numéricos más precisos y eficientes.

La correlación de imágenes digitales como una técnica moderna de análisis de deformaciones fue descrita en el manual clásico de Dally y Riley **[7]**. Más recientemente, se ha dedicado un volumen completo a la teoría y métodos de correlación de imágenes **[8]**.

2.2. Tecnología DIC

La Correlación de Imágenes Digitales (DIC) se trata de una técnica de medición sin contacto (no invasiva) utilizada, entre otros, en el análisis de la deformación de materiales sometidos a diversos tipos de estudios mecánicos. Se fundamenta en el seguimiento de patrones aleatorios aplicados en la superficie de la muestra, habilitando la creación de mapas de deformación de campo completo con alta precisión. Desde sus inicios, ha evolucionado de forma significativa convirtiéndose en un estándar en la caracterización y clasificación de materiales en la industria automotriz, aeroespacial y manufacturera **[9]**.



2.2.1. Área de Cálculo (Faceta)

En DIC, el área de cálculo se divide en pequeñas subregiones llamadas facetas, que son conjuntos de píxeles utilizados para rastrear desplazamientos y deformaciones. Cada faceta

está asociada con un punto (generalmente su centro) para el cual se calculan los desplazamientos, cuanto mayor sea el número de facetas, mayor será la precisión del cálculo. Esto se hace debido a que es inútil buscar la correspondencia de un solo píxel en dos imágenes. De hecho, el valor de escala de grises de un solo píxel puede encontrarse en muchos otros píxeles de la segunda imagen.

Por lo general, se utilizan facetas cuadradas de NxN píxeles, donde N representa el tamaño de la faceta. Las ventajas de las facetas más grandes son: identificación robusta y mejor correlación en imágenes posteriores, medición más precisa de desplazamientos y deformaciones, y menos ruido en los cálculos. Sin embargo, pueden resultar en la pérdida de la información asociada con gradientes de deformación altos. Por otro lado, el costo computacional aumentar proporcionalmente con la extensión del área de cálculo (es decir, con el cuadrado del tamaño de la faceta, N^2). Las facetas más pequeñas permiten observar efector locales y gradientes. Las desventajas de las facetas más pequeñas son: un cálculo más ruidoso del patrón de deformación y un alto riesgo de falta de correlación en algunos dominios **[10]**.

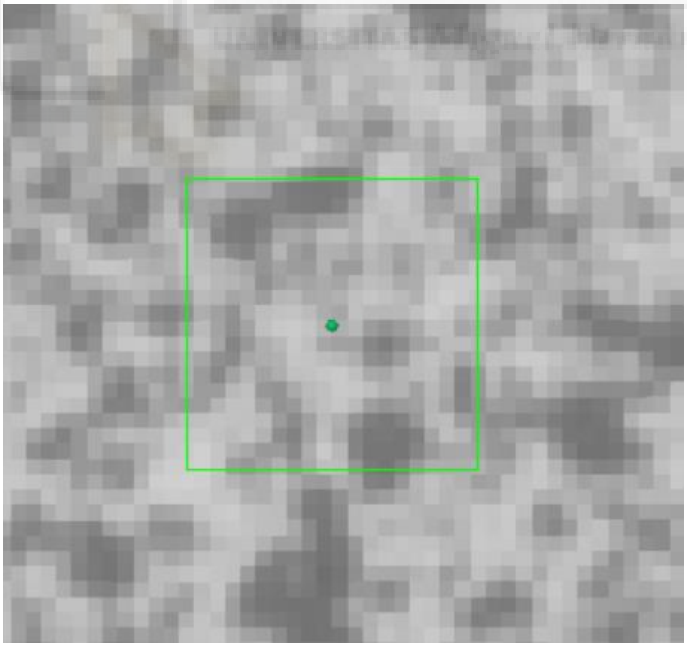


Figura 2.2. Faceta de tamaño 19 x 19 píxeles.

Con el fin de proporcionar cierta redundancia, las facetas se pueden superponer parcialmente por un número de píxeles M, como se muestra en la Figura 2.3. Una superposición entre facetas permite una estimación más robusta de los desplazamientos (a

un mayor coste computacional) cuando las imágenes originales están afectadas por ruido. La máxima superposición posible responde al valor de $M = N - 1$, donde se calcula el número máximo posible de datos para una resolución de imagen dada; esto claramente resuelta en el mayor costo computacional posible.

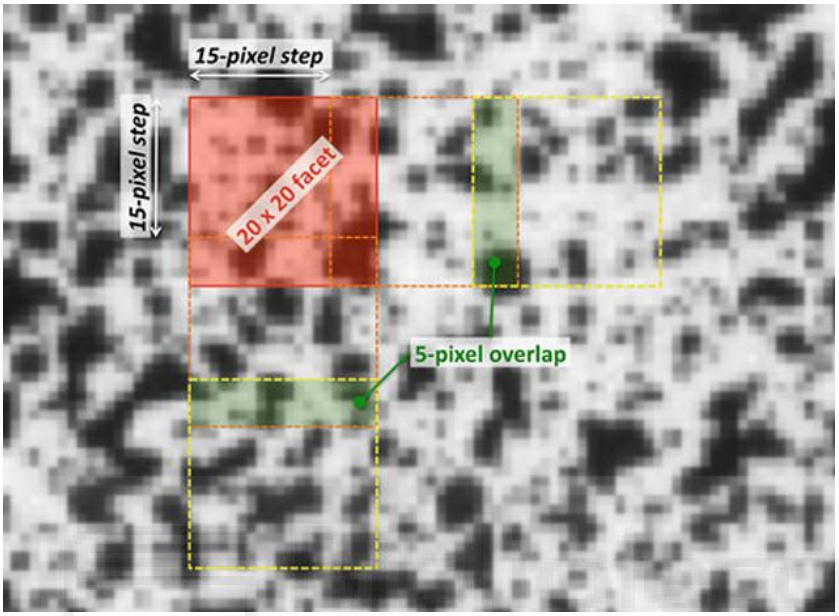


*Figura 2.3. Detalle de una porción de la superficie de las muestras con un patrón estocástico con M = 5 y un paso de 15 píxeles entre facetas **[15]**.*

2.2.2 El Patrón Estocástico (de Moteado)

Para que el método de Correlación Digital de Imágenes (DIC) funcione de manera efectiva y precisa, la superficie del espécimen debe presentar un patrón específico. De hecho, como se explicará en el siguiente apartado, la correlación de imágenes digitales aprovecha la reflexión no uniforme de la luz por parte de la superficie bajo observación. Por lo tanto, una superficie constante y uniforme no ofrecería características ni gradientes de luz para ser rastreados. Para permitir que el software identifique de manera unívoca las regiones en la superficie del espécimen, dicho patrón debe ser aleatorio, de modo que ninguna área se parezca a otra. Además, este patrón debe ser una parte integral del espécimen bajo observación (es decir, debe moverse y deformarse junto con el material subyacente).

En la mayoría de casos, el espécimen de prueba no presenta un patrón de moteado natural. Por esta razón, a menudo se pinta un patrón de moteado en la superficie del espécimen

utilizando una pistola de aerógrafo, aplicación de polvo de tóner o modificando la superficie del material mediante grabado o fotolitografía **[11]**. Cualquiera de estas opciones debe asegurar que la fracción de área cubierta por los moteados negros (o blancos) sea aproximadamente la misma que el área donde el fondo blanco (o negro) es visible, como se muestra en la Figura 2.4.

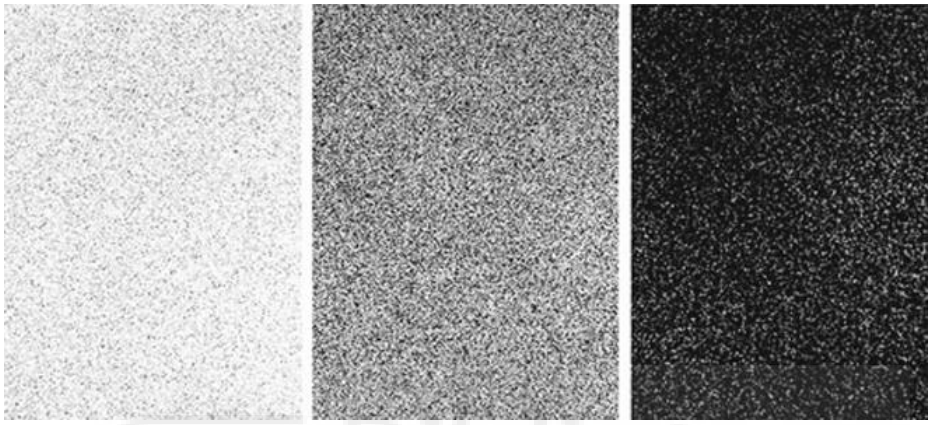
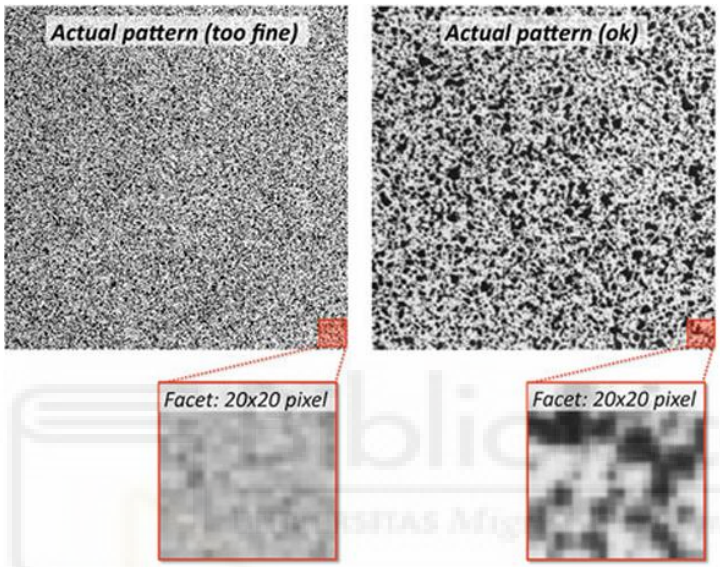


Figura 2.4. Ejemplos de Patrón Estocástico: 1º Demasiado claro; 2º Óptimo; 3º Demasiado Oscuro

Por último, se aplican restricciones al tamaño óptimo del moteado **[12-14]**, como se muestra en la Figura 2.5. Si un moteado es más pequeño que un píxel, no puede detectarse adecuadamente. Para sobremuestrear las características en cada patrón de moteado, el tamaño mínimo del moteado debe estar entre 3 y 5 píxeles.

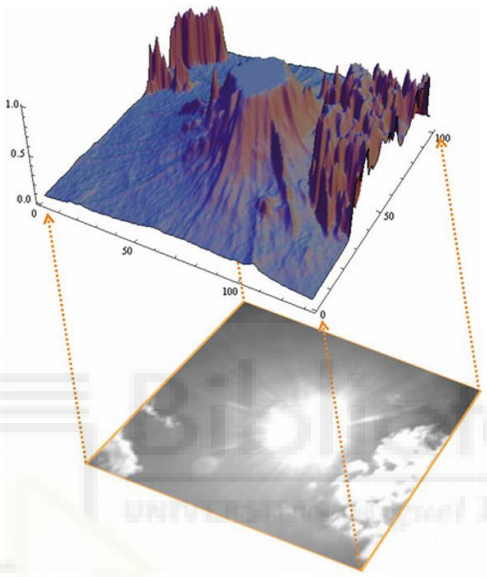
Por el contrario, si los moteados son más grandes que el tamaño mínimo recomendado, se requiere de facetas innecesariamente grandes para rastrearlos. En definitiva, un patrón ideal tiene un tamaño mínimo de moteado entre 3 y 5 píxeles (para ser capturado adecuadamente por la cámara) y una baja dispersión (para evitar moteados excesivamente grandes). Queda claro, pues, el por qué las cámaras de alta resolución pueden ayudar a mejorar la resolución y precisión de las aplicaciones DIC.



2.2.3. Correlación de Imágenes 2D

Aunque el estado del arte de la Correlación Digital de Imágenes (DIC) suele implementarse en su versión tridimensional (es decir, utilizando dos cámaras, lo que permite la reconstrucción espacial completa de los desplazamientos y deformaciones), el principio de funcionamiento del DIC es más fácil de entender comenzando con su versión bidimensional. La versión 2D del DIC utiliza una sola cámara y permite medir los desplazamientos y deformaciones en un plano que es perpendicular a la dirección de observación.

Como se mencionó en el apartado 2.2.1 del presente capítulo, cada faceta incluye un subconjunto finito de píxeles. Cada píxel está asociado con un valor de escala de grises (las imágenes a color no suelen ofrecer ninguna ventaja). En el caso de que la cámara utilizada opere en una base de 8 bits, cada píxel se asigna un valor entre 0 y 255. La Figura 2.6 muestra como cada faceta se describe por una intensidad de luz promedio de escala de grises y por una variación interna bidimensional **[15]**.



Con suerte, este valor promedio de escala de grises y su variabilidad son únicos para cada faceta gracias a la naturaleza aleatoria del patrón estocástico (de moteado) en la superficie del espécimen. Esto permite identificar y rastrear de manera unívoca cada porción de la superficie de la muestra a lo largo de los fotogramas.

Para determinar el desplazamiento promedio en el plano de la muestra, se utilizan funciones de mapeo o funciones de forma **[8]** para localizar un subconjunto inicialmente cuadrado en la imagen de referencia dentro de la siguiente imagen capturada bajo condiciones de carga. Para la traslación de cuerpo rígido, se utilizan funciones de forma de orden cero para representar la traslación de cada punto dentro del subconjunto seleccionado en las direcciones x e y. Sin embargo, las funciones de forma de orden cero no son adecuadas para representar los subconjuntos que están experimentando una combinación de traslación,

rotación, deformación normal y deformación por cortante. Por lo tanto, se utilizan funciones de forma de primer orden y se expresan en la forma de:

$$\zeta_1 = u + \frac{\partial u}{\partial x} \cdot \Delta x + \frac{\partial u}{\partial y} \cdot \Delta y \tag{2}$$

$$\eta_1 = v + \frac{\partial v}{\partial x} \cdot \Delta x + \frac{\partial v}{\partial y} \cdot \Delta y \tag{3}$$

Donde: ζ_1 y η_1 son los desplazamientos totales del subconjunto; u y v son las traslaciones; $\frac{\partial u}{\partial x}$ y $\frac{\partial v}{\partial y}$ son las deformaciones normales; $\frac{\partial u}{\partial y}$ y $\frac{\partial v}{\partial x}$ son las deformaciones por cortante; Δx y Δy son las distancias desde el centro del subconjunto hasta un punto arbitrario dentro del mismo subconjunto en las direcciones x e y, respectivamente. La Figura 2.7 muestra los parámetros de deformación utilizados para representar el desplazamiento promedio en el plano del subconjunto **[16]**.

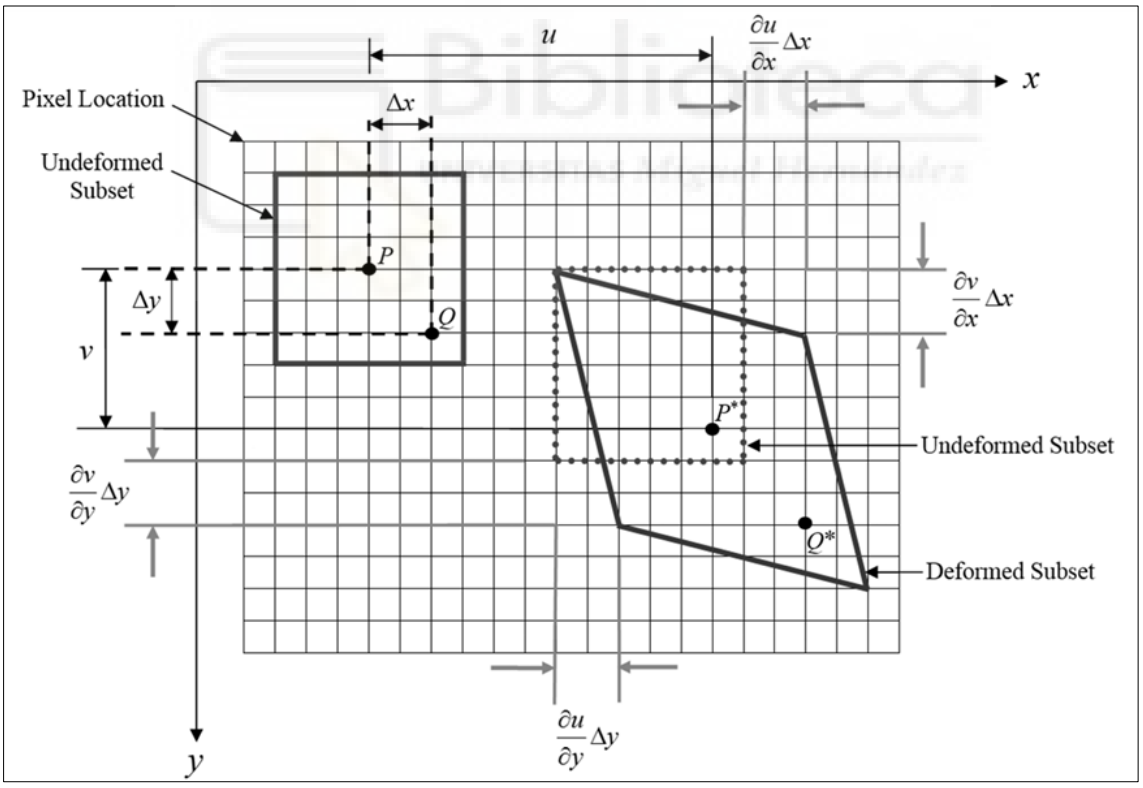


*Figura 2.7. Parámetros de deformación utilizados para representar el desplazamiento promedio en el plano de un subconjunto **[16]**.*

Este método, basado en funciones de forma de primer orden es más robusto y preciso que los enfoques iniciales de interpolación, lo que lo hace esencial para aplicaciones donde se necesita una caracterización detallada del campo de deformaciones. Sin embargo, su implementación requiere una mayor capacidad de procesamiento y un ajuste cuidadoso de los parámetros para garantizar resultados óptimos.

2.2.4. Correlación de Imágenes 3D

Un sistema DIC que consta de dos cámaras permite medir los desplazamientos en tres dimensiones (incluyendo aquellos fuera del plano, en la dirección de las cámaras). Esto permite medir el estado espacial de la deformación, lo cual es particularmente ventajoso para especímenes con geometría compleja.

El primer paso de la correlación de imágenes en 3D consiste en **emparejar las imágenes de dos cámaras**, aprovechando la visión estereoscópica para obtener una descripción tridimensional de la superficie visible del espécimen. De hecho, mientras que una sola cámara permite medir los desplazamientos solo en un plano perpendicular a la dirección de observación, los desplazamientos hacia la cámara (o alejándose de ella) pueden detectarse comparando imágenes de dos cámaras. Para que la visión estereoscópica sea efectiva, la distancia entre las dos cámaras debe ser suficiente, de modo que las direcciones de observación de un mismo punto formen un ángulo de al menos 20º o 25º.

Para un sistema estereoscópico calibrado, se puede utilizar un enfoque optimizado llamado **rectificación**. Basándose en la función de calibración del sistema de cámaras y lentes, las imágenes emparejadas pueden transformarse (mediante transformaciones homográficas) de modo que todas las líneas epipolares se vuelvan horizontales. Por lo tanto, cada punto transformado correspondiente a un punto dado de la superficie del espécimen se encuentra en la misma línea de escaneo en ambas imágenes estereoscópicas emparejadas.

La **correspondencia estereoscópica** es la siguiente fase del procesamiento de imágenes estereoscópicas emparejadas. Consiste en la identificación de los puntos correspondientes en las imágenes emparejadas. En esta fase, se calcula la profundidad de la imagen, es decir, la distancia de cada punto con respecto al sistema de cámaras. Una vez que se ha construido la superficie tridimensional completa del objeto a partir de la correspondencia estereoscópica, se dispone de una descripción tridimensional de la superficie, donde cada punto está asociado con su valor de escala de grises. En esta etapa, se pueden aplicar los algoritmos de correlación descritos anteriormente para el problema plano.

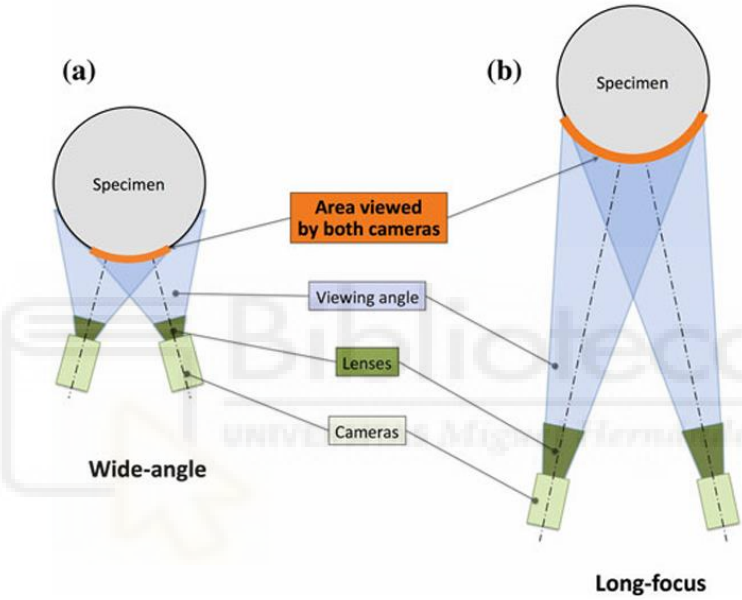


*Figura 2.8. La proporción de espécimen que pueden visualizar ambas cámaras de un sistema DIC 3D depende de la distancia entre el DIC y la muestra, así como de la distancia focal del objetivo **[15]**.*

2.3. Estimación del Flujo Óptico

El flujo óptico es un concepto fundamental en el campo de la visión por computadora que describe la distribución de las velocidades aparentes de los patrones de brillo en una secuencia de imágenes. Este fenómeno surge debido al movimiento de objetos en el mundo real (3D), que induce un movimiento aparente en el plano de las imágenes (2D). Los vectores de velocidad que representan este movimiento se denominan “aparentes” porque pueden originarse tanto por el movimiento del objeto como por el movimiento de la cámara, incluso si el objeto está estático **[18]**. Cuando no hay variaciones significativas en la fuente de luz, el cambio en la posición de un patrón de brillo entre dos o más *frames* consecutivos de un vídeo

puede interpretarse como el movimiento del objeto en la secuencia de imágenes. Este concepto se basa en la detección de bordes y en la definición de movimiento en vídeo, y su análisis requiere la aplicación de restricciones específicas para interpretar correctamente los cambios en los patrones de brillo.

La determinación del flujo óptico es de gran importancia en el análisis de video, ya que los vectores de velocidad calculados proporcionan información sobre el cambio en la disposición espacial de uno o varios objetos de estudio **[18]**. Esto permite identificar regiones con discontinuidades y segmentar la imagen, lo que facilita la delimitación entre objetos y su fondo. Por ejemplo, en aplicaciones prácticas, como el análisis del tráfico, el flujo óptico permite diferenciar automóviles de su entorno y entre sí **[19]**. Además, el flujo óptico ha sido ampliamente utilizado en aplicaciones como la estabilización de vídeo, la compresión de vídeo, el resumen de secuencias y en campos más especializados como la teledetección y las imágenes médicas **[20]**.

Como ya se ha comentado, esta herramienta permite estimar el movimiento independiente de cada píxel entre dos imágenes consecutivas **[19]**. Este concepto está basado en el supuesto que cada píxel que se mueve entre dos imágenes consecutivas mantiene su intensidad aproximadamente constante.

$$I(x+u, y+v, t+1) = I(x, y, t) \tag{4}$$

Si se presentan pequeños desplazamientos esta expresión puede linealizarse aplicando series de Taylor, lo que lleva a:

$$\frac{\partial I(x,y,t)}{\partial x} v_x + \frac{\partial I(x,y,t)}{\partial y} v_y + \frac{\partial I(x,y,t)}{\partial t} = 0 \tag{5}$$

Donde u y v corresponde al flujo óptico y no es más que un campo de desplazamiento de píxeles. El problema con (4) es que existen múltiples soluciones para u y v. En este sentido surgen diversos métodos con el fin de proporcionar una mejora este inconveniente. En este capítulo se estudiarán dos:

i) Algoritmo de Lucas-Kanade.

ii) Método de Horn-Shunck.

***a**. Imagen en escala de grises*

2.3.1. Algoritmo de Lucas-Kanade

El algoritmo desarrollado por Lucas y Kanade aplica una restricción de suavidad local (diseñada para manejar vecindarios pequeños) utilizando un ajuste de mínimos cuadrados en una pequeña ventana de la imagen para el análisis de la traslación del objeto en cuestión **[22]**. Los métodos locales utilizan la información en una vecindad alrededor de un píxel para estimar su movimiento, como el desplazamiento que contienen las imágenes entre dos *frames* cercanos es pequeño y aproximadamente constante dentro de un entorno del punto p en consideración calcula el flujo alrededor de una ventana centrado en el píxel p.

El método de Lucas-Kanade está basado en los siguientes supuestos:

- Las dos imágenes consecutivas se encuentran separadas por un pequeño cambio de tiempo y los objetos mostrados en las mismas presentan pequeños desplazamientos.
- Los objetos mostrados en las imágenes presentan diferentes niveles de gris, que cambian suavemente.

Bajo estos supuestos, Lucas y Kanade pueden asumir que un píxel de intensidad $I(x, y)$ comparte la misma intensidad que los píxeles vecinos y, además, el flujo de imagen v_x, v_y es muy pequeño entre dos instantes cercanos de la imagen para que la ecuación diferencial del flujo óptico pueda mantenerse. Bajo esta premisa a (5) ahora se le añaden n ecuaciones correspondientes a sus n píxeles vecinos.

$$\begin{aligned} I_{x1}v_x + I_{y1}v_y &= -I_{t1} \\ I_{x2}v_x + I_{y2}v_y &= -I_{t2} \\ &\vdots \\ I_{x(n+1)}v_x + I_{y(n+1)}v_y &= -I_{t(n+1)} \end{aligned} \tag{6}$$

Donde I_x, I_y y I_t son las derivadas parciales de la imagen con respecto a la posición (5). Por otro lado, $1, 2, \dots, n+1$ representa los píxeles vecinos al central p. Reescribiendo estas ecuaciones en forma matricial son representadas como $Av = b$, donde:

$$A = \begin{bmatrix} I_{x_1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \\ I_{x(n+1)} & I_{y(n+1)} \end{bmatrix} \tag{7}$$

$$v = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{8}$$

$$b = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{t(n+1)} \end{bmatrix} \tag{9}$$

Como podemos comprobar, tenemos dos incógnitas (v_x, v_y) para n ecuaciones. El sistema será sobredeterminado, pues para que no lo fuera, necesitaríamos exactamente 2 ecuaciones (es decir, 2 píxeles). Sin embargo, no es posible utilizar solo 2 píxeles porque, al estudiar los puntos vecinos alrededor de un punto central formando una ventana cuadrada, el mínimo de píxeles que puede contener la ventana es de 8 (en contacto con el píxel central). Por lo tanto, el mínimo de píxeles para una ventana cuadrada es 9 (3x3), lo que garantiza un sistema sobredeterminado.

Para alcanzar una solución robusta y precisa al ruido se aplican métodos de mínimos cuadrados. $v = (A^T A_v)^{-1} A^T b$, donde A^T es la matriz transpuesta de A.

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x^2 & \sum_i I_x I_y \\ \sum_i I_y I_x & \sum_i I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x I_t \\ -\sum_i I_y I_t \end{bmatrix} \tag{10}$$

Para la solución de mínimos cuadrados, se realiza una versión ponderada donde:

$$v = (A^T W A)^{-1} A^T W b \tag{11}$$

Donde W es la ventana centrada en el píxel en (x, y), y se encuentra en una matriz diagonal de tamaño nxn.

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x^2 & \sum_i w_i I_x I_y \\ \sum_i w_i I_y I_x & \sum_i w_i I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x I_t \\ -\sum_i w_i I_y I_t \end{bmatrix} \tag{12}$$

El tamaño de w_i es generalmente ajustado con una función de tipo gaussiana de la distancia correspondiente entre el píxel central p y el píxel vecino p_i. Finalmente, la ecuación queda como:

$$\sum_{(x,y)\epsilon N} W^2(x,y)\left(I_x u + I_y v + I_t\right)^2 \tag{13}$$

Este algoritmo en particular es el preferido por la mayoría de los investigadores hoy en día y es el que se va a implementar en este trabajo.

2.3.2. Método de Horn-Shunck

Con el fin de solventar el problema de (5), Horn y Shunck proponen minimizar la integral, un método que introduce una restricción global de la suavidad. Con este tipo de método se obtienen campos de desplazamiento densos, usan directamente la ecuación (14) y añaden como restricción global un término de regularización sobre el flujo.

$$I_x u + I_y v + I_t = 0 \tag{14}$$

El problema consiste en tratar de minimizar la suma de los errores en la ecuación para la tasa de cambio de brillo de la imagen (15) y la medida de la salida de la suavidad en el flujo de velocidad (16).

$$\varepsilon_b = E_x u + E_y v + E_t \tag{15}$$

$$\varepsilon_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{16}$$

Donde E_x, E_y y E_t son las derivadas parciales. Por otro lado, α es un factor de regulación que ayuda a suavizar el fujo óptico y evitar soluciones extremas o ruidosas, de modo que se puede expresar el error minimizado como:

$$\varepsilon^2 = \iint (\alpha^2 \varepsilon_c^2 + \varepsilon_b^2) dx dy \tag{17}$$

La minimización se logra con la búsqueda de los valores adecuados para la velocidad de flujo óptico (u, v). Usando el cálculo de la variación se obtiene que:

$$I_x^2 u + I_x I_y v = \alpha^2 \nabla^2 u - I_x I_t \tag{18}$$

$$I_x I_y u + I_y^2 v = \alpha^2 \nabla^2 u - I_y I_t \tag{19}$$

Por medio de la aproximación laplaciana para (u, v), una conveniente aproximación es de la forma: (donde $\bar{u}$ y $\bar{v}$ son las componentes de velocidad de flujo promedio en las direcciones x e y. Representan una estimación inicial o promedio del movimiento)

$$\nabla^2 u \approx k(\bar{u}_{i,j,k} - u_{i,j,k}) \tag{20}$$

$$\nabla^2 v \approx k(\bar{v}_{i,j,k} - v_{i,j,k}) \tag{21}$$

Utilizando esta aproximación se obtiene que:

$$(\alpha^2 + I_x^2) u + I_x I_y v = (\alpha^2 \bar{u} - I_x I_t) \tag{22}$$

$$I_x I_y u + (\alpha^2 + I_y^2) v = (\alpha^2 \bar{v} - I_y I_t) \tag{23}$$

El determinante de la matriz de coeficientes igual a $\alpha^2(\alpha^2 + I_x^2 I_y^2)$, se soluciona para (u, v), y se obtiene que:

$$\left(\alpha^2 + I_x^2 + I_y^2\right) u = +\left(\alpha^2 + I_y^2\right) \bar{u} - I_x I_y \bar{v} - I_x I_t \tag{24}$$

$$\left(\alpha^2 + I_x^2 + I_y^2\right) v = -I_x I_y \bar{u} + (\alpha^2 + I_x^2) \bar{v} - I_y I_t \tag{25}$$

Reescribiendo las ecuaciones (24) y (25) se tiene:

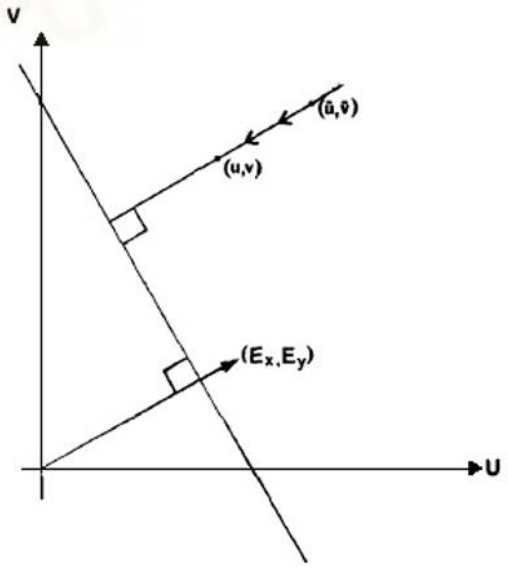
$$\left(\alpha^2 + I_x^2 + I_y^2\right)(u - \bar{u}) = -I_x [I_x \bar{u} + I_y \bar{v} + I_t] \tag{26}$$

$$\left(\alpha^2 + I_x^2 + I_y^2\right)(v - \bar{v}) = -I_y[I_x\,\bar{u} + I_y\,\bar{v} + I_t] \qquad (27)$$

Las ecuaciones (24) y (25) indican cómo se calculan las componentes de la velocidad del flujo (u, v) basándose en las derivadas de la intensidad de la imagen y en las velocidades promedio $\bar{u}$ y $\bar{v}$. Por otro lado, las ecuaciones (26) y (27) muestran cómo la diferencia entre la velocidad de flujo calculada y la velocidad promedio está relacionada con las derivadas de la intensidad de la imagen. Esto indica que el flujo óptico minimiza el error en la estimación del movimiento.

La Figura 2.10 muestra el concepto de la *línea de restricción* en el contexto del flujo óptico. La línea de restricción representa la dirección en la que el movimiento de los píxeles es consistente con el cambio de intensidad de la imagen. La distancia entre el punto estimado (u, v), y la línea de restricción es proporcional al error en la estimación del cambio de brillo.

Es decir, vemos como el flujo óptico calculado tiende a alinearse con la dirección que minimiza el error en la estimación del movimiento, basándose en los cambios de intensidad de la imagen.



Este método funciona bien en presencia de flujo óptico denso. No obstante, no produce buenos resultados en condiciones de mucho ruido. Además, presenta el problema que demanda una alta carga computacional al tener que aplicar in método iterativo de Gauss-Seidel. El algoritmo presentado por Lucas-Kanade introduce otra alternativa computacionalmente más eficiente y de mayor solidez frente al ruido, presente a lo largo de este trabajo.

CAPÍTULO 3

PROCESAMIENTO EN PYTHON

Este capítulo describe el uso de Python y sus librerías especializadas para el análisis de imágenes y videos en ensayos de tracción. Además, el capítulo cubre el procesamiento de imágenes, donde se detallan técnicas como el procesamiento de vídeos, enfocado en la extracción y análisis secuencial de *frames* para medir deformaciones y desplazamientos durante los ensayos.

La combinación de estas herramientas permitió automatizar el análisis, garantizando precisión y eficiencia en los resultados.

3.1. Python

Python **[24]** es un lenguaje de programación creado por Guido Van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses "*Monty Python*". Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipado dinámico, multiplataforma y orientado a objetos.

El uso de Python en este proyecto responde a su versatilidad y potencia en el procesamiento de imágenes y videos, así como en el manejo de datos y la amplia variedad de librerías robustas que ofrece. Sin embargo, en los inicios de este trabajo se trató de analizar los vídeos utilizando Zeiss Inspect Correlate, un software dedicado a la correlación de imágenes digitales. Sin embargo, debido a que la resolución de los vídeos grabados por nuestra cámara no era alta, la creación de superficie en el programa no se ejecutaba correctamente, resultando en una gran cantidad de problemas y limitaciones. A pesar de ser una herramienta potente no permitía establecer ROIs fijas y estudiar únicamente la posición de los puntos que se encontraban dentro de la región. Finalmente, se decidió crear un programa de código abierto que realizara una tarea similar utilizando el Flujo Óptico con el

algoritmo de Lucas-Kanade, que cubriese las necesidades del proyecto y se adaptase a la calidad de los vídeos.

Gracias a las librerías especializadas que hay en Python, se ha podido desarrollar un sistema eficiente y automatizado para el análisis de ensayos de tracción, optimizando la detección y seguimiento de puntos de referencia, el procesamiento de *frames* y la extracción de datos relevantes. La combinación de herramientas como OpenCV, NumPy y Pandas ha permitido implementar algoritmos avanzados con un flujo de trabajo estructurado y preciso.

3.2. Librerías y Funciones Utilizadas

Para el procesamiento y análisis de los ensayos de tracción en video, se han utilizado diversas librerías de Python especializadas en el tratamiento de imágenes y datos. Entre ellas, destacan OpenCV, Numpy, Pandas y Matplotlib.

3.2.1. OpenCV

OpenCV **[25]** es una biblioteca de visión artificial desarrollada por la empresa Intel. Se centra en el procesamiento de imagen en tiempo real, es de código abierto y multiplataforma, ya que puede usarse en MAC OSX, Windows y Linux, conteniendo más de 500 funciones que engloban una gran gama de áreas de proceso de visión: reconocimiento facial, calibración de cámaras, visión robótica… Su primera aparición en versión Alpha tuvo lugar en 1999 y ha tenido una gran evolución usándose en la actualidad en un gran abanico de aplicaciones.

Forma parte de los cimientos de este proyecto, utilizándose para la lectura y manipulación de *frames*, seguimiento de puntos de referencia mediante el algoritmo de Lucas-Kanade, así como para la aplicación de transformaciones en las imágenes.

3.2.2. Numpy

Numpy **[26]** es un proyecto (biblioteca) de código abierto que tiene como objetivo permitir la computación numérica con Python. Fue creado en 2005, basándose en el trabajo inicial de las bibliotecas Numerical y Numarray. Numpy es un software 100% de código abierto, de

uso gratuito para todos y publicado bajo los términos liberales de la licencia BSD modificada y se desarrolla abiertamente en GitHub, a través del consenso de Numpy y de la comunidad científica de Python en general.

Esta librería se ha utilizado para la manipulación eficiente de matrices y arrays, permitiendo realizar operaciones realizar operaciones matemáticas y de indexación necesarias para el cálculo de desplazamientos y transformaciones geométricas.

3.2.3. Pandas

La biblioteca pandas **[27]**, en desarrollo desde 2008, tiene como objetivo cerrar la brecha en la riqueza de herramientas disponibles para el análisis de datos entre Python y las numerosas plataformas de computación estadística y lenguajes de base de datos específicos de dominio. No solo buscamos ofrecer funcionalidades equivalentes, sino también implementar características como alineación automática de datos e indexación jerárquica, que no están fácilmente disponibles de manera tan integrada en otras bibliotecas o entornos de computación.

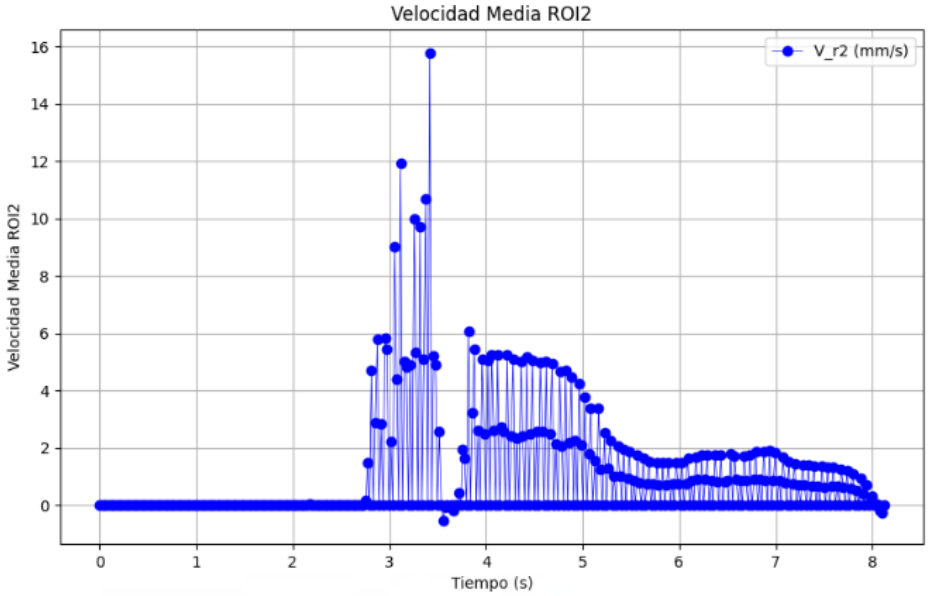
Aunque inicialmente se desarrolló para aplicaciones de análisis de datos financieros, se espera que pandas convierta a Python científico en un entorno de computación estadística más atractivo y práctico tanto para académicos como para profesionales de la industria. El nombre de la biblioteca proviene del término *panel data* (datos de panel), común en estadística y econometría para referirse a conjuntos de datos multidimensionales.

En este proyecto ha sido empleada en la gestión y estructuración de los datos extraídos de los ensayos. Ha facilitado la lectura y escritura de escritura de archivos Excel, asegurando así una organización eficiente de la información obtenida.

3.2.4. Matplotlib

Matplotlib **[28]** es una librería de open source y gratuita para el trazado de gráficas utilizando datos pertenecientes a listas o arrays en leguaje Python y su extensión matemática Numpy. Su integración en Python proporciona un entorno interactivo de investigación y

desarrollo que incluye investigación de datos, adecuado para la mayoría de los usuarios. Proporciona una API diseñada para recordar a Matlab (de ahí su nombre).



Ha sido utilizada para la visualización de resultados, permitiendo graficar desplazamientos, velocidades y otros parámetros obtenidos del análisis de los videos.

3.2. Procesamiento de Imágenes y Vídeos

El procesamiento de imágenes y vídeos mediante la librería de OpenCV en Python se basa en el manejo digital de imágenes, las cuáles se representan numéricamente como matrices multidimensionales de valores denominados píxeles. Para las imágenes a color, la librería emplea comúnmente el modelo BGR (Blue, Green and Red), almacenando estas imágenes en matrices tridimensionales con tres canales, donde cada uno corresponde a un color básico.

Por otro lado, una imagen a escala de grises es almacenada en una matriz bidimensional, donde cada elemento contiene un valor numérico que representa la intensidad luminosa del píxel correspondiente.

En este contexto, es importante destacar que las coordenadas de píxel dentro de una imagen o ventana de visualización siguen una convención matricial estándar, en la que el origen se sitúa en la esquina superior izquierda. Es decir, el punto (0,0) representa el píxel de

la esquina superior izquierda, donde el primer valor corresponde a la posición horizontal (eje X) y el segundo a la posición vertical (eje Y).

En el caso concreto de este proyecto, todas las ventanas emergentes utilizadas para la interacción con el usuario (selección de puntos, eje X, escala, etc.) fueron redimensionadas a un tamaño de 800 x 600 píxeles. Por tanto, las coordenadas válidas abarcan desde:

- X: 0 (izquierda) hasta 799 (derecha).
- Y: 0 (parte superior) hasta 599 (parte inferior). Cabe destacar que en la correlación de imágenes digitales donde se aplica el flujo óptico, es preferible trabajar con imágenes en escala de grises como la que se muestra en la Figura 3.3.

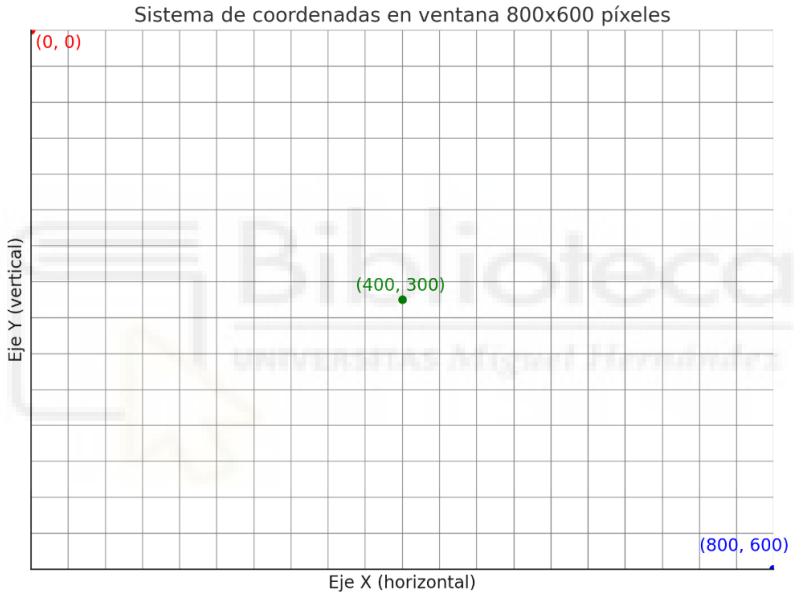


Figura 3.2. Representación del sistema de coordenadas en la ventana de análisis (800x600 píxeles).

Figura 3.3. Probeta BAO-01 a escala de grises.

Por otro lado, el procesamiento de vídeos se realiza secuencialmente, trabajando con imágenes individuales conocidas como fotogramas o frames. Cada uno es leído desde el archivo de vídeo. Posteriormente, cada frame obtenido es susceptible de diversos tratamientos, así como la conversión a escala de grises, redimensionamiento, filtrado, extracción de características, entre otros procesos.

```
video = cv2.VideoCapture(video_path)

# Verify if the video loaded successfully
if not video.isOpened():
    print("Error: no se pudo cargar el video. Verificar ruta de archivo.")
    exit()
else:
    print(f"Video abierto correctamente: {video_path}")

# Get video properties (to work with time, etc.)
fps = int(video.get(cv.CAP_PROP_FPS))  # Frames per second
total_frames = int(video.get(cv.CAP_PROP_FRAME_COUNT))  # Total number of frames
duration = total_frames / fps  # Video duration (in seconds)
print(f"Duración del video: {duration:.2f} segundos, FPS: {fps}")

# Read the first frame to select the point and calculate scale and angle
ret, frame = video.read()
if not ret:
    print("Error al leer el primer frame.")
    exit()
else:
    # Resize the frame to fit the screen
    frame_resized = cv2.resize(frame, (800, 600))
```

Figura 3.4. Main. Procesamiento de un vídeo. Primer frame.

3.3. Calc0pticalFlowPyrLK

La función ***cv2.calcOpticalFlowPyrLK*** es una implementación del método de Lucas-Kanade para el cálculo del flujo óptico entre dos imágenes consecutivas en una secuencia de vídeo. Este algoritmo permite estimar el movimiento de puntos específicos entre dos fotogramas consecutivos.

El algoritmo se basa en la suposición de que la intensidad de los píxeles no varía entre dos fotogramas cercanos en el tiempo y que el movimiento es pequeño.

A partir de esas premisas, Lucas y Kanade propusieron una solución de mínimos cuadrados local para determinar el desplazamiento de píxeles (Capítulo 2.3.1). Esta

implementación en OpenCV se optimiza utilizando una pirámide de imágenes, que permite detectar el movimiento tanto de objetos lentos como rápidos.

```
cv2.calcOpticalFlowPyrLK(prevImg, nextImg, prevPts, winSize, maxLevel[, criteria[, flags[, minEigThreshold]]]) → nextPts, status, err
```

Parámetros de entrada

- **prevImg**: Imagen anterior (grayscale).
- **nextImg**: Imagen siguiente (grayscale).
- **prevPts**: Array de puntos a seguir en la imagen anterior. Debe ser una array de tipo np.float32 con forma (N, 1, 2).
- **winSize**: Tamaño de la ventana de búsqueda en cada nivel de la pirámide, en nuestro caso (800, 600).
- **maxLevel**: Número de niveles de la pirámide, 0 significa que no se usa pirámide.
- **criteria**: criterio de parada del algoritmo.
- **flags**: Parámetros opcionales que modifican el comportamiento (por defecto 0).
- **minEigThreshold**: Umbral mínimo de autovalor para descartar puntos con poca textura.

Parámetros de salida

- **nextPts**: Coordenadas estimadas de los puntos en la siguiente imagen (pix.).
- **status**: Array binario (0 o 1), indica si el flujo fue encontrado para ese punto.
- **err**: Error asociado al cálculo de cada punto (pix.). Cada valor representa típicamente una medida de la calidad del ajuste en función del residuo de mínimos cuadrados entre pirámides.

Es importante comprender cómo se aplica esta función en un caso real, como el de los ensayos realizados en este proyecto. El flujo de trabajo para conectar los vídeos reales con la extracción de información significativa mediante esta función puede resumirse en los siguientes pasos:

1. Captura y procesado del vídeo en frames.

2. Selección de puntos de interés (prevPts).
3. Cálculo del flujo óptico.
4. Análisis de desplazamientos.
5. Repetición sobre la secuencia completa.

CAPÍTULO 4

DESARROLLO DEL SISTEMA

En este capítulo se describe el desarrollo completo del sistema implementado para el análisis automatizado de ensayos de tracción. Se abordan desde la adquisición y preprocesamiento de vídeos, hasta la selección de parámetros clave como el eje de referencia y la escala. También se detallan los métodos de creación de regiones de interés (ROIs), la distribución de puntos de seguimiento y el uso del algoritmo de Lucas-Kanade para calcular desplazamientos y velocidades. Finalmente, se presentan mecanismos de detección de rotura, identificación de inicio de movimiento y exportación de resultados.

4.1. Descripción de los ensayos reales

Los ensayos realizados en este trabajo tienen como objetivo estudiar el comportamiento de distintos materiales cuando son sometidos a una tracción controlada sobre un pin cilíndrico, con un cierto ángulo de contacto. Este tipo de ensayos reciben el nombre de **Bending Under Tension** (BUT) y permite analizar la interacción entre el material y el pin, así como las posibles deformaciones, desplazamientos y comportamientos tribológicos (como adherencia o fricción), bajo diferentes condiciones de carga y geometría.

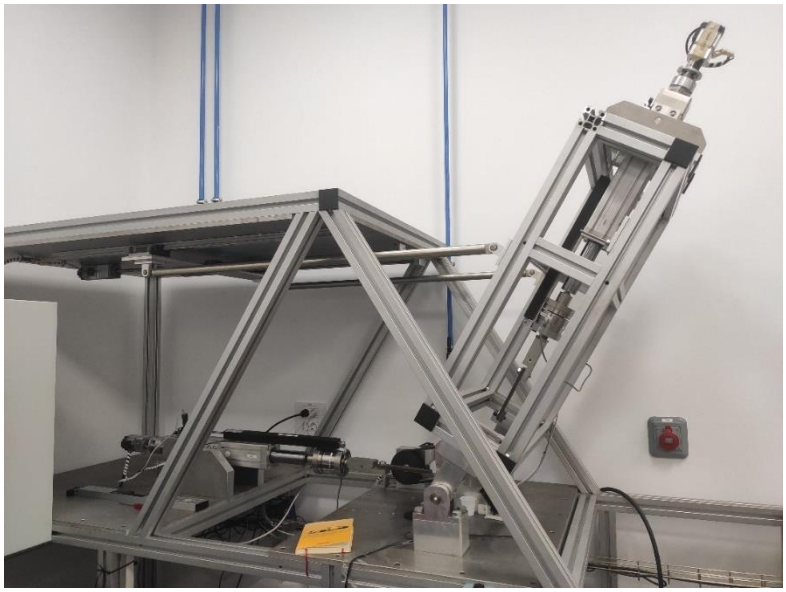


Figura 4.1. Vista general del banco de ensayos de tracción con ángulo variable

El objetivo fundamental es analizar la respuesta de los materiales BAO y TRIP_590 cuando son traccionados sobre un pin fijo o móvil, evaluando cómo distintos factores influyen en su comportamiento. Los datos obtenidos permitirán caracterizar la interacción material-pin y extraer conclusiones útiles para la industria automotriz.

Entre los factores clave que se estudian se encuentran:

- El **ángulo** de contacto entre el material y el pin (30º, 45º o 60º).
- La **velocidad** de tracción, que afecta directamente al comportamiento dinámico (3, 7 o 15 mm/s).
- La **temperatura** del material durante el ensayo (25 o 150 ºC).
- La **rotación** del pin, para estudiar condiciones más complejas de contacto (0 o 5, fijo o libre).
- El tipo de **lubricante** entre probeta y pin (Ninguno, Líquido o Sólido).
- Las **propiedades** del material sometido a ensayo (elasticidad, rigidez, etc).

Para llevar a cabo los ensayos se utilizó un sistema en el que el material se desplaza longitudinalmente mientras mantiene contacto con un pin cilíndrico. El extremo izquierdo de la probeta se mantiene unido a un cilindro fijo mientras que el extremo derecho es traccionado por un cilindro rotacional. La Figura 4.1 muestra el sistema real utilizado.

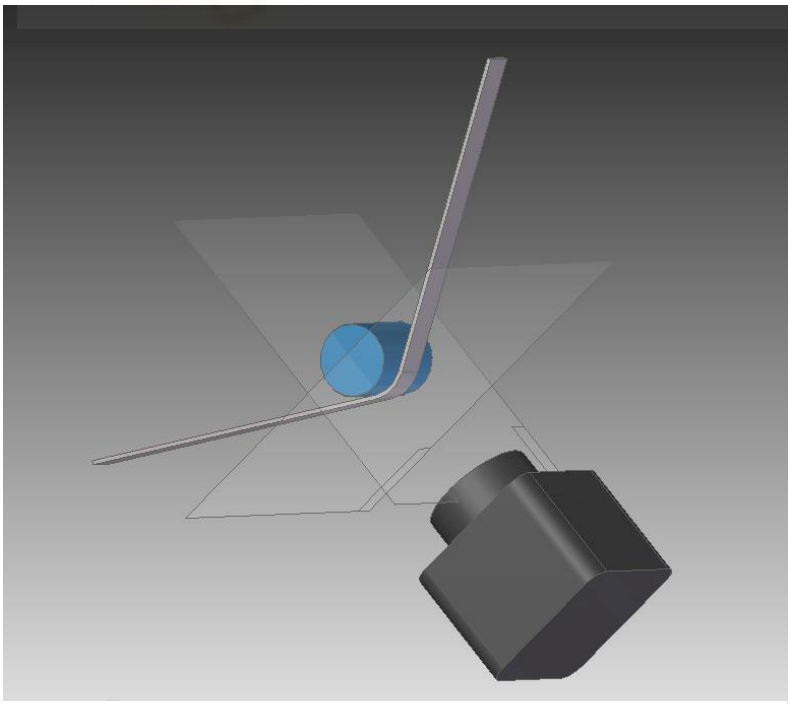


Figura 4.2. Representación 3D del banco de ensayos.

El sistema se graba lateralmente con una cámara fija, lo que permite capturar la interacción entre el material y el pin. En la Figura 4.2 puede observarse el montaje tridimensional del sistema y la geometría del ensayo. La cámara se posiciona de modo que se obtenga una vista clara del plano donde se produce la deformación y el movimiento a estudiar. Como se ha comentado a lo largo del Capítulo 2, de esta forma conseguimos extraer datos en posiciones de la probeta experimental donde no se podrían obtener con sensores convencionales.

En la Figura 4.2 se ilustran las variables geométricas implicadas, las más destacas son:

- $\boldsymbol{\alpha}(alpha)$: ángulo de contacto con el pin.
- $\boldsymbol{\theta}(theta)$: curvatura local en la zona de contacto (despreciable).
- $\boldsymbol{Dp}$: diámetro del pin.
- $\boldsymbol{x}$ y $\boldsymbol{x'}$: desplazamiento local sobre plano y desplazamiento sobre curvatura, respectivamente.
- $\boldsymbol{Z1}, \boldsymbol{Z2}$ y $\boldsymbol{X0}$: distancias utilizadas para calibración y análisis espacial.
- **Plano** imagen cámara: superficie proyectada sobre la cual se extraen datos visuales.

Estas variables son fundamentales para interpretar correctamente los resultados obtenidos del vídeo y para aplicar posteriormente los algoritmos de visión por computador.

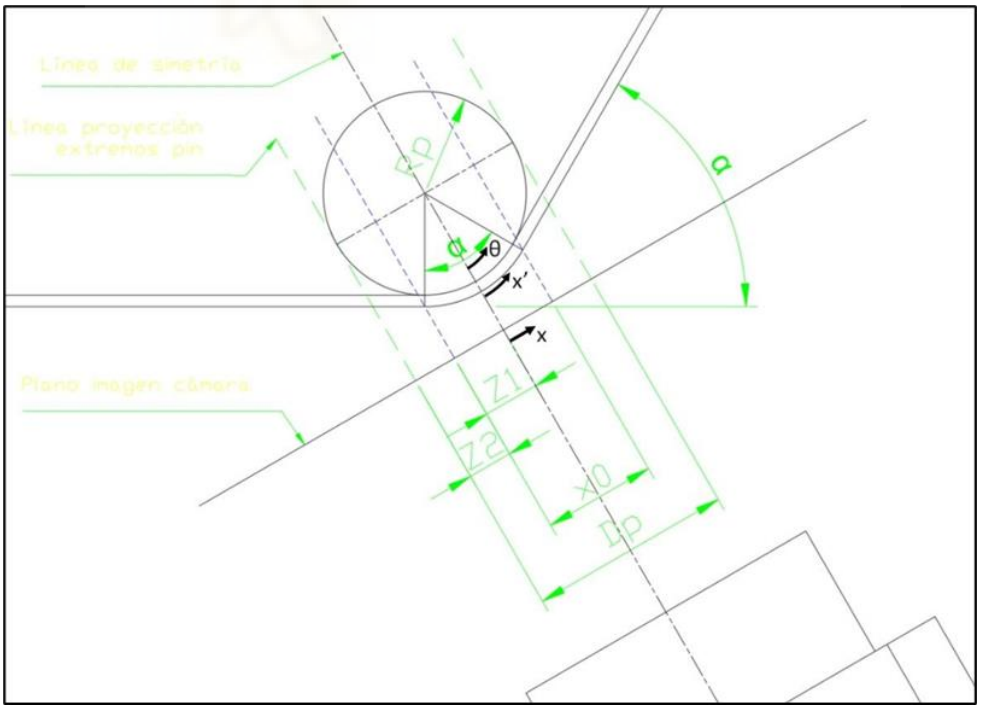


Figura 4.3. Vista lateral 2D del experimento con variables espaciales.

4.2. Adquisición de Datos y Preprocesamiento de Vídeos

Para poder analizar correctamente los ensayos mediante técnicas de visión por computador, ha sido necesario estructurar y preparar tanto los datos experimentales como los vídeos grabados durante las pruebas. Este apartado detalla el proceso seguido para la adquisición de datos y el tratamiento aplicado a los vídeos antes de ser utilizados por el sistema.

4.2.1. Adquisición de Datos

Los parámetros físicos correspondientes a cada ensayo se almacenan en un archivo Excel llamado *datos_generales.xlsx*. Cada hoja del documento contiene todos los ensayos de cada material: BAO o TRIP_590.

Cada fila representa un ensayo individual, identificado por su número de ensayo, mientras que las columnas contienen los distintos valores experimentales, tales como: ancho y profundidad real de la probeta, ángulo de contacto, temperatura, lubricante, rotación, velocidad del ensayo y diámetro del pin. Todos ellos se leen y guardan en el programa bien para mostrarlos por pantalla o bien por ser esenciales en partes del código para llevar a cabo cálculos específicos. Una vista parcial de esta tabla se encuentra en la Tabla 4.1.

Ancho	Grosor	Probeta	Lubricante	Ángulo	Velocidad	Diámetro pin	Temp.	Rotación
8,4	1,4	1	Líquido	60	3	26,1	25	0
9,9	1,5	2	Líquido	60	3	19,1	150	0
8,3	1,4	3	Líquido	60	15	19,1	25	5
8,3	1,5	4	Sólido	60	3	22,1	25	0
8,5	1,5	5	Sólido	60	7	22,1	150	5
8,1	1,4	6	Sólido	60	7	26,1	25	5
8,6	1,5	7	Líquido	30	15	22,1	150	5
8,5	1,5	8	Ninguno	30	7	19,1	150	0
8,5	1,5	9	Líquido	45	7	19,1	150	0
8,3	1,4	10	Ninguno	45	15	26,1	25	5
8,1	1,4	11	Líquido	45	15	22,1	25	0
8,3	1,5	12	Líquido	45	15	19,1	150	5
8,6	1,5	13	Líquido	30	15	19,1	150	0
8,6	1,5	14	Ninguno	60	15	19,1	150	5
8,3	1,4	15	Ninguno	45	15	22,1	150	0

Tabla 4.1. Tabla parcial de la hoja BAO en datos generales.xlsx.

El sistema desarrollado en Python permite seleccionar qué ensayo se desea analizar, accediendo directamente a la fila correspondiente en Excel. En la Figura 4.4 se muestra un fragmento de código donde se accede a estos datos y se almacenan en función del tipo de material y del número de ensayo.

```python
# Pedir al usuario que seleccione el material y número del ensayo
    type_video = input("Introduce el material que quieres estudiar: \n1. BAO\n2. TRIP 590\n(1 o 2): ")
    if type_video == "1":
        material = "BAO"
    elif type_video == "2":
        material = "TRIP_590"
    else:
        print("ERROR: material no valido")
        exit()
    video_number = input("Introduce el número del video que quieres abrir: ")

# Read data video
    datos_ensayo = leer_datos_ensayo(int(video_number), material)
    if datos_ensayo:
        # Mostrar los datos al usuario
        print("\nDatos del ensayo seleccionado:")
        for clave, valor in datos_ensayo.items():
            print(f"{clave.replace('_', ' ').capitalize()}: {valor}")

        real_length_mm = float(datos_ensayo['ancho_inicial'])
        radio_pin = float(datos_ensayo['diametro_pin']) / 2
        angulo_traccion = int(datos_ensayo['angulo'])
        vel_ensayo = int(datos_ensayo['velocidad'])
```

Figura 4.4. Main. Identificación ensayo. Muestra por pantalla y guardado de variables.

Los datos almacenados en *video_number* y *material* son cruciales a lo largo de todo el código, tanto para la lectura como para la exportación de los resultados del experimento.

Además, en el Capítulo 4.6 (Detección de Rotura de Probeta), es importante conocer valores críticos como la velocidad del ensayo. Por otro lado, en el Capítulo 4.3 se hará énfasis en la importancia de valores como el ángulo de contacto y el diámetro del pin para posicionar de forma automática las ROIs.

En la Figura 4.5 se muestra la función *leer_datos_ensayo*, creada para buscar y guardar todos los datos del ensayo en esa hoja de cálculo. Para ello, se ha hecho uso de la biblioteca *pandas*, que como bien se explica en el Capítulo 3.2.3, nos permite un manejo sencillo y muy eficaz de datos con archivos .xlsx.

```python
def leer_datos_ensayo(numero_ensayo, material):
    """
    Lee los datos del Excel según el número de ensayo.

    Args:
        numero_ensayo (int): Número de la probeta/ensayo a buscar.
        material : tipo de material a estudiar.

    Returns:
        dict: Diccionario con los datos relevantes del ensayo.
    """
    excel_path = project_root / 'datos_generales.xlsx'
    try:

        # Leer el archivo Excel
        datos = pd.read_excel(excel_path, material)

        # Buscar la fila correspondiente al número de ensayo
        fila_ensayo = datos.loc[datos['probeta'] == numero_ensayo]

        if fila_ensayo.empty:
            print(f"No se encontró el ensayo con número de probeta {numero_ensayo}.")
            return None

        # Extraer los valores relevantes
        datos_ensayo = {
            'ancho_inicial': fila_ensayo['ancho inicial'].values[0],
            'grosor_inicial': fila_ensayo['grosor inicial'].values[0],
            'lubricante': fila_ensayo['lubricante'].values[0],
            'angulo': fila_ensayo['angulo'].values[0],
            'velocidad': fila_ensayo['velocidad'].values[0],
            'diametro_pin': fila_ensayo['diametro pin'].values[0],
            'temperatura': fila_ensayo['temp.'].values[0],
            'rotacion': fila_ensayo['rotacion'].values[0]
        }

        return datos_ensayo

    except FileNotFoundError:
        print(f"No se encontró el archivo: {excel_path}")
    except Exception as e:
        print(f"Error al leer los datos del Excel: {e}")
    return None
```

Figura 4.5. Data Processing. Función leer_datos_ensayo.

4.2.2. Preprocesamiento de Vídeos

Los vídeos obtenidos de los ensayos no podían ser utilizados directamente, ya que presentaban diversos problemas técnicos que afectaban al análisis automático. Por este motivo, se llevó a cabo un preprocesamiento en tres fases:

a. Unificación de la tasa de muestreo (FPS)

Los sensores de los ensayos registran datos con una frecuencia de 0,02 segundos (50 Hz). Sin embargo, los vídeos originales se grabaron con diferentes tasas: 30 FPS o 60 FPS, lo que dificultaba la sincronización exacta entre vídeos y datos. Para solucionarlo, se utilizaron herramientas de edición como CapCut para convertir todos los vídeos a 50 FPS, mediante duplicación o interpolación de datos.

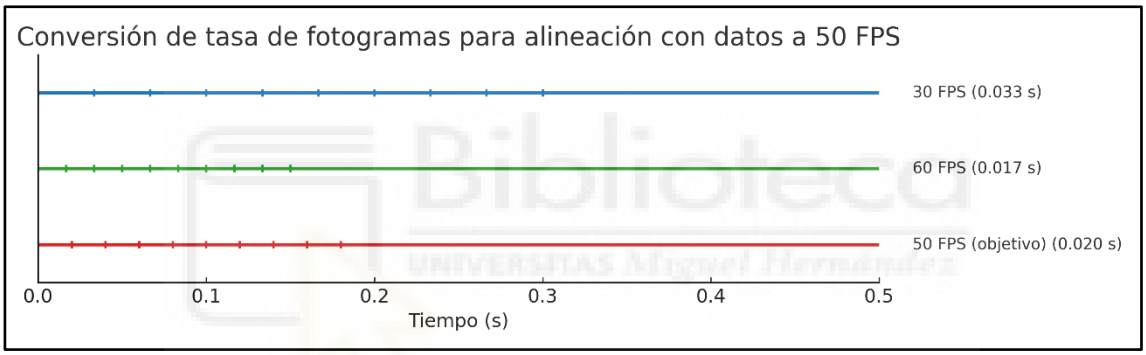


Figura 4.6. Comparación de tasas de fotogramas originales (30 y 60 FPS) con la tasa unificada de 50 FPS utilizada para alinear el análisis visual con la frecuencia de los sensores (0,02s).

b. Recorte de duración del vídeo

Aunque el programa es capaz de detectar el movimiento inicial y la rotura de la probeta, finalizando el ensayo 1 segundos después de la rotura, hay algunos vídeos en los que el tiempo que transcurre entre el inicio del vídeo y el inicio del estiramiento es muy grande.

Como esa información es innecesaria y aporta una mayor carga computacional, se optó por recortar los vídeos 1.5 segundos previo al inicio del estiramiento y 1.5 segundos posterior a la rotura.

De esta forma se mejora la eficiencia del análisis y se reduce el volumen de datos procesados.

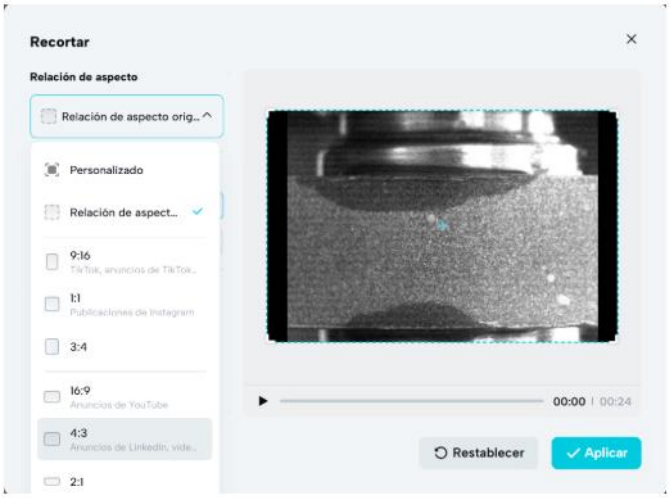
c. Eliminación de bandas negras mediante recorte proporcional

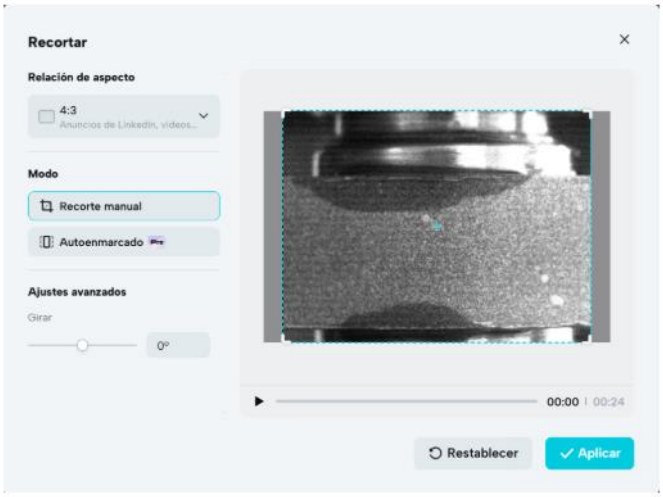
Durante el procesamiento en CapCut, se detectó la aparición de bandas negras laterales en algunos vídeos, debido a la diferencia de resoluciones originales. Se identificaron dos tipos de calidad:

- Vídeos en **alta resolución** (**2560x1920**), que se exportaban correctamente a 1440x1080 sin bandas.
- Vídeos en **baja resolución** (**720x480**), que al ser escalados a 1080p generaban bandas negras laterales.

El problema de las bandas es crítico en este tipo de análisis, pues alteran la geometría de la imagen y afectan a la calibración espacial necesaria para convertir píxeles en unidades físicas reales. Además, pueden confundir los algoritmos de detección de características, generando falsos positivos o impidiendo el correcto seguimiento de puntos reales del material.

Este problema se debía a que CapCut escalaba los vídeos proporcionalmente sin recortar, lo que provocaba que los vídeos con distinta relación de aspecto no se adaptaran correctamente al nuevo formato. La solución aplicada fue realizar un recorte manual en CapCut a los vídeos de baja resolución estableciendo una proporción de 4:3 lo que eliminaba las bandas negras y unifica visualmente todos los vídeos.





4.3. Definición de Parámetros por el Usuario

El programa inicia con la visualización del primer fotograma del vídeo, que será utilizado por el usuario con el fin de definir manualmente dos parámetros esenciales para su posterior análisis: el eje X y el ancho de la probeta experimental.

4.3.1. Definición de eje X de la probeta

La primera acción que debe realizar el usuario es definir el eje horizontal de la probeta, que será utilizado como referencia de ángulo para determinar la rotación del sistema respecto al plano de la imagen. Esta rotación se utiliza tanto para corregir la orientación de los vectores de movimiento como para distribuir automáticamente los puntos de interés a lo largo de la probeta en pasos posteriores del programa.

Para ello, el usuario debe hacer clic sobre el extremo superior izquierdo visible de la probeta y posteriormente sobre el extremo superior derecho, como se puede observar en la Figura 4.8.

El programa detecta esos clics mediante una función personalizada llamada *select_axis* que escucha eventos del ratón sobre la ventana emergente utilizando OpenCV. Cada punto seleccionado se guarda y se representa visualmente:

- Al hacer clic, se dibuja un círculo azul en el punto.
- Cuando se seleccionan dos puntos, se traza una línea azul que representa el eje X definido. A partir de esta línea, se calcula el ángulo de inclinación respecto al eje horizontal mediante trigonometría básica.

```python
# Paso 1: Seleccionar dos puntos para definir el eje X de la probeta
def select_axis(frame, out):
    points = []
    frame_resized = cv.resize(frame, (800, 600))
    display_frame = frame_resized.copy()

    def click_event(event, x, y, flags, param):
        nonlocal frame_resized
        if event == cv.EVENT_LBUTTONDOWN:
            points.append((x, y))
            cv2.circle(display_frame, (x,y), 5, (178,114,0), -1)
            if len(points) == 2:
                cv2.line(display_frame, points[0], points[1], (178, 114, 0), 2)

    cv2.namedWindow('Define Material X-Axis')
    cv2.setMouseCallback('Define Material X-Axis', click_event)
    print("Selecciona dos puntos para definir el eje X de la probeta.")

    while True:
        out.write(display_frame)
        cv2.imshow('Define Material X-Axis', display_frame)
        if cv2.waitKey(20) & 0xFF == 13:
            break

    cv2.destroyWindow('Define Material X-Axis')

    if len(points) != 2:
        raise ValueError("Debes seleccionar exactamente dos puntos.")
    return points


# Paso 2: Calcular el ángulo de rotación
def calculate_angle(point1, point2):
    dx = point2[0] - point1[0]
    dy = point1[1] - point2[1]
    angle = np.arctan2(dy, dx)  # Ángulo en radianes
    return angle
```

Figura 4.9. Dic Processing. Funciones select_axis y calculate_angle.

4.3.2. Definición visual del ancho de la probeta

Una vez establecido el eje de referencia, el siguiente paso es definir gráficamente el ancho de la probeta. Para ello, el usuario debe seleccionar dos puntos sobre el pin que actúa como soporte del material. Es decir, se debe marcar el ancho en el cruce entre probeta y extremo izquierdo del pin, como se muestra en la Figura 4.10.

Esto permite conocer la altura de la probeta en píxeles, medida vertical que se usará posteriormente para calcular la escala $\frac{mm}{pix}$.

Debemos conectar de algún modo las distancias en píxeles con las reales en mm. Para ello, se decidió utilizar el ancho de probeta con el fin de poder calcular una escala ($\frac{mm}{pix}$) que nos sirviera a lo largo de todo el programa para poder extraer datos en magnitudes reales.

Este cálculo se realiza relacionando el valor en píxeles con el valor real del ancho (*ancho_probeta*) leído desde el Excel correspondiente a ese ensayo.

Definir con precisión este parámetro es crítico, ya que cualquier error en la estimación de la escala se traducirá en errores proporcionales en la interpretación de desplazamientos, deformaciones o cualquier otra magnitud física extraída a partir del vídeo.

```python
def calculate_scale(frame, out, real_length_mm):
    """
    This function allows the user to manually select two points in the image
    by clicking on it. Once two points are selected, it draws a line between
    them and displays their coordinates. These points are used to calculate
    the distance in pixels and determine the scale (mm/pixel)
    """
    frame_resized = cv.resize(frame, (800, 600))
    display_frame = frame_resized.copy()

    points = []

    # Función para capturar puntos clickeados
    def select_point(event, x, y, flags, param):
        nonlocal frame_resized
        if event == cv.EVENT_LBUTTONDOWN:
            points.append((x, y))
            cv2.circle(display_frame, (x, y), 5, (0, 94, 213), -1)
            if len(points) == 2:
                cv2.line(display_frame, points[0], points[1], (0, 94, 213), 2)
                print(f"Primer punto: {points[0]}")
                print(f"Segundo punto: {points[1]}")

    # Configurar la ventana para seleccionar puntos
    cv2.namedWindow("Define Material Width")
    cv2.setMouseCallback("Define Material Width", select_point)
    while True:
        out.write(display_frame)
        cv2.imshow("Define Material Width", display_frame)
        if cv2.waitKey(20) & 0xFF == 13:
            break

    cv2.destroyWindow("Define Material Width")

    # Calcular la distancia entre los puntos en píxeles
    if len(points) == 2:
        x1, y1 = points[0]
        x2, y2 = points[1]
        pixel_distance = np.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)
        print(f"Distancia en píxeles entre los puntos seleccionados: {pixel_distance:.2f} píxeles")

        # Calcular la escala en mm/píxel
        scale = real_length_mm / pixel_distance
        print(f"Escala: {scale:.4f} mm/píxel")
        return scale, points
    else:
        print("No se han seleccionado dos puntos.")
        return None
```

Figura 4.11. Dic Processing. Función del cálculo de la escala.

Si el extremo izquierdo del pin no está visible se deberá indicar el ancho de la probeta en el borde izquierdo de la ventana, como se muestra en la Figura 4.12.

Este sistema mixto, donde el usuario interviene al inicio para definir las referencias espaciales, garantiza una mayor flexibilidad y precisión en los análisis posteriores, sin depender exclusivamente de algoritmos automáticos que podrían fallar en entornos con ruido visual o geometrías variables.

Figura 4.12. Definición del ancho de probeta con mala visibilidad del extremo izquierdo del pin en TRIP-18 (línea naranja).

El fragmento del archivo *main.py* que controla estas funciones se puede encontrar en la Figura 4.13.

```python
# Selección del eje X (angulo)
axis_points = select_axis(frame, out)
angle = calculate_angle(*axis_points)
print(f"Ángulo del eje X respecto al horizontal: {np.degrees(angle):.2f} grados")

# Selección ancho de probeta (escala)
print("\nSelecciona dos puntos para definir el ancho de la probeta. De esta forma, podremos calcular la escala.")
scale, width = calculate_scale(frame, out, real_length_mm=real_length_mm)
if scale == None:
    print("No se ha podido calcular la escala, verifique los parámetros.")
    exit()
else:
    print("Escala calculada.")
```

Si realizamos estos pasos, el programa nos mostrará por pantalla los resultados de los valores que hemos ido calculando. Además, se habrá guardado posiciones espaciales esenciales para posteriores pasos. La Figura 4.14 muestra lo que el programa nos devuelve por pantalla hasta este punto, se ha utilizado como ejemplo el ensayo BAO-46.

```
Introduce el material que quieres estudiar:
1. BAO
2. TRIP 590
(1 o 2): 1
Introduce el número del video que quieres abrir: 46
Video abierto correctamente: C:\aaUniversidad\Vision_por_computador\deteccion_keypoints\BAO\videos\BAO_46_50FPS.mp4

Datos del ensayo seleccionado:
Ancho inicial: 8.2
Grosor inicial: 1.4000000000000001
Lubricante: Ninguno
Angulo: 45.0
Velocidad: 3.0
Diametro pin: 22.1
Temperatura: 25.0
Rotacion: 0.0
Duración del video: 20.28 segundos, FPS: 50

Selecciona dos puntos para definir el eje X de la probeta.
Ángulo del eje X respecto al horizontal: 4.30 grados

Selecciona dos puntos para definir el ancho de la probeta. De esta forma, podremos calcular la escala.
Primer punto: (59, 237)
Segundo punto: (89, 563)
Distancia en píxeles entre los puntos seleccionados: 327.38 píxeles
Escala: 0.0250 mm/pixel
Escala calculada.
```

4.4. Creación de Regiones de interés (ROIs)

En análisis de vídeo mediante visión por computador, una **Región de Interés (ROI)** es una zona específica de una imagen en la que se desea centrar el procesamiento o la extracción de información. En este proyecto, las ROIs se utilizan para monitorizar el movimiento del material en zonas clave del contacto entre la probeta y el pin durante el ensayo de tracción.

El objetivo de definir estas ROIs es calcular la velocidad media de todos los puntos de interés que se encuentren dentro de ellas, fotograma a fotograma. Posteriormente, se comparan las velocidades medias de la ROI izquierda y la ROI derecha, lo que permite identificar diferencias de movimiento y, por tanto, deformaciones relativas del material en las zonas de contacto/no contacto. Esta comparación es fundamental para analizar el comportamiento mecánico de la probeta.

Una vez hemos indicado el eje X y el ancho de la probeta, el usuario debe escoger entre dos modos de creación de ROIs.

4.4.1. ROIs automáticas

Este método genera las ROIs de forma totalmente automatizada utilizando ecuaciones geométricas, partiendo de los puntos ya definidos en el extremo izquierdo del pin. Para poder obtener unas ROIs dibujadas que sean siempre perpendiculares a la probeta, corregidas automáticamente en caso de que haya algún error y con una anchura variable, se ha desarrollado una función llamada *calcular_rois*, que utiliza el siguiente procedimiento:

```
def calcular_rois(punto_pin_1, punto_pin_2, rp, alpha, frame, scale, angle, out):
    """
    Calcula las posiciones de las ROIs automáticamente.

    :param punto_pin_1: Primer punto seleccionado por el usuario sobre el extremo del pin (x, y)
    :param punto_pin_2: Segundo punto seleccionado por el usuario sobre el extremo del pin (x, y)
    :param rp: Radio del pin (mm)
    :param alpha: Ángulo de tracción de la probeta (Cuidado!!!en grados)
    :param frame: Imagen donde se mostrarán las ROIs
    :param scale: Escala de mm/pix
    :param angle: Ángulo de rotacion de la probeta con respecto eje X universal (Cuidado!!! en radianes)
    :return: Coordenadas de las ROIs en formato (A, C, D, B) siendo la distribución de la siguiente forma->  [A  C]
                                                                                                             [B  D]
    """
```

Figura 4.15. Dic Processing. Encabezado función calcular_rois.

I. Conversión de unidades y cálculo de distancias base

La función convierte el ángulo de tracción *alpha* de grados a radianes, ya que las funciones trigonométricas de Numpy trabajan en radianes. Luego se calculan las distancias desde el extremo del pin hasta el eje central de cada ROI.

```
    arreglado = False
    # Convertir alpha a radianes
    alpha_rad = np.radians(alpha)
    roi_width = 150
    roi_height = 100 # Además del ancho de la probeta y para cada lado

    # Calcular las distancias de las lineas de contacto con pin desde extremo pin izq
    d_roi1 = rp - (rp + 1.2) * np.sin(alpha_rad / 2) #mm
    d_roi2 = rp + (rp + 1.2) * np.sin(alpha_rad / 2) #mm

    # Paso de mm a pix
    d_roi1 = d_roi1/scale
    d_roi2 = d_roi2/scale
```

La Figura 4.17. muestra las distancias d_{roi1} y d_{roi2} con respecto al extremo izquierdo del pin.

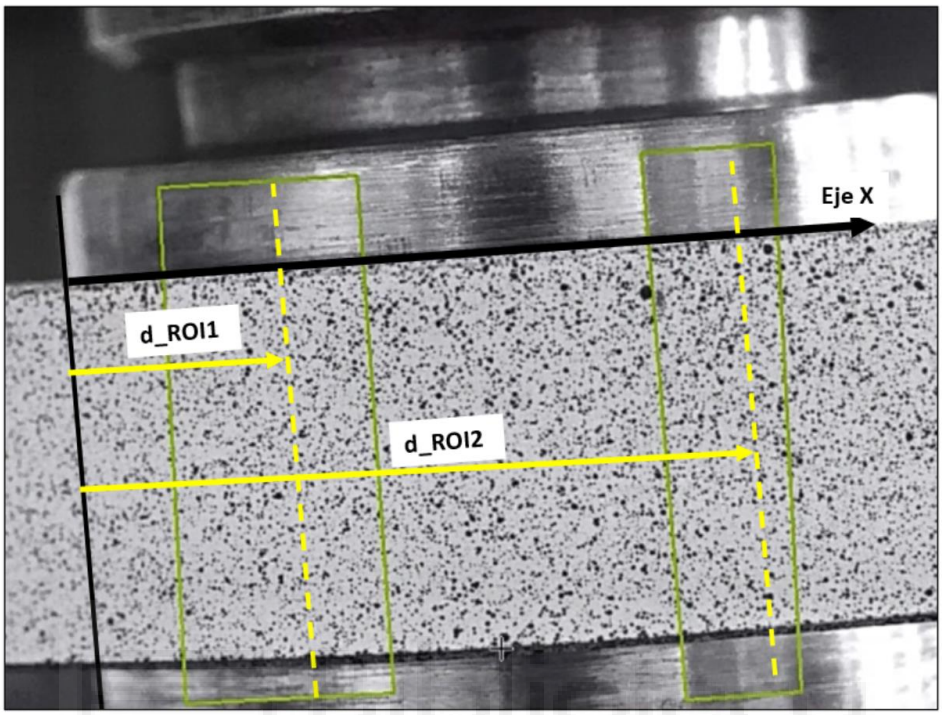


Figura 4.17. Representación de las distancias de los ejes centrales de cada ROI desde el extremo izquierdo del pin.

$$d_{roi1} = rp - (rp + 1.2) \cdot sen\left(\frac{\alpha}{2}\right) \approx rp \cdot \left(1 - sen\left(\frac{\alpha}{2}\right)\right) \quad (28)$$

$$d_{roi2} = rp + (rp + 1.2) \cdot sen\left(\frac{\alpha}{2}\right) \approx rp \cdot \left(1 + sen\left(\frac{\alpha}{2}\right)\right) \quad (29)$$

Como ya se ha comentado, (28) y (29) describen la posición horizontal (en el eje X del material) donde se deben colocar las ROIs izquierda y derecha respecto al extremo izquierdo del pin. Ambas se han deducido aplicando trigonometría en un sistema circular, proyectando desde el centro del pin hacia los laterales, y considerando la inclinación angular del ensayo (α). Sirven para posicionar las ROIs de forma simétrica, alineada y fuera de la zona de contacto directa, facilitando el análisis de deformación.

II. Cálculo del eje central de cada ROI

A partir de d_{roi1} y d_{roi2}, se calculan las coordenadas de los ejes centrales de las ROIs mediante trigonometría, considerando la rotación de la probeta (variable *angle* en radianes).

```
# Conversión por el ángulo de giro de la probeta (dist. entre extremo pin y ejes centros ROIs corregida)
    x1 = np.cos(angle) * d_roi1
    y1 = np.sin(angle) * d_roi1
    x2 = np.cos(angle) * d_roi2
    y2 = np.sin(angle) * d_roi2
```

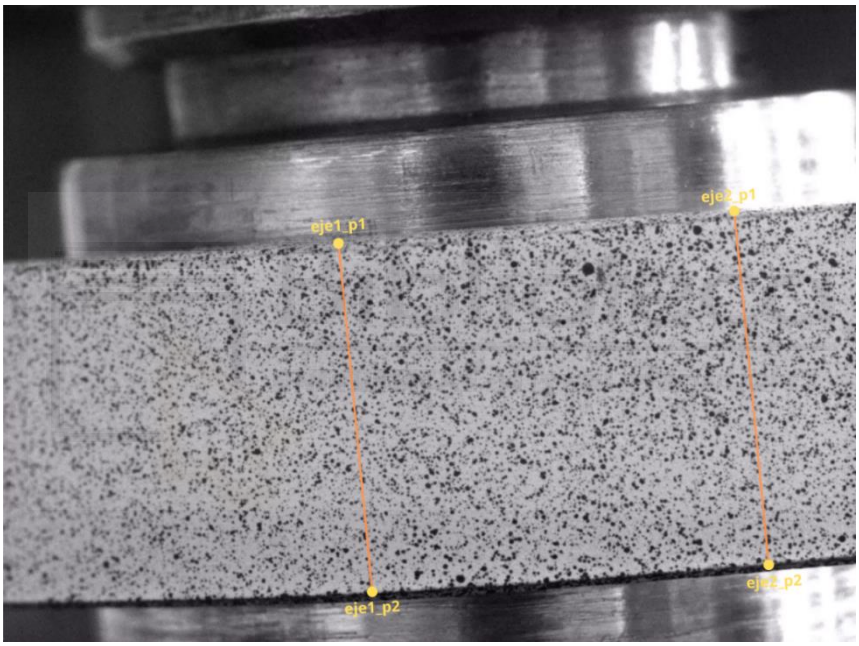


Figura 4.19. Ejes centrales de cada ROI (líneas naranjas).

III. Proyección extendida de los ejes

Para asegurar que las ROIs se dibujen completas en pantalla, los ejes se extienden verticalmente en la dirección perpendicular de la probeta fuera de la pantalla.

```
# Ejes centrales más largos
    larg_x_sup = (np.cos(np.radians(90) - angle) * (300))
    larg_y_sup = (np.sin(np.radians(90) - angle) * (300))
    larg_x_inf = (np.sin(angle) * (300))
    larg_y_inf = (np.cos(angle) * (300))
```

Figura 4.20. Dic Processing. Proyección extendida de los ejes. Función calcular_rois.

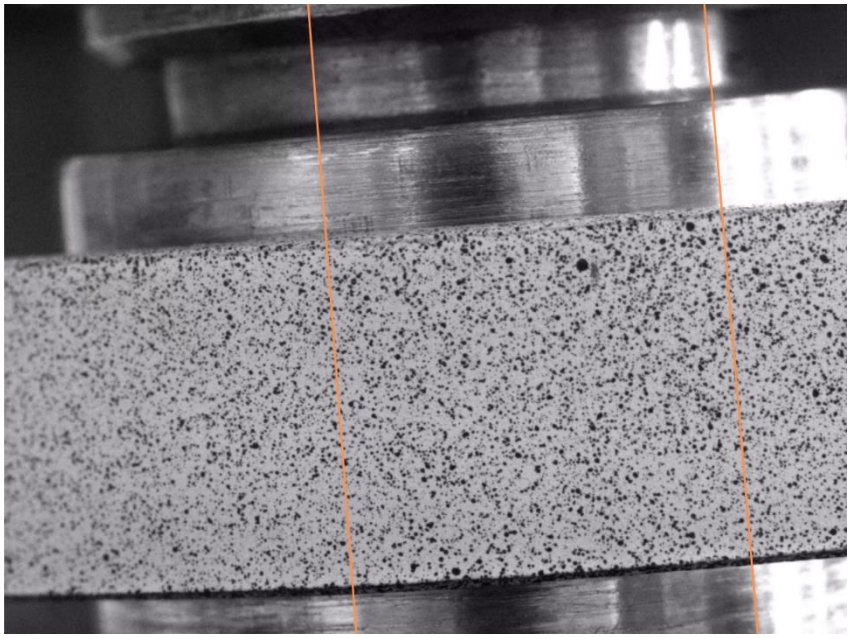


Figura 4.21. Ejes centrales de cada ROI extendidos (líneas naranjas).

IV. Definición de las esquinas de las ROIs

Cada ROI es un rectángulo definido por cuatro esquinas. Estas se obtienen sumando y restando vectores al eje central para calcular desplazamientos en X e Y. La Figura 4.22 muestra cuál es el orden y la enumeración de estas esquinas.

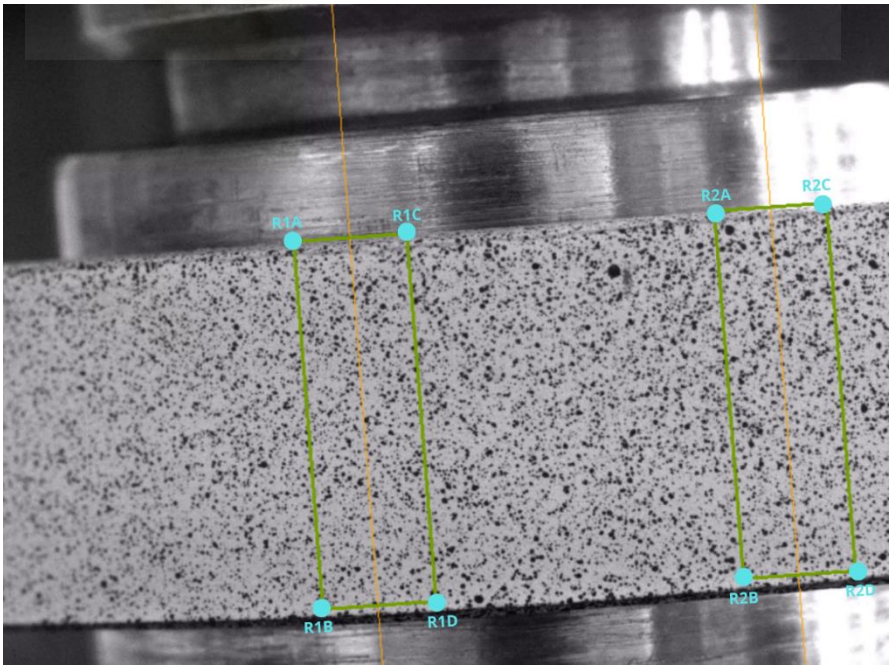


Figura 4.22. Enumeración de las esquinas de cada ROI y primera vista de ROIs sin extender.

Durante la implementación de las ROIs automáticas surgieron una serie de ajustes y mejoras necesarias para garantizar su funcionalidad en condiciones reales de ensayo. Uno de los primeros problemas destacados fue que, al comenzar la tracción, la probeta puede desplazarse ligeramente en el eje vertical, provocando que una parte del material quede fuera de la región inicialmente definida. Para solventarlo, se decidió ampliar verticalmente las ROIs, extendiéndolas por encima y por debajo del eje de contacto.

La Figura 4.23 muestra cómo unas ROIs sin extender pueden provocar un desajuste. Además, se observa que la ampliación vertical permite admitir ahora todos los puntos a pesar de ese ligero movimiento en el eje Y de la probeta de ensayo.

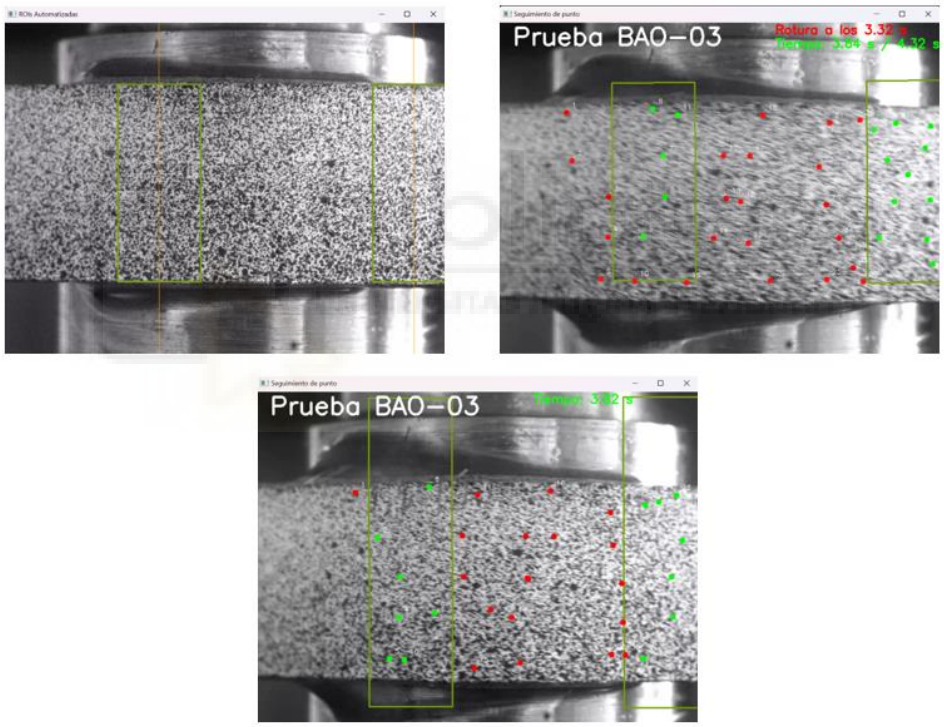


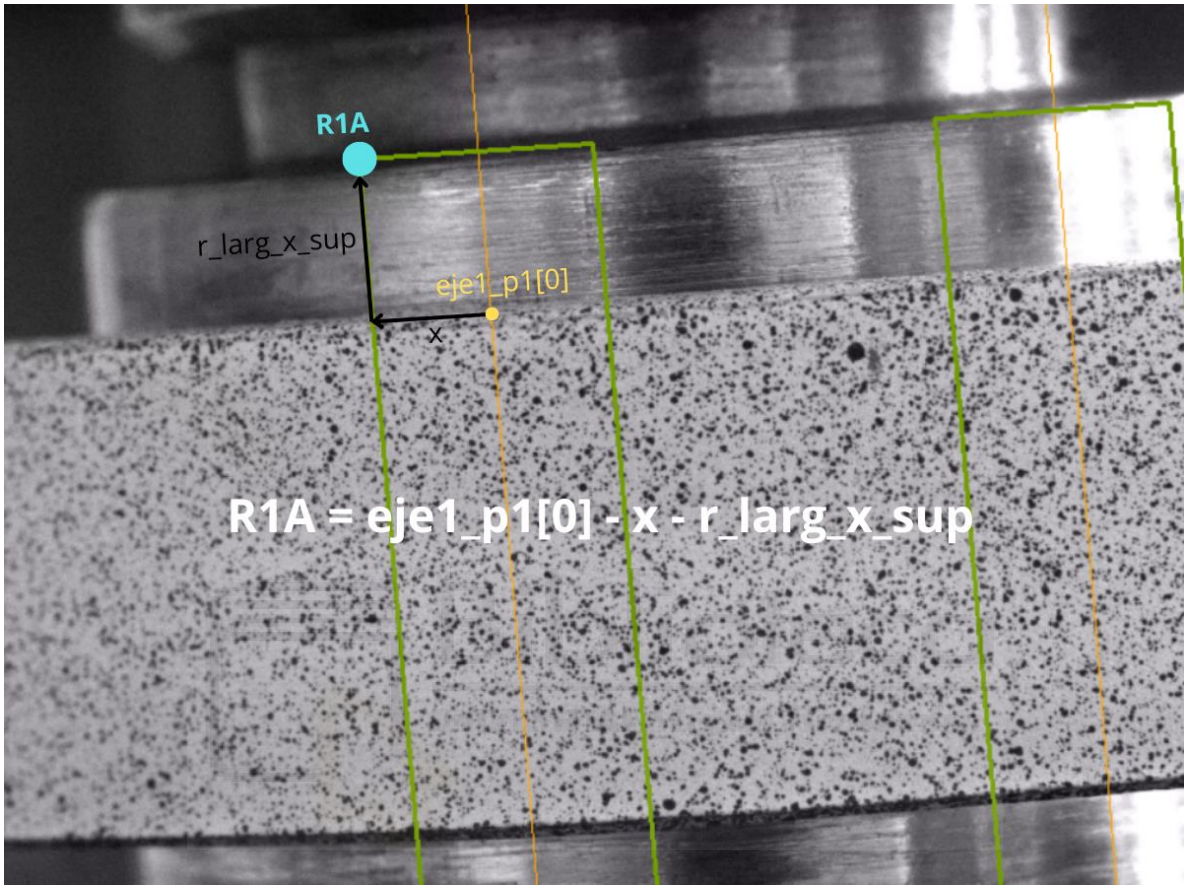
Figura 4.23. Ejemplo de desajuste por la no extensión vertical de ROIs. BAO-03.

El fragmento de código que controla la creación cada esquina se encuentra en la Figura 4.24. En ella podemos observar como en función de la orientación de la probeta (positiva o negativa) debemos definir una serie ecuaciones u otras, esto se debe a que en la matriz de píxeles que componen la ventana, el eje X aumenta hacia la derecha y el eje Y lo hace hacia abajo. En la Figura 4.22 podemos ver como la orientación de la probeta es positiva con lo que la esquina R1C tendrá mayor X, pero menor Y, y así sucesivamente.

```
        # Vectores de desplazamiento para aumentar la altura de las ROIs
        r_larg_x_sup = (np.cos(np.radians(90) - angle) * (roi_height))
        r_larg_y_sup = (np.sin(np.radians(90) - angle) * (roi_height))
        r_larg_x_inf = (np.sin(angle) * (roi_height))
        r_larg_y_inf = (np.cos(angle) * (roi_height))

        # Vectores de desplazamiento desde punto de contacto de eje central hasta esquina de ROI
        x = (roi_width / 2) * np.cos(angle)
        y = (roi_width / 2) * np.sin(angle)

        # Dependiendo de la orientación usaremos unas ecuaciones u otras.
        if angle >= 0:

            # Extremos ejes en contacto con el pin
            eje1_p1 = (punto_pin_1[0] + x1, punto_pin_1[1] - y1)
            eje1_p2 = (punto_pin_2[0] + x1, punto_pin_2[1] - y1)
            eje2_p1 = (punto_pin_1[0] + x2, punto_pin_1[1] - y2)
            eje2_p2 = (punto_pin_2[0] + x2, punto_pin_2[1] - y2)

            # Extremos ejes fuera de pantalla (sirven para pintar únicamente)
            contact1_p1 = (int(eje1_p1[0] - larg_x_sup), int(eje1_p1[1] - larg_y_sup))
            contact1_p2 = (int(eje1_p2[0] + larg_x_inf), int(eje1_p2[1] + larg_y_inf))
            contact2_p1 = (int(eje2_p1[0] - larg_x_sup), int(eje2_p1[1] - larg_y_sup))
            contact2_p2 = (int(eje2_p2[0] + larg_x_inf), int(eje2_p2[1] + larg_y_inf))

            # Esquinas superiores de ambas ROIs
            R1A = (int(eje1_p1[0] - x - r_larg_x_sup), int(eje1_p1[1] + y - r_larg_y_sup))
            R1C = (int(eje1_p1[0] + x - r_larg_x_sup), int(eje1_p1[1] - y - r_larg_y_sup))
            R2A = (int(eje2_p1[0] - x - r_larg_x_sup), int(eje2_p1[1] + y - r_larg_y_sup))
            R2C = (int(eje2_p1[0] + x - r_larg_x_sup), int(eje2_p1[1] - y - r_larg_y_sup))

            # Esquinas inferiores de ambas ROIs
            R1B = (int(eje1_p2[0] - x + r_larg_x_inf), int(eje1_p2[1] + y + r_larg_y_inf))
            R1D = (int(eje1_p2[0] + x + r_larg_x_inf), int(eje1_p2[1] - y + r_larg_y_inf))
            R2B = (int(eje2_p2[0] - x + r_larg_x_inf), int(eje2_p2[1] + y + r_larg_y_inf))
            R2D = (int(eje2_p2[0] + x + r_larg_x_inf), int(eje2_p2[1] - y + r_larg_y_inf))

        else:
            eje1_p1 = (punto_pin_1[0] + x1, punto_pin_1[1] + y1)
            eje1_p2 = (punto_pin_2[0] + x1, punto_pin_2[1] - y1)
            eje2_p1 = (punto_pin_1[0] + x2, punto_pin_1[1] + y2)
            eje2_p2 = (punto_pin_2[0] + x2, punto_pin_2[1] - y2)

            contact1_p1 = (int(eje1_p1[0] + larg_x_sup), int(eje1_p1[1] - larg_y_sup))
            contact1_p2 = (int(eje1_p2[0] - larg_x_inf), int(eje1_p2[1] + larg_y_inf))
            contact2_p1 = (int(eje2_p1[0] + larg_x_sup), int(eje2_p1[1] - larg_y_sup))
            contact2_p2 = (int(eje2_p2[0] - larg_x_inf), int(eje2_p2[1] + larg_y_inf))

            R1A = (int(eje1_p1[0] - x + r_larg_x_sup), int(eje1_p1[1] - y - r_larg_y_sup))
            R1C = (int(eje1_p1[0] + x + r_larg_x_sup), int(eje1_p1[1] + y - r_larg_y_sup))
            R2A = (int(eje2_p1[0] - x + r_larg_x_sup), int(eje2_p1[1] - y - r_larg_y_sup))
            R2C = (int(eje2_p1[0] + x + r_larg_x_sup), int(eje2_p1[1] + y - r_larg_y_sup))

            R1B = (int(eje1_p2[0] - x - r_larg_x_inf), int(eje1_p2[1] - y + r_larg_y_inf))
            R1D = (int(eje1_p2[0] + x - r_larg_x_inf), int(eje1_p2[1] + y + r_larg_y_inf))
            R2B = (int(eje2_p2[0] - x - r_larg_x_inf), int(eje2_p2[1] - y + r_larg_y_inf))
            R2D = (int(eje2_p2[0] + x - r_larg_x_inf), int(eje2_p2[1] + y + r_larg_y_inf))
```

Figura 4.24. Dic Processing. Definición de las esquinas de las ROIs.

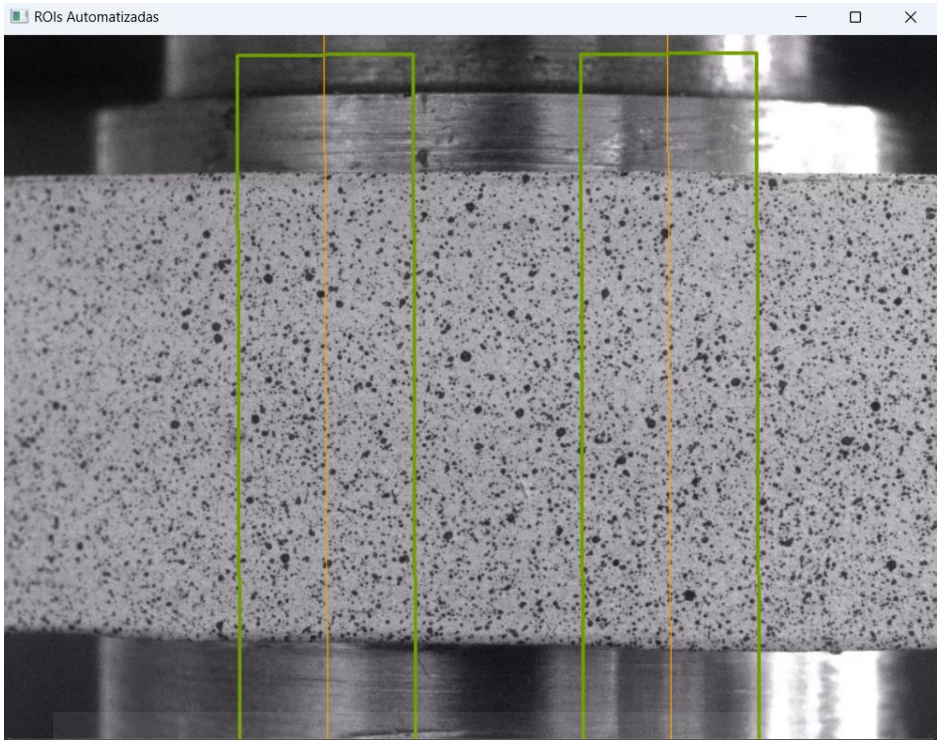
Con el fin de poder entender mejor el código que permite la creación de estas ROIs a partir de la definición de sus esquinas en función de la orientación de la probeta se ha creado la Figura 4.25, donde se pueden observar los pasos para crear la esquina R1A.



Como se puede ver, centrándonos únicamente en las coordenadas x, si partimos de la coordenada x del punto *eje1_p1* y le restamos el valor *x* tendríamos la esquina de la ROI sin ampliar verticalmente, pero como se ha explicado previamente es necesario hacerlo. Para ello, se vuelve a restar el valor de *r_larg_x_sup*, de esta forma llegamos a obtener el valor de la coordenada x del punto R1A.

Esta ha sido la forma de proceder con el resto de los puntos, pero se ha de tener en cuenta la orientación de la probeta para modificar los signos positivo o negativo.

La Figura 4.26 muestra un ejemplo de creación automática de ROIs sin problemas posteriores para el ensayo TRIP-26.



V. Correcciones geométricas automáticas

A pesar de que el cálculo de las ROIs automáticas se basa en una formulación geométrica sólida y teóricamente correcta, se observó que en la práctica estas regiones podían no posicionarse de forma completamente precisa en la imagen. Las distancias teóricas calculadas (d_{roi1} y d_{roi2}) no siempre coincidían visualmente con la localización esperada de las zonas de contacto/no contacto en los vídeos grabados.

Esta discrepancia, aunque leve, podía provocar que:

- La ROI derecha quedase parcialmente fuera del área visible de la imagen.
- Ambas ROIs se superpusiesen entre sí.
- Alguna de sus esquinas estuviera fuera de la ventana de análisis, provocando errores de lectura o valores nulos.

Resulta crucial que exista una distancia entre el margen derecho de la ventana y el fin de la ROI derecha para que los puntos puedan salir de esta región. En caso contrario, los puntos quedarán siempre dentro al borde de la imagen por no poder continuar debido a un límite físico, reduciendo así la velocidad media de las ROI.

Se identificaron dos causas principales que justifican estas desviaciones entre la teoría y la imagen:

- **Variaciones de zoom o escala entre vídeos.**

  Aunque el programa recalcula la escala en cada experimento, si el zoom aplicado por la cámara durante la grabación cambia, las proporciones aparentes en la imagen pueden variar ligeramente. Esto puede afectar la alineación esperada entre el modelo geométrico y el vídeo real.

- **Diferencias en el ángulo de grabación.**

  Más importante aún es la posible falta de repetitividad en el posicionamiento de la cámara. En algunos ensayos, la cámara no estaba perfectamente perpendicular a la probeta ni centrada con respecto al pin. Esto genera una proyección oblicua que distorsiona ligeramente las distancias y provoca que lo que se calcula como centrado respecto al pin no se vea así en la pantalla. La Figura 4.27 muestra este posible inconveniente.

Por tanto, se optó por priorizar la validez visual y operativa de las ROIs frente a la rigidez teórica. Un pequeño error de posición en píxeles es preferible a una ROI mal colocada o fuera de imagen que impida obtener datos válidos.

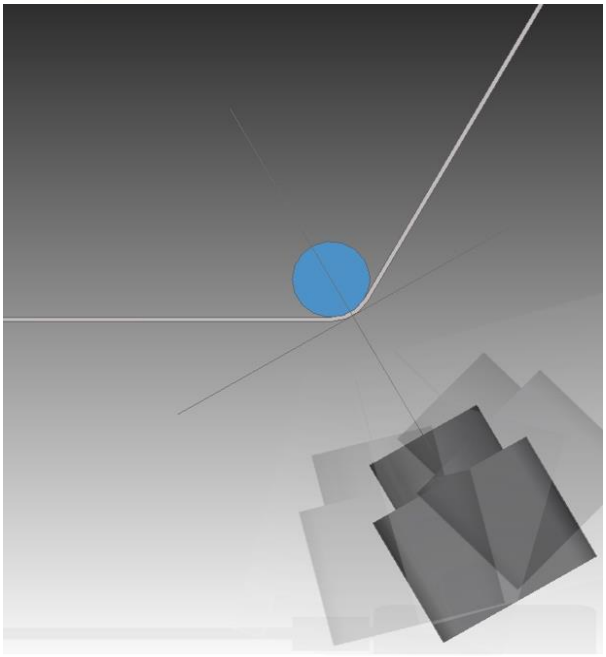


Figura 4.27. Posible variación en el ángulo de grabación.

Con el fin de garantizar la validez de las ROIs generadas, se implementó un sistema de verificación y corrección en tres fases, aplicadas secuencialmente si se detecta un problema. Este sistema está contenido dentro de un bucle *while not arreglado*, lo que permite recalcular automáticamente hasta que se obtenga una solución válida.

1. Evitar solapamiento entre ROIs

Si la distancia entre d_{roi1} y d_{roi2} es menor que el ancho de una ROI significa que se superponen. La solución adoptaba es desplazar ambas en direcciones opuestas una distancia igual a la mitad del ancho de la ROI para cada una.

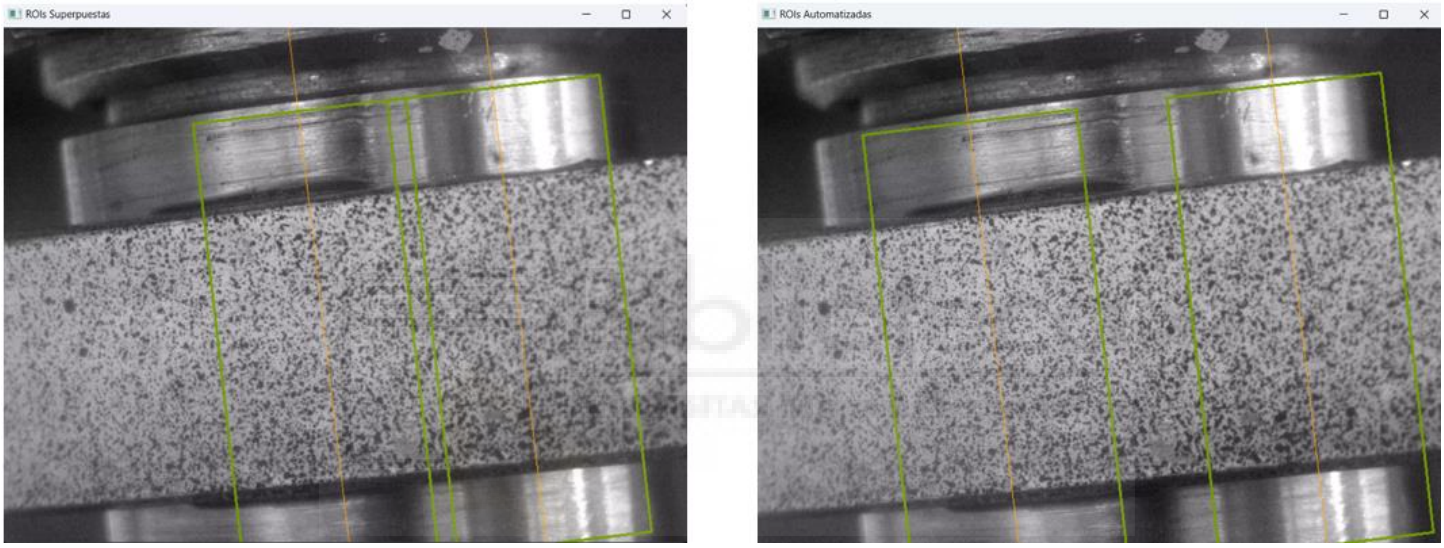


Figura 4.28. Ejemplo de superposición de ROIs junto a la solución adoptada.BAO-31.

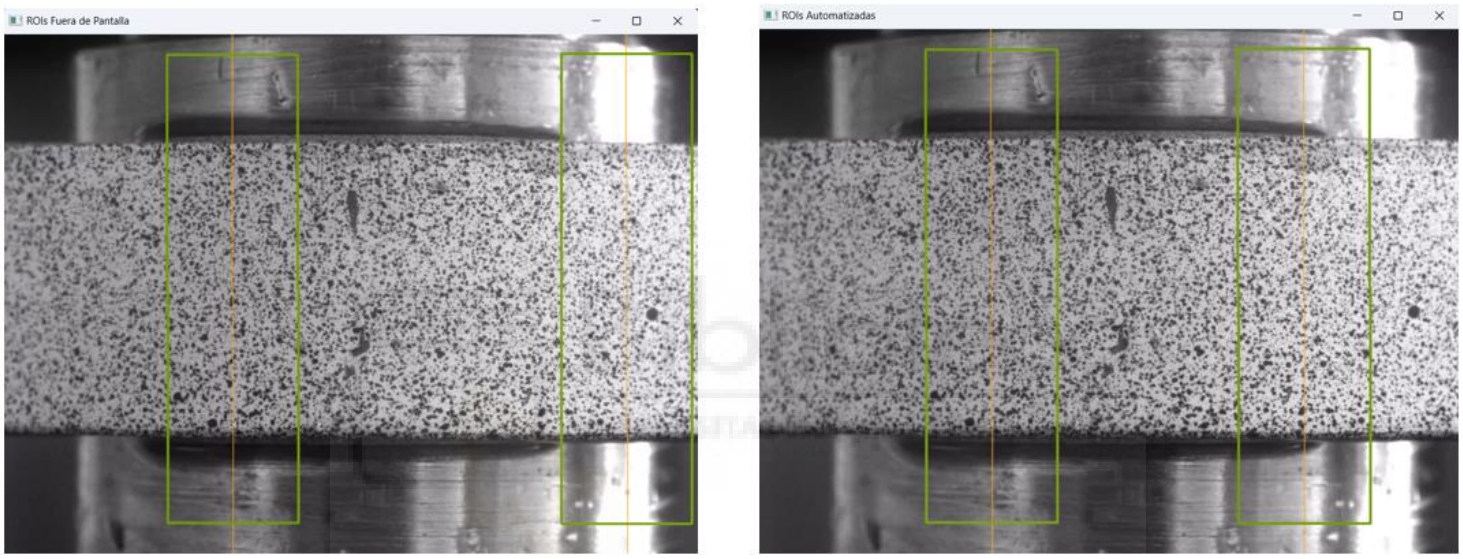
Los ensayos con mayor riesgo de superposición de ROIs son aquellos con menor ángulo de tracción (30º) y con menor diámetro de pin (19.1 mm). Aunque cabe destacar que, con un ancho de ROI de 150 píxeles no ha habido ningún caso de solapamiento en los 114 ensayos estudiados. Para este caso se ha aumentado el ancho de las ROIs hasta los 250 píxeles cada una.

2. Evitar ROIs fuera del límite derecho establecido

Si d_{roi2} (ROI derecha) supera el límite establecido se considera inválida. Dicho límite es el borde de la imagen (800 píxeles) menos la mitad del ancho de ROI (75 píxeles) menos un margen de seguridad (25 píxeles); es decir, si $d_{roi2} > 700$ píxeles se considera que la ROI derecha está demasiado cercana al borde derecho para que los puntos puedan salir de ella con seguridad.

Para solventar este inconveniente, se recalculan las distancias utilizando una proporción de ¼ y ¾ del espacio disponible desde el extremo izquierdo del pin hasta el borde derecho de la pantalla, para mantenerlas bien distribuidas y dentro del marco.

La Figura 4.29 muestra un ejemplo en el que la ROI derecha no está fuera de pantalla, pero el margen entre el final de la región y el borde derecho de la pantalla es muy pequeño, con lo que se decide utilizar la proporción de distancias previamente explicada.



Los ensayos con mayor riesgo de que la ROI derecha se salga de los límites son aquellos con un mayor ángulo de tracción (60º) y con mayor diámetro de pin (26.1 mm). Esto se debe a que son los casos con las ROIs más separadas y es muy probable que ROI2 se salga de los límites.

3. Esquina fuera de la imagen

Si las esquinas R2C o R2D (las de la ROI2) tienen coordenadas $x > 800$ píxeles, una parte del polígono queda fuera de imagen. Esto puede ocurrir cuando el ángulo de orientación de la probeta es muy pronunciado, generando que d_{roi2} cumpla con las especificaciones del requisito número 2 pero que alguna esquina se encuentre fuera de la imagen.

Para poder hacer frente a esta casuística se decide desplazar ambas ROIs una distancia f_{corr}, que depende del ángulo de orientación de la probeta (conocido gracias a la introducción del eje X por el usuario, Capítulo 4.3.1). La Figura 4.30 muestra un boceto con la geometría básica implementada para el cálculo de esta distancia.

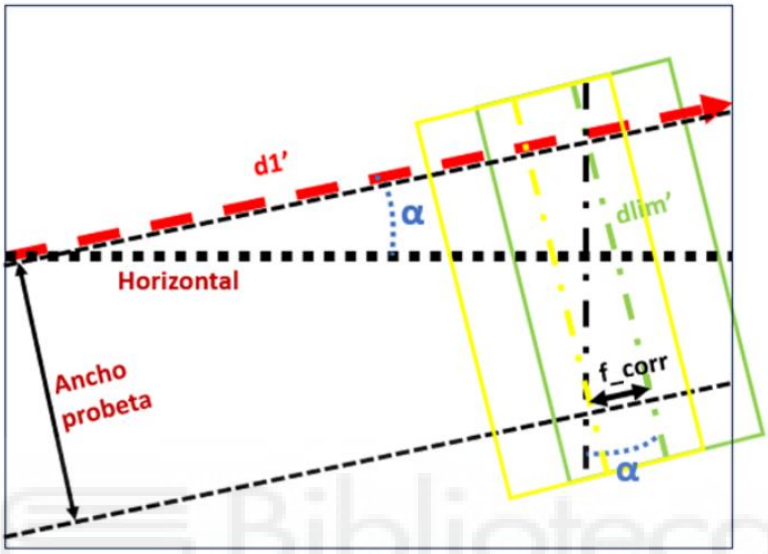
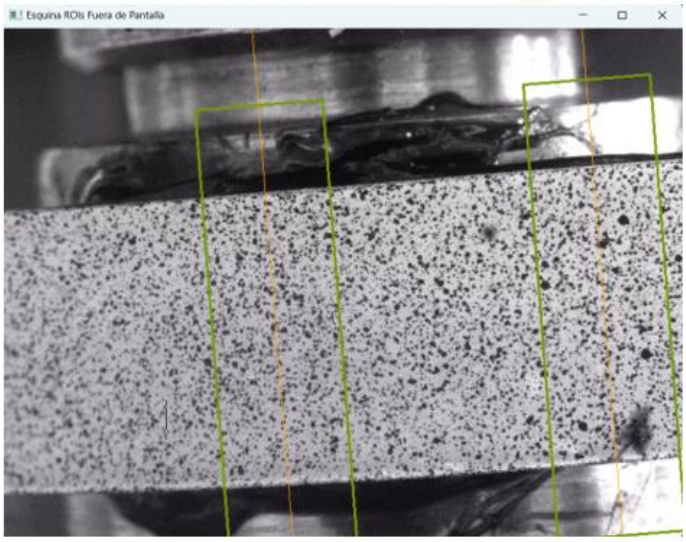


Figura 4.30. Boceto con geometría básica para el cálculo de f_corr.

$$f_{corr} = \tan(\alpha) \cdot ancho\_probeta \quad (30)$$



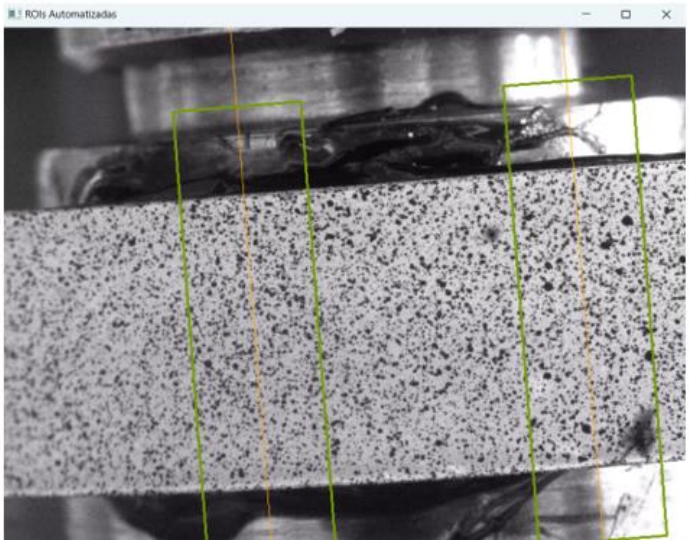


Figura 4.31. Ejemplo de esquina inferior ROI derecha fuera de pantalla. BAO-53.

Como se puede observar en la Figura 4.30, cuanto mayor sea α, mayor será el área de la región que queda fuera de la pantalla (ROI verde). Para asegurarse que toda la región se encuentra dentro de la ventana se ha de utilizar esta corrección cuando sea necesario, como ocurre en la Figura 4.31.

```
arreglado = False
  while(arreglado == False):
        arreglado = True
        if ((d_roi2-d_roi1) < roi_width) and arreglado == True:
            frame0 = frame.copy()
            pintar_rois(contact1_p1, contact1_p2, contact2_p1, contact2_p2, R1A, R1B, R2A, R2B, R1C, R1D, R2C, R2D, frame0)
            print("Cuidado. Superposición de ROIs. Recalculando....")
            d_roi2 = d_roi2 + roi_width
            d_roi1 = d_roi1 - roi_width
            cv2.namedWindow("ROIs Superpuestas")
            while True:
                out.write(frame0)
                cv2.imshow("ROIs Superpuestas", frame0)
                if cv2.waitKey(20) & 0xFF == 13:
                    break
            cv2.destroyWindow("ROIs Superpuestas")
            print("Recalculado.")
            arreglado = False

        desplazamiento = np.cos(angle)*(800-punto_pin_1[0])
        if (d_roi2 > (np.cos(angle)*(800-punto_pin_1[0])) - (roi_width/2) - 25) and arreglado == True:
            frame1 = frame.copy()
            pintar_rois(contact1_p1, contact1_p2, contact2_p1, contact2_p2, R1A, R1B, R2A, R2B, R1C, R1D, R2C, R2D, frame1)
            print("La ROI2 está fuera de los límites. Recalculando...")
            d_roi2 = desplazamiento*3/4
            d_roi1 = desplazamiento/4
            cv2.namedWindow("ROIs Fuera de Pantalla")
            while True:
                out.write(frame1)
                cv2.imshow("ROIs Fuera de Pantalla", frame1)
                if cv2.waitKey(20) & 0xFF == 13:
                    break
            cv2.destroyWindow("ROIs Fuera de Pantalla")
            print("Recalculado.")
            arreglado = False

        f_corr = (np.cos(angle)*abs(punto_pin_2[1]-punto_pin_1[1]))*np.tan(angle)
        if ((R2C[0] > 800) or R2D[0] > 800) and arreglado == True:
            frame2 = frame.copy()
            pintar_rois(contact1_p1, contact1_p2, contact2_p1, contact2_p2, R1A, R1B, R2A, R2B, R1C, R1D, R2C, R2D, frame2)
            print("Una de las dos esquinas derechas de ROI2 se encuentra fuera de la pantalla. Recalculando...")
            d_roi2 = d_roi2 - f_corr
            d_roi1 = d_roi1 - f_corr
            cv2.namedWindow("Esquina ROIs Fuera de Pantalla")
            while True:
                out.write(frame2)
                cv2.imshow("Esquina ROIs Fuera de Pantalla", frame2)
                if cv2.waitKey(20) & 0xFF == 13:
                    break
            cv2.destroyWindow("Esquina ROIs Fuera de Pantalla")
            print(f"Recalculado. Se han desplazado hacia la izquierda ambas regiones {f_corr:.2f} pix.")
            arreglado = False
```

VI. Dibujar ROIs

Por último, una vez validadas, se dibujan en pantalla usando la función *pintar_rois*, que traza los contornos y los ejes centrales.

```python
def pintar_rois(contact1_p1, contact1_p2, contact2_p1, contact2_p2, R1A, R1B, R2A, R2B, R1C, R1D, R2C, R2D, framei):
    """
        Dibuja las ROIs y sus ejes centrales automáticamente.

        :param contact1_p1: Primer punto del eje 1 extendido.
        :param contact1_p2: Segundo punto del eje 1 extendido.
        :param contact2_p1: Primer punto del eje 2 extendido.
        :param contact1_p2: Segundo punto del eje 2 extendido.
        :param R1A, R1B, R1C, R1D: Coordenadas de los vértices de la ROI1.
        :param R2A, R2B, R2C, R2D: Coordenadas de los vértices de la ROI2.
        :param framei: Imagen donde se mostrarán las ROIs.
        :return: ROIS como rectángulos
        """
    cv2.line(framei, contact1_p1, contact1_p2, (0, 165, 255), 1)  # Línea en ROI1
    cv2.line(framei, contact2_p1, contact2_p2, (0, 165, 255), 1)  # Línea en ROI2

    roi1 = [R1A, R1C, R1D, R1B]
    roi2 = [R2A, R2C, R2D, R2B]
    roi1 = np.array(roi1, dtype=np.int32)
    roi2 = np.array(roi2, dtype=np.int32)

    cv2.polylines(framei, [roi1], isClosed=True, color=(0, 158, 115), thickness=2)
    cv2.polylines(framei, [roi2], isClosed=True, color=(0, 158, 115), thickness=2)

    return roi1, roi2
```


4.4.2 ROIs manuales

Además del sistema automático, el programa ofrece la posibilidad de definir manualmente las Regiones de Interés (ROI) en aquellos casos en los que el usuario desee un mayor control o cuando la geometría del ensayo no se ajuste adecuadamente al modelo automático.

El proceso es simple e interactivo: al seleccionar esta opción, el programa muestra el primer fotograma del vídeo y solicita al usuario que haga clic sobre cuatro puntos para definir la ROI como un polígono cuadrilátero. El orden en que se seleccionen los vértices debe ser el siguiente:

- Esquina superior izquierda (RA).
- Esquina superior derecha (RC).
- Esquina inferior derecha (RD).
- Esquina inferior izquierda (RB).

Esto garantiza que la forma de la ROI sea coherente, y que el algoritmo de seguimiento posterior pueda interpretar correctamente los límites del área.

```python
def draw_roi(event, x, y, flags, param):
    """
    Callback para capturar los clics del ratón y dibujar los puntos.
    """
    global points
    if event == cv.EVENT_LBUTTONDOWN:
        points.append((x, y))
        cv2.circle(param, (x, y), 5, (0, 158, 115), -1)

def seleccionar_roi_con_puntos(frame, out, color):
    """
    Permite al usuario seleccionar una ROI dibujando un polígono a partir de 4 puntos.
    """
    global points
    points = []
    temp_frame = frame.copy()

    print("Haz clic en 4 puntos para definir la ROI. Pulsa 'c' para confirmar.")

    cv2.imshow("Seleccionar ROI", temp_frame)
    cv2.setMouseCallback("Seleccionar ROI", draw_roi, param=temp_frame)

    while True:
        out.write(temp_frame)
        cv2.imshow("Seleccionar ROI", temp_frame)
        key = cv.waitKey(1) & 0xFF

        if len(points) == 4:
            print("ROI confirmada.")
            break

    roi_contour = np.array(points, dtype=np.int32)
    cv2.polylines(frame, [roi_contour], isClosed=True, color=color, thickness=2)

    cv2.destroyWindow("Seleccionar ROI")
    return roi_contour
```

Figura 4.34. Dic Processing. Función seleccionar_roi_con_puntos y callback de la función draw_roi.

4.5. Distribución de Puntos Característicos

Una de las partes más importantes del sistema de análisis es la correcta definición de los puntos de interés que se utilizarán para estimar el movimiento del material a lo largo del ensayo. Estos puntos serán los que se rastreen entre fotogramas consecutivos mediante las técnicas de flujo óptico, por lo que su distribución inicial debe ser robusta, precisa y representativa del área de interés.

Por todo ello, se decidió desarrollar un sistema que permita elegir entre una distribución automática o una manual de estos puntos, en función de las condiciones del experimento y de los requisitos de análisis.

4.5.1 Distribución automática basada en la geometría de la probeta

La distribución automática se genera en forma de malla rectangular de puntos, alineada con su eje longitudinal y perfectamente adaptada a su orientación angular, ya que sigue la geometría definida previamente por el usuario (Capítulo 4.3).

Para definir esta malla, el programa utiliza como referencia:

- El **eje X de la probeta**, definido por dos clics al inicio del programa.
- El **ancho de probeta**, también definido al inicio del programa.
- El **ancho de las ROIs**, para garantizar que en todo momento haya puntos dentro de ellas.

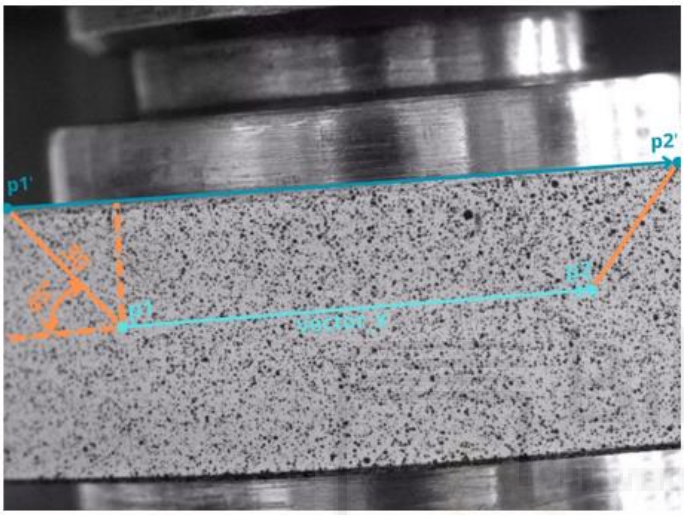
Para calcular la matriz de puntos se utilizan dos vectores, el vector que define el eje X de la probeta y el vector que define el ancho de la probeta. Con estos dos vectores y un valor para el número de columnas y filas sería suficiente. Sin embargo, se han llevado a cabo algunas especificaciones extra para asegurar el correcto funcionamiento en situaciones reales.

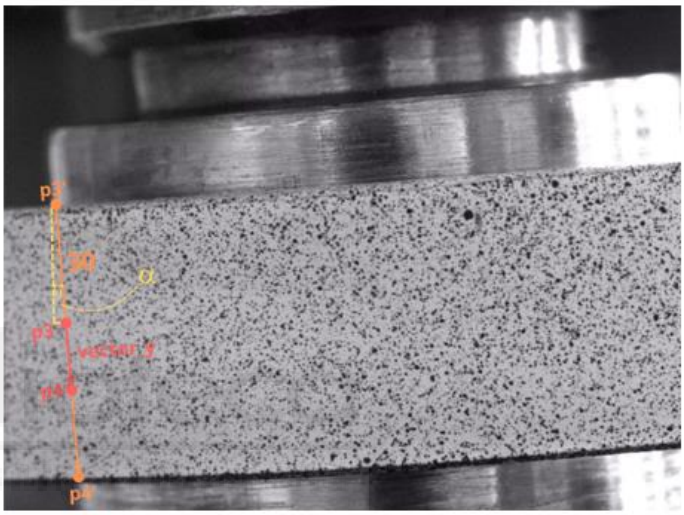
I. Reducción del área de distribución

En un primer momento, se diseñó el sistema para funcionar directamente utilizando los vectores recibidos por el usuario, pero al definir el eje X sobre el extremo superior de la probeta, la primera fila de la matriz y la última se encontraban demasiado cerca

del borde de la probeta (zonas que suele tener menos textura o quedan mal definidas por sombras o brillos).

Para evitar esta casuística se decidió introducir un parámetro de reducción diagonal (45º) para el vector que define el eje de la probeta (*vector_x*) y otro en función de la orientación de la probeta (α) para el vector que define el ancho (*vector_y*), ambos de valor de 30 píxeles. Esto garantiza que los puntos no queden al filo de la probeta y que se mantenga una zona de seguridad visual alrededor del área efectiva de análisis.





En la Figura 4.35 podemos observar cómo se diseñó la reducción de estos vectores. Para que se pueda apreciar visualmente se ha ampliado el tamaño y no está a escala.

```
reduccion = 30 #Píxeles que se quiere reducir el área de distribución de puntos para que no se sitúen en los bordes
p1_x, p1_y = axis_points[0] # Primer punto del eje X (x,y)
p1 = (p1_x + reduccion*np.cos(np.radians(45)), p1_y + reduccion*np.sin(np.radians(45)))
p2_x, p2_y = axis_points[1] # Segundo punto del eje X
p2 = (p2_x - reduccion*np.sin(np.radians(45)), p2_y + reduccion*np.cos(np.radians(45)))
p3_x, p3_y = width[0] # Primer punto del ancho
p3 = (p3_x + reduccion*np.sin(angle), p3_y + reduccion*np.cos(angle))
p4_x, p4_y = width[1] # Segundo punto del ancho
p4 = (p4_x - reduccion*np.cos(np.radians(90)-angle), p4_y - reduccion*np.sin(np.radians(90)-angle)), dtype=np.int32)
```

De esta forma, tenemos definidos los puntos que conforman los vectores definitivos, conformando así una matriz de puntos que nos asegura un estudio eficaz y seguro de la superficie de la probeta

II. Cálculo de la densidad de la matriz

Es muy importante decidir cuál va a ser la cantidad de puntos que van a conformar dicha matriz debido a que en todo momento debe haber puntos dentro de ambas ROIs.

Para asegurar este aspecto, se calcula el ancho de las ROIs izquierda y derecha, y se selecciona el menor de ambos. Este valor sirve para establecer una distancia de separación máxima entre columnas que asegure que siempre haya puntos dentro de ambas ROIs.

El número de columnas se calcula como el cociente entre el módulo del *vector_x* y el mínimo ancho de ambas ROIs. Una vez calculado el número de columnas justas que aseguran que siempre haya puntos dentro de ambas ROIs, se aplica un margen de seguridad para asegurarse de que esto se cumpla, en cualquier caso. Dicho margen consta de 3 columnas extras y 1 optativa si el decimal del primer resultado de *N_columns* es menor que 5. Sirva como ejemplo el ensayo BAO-46:

$p2[0] - p1[0] = 756.6\ pix$

$min\_ancho\_roi = 150\ pix$

$N\_columns = 756.6/150 = 5.01$

$N\_columns = int(5.01 + 3 + 1) = 9$

$N\_rows = 5$

De esta forma, nuestra matriz estará conformada por 45 puntos distribuidos en 5 filas y 9 columnas. Nótese como el número de filas es variable y no afecta a la condición de siempre puntos en el interior de las regiones ya que la probeta se desplazada longitudinalmente.

```
# Parámetros para generar puntos
ancho_roi1 = roi1[1][0] - roi1[0][0]
ancho_roi2 = roi2[1][0] - roi2[0][0]
min_ancho_rois = min(int(ancho_roi1), int(ancho_roi2))
largo_probeta = p2[0] - p1[0]

N_columns = largo_probeta/min_ancho_rois
N_columns = int(N_columns) + 3 + (1 if N_columns % 1 < 0.5 else 0) # Si parte decimal menor de 0.5 en N_columns se suma 1

N_rows = 5  # Número de filas de puntos
```

III. Generación de puntos

Una vez calculadas las dimensiones del área y el número de columnas y filas, se llama a la *función generar_puntos_automaticamente*, que interpola cada punto entre los extremos *p1-p2* (horizontal) y *p3-p4* (vertical), generando así la malla rectangular perfectamente alineada con la probeta.

```python
def generar_puntos_automaticamente(p1, p2, p3, p4, N_columns, N_rows):
    """
    Genera puntos distribuidos uniformemente en columnas sobre el área de la probeta.

    Parámetros:
    - p1, p2: Extremos del eje X de la probeta.
    - p3, p4: Extremos del ancho de la probeta.
    - N_columns: Número de columnas de puntos.
    - N_rows: Número de filas de puntos.

    Retorna:
    - Lista de coordenadas [(x, y)] para los puntos generados.
    """
    # Calcular vectores de base a partir de los puntos
    vector_x = (p2[0] - p1[0], p2[1] - p1[1])  # Vector a lo largo del largo de la probeta
    vector_y = (p4[0] - p3[0], p4[1] - p3[1])  # Vector a lo largo del ancho de la probeta

    # Generar los puntos distribuidos
    puntos = []
    for col in range(N_columns):
        for row in range(N_rows):
            # Calcular posición interpolando entre los extremos
            x = p1[0] + (col / (N_columns - 1)) * vector_x[0] + (row / (N_rows - 1)) * vector_y[0]
            y = p1[1] + (col / (N_columns - 1)) * vector_x[1] + (row / (N_rows - 1)) * vector_y[1]
            puntos.append((x, y))
    return puntos
```

Esta función asegura que todos los puntos queden distribuidos uniformemente, sin depender de la orientación angular del ensayo.

La Figura 4.39 muestra un ejemplo de matriz de puntos para el ensayo BAO-46, como se puede observar, todos los puntos aparecen en color rojo y sin ninguna numeración, en el Capítulo 4.6 veremos cómo al entrar los puntos dentro de las ROIs su color cambia a verde, cada punto está enumerado y se implementa un contador de tiempo.

Figura 4.39. Matriz de puntos generada. BAO-46.

4.5.2. Distribución manual

Además del modo automático, el sistema también permite al usuario indicar manualmente los puntos característicos que desee seguir durante el análisis. Esto se hace mediante una interfaz interactiva en la que el usuario puede ir haciendo clics sobre la imagen para definir puntos de interés.

Cada clic:

- Almacena la coordenada (x,y).
- Dibuja un círculo rojo sobre la imagen.
- Numera el punto para facilitar el seguimiento.

La selección termina cuando se pulsa ‘enter’ o la tecla ‘q’.

Este método es útil en ensayos donde:

- La geometría de la probeta no es estándar o está parcialmente oculta.
- Se requiere enfocar el análisis en zonas específicas (grietas, discontinuidades…)
- Hay ROIs personalizadas no alineadas con la malla regular.

```python
def seleccionar_puntos_manual(frame, out):
    """
    Permite al usuario seleccionar manualmente puntos en el frame haciendo clic con el ratón.
    Los puntos seleccionados se quedarán visibles en la imagen con un tamaño más grande.

    Args:
        frame: La imagen o el frame sobre el cual se seleccionarán los puntos.

    Returns:
        list: Lista de puntos seleccionados [(x, y), ...].
    """
    points = []

    def click_event(event, x, y, flags, param):
        """
        Callback para capturar clics del ratón y dibujar puntos grandes en la imagen.
        """
        nonlocal frame
        if event == cv.EVENT_LBUTTONDOWN:
            point = (x, y)
            points.append(point)
            print(f"Punto añadido: {point}")

            # Dibujar el punto grande en la imagen
            cv2.circle(frame, point, 3, (50, 17, 230), -1)  # Color rojo
            cv2.putText(frame, f"{len(points)}", (x + 10, y - 10), cv.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 1)

            # Actualizar la ventana con el punto añadido
            cv2.imshow("Seleccionar puntos", frame)

    cv2.setMouseCallback("Seleccionar puntos", click_event)  # Vincular el callback de ratón
    cv2.namedWindow("Seleccionar puntos")

    print("Haz clic para seleccionar puntos. Pulsa 'q' cuando termines.")
    while True:
        out.write(frame)
        cv2.imshow("Seleccionar puntos", frame)
        key = cv.waitKey(1) & 0xFF
        if key == ord('q'):
            break

    cv2.destroyWindow("Seleccionar puntos")
    return points
```

Figura 4.40. Dic Processing. Función seleccionar_puntos_manual.

En la Figura 4.41 se puede observar cómo podría ser una disposición manual de puntos a lo largo de la probeta, centrando el estudio sobre una zona específica o posicionando los puntos de una manera concreta debido a ciertos impedimentos geométricos.

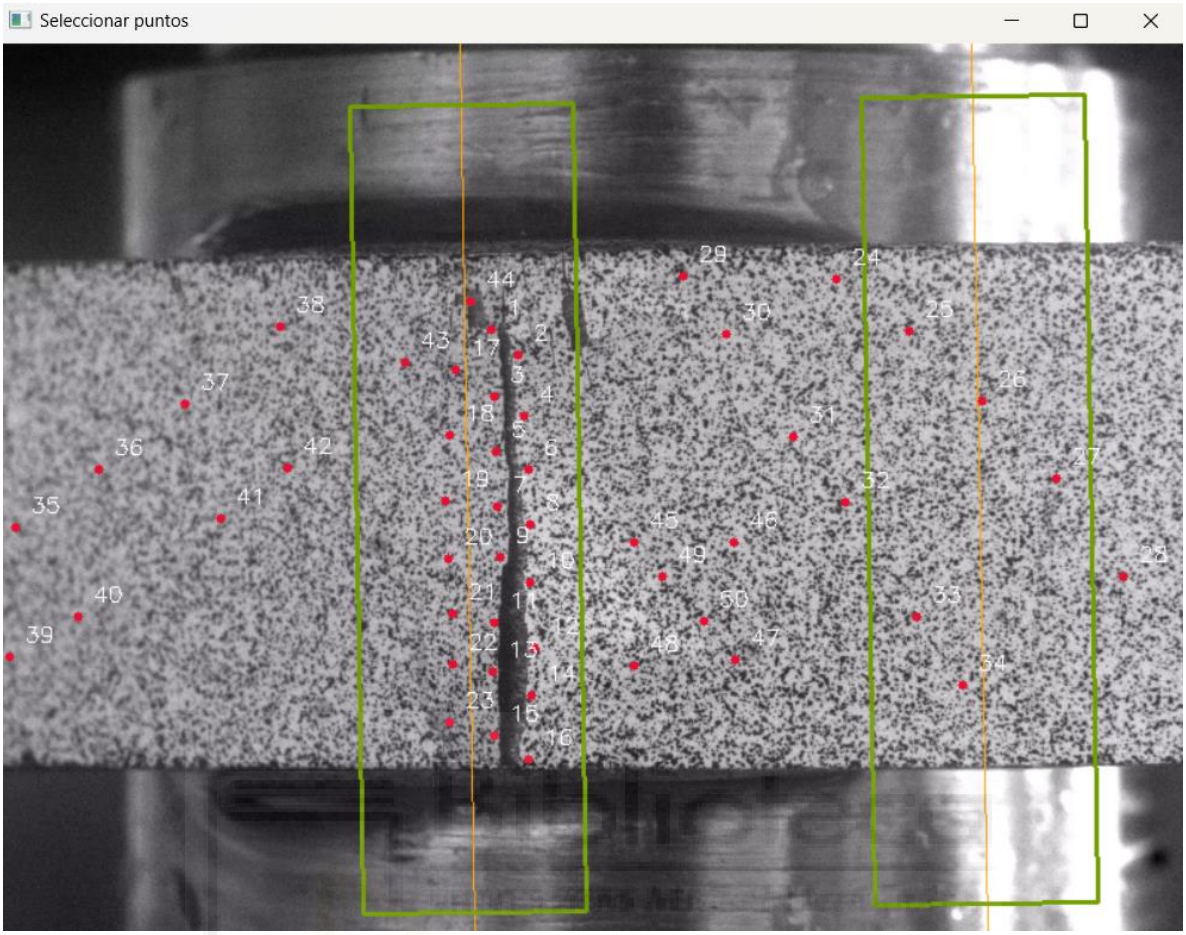


Figura 4.41. Ejemplo de disposición de puntos manual para centrar el estudio sobre una grieta. BAO-37.

4.6. Seguimiento de Puntos con Flujo Óptico

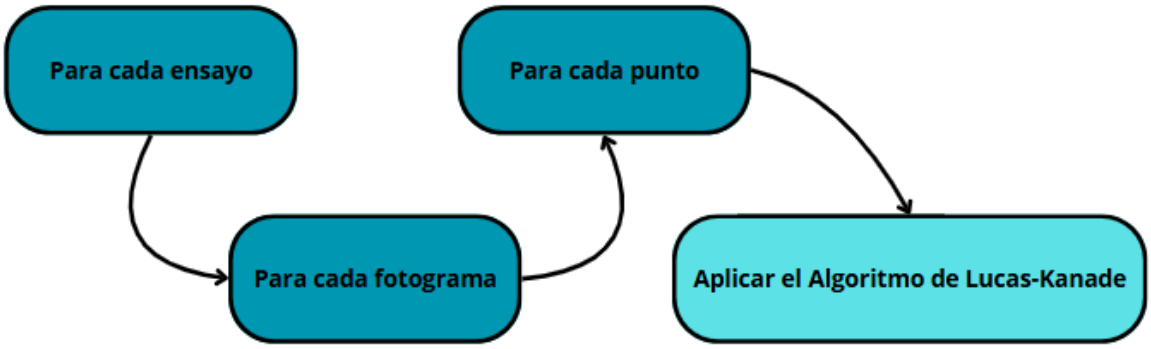
El objetivo de esta sección es realizar el seguimiento espacial de los puntos característicos distribuidos previamente sobre la probeta durante el ensayo. Para ello, se utiliza el algoritmo Lucas-Kanade Piramidal a través de la función ***cv2.calcOpticalFlowPyrLK*** (Capítulo 3.3). Este método permite estimar el desplazamiento de cada punto entre fotogramas consecutivos con alta precisión, incluso en presencia de movimiento rápido o iluminación variable.

El sistema está diseñado para realizar múltiples ejecuciones consecutivas de un mismo ensayo (por defecto, 10), permitiendo así suavizar el ruido propio de los vídeos mediante el cálculo de medias.

4.6.1. Arquitectura del seguimiento

Para cada ensayo se estructura un bucle anidado en tres niveles:

- **Bucle externo**: itera sobre el número de repeticiones (*num_ensayos*).
- **Bucle medio**: recorre todos los fotogramas del vídeo.
- **Bucle interno**: analiza individualmente cada punto.



4.6.2. Detalles del seguimiento

Antes de comenzar el primer ensayo, se realiza una copia de la posición de los puntos originales (*initial_points*) para poder reiniciar su posición entre repeticiones. Se definen estructuras de datos (listas y diccionarios) para almacenar:

- La evolución de los puntos en **píxeles** y en **milímetros corregidos**.
- Las **velocidades** para cada punto.
- Los **errores** de seguimiento estimados por Lucas-Kanade.
- La media de estos valores en cada ROI, en cada fotograma.

En cada iteración del bucle interno (por punto), se utiliza el siguiente fragmento visto en el Capítulo 3.3, en concreto la Figura 3.5.

Si el seguimiento es exitoso, se calcula el desplazamiento horizontal (*dx*) utilizando las nuevas coordenadas del punto y las del fotograma anterior, y se actualizan las nuevas coordenadas de ese punto.

Si no fue exitoso, el diferencial de desplazamiento (*dx*) será nulo y las nuevas coordenadas serán las del fotograma anterior (como si el punto no se hubiera desplazado).

```python
for i, center_point in enumerate(points):

    # Realizar el seguimiento empleando el metodo de Lucas-Kanade Piramidal
    flow, status, err = cv.calcOpticalFlowPyrLK(
        gray_prev, gray_curr, np.array([center_point], dtype=np.float32), None
    )
    """
    flow: contiene las coordenadas calculadas para los puntos en el frame actual
    status: array binario que indica si el seguimiento del punto fue existoso(1) o no(0)
    err: representa el error de prediccion para cada punto, basado en la precision del modelo de flujo óptico
    """
    if flow is not None and status[0] == 1:
        dx = flow[0][0] - points[i][0] # Se usa para obtener la velocidad
        points[i] = tuple(flow[0])  # Actualizar las coordenadas del punto
    else:
        dx = points[i][0] - points[i][0]
        points[i] = tuple(points[i])
```

Figura 4.43. Main. Gestión del seguimiento de los puntos exitoso/defectuoso.

Posteriormente, se calcula el desplazamiento horizontal corregido (*dx_corr*) y la velocidad en $\frac{mm}{s}$. Donde $\emptyset$ es el ángulo de la probeta, $scale$ es la escala en $\frac{mm}{pix}$, y fps los fotogramas por segundo.

$$dx_{corr} = \frac{dx}{\cos(\emptyset)} \cdot scale \tag{31}$$

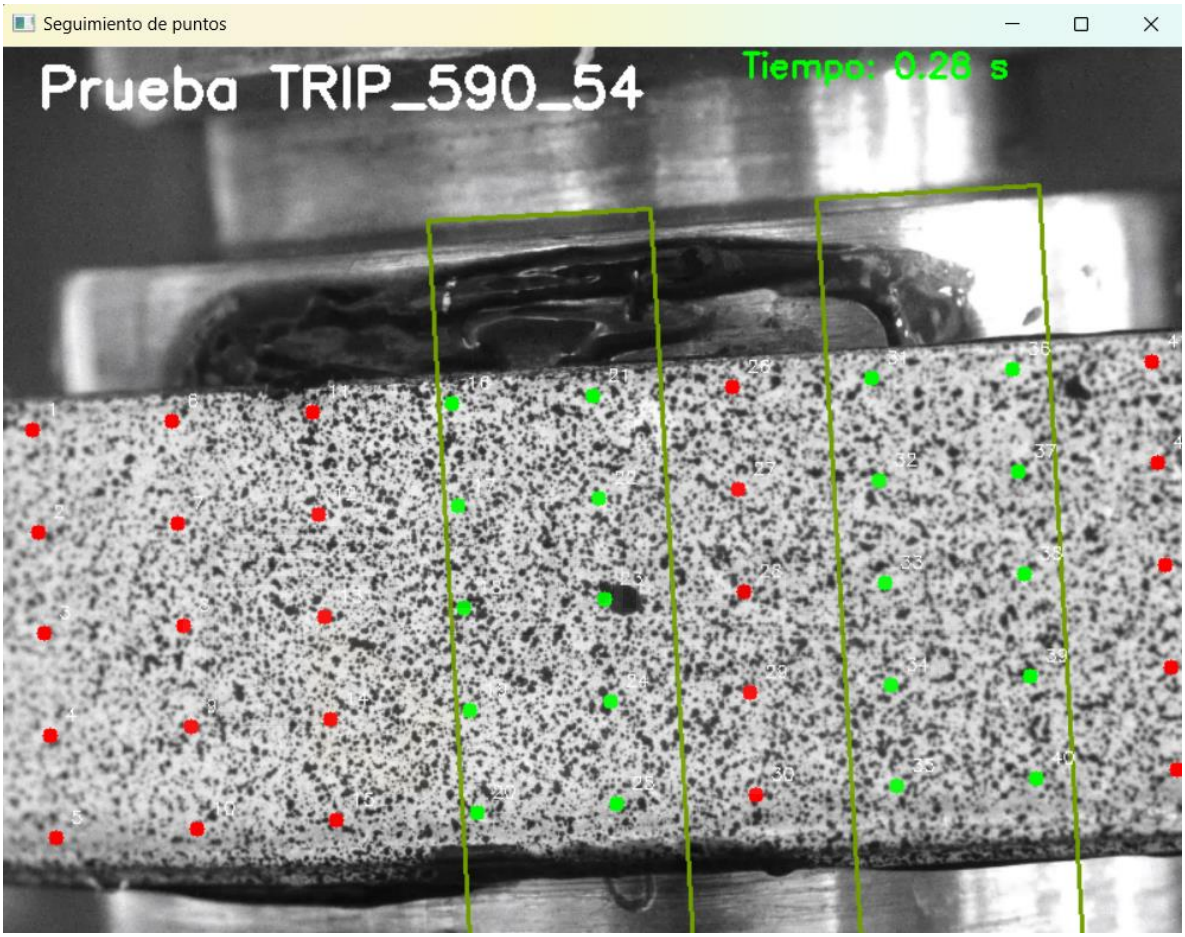
$$v = dx_{corr} \cdot fps \tag{32}$$

Estas velocidades se almacenarán en diferentes listas en función de la posición del punto, es decir, si el punto está dentro de ROI1, su velocidad se almacenará en la lista de velocidades de ROI1, de la que posteriormente se hará una media.

```python
dentro_roi1 = cv.pointPolygonTest(roi1, center_point, False) > 0
dentro_roi2 = cv.pointPolygonTest(roi2, center_point, False) > 0
if dentro_roi1:
    color = (0, 255, 0)  # Green
    velocidades_puntos_roi1.append(velocidad)
    err_roi1.append(err[0][0])
    mm_corrected_roi1.append(keypoints_x_mm_corrected[i][-1])
elif dentro_roi2:
    color = (0, 255, 0) # Green
    velocidades_puntos_roi2.append(velocidad)
    err_roi2.append(err[0][0])
    mm_corrected_roi2.append(keypoints_x_mm_corrected[i][-1])
else:
    color = (0, 0, 255) # Red
    err_out.append(err[0][0])
    velocidades_puntos_out.append(velocidad)

# Display the points and his identifier
cv.circle(frame_visual, (int(points[i][0]), int(points[i][1])), 5, color, -1)
cv.putText(frame_visual, f"{i + 1}", (int(points[i][0]) + 10, int(points[i][1]) - 10), cv.FONT_HERSHEY_SIMPLEX, 0.4, (255, 255, 255),1)
```

Figura 4.44. Main. Gestión de la posición de la posición de cada punto.

Para detectar si el punto está dentro de alguna región se ha utilizado la función ***cv2.pointPolygonTest***, que permite comprobar si un punto pertenece a un polígono (en este caso, las ROIs). Si se detecta que el punto está dentro de alguna ROI se cambia su color de rojo a verde mejorando así la visualización del vídeo.



4.6.3. Separación de ventanas: seguimiento vs visualización

Una mejora crítica implementada fue la separación entre:

- **Ventana de seguimiento** (*gray_prev*, *gray_curr*): se utilizan imágenes en escala de grises, sin ningún cambio tipo de anotación visual, para que el algoritmo de flujo óptico no se vea alterado por cambios de intensidad provocados por las superposiciones gráficas.
- **Ventana de visualización** (*frame_visual*): donde se pintan las ROIs, los puntos, los textos, tiempo y otros indicadores.

Esta decisión fue clave tras observar que, en vídeos de baja calidad, algunos puntos dejaban de ser detectados si estaban en el borde de una ROI o si se pintaban elementos encima de ellos, ya que modificaban la intensidad de los píxeles y afectaban al seguimiento.

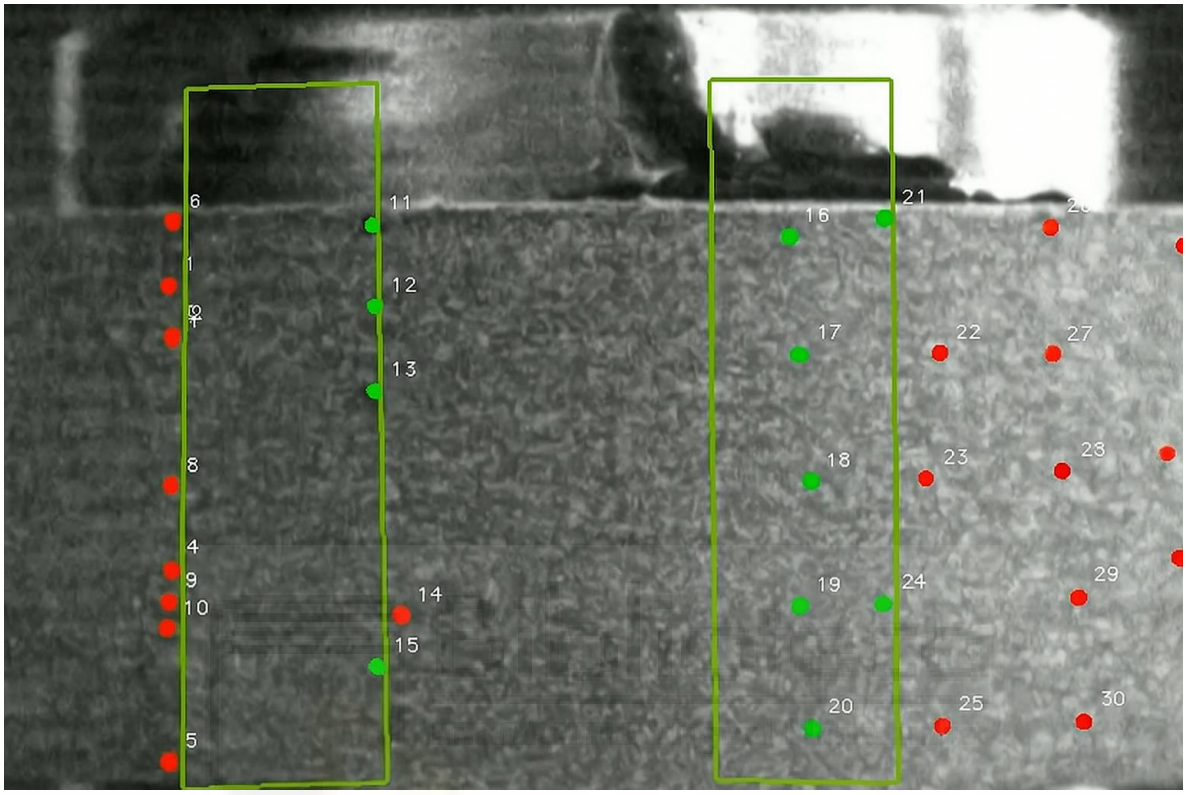


Figura 4.46. Ejemplo de mal seguimiento en bordes ROI1.

4.6.4. Gestión de múltiples ensayos

Para mejorar la robustez frente al ruido, se ejecutan varios ensayos consecutivos (*num_ensayos*). Al finalizar cada uno se reinicia la posición de los puntos iniciales con un pequeño desplazamiento de $\pm$ 6 píxeles.

```
points = [(p[0] + np.random.uniform(-6, 6), p[1] + np.random.uniform(-6, 6)) for p in initial_points]
```

Figura 4.47. Main. Desplazamiento aleatorio de la posición inicial de los puntos para el comienzo de un nuevo ensayo.

Esta variación permite simular distintos inicios de punto de vista y mejorar la media de comportamiento del material, mitigando errores introducidos por píxeles aislados o condiciones lumínicas.

4.7. Detección de Rotura de la Probeta

La detección de la rotura de la probeta se ha implementado como un proceso automático basado en la evolución de los valores de error y velocidad obtenidos a partir del algoritmo de Lucas-Kanade. Esta detección se realiza en tiempo real durante el seguimiento de los puntos característicos, permitiendo detener el procesamiento del vídeo inmediatamente tras la rotura, y dejando un segundo adicional para su visualización.

4.7.1. Umbrales de error y velocidad

El criterio de detección se basa en los incrementos de error (*delta_e*) y variaciones de velocidad (*delta_v*) en los puntos contenidos dentro de las regiones de interés. Para ello, se definen dos umbrales:

- ***umbral_error***: representa un salto significativo en el error del seguimiento.
- ***umbral_velocidad***: define una aceleración anómala que podría indicar rotura.

Ambos umbrales están adaptados a la velocidad de ensayo, ya que en ensayos a altas velocidades (por ejemplo, 15 mm/s) se producen incrementos bruscos de velocidad al inicio que podrían confundirse con una rotura si no se ajustan correctamente. A continuación, se muestra cómo se seleccionan estos umbrales en función de la velocidad del ensayo.

```
# Definición del umbral de error y velocidad para detectar rotura
    if vel_ensayo == 3:
        umbral_error = 8
        umbral_velocidad = vel_ensayo * 4
    elif vel_ensayo == 7:
        umbral_error = 6
        umbral_velocidad = vel_ensayo * 2.5
    elif vel_ensayo == 15:
        umbral_error = 7
        umbral_velocidad = vel_ensayo * 2.5
    else:
        print("ERROR: Velocidad no válida")
        exit()
```

Figura 4.48. Main. Selección de umbrales en función de la velocidad de ensayo.

4.7.2. Lógica de detección en tiempo real

Durante el seguimiento de puntos, en cada fotograma se evalúa si alguno de los siguientes criterios se cumple:

- Un incremento brusco de error y velocidad media en cualquiera de las ROIs.
- Una velocidad extremadamente elevada en cualquiera de las regiones (más de 200 mm/s)

Cuando uno de estos casos ocurre, se activa el evento rotura y se guarda el instante temporal correspondiente (*tiempo_rotura*). A partir de ese momento, se muestra un contador con el tiempo transcurrido hasta la rotura, seguido de un segundo adicional para permitir observar visualmente la fractura sin procesar más vídeo innecesariamente.

```
if len(error_promedio_out) >= 2:
    delta_e1 = error_promedio_roi1[-1] - error_promedio_roi1[-2]
    delta_e2 = error_promedio_roi2[-1] - error_promedio_roi2[-2]
    delta_v1 = velocidades_roi1[-1] - velocidades_roi1[-2]
    delta_v2 = velocidades_roi2[-1] - velocidades_roi2[-2]
else:
    delta_e1 = 0
    delta_e2 = 0
    delta_v1 = 0
    delta_v2 = 0

# Detección de la rotura en la probeta
if rotura_detectada == False and (
        (delta_e1 > umbral_error and abs(delta_v1) > umbral_velocidad) or (
        delta_e2 > umbral_error and abs(delta_v2) > umbral_velocidad) or (
        abs(delta_v1) > 200 or abs(delta_v2) > 200)):
    tiempo_rotura = frame_idx / fps
    rotura_detectada = True
    print(f"Rotura detectada en el segundo: {tiempo_rotura:.2f} s")
    indice_rotura = frame_idx

# Posicionamiento de textos sobre el video
cv.putText(frame_visual, f"Prueba {material}_{video_number}", (25,40), cv.FONT_HERSHEY_SIMPLEX, 1.25, (255, 255, 255), 3)
if rotura_detectada == False:
    cv.putText(frame_visual, f"Tiempo: {frame_idx / fps:.2f} s", (500,20), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
else:
    tiempo_contador = tiempo_rotura + 1
cv.putText(frame_visual, f"Rotura a los {tiempo_rotura:.2f} s", (500, 20), cv.FONT_HERSHEY_SIMPLEX, 0.75,(0, 0, 255), 2)
cv.putText(frame_visual, f"Tiempo: {frame_idx / fps:.2f} s / {tiempo_contador:.2f} s", (500,45), cv.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255,
0), 2)
if tiempo_contador == frame_idx / fps:
    break
```

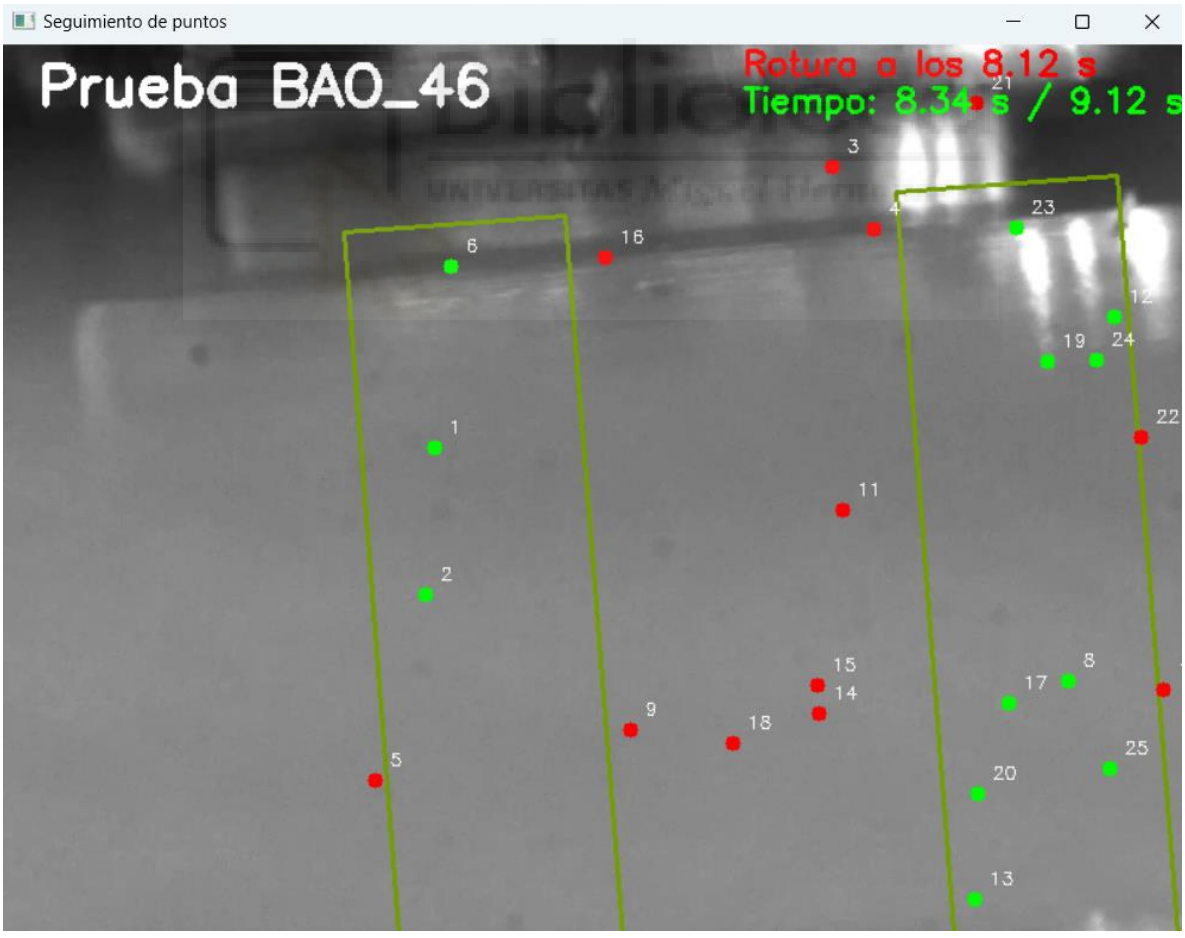
Esta detección inmediata permite una mayor eficiencia en el procesamiento, especialmente en vídeos largos o de alta resolución.

4.7.3. Consideraciones e inconvenientes

La fiabilidad de la detección depende en gran medida de la calidad del vídeo. En algunos casos, especialmente en vídeos con baja resolución o alto ruido, los incrementos de error pueden ser falsos positivos o estar muy dispersos. Por ello, es posible que se requiera ajustar manualmente los umbrales en función de la calidad del vídeo y el comportamiento del material.

Del mismo modo, pequeños errores de alineación, falta de contraste o variaciones en la iluminación pueden afectar a la sensibilidad del seguimiento y, por tanto, a la correcta identificación del instante de rotura.

Sin embargo, esta detección automática ha sido muy eficaz a lo largo de todos los ensayos de este proyecto y su implementación aporta un gran valor a su viabilidad.



4.8. Detección de movimiento (t_0)

El instante t_0 representa el momento exacto en el que la probeta inicia su movimiento, es decir, cuando comienza la tracción efectiva. Este momento se detecta de forma **a posteriori**, una vez finalizado el seguimiento de puntos y tras haber detectado el instante de rotura.

Detectar con precisión el t_0 es fundamental para poder analizar únicamente el intervalo útil del ensayo, descartando los segundos previos (sin movimiento) y evitando errores producidos tras la rotura, momento en el cual los puntos se desplazan de forma abrupta y descontrolada.

La detección se basa en tres listas generadas durante el rastreo de puntos:

- medias_mm_corrected_roi1: desplazamiento medio de los puntos dentro de ROI1.
- medias_mm_corrected_roi2: desplazamiento medio de los puntos dentro de ROI2.
- velocidades_roi2: velocidad media de los puntos en ROI2.

Se evalúan los desplazamientos absolutos y las velocidades corregidas para identificar el primer fotograma en el que ambas regiones presentan un desplazamiento o velocidad positiva mantenida.

```python
def detectar_t0(lista1, lista2, lista3, indice_rotura):
    """
    Detecta t0 basado en la velocidad de los puntos dentro de ROI_2 y recorta las listas.

    :param lista1: Lista de desplazamientos medios asociadas a los puntos dentro de ROI_1.
    :param lista2: Lista de desplazamientos medios asociadas a los puntos dentro de ROI_2.
    :param lista3: Lista de velocidades medias de la ROI2
    :param indice_rotura: fila donde se ha detectado la rotura, para evaluar únicamente hasta ese punto, pues posterior a la rotura los
           puntos se mueven drásticamente al perder la probeta.
    :return: t0 (nº de fotograma).
    """
    t0 = None
    for i in range(len(lista1)):
        if (all(val >= 0.01 for val in lista1[i:indice_rotura]) and all(val >= 0.01 for val in lista2[i:indice_rotura]))
            or all(val >= 0 for val in lista3[i:indice_rotura]):
            t0 = i + 1
            break

    if t0 is None:
        print("No se detectó un t0 válido.")
        return lista1, lista2, lista3, t0
    else:
        return t0
```

El valor t_0 se define como aquel **índice a partir del cual todos los desplazamientos** (o velocidades) **son positivos** hasta la rotura. En el caso de los desplazamientos han de ser mayores de 0.01 mm. Esto garantiza que no se considera un falso inicio debido a oscilaciones o ruido antes del movimiento real.

Importante: para que esta condición tenga sentido, es imprescindible recortar las listas hasta el momento de rotura, ya que los datos posteriores pueden estar contaminados por el colapso de la probeta. Por ejemplo, si la rotura ocurre a los 8.0 segundos, pero se detecta a los 9.0 segundos, el desplazamiento a los 8.5 segundos puede ser negativo por el movimiento caótico de los puntos, lo que invalidaría la detección si no se recortan los datos correctamente.

Por otro lado, debe haber siempre **algún punto dentro** de ambas ROIs para que se detecte correctamente. De no ser así, el desplazamiento medio en ese instante es nulo ($< 0.01\ mm$) y falla la detección automática.

4.9. Exportación de Datos a Excel y Grabación de Vídeos

Una vez finalizado el seguimiento de los puntos y realizadas las detecciones de t_0 y de rotura de la probeta, se lleva a cabo la exportación de los datos y la grabación de los vídeos procesados. Esta etapa tiene como objetivo almacenar de forma estructurada y reutilizable toda la información relevante extraída del análisis.

4.9.1. Organización de los datos exportados a Excel

Para cada ensayo se generan dos archivos de Excel distintos:

- {*material*}_{*video_number*}.xlsx (por ejemplo, BAO_46.xlsx)
- {*material*}_{*video_number*}_recortado.xlsx

Ambos contienen la misma estructura de hojas, pero en el segundo archivo todas las listas han sido recortadas automáticamente desde t_0 hasta el instante de rotura, usando como referencia los tiempos detectados en el primer ensayo.

➡ **Estructura de hojas:**

1. **Primera hoja (Med_x10)**

   Contiene los valores promedio de todos los ensayos realizados (por defecto, 10 repeticiones). Se presentan en 4 columnas:

   - Tiempo (s)
   - Velocidad media de ROI1 (mm/s)
   - Velocidad media de ROI2 (mm/s)
   - Diferencia de velocidades (ROI2 – ROI1) (mm/s)

2. **Hojas individuales por ensayo (E1, E2, …, E10)**

   Cada hoja contiene los datos de un ensayo individual y está estructurado de la misma manera que la primera hoja, pero además incluye:

   - Columna en blanco separadora
   - Desplazamiento de cada punto de interés (P1 (mm), P2 (mm), …), correspondiente a su posición corregida a lo largo del ensayo.

Estas hojas permiten tanto un análisis global del comportamiento medio del material como una revisión detallada de cada repetición individual. Como bien se comentó a lo largo del Capítulo 3.1, este proceso ha sido realizado utilizando *pandas* y *openpyxl*.

4.9.2. Grabación de los vídeos de ensayo

Durante el primer ensayo, se realiza también la grabación de un vídeo que documenta de forma visual todo el proceso seguido por el algoritmo. Esta grabación no solo tiene un valor explicativo y de validación, sino que también constituye una herramienta fundamental para comprobar el comportamiento de los puntos y las regiones de interés.

Por eso, se documenta todo el proceso inicial desde la indicación de eje X y ancho de probeta y la creación de ROIs y puntos hasta la rotura del primer ensayo.

La grabación se gestiona mediante el objeto ***cv2.VideoWriter*** de OpenCv. El siguiente fragmento de código muestra cómo se configura.

```
fourcc = cv.VideoWriter_fourcc(*'XVID')
video_filename = project_root / 'Record' / f'{material}'/ f"{material}_{video_number}.avi"
out = cv.VideoWriter(video_filename, fourcc, fps, (800, 600))
```

La grabación del vídeo se realiza frame a frame, utilizando el método ***out.write(frame)*** para el fotograma que se quiera grabar. Al finalizar el primer ensayo, el vídeo se cierra con ***out.release()***.

CAPÍTULO 5

RESULTADOS Y VALIDACIÓN

En este capítulo se analiza el rendimiento del sistema desarrollado a partir de los resultados obtenidos de distintos ensayos de tracción. Dado que no se dispone de un sistema de medida externo sincronizado con la grabación, la validación se basa en la coherencia interna de los datos, en comparaciones indirectas con registros de sensores y en el análisis de variables clave como el inicio del movimiento, la rotura, la velocidad y la dispersión de puntos. Todo esto permite evaluar el grado de fiabilidad del sistema y su utilidad.

5.1. Análisis comparativo de los ensayos

En este apartado del Capítulo 5 de Resultados y Validación se presenta un análisis comparativo de los diferentes ensayos realizados. El objetivo principal es poder identificar qué configuraciones experimentales otorgan un resultado más fiable y consistente en términos de seguimiento y ajuste de la curva de velocidad. Para ello, se ham clasificado y analizado los ensayos en función del diámetro del pin (19,1 mm, 22,1 mm y 26,1 mm), incluyendo también otras variables como la calidad del vídeo, la velocidad del ensayo, la temperatura, el tipo de lubricante y el ángulo de tracción.

Para cada grupo de ensayos, se han recopilado valores de error típico y coeficiente de determinación R^2 obtenidos a partir del ajuste de lineal de la curva de velocidad de ROI1. De esta forma, es posible evaluar el grado de linealidad y estabilidad del comportamiento medido por el sistema de visión por computador.

Las Tablas 5.1, 5.2 y 5.3 recogen los valores obtenidos del estudio de las gráficas de velocidad correspondientes a la región de interés izquierda. Como bien se ha comentado, las Tablas vienen organizadas en función del diámetro de pin, variando el tipo de lubricante, la velocidad de ensayo y la temperatura, mientras que el ángulo de tracción se deja libre.

En total, se han graficado y estudiado 47 configuraciones diferentes, de las cuales 30 son del material BAO y 17 del material TRIP. Cabe recordar que, el número de ensayo recoge las mismas configuraciones de los parámetros para los dos materiales.

Diámetro de pin igual a 19,1 mm								
Calidad	Velocidad	Lubricante	Temperatura	Ángulo	Diametro pin	Ensayo	error típico	R^2
2560x1920	3	Ninguno	25	60	19,1	BAO-35	0,439	0,65450
2560x1920	7	Ninguno	25	45	19,1	BAO-26	1,6672	0,1307
2560x1920	15	Ninguno	25	30	19,1	BAO-40	2,046	0,04200
2560x1920	3	Líquido	25	30	19,1	BAO-31	0,806	0,36250
2560x1920	7	Líquido	25	30	19,1	BAO-21	1,452	0,31140
2560x1920	15	Líquido	25	60	19,1	BAO-03	2,791	0,04550
	3	Sólido	25		19,1			
2560x1920	7	Sólido	25	30	19,1	TRIP-42	1,418	0,48090
2560x1920	15	Sólido	25	45	19,1	TRIP-49	2,044	0,40640
2560x1920	3	Ninguno	150	45	19,1	TRIP-34	1,198	0,1544
2560x1920	7	Ninguno	150	30	19,1	TRIP-08	1,992	0,49050
720x480	15	Ninguno	150	60	19,1	BAO-14	1,029	0,02190
720x480	3	Líquido	150	60	19,1	BAO-02	0,209	0,75170
720x480	7	Líquido	150	45	19,1	BAO-09	1,062	0,01950
720x480	15	Líquido	150	30	19,1	TRIP-13	2,027	0,33340
720x480	3	Sólido	150	30	19,1	BAO-19	0,512	0,50440
720x480	7	Sólido	150	60	19,1	BAO-25	0,852	0,0171

Tabla 5.1. Resultados de error típico y coeficiente de determinación para diámetro de 19,1 mm.

Diámetro de pin igual a 22,1 mm								
Calidad	Velocidad	Lubricante	Temperatura	Ángulo	Diametro pin	Ensayo	error típico	R^2
2560x1920	3	Ninguno	25	45	22,1	BAO-46	0,800	0,54280
2560x1920	7	Ninguno	25	30	22,1	TRIP-29	1,255	0,59840
2560x1920	15	Ninguno	25	60	22,1	TRIP-57	3,160	0,38170
	3	Líquido	25					
720x480	7	Líquido	25	45	22,1	TRIP-36	0,494	0,68920
2560x1920	15	Líquido	25	45	22,1	TRIP-11	2,653	0,24850
2560x1920	3	Sólido	25	30	22,1	BAO-39	0,663	0,61580
2560x1920	7	Sólido	25	45	22,1	TRIP-53	1,204	0,32790
2560x1920	15	Sólido	25	30	22,1	TRIP-54	1,668	0,63600
2560x1920	3	Ninguno	150	30	22,1	BAO-43	0,864	0,05350
720x480	7	Ninguno	150	45	22,1	BAO-33	0,535	0,58320
2560x1920	15	Ninguno	150	30	22,1	TRIP-15	3,000	0,20780
720x480	3	Líquido	150	60	22,1	BAO-24	0,422	0,13950
2560x1920	7	Líquido	150	30	22,1	TRIP-47	1,141	0,58440
720x480	15	Líquido	150	30	22,1	BAO-07	0,108	0,84730

Tabla 5.2. Resultados de error típico y coeficiente de determinación para diámetro de 22,1 mm.

Diámetro de pin igual a 26,1 mm								
Calidad	**Velocidad**	**Lubricante**	**Temperatura**	**Ángulo**	**Diametro pin**	**Ensayo**	**error típico**	**R^2**
	3	Ninguno	25					
2560x1920	7	Ninguno	25	45	**26,1**	**BAO-27**	1,977	0,15910
	15	Ninguno	25					
2560x1920	3	Líquido	25	**60**	**26,1**	**BAO-01**	0,728	0,46300
2560x1920	7	Líquido	25	45	**26,1**	**BAO-28**	1,384	0,21130
2560x1920	15	Líquido	25	**30**	**26,1**	**BAO-23**	3,431	**0,01280**
2560x1920	3	Sólido	25	45	**26,1**	**BAO-45**	0,403	**0,63770**
	7	Sólido	25					
2560x1920	15	Sólido	25	60	**26,1**	**BAO-55**	2,515	0,35210
	3	Ninguno	150					
720x480	7	Ninguno	150	60	**26,1**	**BAO-17**	0,886	**0,01540**
2560x1920	15	Ninguno	150	**30**	**26,1**	**BAO-22**	2,746	**0,09500**
2560x1920	3	Líquido	150	45	**26,1**	**TRIP-38**	0,990	**0,52350**
720x480	7	Líquido	150	60	**26,1**	**TRIP-41**	1,242	0,45720
	15	Líquido	150					
720x480	3	Sólido	150	60	**26,1**	**BAO-51**	0,345	**0,00270**
2560x1920	7	Sólido	150	**30**	**26,1**	**TRIP-18**	1,908	0,4425
720x480	15	Sólido	150	**30**	**26,1**	**TRIP-56**	2,667	**0,02240**

Tabla 5.3. Resultados de error típico y coeficiente de determinación para diámetro de 26,1 mm.

En **pines de 22,1 mm** destacan los ensayos con lubricante líquido, ángulo de 30º, resolución 720x480 y temperatura elevada (150 ºC), como el ensayo BAO-07, que muestra el mayor coeficiente de determinación $R^2 = 0{,}8473$ y un error típico muy bajo. Esta combinación parece facilitar una mayor linealidad en la evolución de la velocidad, probablemente debido a una menor fricción irregular y una tracción más uniforme.

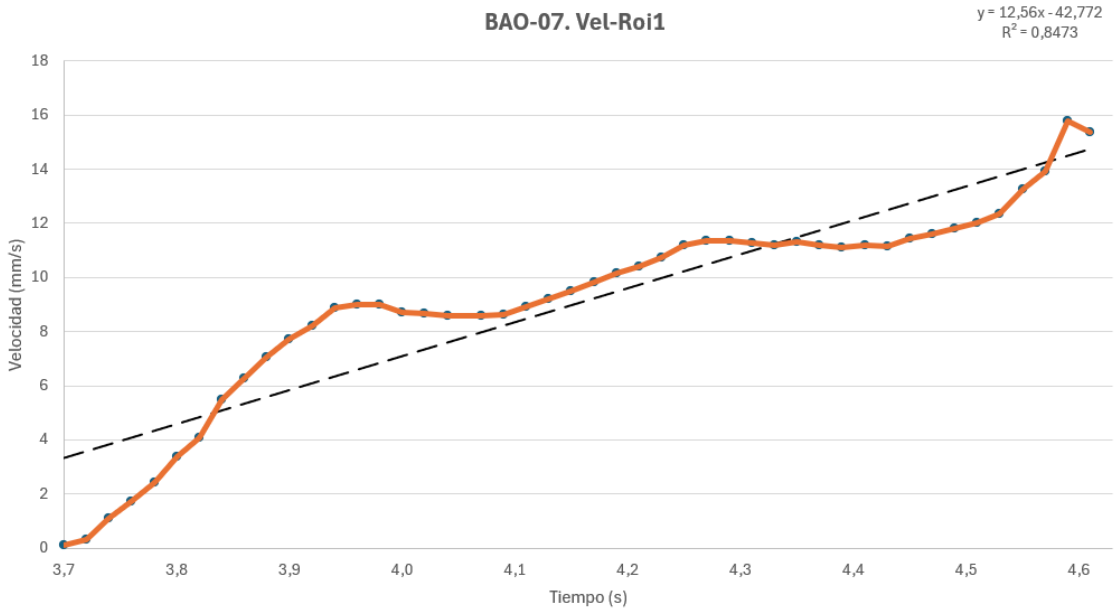
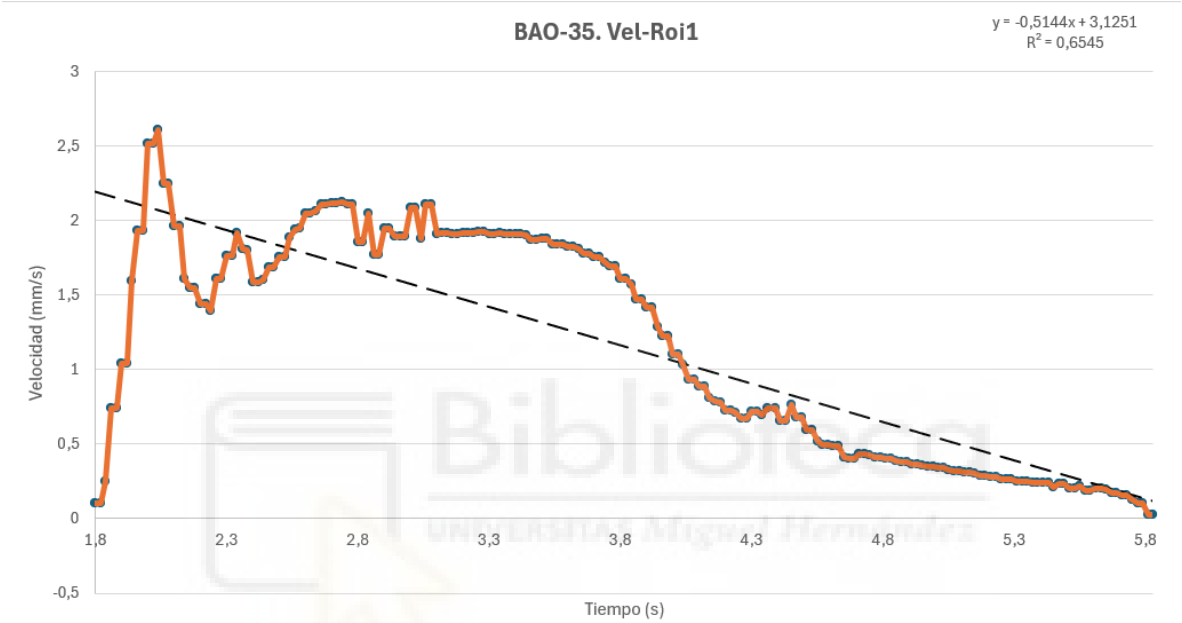


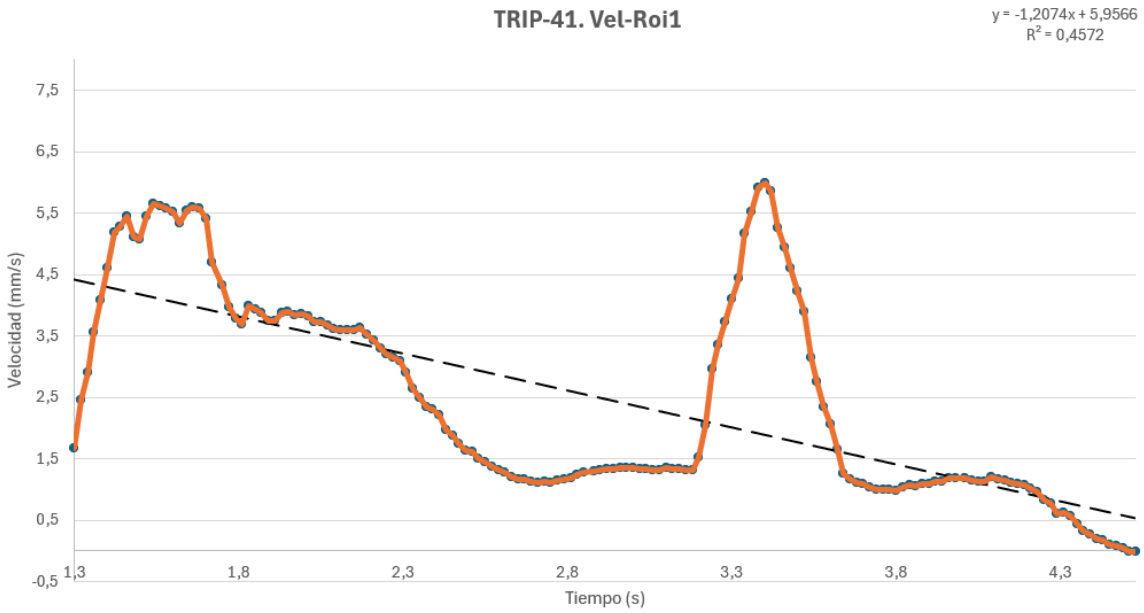
Figura 5.1. Gráfica de velocidad media ROI1 suavizada en BAO-07

En contraste, con resoluciones más altas (2560x1920) y condiciones similares, los valores de R^2 tienden a ser más bajos, lo que podría indicar que el mayor nivel de detalle no necesariamente contribuye a una mejor representación de linealidad.

En **pines de 19,1 mm** los mejores resultados se observan en configuraciones a temperatura ambiente, lubricante sólido y ángulo de 30º o 45º, como el ensayo BAO-35 con $R^2 = 0{,}6545$. Sin embargo, en general, este diámetro presenta más dispersión en los resultados.



Los ensayos con **pines de 26,1 mm** no parecen ofrecer un comportamiento más lineal, y los valores de R^2 son generalmente bajos, salvo algunos casos como el ensayo TRIP-41. Sin embargo, aún en este caso, se puede apreciar en la Figura 5.3 como, debido a fricciones grandes entre el pin y la probeta, se observan tirones alrededor de los 3,3s perdiendo de esta forma linealidad y ajuste con el coeficiente de determinación.



En general, los ensayos realizados con ángulos de 30º, lubricante líquido y pines de 22,1 mm han mostrado mayor linealidad en los resultados, con coeficientes R^2 superiores al 0,8 y errores típicos inferiores al 0,2. Por tanto, estas condiciones podrían considerarse las más favorables dentro del rango estudiado.

5.2. Evaluación del Seguimiento de Puntos

Con el objetivo de evaluar la fiabilidad dl sistema de seguimiento implementado, se han analizado dos métricas fundamentales: la **dispersión de los puntos** y el **error de seguimiento**. Estos valores permiten estudiar la estabilidad del algoritmo y su capacidad para mantener un seguimiento preciso durante el desarrollo del ensayo, incluso ante posibles perturbaciones como vibraciones o desenfoques.

La Tabla 5.4 recoge los valores medios absolutos de dispersión y error para ambas regiones de interés (ROI1 y ROI2) en una selección representativa de ensayos que incluyen diferentes configuraciones de calidad, velocidad de ensayo y condiciones de prueba.

Calidad	Velocidad	Ensayo	Disp. ROI1	Disp. ROI2	Err ROI1	Err ROI2
2560x1920	3	TRIP-46	0,058	0,061	2,583	2,532
2560x1921	7	BAO-26	0,082	0,088	2,015	2,188
2560x1922	15	BAO-40	0,071	0,089	2,060	2,207
720x480	3	BAO-02	0,039	0,064	7,683	9,981
720x481	7	TRIP-33	0,080	0,076	2,343	3,289
720x482	15	TRIP-14	0,162	0,079	3,209	4,856

Tabla 5.4. Resultados de dispersión y error de seguimiento promedio absoluto.

De igual manera, la Figura 5.4 representan visualmente estos valores medios para facilitar la comparación entre ensayos.

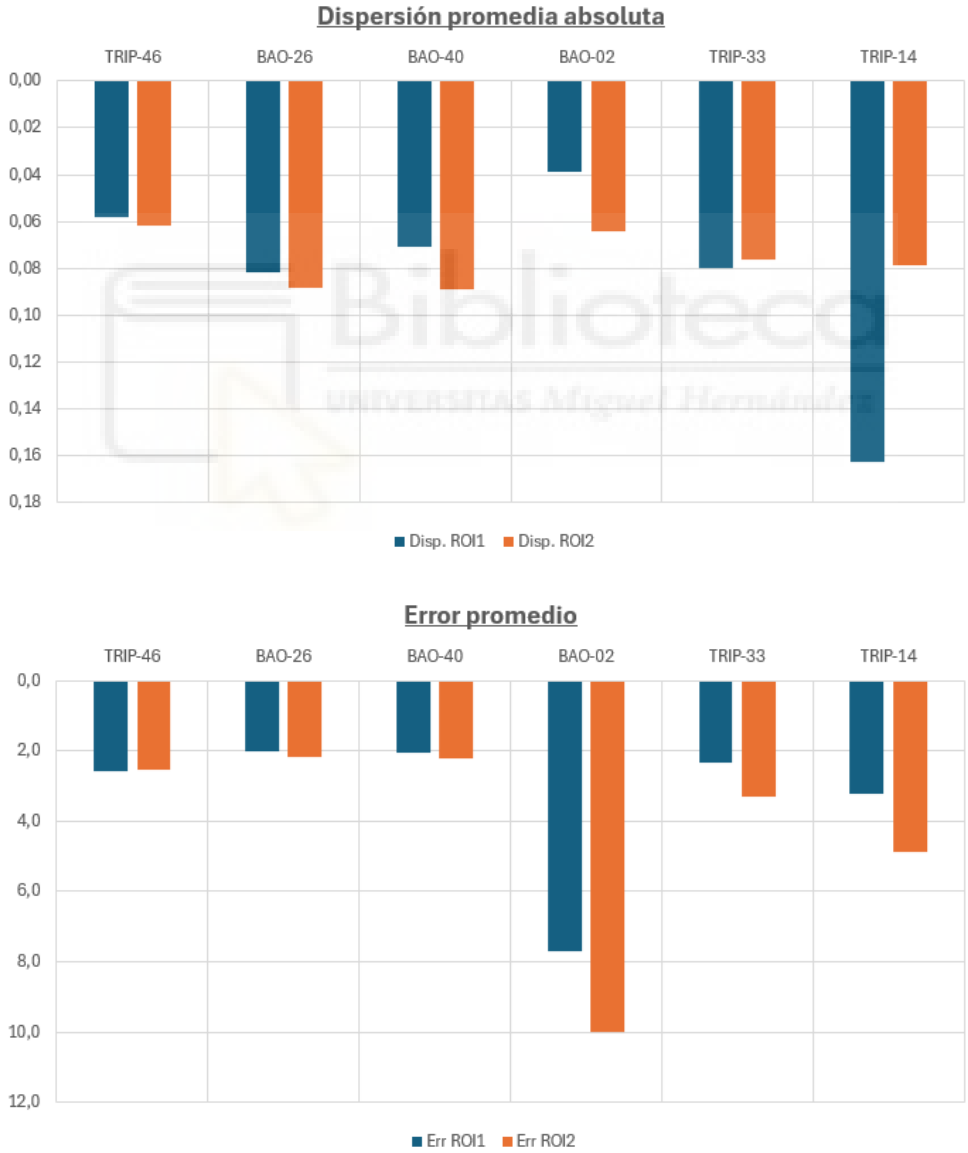


Figura 5.4. a) Comparativa de dispersión promedia absoluta, b) error promedio. Ambos en ensayos con diferente calidad y velocidad de ensayo.

La evaluación de la calidad del seguimiento mediante análisis de la dispersión y el error proporciona una visión clara del rendimiento del sistema de detección y seguimiento de puntos en diferentes condiciones de ensayo. Ambas gráficas de barras muestran como existe una variabilidad significativa en los valores medios de dispersión y error entre los distintos ensayos.

Una dispersión baja en las coordenadas de los puntos indica una buena estabilidad en la localización de los puntos de interés a lo largo del tiempo. En otras palabras, si los puntos no varían significativamente su posición relativa de un frame a otro, se puede concluir que el seguimiento es fiable y robusto.

En cuanto al error, se ha utilizado el valor proporcionado por la función de Lucas-Kanade Piramidal. Este error representa una medida del residuo de la estimación óptica entre dos imágenes consecutivas y se expresa en unidades de píxeles. Es importante destacar que aunque este valor no es directamente el desplazamiento entre la posición estimada y real, sí refleja la calidad del ajuste realizado en la pirámide de imágenes. Un valor pequeño indica un seguimiento más seguro, mientras que valores elevados podrían deberse a desenfoques, pérdida de textura o deformaciones abruptas (como puede ocurrir a una velocidad de ensayo de 15 mm/s).

En general, los ensayos con mejor resolución y condiciones de iluminación estables tienden a presentar menor dispersión y error. Además, se observa que las configuraciones que combinan alta calidad de vídeo con velocidad moderada ofrecen un seguimiento más limpio. Por el contrario, ensayos con mala calidad de vídeo o configuraciones experimentales desfavorables presentan oscilaciones y mayor ruido.

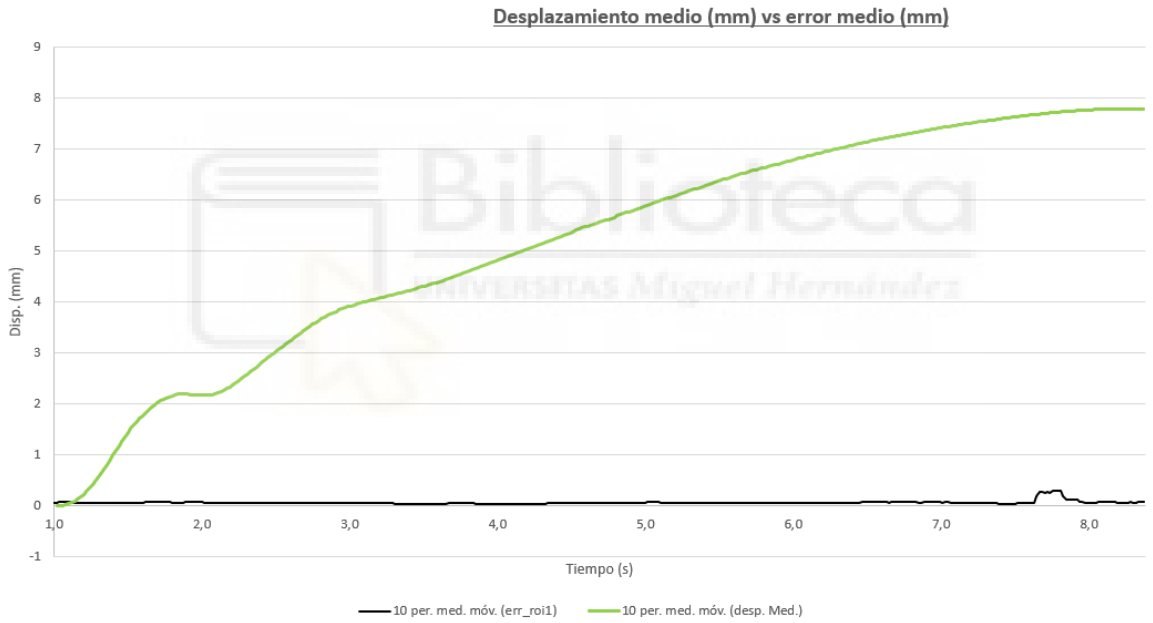
A modo de ejemplo ilustrativo se presentan las gráficas de dispersión y error correspondientes al ensayo TRIP-46.

Figura 5.6. Error de seguimiento de los puntos de interés para el ensayo TRIP-46.

En esta prueba se puede comprobar como, durante la mayor parte del ensayo, las curvas de dispersión permanecen estables y cercanas a cero, lo que indica una correcta alineación temporal entre los puntos detectados. Únicamente hacia el final del ensayo se aprecia un incremento puntual de la dispersión, coincidiendo probablemente con la fase de rotura del material.

En cuanto al error de seguimiento, las gráficas reflejan una evolución controlada y coherencia entre ambas regiones de interés, con un pico puntual al final del ensayo que también podría relacionarse con un movimiento abrupto en la muestra.

En la Figura 5.7 se muestra una gráfica comparativa entre el desplazamiento medio del ensayo TRIP-46 y el error medio a lo largo del tiempo, ambos en mm.



Por otro lado, el ensayo BAO-02 es un claro ejemplo de un vídeo de mala calidad con más ruido. En este tipo de ensayos puede apreciarse cómo el error medio es significativamente mayor que en el resto de los ensayos, alcanzando valores por encima de 10 píxeles durante una gran parte del seguimiento, especialmente en ROI2. Esta situación se traduce en una mayor incertidumbre sobre la posición real de los puntos, afectando de forma directa a la calidad de las velocidades obtenidas.

Figura 5.9. Error de seguimiento de los puntos de interés para el ensayo BAO-02.

Del mismo modo, la dispersión en este ensayo también muestra un patrón claramente desfavorable: los valores de dispersión se alejan del cero de forma brusca en torno al instante de rotura, lo cual indica una pérdida de la estabilidad en el seguimiento. Este fenómeno puede estar relacionado con factores como la calidad de imagen, el desenfoque o el contraste deficiente.

En conjunto, el análisis del error y la dispersión no solo permite evaluar la robustez del algoritmo de seguimiento, sino que también ayuda a identificar posibles limitaciones en los datos de entrada y en la calidad del experimento. Ensayos como BAO-02 dejan patente que una imagen de baja resolución puede comprometer significativamente la fiabilidad del sistema de medición.

5.3. Análisis de la Detección de Rotura y t0

Con el objetivo de validar la fiabilidad del sistema desarrollado para la detección de los instantes t_0 y t_{rotura}, se ha llevado a cabo una comparación directa con los valores obtenidos a partir de los sensores externos presentes en los ensayos.

Para ello, se han estudiado todas las muestras existentes tanto del material BAO como del material TRIP. En cada caso, se ha registrado el valor del delta temporal ($\Delta t = t_{rotura} - t_0$) calculado por el sistema, así como el valor correspondiente extraído de los sensores. Los resultados se agrupan en dos tablas diferencias, una por cada material (BAO y TRIP) e incluyen la calidad del vídeo original, la velocidad de ensayo, y los valores de t_0, t_{rotura} y delta para los experimentos procesados por el algoritmo, mientras que para los datos de los sensores se tiene únicamente el delta de tiempo correspondiente entre la detección de inicio de movimiento y el tiempo de rotura.

Además, se calcula la diferencia absoluta en segundos entre el delta del algoritmo y el delta de los sensores y se obtiene su error porcentual. Al final, se realiza una media del error en vídeos de 30 FPS y 60 FPS, obteniendo una diferencia notable.

probeta	velocidad	t_rotura (s)	t0	dt Sistema (s)	dt Sensor (s)	Dif. Abs. (s)	Resolución	FPS	Error Abs. (%)
1	3	6,92	2,11	4,81	7,44	2,63		30	35,35
2	3	11,74	7,15	4,59	4,58	**0,01**		60	**0,22**
3	15	4,32	3,21	1,11	1,76	0,65		30	36,93
4	3	6,68	2,55	4,13	7,2	3,07		30	42,64
5	7	5,92	3,55	2,37	2,4	**0,03**		60	**1,25**
6*	7					0			
7	15	4,6	3,67	0,93	0,9	**0,03**		60	**3,33**
8	7	4,7	3,21	1,49	1,4	**0,09**		60	**6,43**
9	7	6,84	4,19	2,65	2,68	**0,03**		60	**1,12**
10*	15								
11	15	3,26	2,07	1,19	1,76	0,57		30	32,39
12	15	4,96	3,88	1,08	1,1	**0,02**		60	**1,82**
13	15			0				60	
14	15	3,48	2,45	1,03	1,1	**0,07**		60	**6,36**
15	15	1,26	0,5	0,76	1,22	0,46		30	37,70
16	3	4,08	1,77	2,31	3,42	1,11		30	32,46
17	7	3,26	1,55	1,71	1,72	**0,01**		30	**0,58**
18	7	2,58	1,45	1,13	1,98	0,85		30	42,93
19	3	7,8	2,57	5,23	5,5	**0,27**		60	**4,91**
20	3	6,68	1,58	5,1	5,1	**0**		60	**0,00**
21	7	4,32	2,01	2,31	3,6	1,29		30	35,83
22	15	2,28	1,73	0,55	0,88	**0,33**		30	37,50
23	15	2,48	1,37	1,11	1,72	0,61		30	35,47
24	3	5,86	1,93	3,93	4,08	**0,15**		60	**3,68**
25	7	4,06	1,94	2,12	2,14	**0,02**		60	**0,93**
26	7	4,62	2,45	2,17	3,18	1,01		30	31,76
27	7	4,02	1,76	2,26	3,54	1,28		30	36,16
28	7	3,88	1,91	1,97	2,98	1,01		30	33,89
29	7	4,26	2,42	1,84	3,18	1,34		30	42,14
30 (42)	3								
31	3	6,96	2,11	4,85	7,5	2,65		30	35,33
32	7	4,44	2,13	2,31	2,28	**0,03**		60	**1,32**
33	7	4,04	2,03	2,01	2,34	0,33		60	14,10
34	3	5,86	1,51	4,35	4,38	**0,03**		60	**0,68**
35	3							30	
36	7	4,12	2,35	1,77	2,96	1,19		30	40,20
37	3	7,02	2,01	5,01	7,48	2,47		30	33,02
38	3	4,62	2,17	2,45	4,08	1,63		30	39,95
39	3	6,82	2,25	4,57	7,66	3,09		30	40,34
40	15	5,66	4,38	1,28	2	0,72		30	36,00
41	7	5,68	2,83	2,85	2,84	**0,01**		60	**0,35**
42	7	4,66	2,67	1,99	3,02	1,03		30	34,11
43	3	4,28	1,31	2,97	4,34	1,37		30	31,57
44	15	3,14	1,97	1,17	1,16	**0,01**		60	**0,86**
45	3	5,22	1,35	3,87	5,98	2,11		30	35,28
46	3	8,12	2,77	5,35	7,98	2,63		30	32,96
47	7	3,18	1,77	1,41	2,18	0,77		30	35,32
48	7	3,06	1,25	1,81	2,7	0,89		30	32,96
49	15	2,46	1,55	0,91	1,38	0,47		30	34,06
50	15	2,66	1,91	0,75	1,24	0,49		30	39,52
51	3	7,28	0,86	6,42	6,46	**0,04**		60	**0,62**
52*	15								
53	7	4,66	2,65	2,01	3,4	1,39		30	40,88
54	15	2,72	1,85	0,87	1,5	0,63		30	42,00
55	15	3,02	1,99	1,03	1,56	0,53		30	33,97
56	15	3,1	2,11	0,99	1	**0,01**		60	**1,00**
57	15	2,36	1,53	0,83	1,56	0,73		30	46,79

Tabla 5.5. Resultados de diferencia de deltas entre algoritmo y sensores en material BAO.

probeta	velocidad	t_rotura (s)	t0	dt Sistema (s)	dt Sensor (s)	Dif. Abs. (s)	Resolución	FPS	Error Abs. (%)
1	3	6,9	1,1	5,8	8,7	2,9		30	33,33
2	3	7,86	0,96	**6,9**	**6,8**	**0,1**		60	**1,47**
3	15	2,62	1,17	1,45	2,22	0,77		30	34,68
4	3	7,48	1,4	6,08	9,36	3,28		30	35,04
5	7	4,86	0,98	**3,88**	**3,78**	**0,1**		60	**2,65**
6	7	4,82	0,94	**3,88**	**3,86**	**0,02**		60	**0,52**
7	15	2,92	1,41	**1,51**	**1,54**	**0,03**		60	**1,95**
8	7	1,44	0,43	1,01	3,26	2,25		51	69,02
9	7	5,2	1,63	**3,57**	**3,64**	**0,07**		60	**1,92**
10	15	3,88	1,69	**2,19**	**2,2**	**0,01**		60	**0,45**
11	15	2,32	0,88	1,44	2,18	0,74		30	33,94
12	15	2,72	1,29	**1,43**	**1,6**	**0,17**		60	**10,63**
13	15	2,86	1,13	**1,73**	**1,8**	**0,07**		60	**3,89**
14	15	3,3	1,69	**1,61**	**1,58**	**0,03**		60	**1,90**
15	15	2,38	1,36	1,02	1,54	0,52		30	33,77
16	3	9,2	1,17	**8,03**	**8,24**	**0,21**		60	**2,55**
17	7	3,26	1,45	1,81	3,06	1,25		30	40,85
17_rep	7	3,82	0,74	**3,08**	**3,08**	**0**		30	**0,00**
18	7	3,02	1,01	2,01	3,44	1,43		30	41,57
19	3	9,06	0,98	**8,08**	**8,14**	**0,06**		60	**0,74**
20	3	9,34	0,98	**8,36**	**8,38**	**0,02**		60	**0,24**
21	7	4,36	0,9	3,46	5,28	1,82		30	34,47
22	15	1,72	0,91	0,81	1,32	0,51		30	38,64
23	15	2,92	1,11	1,81	2,72	0,91		30	33,46
24	3							60	
25	7	4,7	1,23	**3,47**	**3,44**	**0,03**		60	**0,87**
26	7	3,92	0,8	3,12	4,76	1,64		30	34,45
27	7	3,92	0,88	3,04	4,52	1,48		30	32,74
28	7	3,82	0,9	2,92	4,24	1,32		30	31,13
29	7	4,66	1,39	3,27	5,64	2,37		30	42,02
30	3			0				30	
31	3	7,28	1,24	6,04	9,22	3,18		30	34,49
32	7	5,2	1,75	**3,45**	**3,46**	**0,01**		60	**0,29**
33	7	3,52	0,56	2,96	3,66	0,7		30	19,13
34	3	5,9	0,58	5,32	8,5	3,18		30	37,41
35	3	8,02	1,05	6,97	10,76	3,79		30	35,22
36	7	6,26	0,86	**5,4**	**5,42**	**0,02**		30	**0,37**
37	3	7,48	0,6	6,88	10,48	3,6		30	34,35
38	3	5,06	0,5	4,56	7,96	3,4		30	42,71
39	3	6,76	0,56	6,2	10,42	4,22		30	40,50
40	15	2,72	0,97	1,75	2,64	0,89		30	33,71
41	7	4,52	1,27	**3,25**	**3,34**	**0,09**		60	**2,69**
42	7	4,36	1,04	3,32	5,04	1,72		30	34,13
43	3	4,52	0,67	3,85	5,84	1,99		30	34,08
43_BIS	3	5,32	0,64	4,68	7,02	2,34		30	33,33
44	15	3,62	1,69	**1,93**	**1,96**	**0,03**		60	**1,53**
45	3	8,48	0,84	7,64	11,6	3,96		30	34,14
46	3	8,4	0,98	7,42	11,18	3,76		30	33,63
47	7	3,5	1,05	2,45	3,7	1,25		30	33,78
48	7	3,86	0,7	3,16	4,78	1,62		30	33,89
49	15	2,36	0,67	1,69	2,48	0,79		30	31,85
50	15	3,12	1,45	1,67	2,46	0,79		30	32,11
51	3	9,18	1,42	**7,76**	**7,76**	**0**		60	**0,00**
52	15	2,62	1,05	**1,57**	**3,4**	1,83		30	53,82
53	7	3,06	0,81	2,25	3,6	1,35		30	37,50
54	15	3,1	1,21	1,89	2,86	0,97		30	33,92
55	15	2,26	0,79	1,47	2,28	0,81		30	35,53
56	15	2,98	0,99	**1,99**	**1,94**	**0,05**		60	**2,58**
57	15	2,36	0,96	1,4	2,44	1,04		30	42,62

Tabla 5.6. Resultados de diferencia de deltas entre algoritmo y sensores en material TRIP.

Al analizar los resultados, se observa un comportamiento muy diferente entre los vídeos que fueron grabados originalmente a 60 FPS y aquellos grabados a 30 FPS. En los vídeos de 60 FPS, el ajuste entre los valores de delta extraídos por el sistema y los sensores es excelente, con errores absolutos medios muy bajos (aproximadamente 2 centésimas de segundo), lo cual valida la precisión del sistema propuesto.

Sin embargo, en los vídeos originalmente grabados a 30 FPS, los resultados muestran una discrepancia considerable. Inicialmente se pensó que esta desviación podía estar causada por el proceso de interpolación aplicado a estos vídeos, los cuales fueron preprocesados para aumentar su tasa de muestre hasta 50 FPS mediante técnicas de duplicación o interpolación de frames.

No obstante, al realizar un análisis más detallado de estos ensayos, se ha detectado una anomalía adicional: los valores de delta obtenidos a partir de los sensores parecen inconsistentes con la propia duración de los vídeos originales. Por ejemplo, en el ensayo BAO-01 (30 FPS), el vídeo original tiene una duración total de 10 segundos, y el sistema detecta correctamente un t_0 en torno a los 2,11s y una rotura a 6,92s, lo cual concuerda visualmente con la observación directa del vídeo. Sin embargo, el delta proporcionado por los sensores es de 7,44s, valor que no puede sostenerse ni cronológica ni visualmente dentro de la duración total del vídeo. Este fenómeno se ha reproducido de forma sistemática en todos los ensayos de 30 FPS, pero no en los de 60 FPS, lo que permite descartar que el error provenga exclusivamente del sistema de análisis por visión artificial o de la interpolación aplicada. Todo parece indicar que la causa más probable es una desincronización entre la grabación de vídeo y el sistema de adquisición de datos de los sensores, específicamente en los ensayos realizados a 30 FPS.

Porcentaje de error entre deltas de tiempo

BAO	
Error Promedio 60 FPS	**2,72%**
Error Promedio 30 FPS	**35,70%**
Nº Ensayos a 60 FPS	**19**
Nº Ensayos a 30 FPS	**40**
BAO	

TRIP	
Error Promedio 60 FPS	**2,02%**
Error Promedio 30 FPS	**33,51%**
Nº Ensayos a 60 FPS	**19**
Nº Ensayos a 30 FPS	**40**
TRIP	

Número de ensayos a 60 FPS o a 30 FPS

Tabla 5.7. Recopilación de errores medios a diferentes FPS para cada material.

5.4. Corrección de la curvatura

A lo largo del análisis de desplazamientos en los ensayos, se planteó la posibilidad de aplicar una corrección geométrica para tener en cuenta la curvatura del pin de contacto. Esta curvatura provoca que los puntos a estudiar no se muevan por una superficie plana, sino sobre una curva, lo que introduce una distorsión entre en el valor real de desplazamiento observado desde el plano de la cámara.

Para calcular esta variación, se desarrolló un modelo basado en la proyección de los puntos sobre la semicircunferencia que forma el contorno del pin, según su diámetro (D_p) y el ángulo de contacto ($\boldsymbol{\alpha}$), tal y como se esquematiza en la Figura 4.3. En este modelo existen dos zonas diferenciadas:

- **Zona Z1**: región en contacto con el pin donde la corrección depende del ángulo de apoyo α. En este caso, la corrección geométrica se describe mediante:

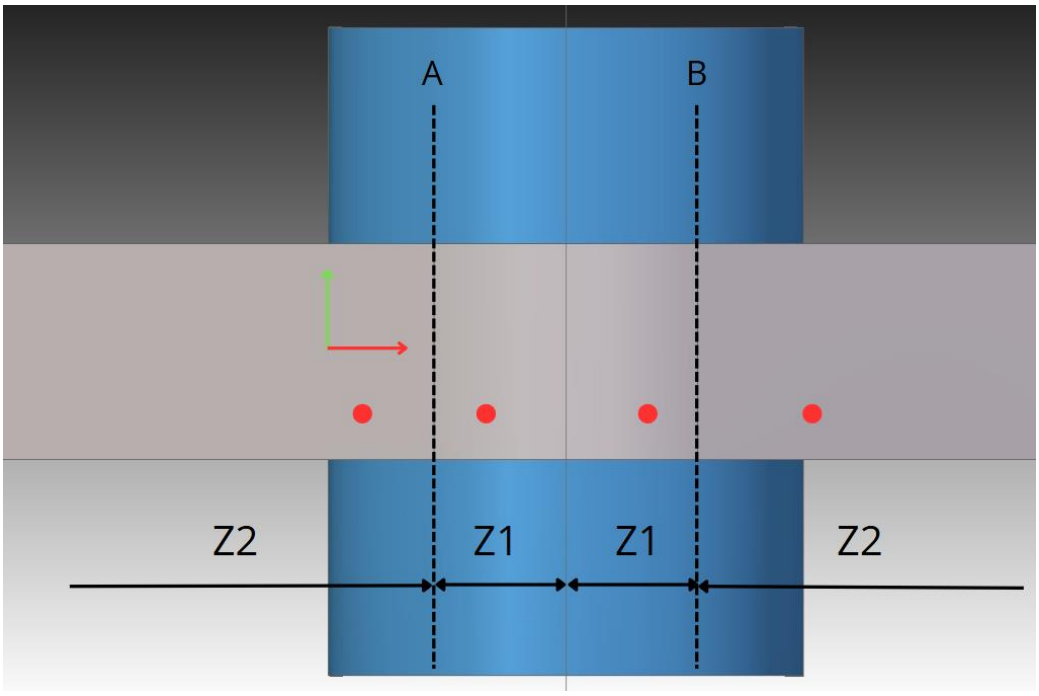
$$x' = (R_p + t_p) \cdot \arcsin\left(\frac{x}{R_p + t_p}\right) \tag{33}$$

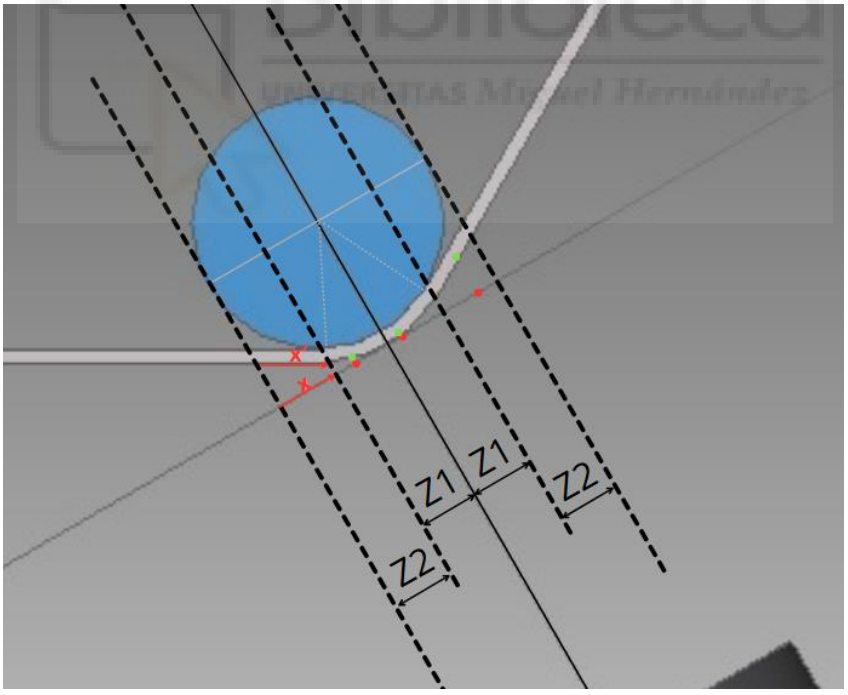
- **Zona Z2**: desplazamientos fuera del contacto directo con el pin, donde la trayectoria es recta y se puede aplicar una corrección usando la fórmula:

$$x' = \frac{x}{\sin\left(\frac{\alpha}{2}\right)} + (R_p + t_p) \cdot \left(\frac{\alpha}{2} - 1\right) \tag{34}$$

donde:

- $\boldsymbol{R_p}$ es el radio del pin.
- $\boldsymbol{t_p}$ es el espesor de la probeta, aproximado a 1 mm.
- $\boldsymbol{x}$ es el desplazamiento observado en la imagen.
- $\boldsymbol{x'}$ es el desplazamiento real corregido


A
B
Z2
Z1
Z1
Z2


Z1
Z1
Z2
Z2

Para analizar la magnitud de esta corrección, se han calculado los valores límite de ambas zonas y se han calculado las dos fórmulas a una serie de valores para analizar cómo sería la variación. La Tabla 5.8 recoge los valores de ayuda para el cálculo de la corrección de la curvatura.

tp [mm]	1								
Dp [mm]	19,1			22,1			26,1		
Ángulo α [°]	30	45	60	30	45	60	30	45	60
Ángulo α [rad]	0,5236	0,7854	1,0472	0,5236	0,7854	1,0472	0,5236	0,7854	1,0472
Valor límite de x l1=(Rp+tp)* sen(α/2)	2,731	4,037	5,275	3,119	4,611	6,025	3,636	5,377	7,025
(Rp+tp)	10,55			12,05			14,05		
α/2	15	22,5	30	15	23	30	15	22,5	30
sen(α/2)	0,259	0,383	0,5	0,259	0,383	0,5	0,259	0,383	0,5
α/2-1	-0,476	-0,215	0,047	-0,476	-0,215	0,047	-0,476	-0,215	0,047

Por otro lado, la Figura 5.11 muestra una comparación en los desplazamientos en milímetros para los puntos dentro de la zona 1 (Z1), la zona de contacto. Como se puede observar, la diferencia es mínima e insignificante. Además, se ha de tener en cuenta que la distancia máxima de Z1 será de 7,025 mm para un ensayo con diámetro de pin de 26,1 mm y ángulo de tracción de 60º.

x	x'
0	0
0,1	0,1000015
0,2	0,20001198
0,3	0,30004045
0,4	0,4000959
0,5	0,50018737
0,6	0,60032391
0,7	0,70051464
0,8	0,80076867
0,9	0,90109521
1	1,00150351
1,1	1,10200288
1,2	1,20260272
1,3	1,30331252
1,4	1,40414183
1,5	1,50510033
1,6	1,6061978
1,7	1,70744414
1,8	1,80884938
1,9	1,91042367
2	2,01217735
2,1	2,11412086
2,2	2,21626487
2,3	2,31862018
2,4	2,4211978
2,5	2,52400897

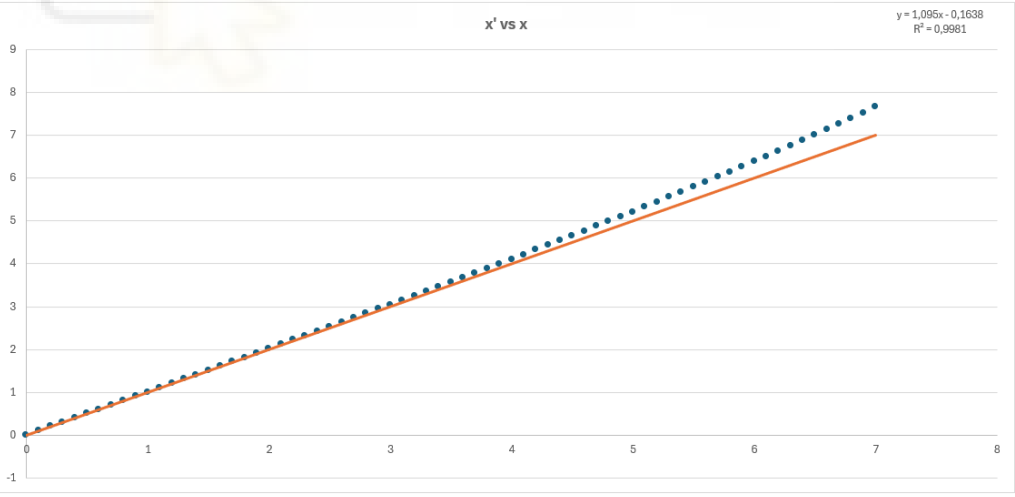


Figura 5.12. Comparación en los desplazamientos para los puntos dentro de la zona 1 (Z1) en la corrección de la curvatura.

Aunque desde un punto de vista teórico la corrección de la curvatura aporta un punto extra en cuanto a precisión del comportamiento del material, los resultados obtenidos indican que la diferencia introducida es insignificante respecto del desplazamiento total y a la resolución del sistema. Por tanto, se concluye que el impacto en los resultados es despreciable y no justifica la complejidad añadida al proceso de postprocesado.

CAPÍTULO 6

CONCLUSIONES Y LÍNEAS FUTURAS

6.1. Análisis Crítico del Sistema

Todos los objetivos propuestos para la realización de este trabajo se han llevado a cabo de forma satisfactoria y es un orgullo haber podido realizarlo. Gracias a la realización de este proyecto se ha podido aprender el funcionamiento de un sistema completo de análisis basado en visión por computador, desde el tratamiento inicial de imágenes hasta la extracción y exportación de resultados. El desarrollo ha permitido integrar conocimientos de programación en Python, uso de librerías como OpenCV, así como de metodologías de análisis experimental en ensayos de tracción.

El sistema implementado permite automatizar, en gran medida, el seguimiento de puntos sobre probetas sometidas a carga, calculando velocidades locales en regiones de interés previamente definidas y destacando con fiabilidad eventos clave como el inicio del movimiento o la rotura del material. Además, se ha diseñado un flujo de trabajo flexible que admite la interacción del usuario cuando es necesario, sin comprometer la coherencia del procesamiento automático.

A lo largo del desarrollo, se han superado numerosos retos técnicos relacionados con la calidad variable de los vídeos, la sensibilidad del algoritmo del flujo óptico y la necesidad de asegurar un tratamiento homogéneo en todos los ensayos. Todo ello ha contribuido a consolidar una herramienta sólida, abierta a mejoras futuras, y con gran potencial para ser aplicada en otros contextos de análisis experimental de materiales.

En definitiva, este trabajo no solo ha permitido alcanzar los objetivos técnicos planteados, sino que también ha servido como una valiosa experiencia de aprendizaje en el ámbito de la visión por computador y en el desarrollo de programas mediante programación con lenguaje Python. La combinación de programación, tratamiento de vídeo y análisis físico ha demostrado ser una vía eficaz para abordar problemas complejos de forma estructurada. Aunque el sistema aún presenta áreas susceptibles de mejora, su implementación constituye una base sólida y versátil sobre la que seguir construyendo. Esta experiencia deja abiertas

múltiples puertas para el futuro, tanto en la mejora de la herramienta como en su aplicación a nuevos entornos experimentales y profesionales, como por ejemplo con el uso de varias cámaras para realizar un análisis 3D.

6.2. Posibles Mejoras

Una de las posibles líneas de mejora más interesantes podría ser la implementación de un **sistema multicámara** que permita realizar un análisis tridimensional (3D) del comportamiento de la probeta a lo largo del ensayo. Actualmente, el análisis se limita a dos dimensiones (2D), al incluir cámaras en diferentes ángulos sería posible reconstruir las trayectorias espaciales de los puntos mediante técnicas de fotogrametría, lo que permitiría capturar componentes de desplazamientos fuera del plano de la imagen y caracterizar deformaciones complejas. Esta mejora sería especialmente útil en este tipo de ensayos debido a que todos ellos tienen como variable el ángulo de tracción, pudiendo estudiar así la flexión o torsión del material, dándole un valor añadido al estudio realizado.

Otra línea futura relevante es la mejora de la robustez del seguimiento de puntos, especialmente en vídeos de baja calidad. Este punto podría ser abordado mediante la implementación de una nueva técnica de flujo óptico, como **DeepFlow** o modelos de seguimiento basados en **redes neuronales convolucionales** (CNN), que han demostrado una mayor resistencia frente al ruido y a variaciones bruscas en la textura o iluminación. Del mismo modo, una integración más profunda de técnicas de inteligencia artificial permitiría mejorar el rastreo o clasificar los ensayos según los resultados.

En términos visuales y de uso, una mejora importante sería el desarrollo de una **interfaz gráfica de usuario** (GUI), que permita al usuario cargar vídeos, ajustar parámetros, visualizar los resultados e incluso exportar los datos desde un entorno más cercano. Esta funcionalidad convertiría el sistema en una herramienta más accesible, tanto para usuarios con perfil técnico como para personal investigador sin experiencia en la programación.

En conjunto, todas estas propuestas permitirían transformar desde una solución funcional hacia una herramienta completa, versátil y adaptable a una amplia variedad de ensayos mecánicos experimentales.

CAPÍTULO 7

BIBLIOGRAFÍA

[1]: Peter WH, Ranson WF. Digital imaging techniques in experimental stress analysis. Optical Engineering;Vol 21 (3): 427-31 (1982).

[2]: I. M. Castillo-González, A. E. Peñaranda-Domínguez, and J. G. Díaz-Rodríguez, Implementation of the Digital Image Correlation Technique with Open Software, AbI Research, Administration and Engineering Journal, vol. 8, no. 3, pp. 25–32, Sep.–Dec. 2020. DOI: 10.15649/2346030X.875.

[3]: E. López-Alba, F. A. Díaz, R. Dorado, R. López-García, (2010), Análisis de deformaciones en probetas planas mediante correlación digital de imágenes. Disponible en: http://www.uclm.es/actividades/2010/CongresoIM/pdf/cdarticulos/109.pdf

[4]: Michael A.Sutton, Jean-José Orteu, Hubert W. Schreier, (Jun 01, 1982), Image Correlation for Shape, Motion and Deformation Measurements.

[5]: LuoPF, Chao YJ, Sutton MA, Peters WH (1993) Accurate measurement of three dimensional deformations in deformable and rigid bodies using computer vision. Exp Mech 32(2):123–132

[6]: Schreier HW, Braasch JR, Sutton MA (2000) Systematic errors in digital image correlation caused by intensity interpolation. Opt Eng 39:2915–2921

[7]: Dally JW, Riley WF (2005) Experimental stress analysis, 4th edn. College House Enterprises Llc, Knoxville

[8]: Sutton MA, Orteu JJ, Schreier H (2009) Image correlation for shape motion and deformation measurements: basic concepts theory and applications. Springer Science and Business Media, New York: Springer, 88.

[9]: Agha, A., & Abu-Farha, F. (2023). A Method for Measuring In-Plane Forming Limit Curves Using 2D Digital Image Correlation. SAE International Journal of Materials and Manufacturing, 16(3), 2023.

[10]: GOM Correlate 2022 y INSPECT Correlate 2023: Manual de usuario y guía técnica. ZEISS, 2023. Disponible: https://www.zeiss.com.

[11]: Barranger Y, Doumalin P, Dupre J C, GermaneauA (2010) Digital image correlation accuracy: influence of kind of speckle and recording setup. Eur Phys J Web Conf 6:31002

[12]: Lionello G,Cristofolini L (2014) A practical approach to optimizing the preparation of speckle patterns for digital-image correlation. Meas Sci Technol 25(10):107001

[13]: Reu P (2012) Hidden components of DIC: calibration and shape function—part 1. Exp Tech 36(2):3–5

[14]: Zhou P, Goodson KE (2001) Subpixel displacement and deformation gradient measurement using digital image/speckle correlation. Opt Eng 40(8):1613–1620

[15]: A. Freddi, G. Olmi, and L. Cristofolini, Experimental Stress Analysis for Materials and Structures: Stress Analysis Models for Developing Design Methodologies. Springer Series in Solid and Structural Mechanics, vol. 4. Cham, Switzerland: Springer, 2015.

[16]: S.-W. Khoo, S. Karuppanan, and C.-S. Tan, A review of surface deformation and strain measurement using two-dimensional digital image correlation, Metrol. Meas. Syst., vol. 23, no. 3, pp. 461–480, 2016. doi: 10.1515/mms-2016-0030.

[17]: Sutton, M.A., Wolters, W.J., Peters, W.H., Ranson, W.F., McNeill, S.R. (1983). Determination of displacements using an improved digital image correlation method. Image Vis. Comput., 1(3), 133−139.

[18]: B. K. P. Horn and B. G. Schunck, Determining optical flow, Artificial Intelligence, vol. 17, no. 1-3, pp. 185-203, 1980.

[19]: M. Shah, Optical flow estimation, in Computer Vision: Algorithms and Applications, 2011, pp. 201-220.

[20]: R. Szeliski, Computer Vision: Algorithms and Applications. Springer, 2010.

[21]: D. Forsyth and J. Ponce, Computer Vision, A Modern Approach. Pearson Education, 2012.

[22]: B. Lucas y K. Take, An Iterative Image Registration Technique with an Application to Stereo Vision, en Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI), Vancouver, 1981.

[23]: Horn and S. B, Optical flow artificial intelligence, Artificial Intelligence Laboratory, pp. 185–203, 1981.

[24]: Python. (s.f.). General Python FAQ. Disponible: https://docs.python.org/3/faq/general.html#why-is-it-called-python

[25]: OpenCV. (s.f.). About. Disponible: https://opencv.org/about/.

[26]: NumPy. (s.f.). About us. Disponible: https://numpy.org/about/.

[27]: W. McKinney, pandas: a python data analysis library. [Online]. Disponible: http://pandas.sourceforge.net

[28]: J. Hunter, et al., matplotlib: Python plotting, http://matplotlibsourceforge.net/.

CAPÍTULO 8

APÉNDICES

Todo el código desarrollado para el tratamiento de vídeos, el procesamiento de datos y la generación de gráficas se encuentra disponible públicamente en el siguiente repositorio de GitHub:

Repositorio del proyecto:

https://github.com/jguilabert/DIC-tracking-system

En él se incluyen:

- Scripts de preprocesado de vídeo y detección de puntos.
- Procesamiento de desplazamientos y cálculo de velocidades.
- Organización del proyecto con carpetas para datos, resultados y grabaciones.

Este repositorio permite garantizar la transparencia, reproducibilidad y escalabilidad del sistema desarrollado, además se facilita su posible reutilización en estudios futuros.

Recursos adicionales: (Ejemplo completo para ensayo TRIP-48)

- **Ejemplo completo para ensayo TRIP-48:** **https://drive.google.com/drive/folders/1iXb_e9Va_aZQUDeKHzbHhcnSHX9x87Dv?usp=sharing**
- **Hojas de cálculo dedicadas a la lectura y al análisis de resultados:** **https://drive.google.com/drive/folders/147NLexpO0iBkQSTQjViJNtvI5gQTW2vA?usp=drive_link**
- **Imágenes relacionadas con el ensayo de tracción:** **https://drive.google.com/drive/folders/1RnzhI9uU7jgGAfULE0d13sEoJEDhJWAn?usp=sharing**