



UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

**DESARROLLO DE
HERRAMIENTA DE
PROCESADO DE
IMÁGENES PARA SU USO EN
LA MECÁNICA
EXPERIMENTAL.**

Alumno: Luis Granados Fernández

Tutor: Luis A. Felipe Sesé

Depto.: Ingeniería Mecánica y Minera

Febrero, 2024

INDICE

1. INTRODUCCIÓN.....	6
2. OBJETIVO	11
3. FUNDAMENTOS TEORICOS	12
1.1 LA VISIÓN ARTIFICIAL	12
3.2 REPRESENTACIÓN ARTIFICIAL DE LA REALIDAD. LA VISIÓN ARTIFICIAL	14
3.3 CORRELACIÓN DIGITAL DE IMÁGENES.....	24
3.4 FUNDAMENTOS PARA LA PROGRAMACIÓN	29
3.5 OPEN CV	32
4. METODOLOGÍA	47
4.1 ALGORITMOS DE PROCESAMIENTO DE IMAGEN	47
4.2 METODOLOGÍA EXPERIMENTAL.....	55
5. RESULTADOS.....	78
5.1 ENSAYO 1. CÁLCULO DEL ÁNGULO DE ALINEACIÓN.....	78
5.2 ENSAYO 2. CÁLCULO DEL ÁNGULO GIRADO	80
5.3 ENSAYO 3. EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERPENDICULARIDAD CON DIC	81
5.4 ENSAYO 4. EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA ANTE DESPLAZAMIENTOS VERTICALES	86
6. CONCLUSIONES	95

7. BIBLIOGRAFÍA	96
Listado de Gráficos.....	101
Tabla De Ilustraciones	102
ANEXOS	108

RESUMEN

En el siguiente documento se aborda el desafío que supone, para la ingeniería mecánica experimental, la realización de ensayos sobre desplazamientos y deformaciones de materiales. Dada la importancia de la exactitud en los ensayos, y la dificultad que comporta lograr la perpendicularidad entre la cámara y el objeto, se hace evidente la necesidad de idear y desarrollar herramientas que permitan corregir los desvíos, que se pudieran producir, por captación errónea de la perspectiva. Así pues, a través de la bibliografía publicada, referente a dicho proceso, se analiza y evalúa el procesamiento de imágenes y la visión computarizada. También se explora en profundidad la correlación digital de imágenes en 2D para analizar desplazamientos y deformaciones en materiales, una técnica crucial en diversas aplicaciones de ingeniería. Este TFG se sumerge en los fundamentos teóricos de la visión artificial, incluyendo el modelo físico y fisiológico de la luz y las técnicas avanzadas de detección de bordes y cambio de perspectiva. Se propone una solución basada en el desarrollo de algoritmos sofisticados para medir ángulos de rotación y cambios de perspectiva, y el desarrollo de interfaces destinadas a facilitar su uso; proporcionando herramientas valiosas para la evaluación de estructuras y materiales. Además, se incluyen ensayos experimentales orientados a demostrar la aplicabilidad y eficacia de estos métodos en escenarios del mundo real. Por último, se analizan las conclusiones a las que dichos ensayos dan respuesta y su utilidad en el campo de experimentación elegido.

PALABRAS CLAVE: DIC, Visión artificial, Deformación de materiales, detección de bordes, ángulos de rotación, logaritmo, ensayo, perspectiva, procesamiento de imágenes, interfaz

ABSTRACT

The following document addresses the challenge that carrying out tests on displacements and deformations of materials poses for experimental mechanical engineering. Given the importance of accuracy in testing, and the difficulty involved in achieving perpendicularity between the camera and the object, the need to devise and develop tools that allow correcting deviations that could occur due to incorrect perspective capture becomes evident. Thus, through the published bibliography, referring to said process, image processing and computer vision are analyzed and evaluated. The digital correlation of 2D images is also explored in depth to analyze displacements and deformations in materials, a crucial technique in various engineering applications. This TFG delves into the theoretical foundations of computer vision, including the physical and physiological model of light and advanced edge detection and perspective change techniques. A solution is proposed based on the development of sophisticated algorithms to measure rotation angles and perspective changes, and the development of interfaces intended to facilitate their use, providing valuable tools for the evaluation of structures and materials. In addition, experimental tests are included aimed at demonstrating the applicability and effectiveness of these methods in real-world scenarios. Finally, the conclusions to which these tests respond and their usefulness in the chosen field of experimentation are analyzed.

KEY WORDS: DIC (Digital Image Correlation), Artificial Vision, Material Deformation, Edge Detection, Rotation Angles, Logarithm, Test, Perspective, Image Processing, interface.

1. INTRODUCCIÓN

La mecánica experimental es un campo de la ingeniería que se enfrenta a constantes cambios y desafíos. No debemos olvidar que, el objetivo de esta rama de la ciencia es investigar y desarrollar técnicas, y analizar datos, que ayuden a comprender mejor los materiales y sistemas mecánicos. Su propósito es lograr un difícil equilibrio entre innovación y precisión, impulsando la eficacia de los procesos y el progreso industriales y, por tanto, de toda la sociedad. Gran cantidad de ámbitos se benefician de las aportaciones que la ingeniería experimental proporciona, dando soluciones eficientes a problemas que se plantean en el día a día. Esto se logra mediante ensayos de componentes y medidas lo más fidedigna posibles, evitando simplificaciones de cálculos analíticos o simulaciones [1].

Una de las herramientas que mayor cantidad implementaciones está aportando al mundo de la ingeniería es la visión artificial. Se trata de una sub-rama de la inteligencia artificial en la que se utilizan equipos informáticos para el análisis, al más alto nivel, de imágenes digitales. Se caracteriza por su índole sumamente multidisciplinar, ya que utiliza técnicas y conocimientos procedentes de entornos tan dispares como el cálculo numérico, la estadística, el proceso de señales, la física óptica, la geometría o la infografía [2].

La Visión Artificial se ha utilizado en gran número de aplicaciones, abarcando problemas que van desde la detección de imperfecciones en la fabricación de circuitos electrónicos y la búsqueda de yacimientos minerales a partir de imágenes por satélite; pasando por el diagnóstico automático de tumores en muestras médicas o por el reconocimiento de caras a partir de las imágenes obtenidas por una cámara de seguridad. La enumeración de las posibles aplicaciones de la visión artificial sería interminable, ya que estamos hablando de intentar imitar, vía ordenador, las capacidades del sentido de la vista, el que más información aporta a los humanos [2].

Otro interesante campo en el que la ingeniería mecánica experimental se ha sumergido con gran aprovechamiento y logrando un gran auge, es el análisis de desplazamientos y deformaciones de materiales mediante la utilización de técnicas de procesamiento de imágenes. El uso de estas técnicas se aplica a sectores que suponen un desafío, como permitir el uso de DIC (correlación digital de imágenes) durante un tiempo prolongado y sin

necesidad de instalaciones permanentes en áreas de difícil acceso, o instalaciones que, por sus características o tamaño, no pueden estudiarse en un laboratorio [3].

Como ejemplo para ilustrar las múltiples aplicaciones de esta técnica en la ingeniería, podemos proponer la medición de la apertura de grietas. En este caso, el objetivo es analizar las variaciones que se producen en una estructura, como una barra de hormigón armado, al ser sometida a una prueba de carga (Ilustración 1). A simple vista, es fácil localizar la ubicación de la grieta principal. Sin embargo, también pueden estar presentes otras grietas más pequeñas que no se ven tan fácilmente. El uso de un tinte penetrante, Ilustración 2, permite identificar más grietas, aunque de una manera complicada y que requiere mucha mano de obra. Al utilizar DIC, con una imagen tomada antes de que se produjera la fisura y otra después, se puede ver la extensión total de la apertura de la fisura (Ilustración 3). Esto tiene la ventaja de identificar las fisuras que se han abierto sin contacto y de forma económica. Puede proporcionar mediciones precisas de la apertura de grietas, incluso si los bordes de estas no están bien definidos, y proporciona información útil, como, por ejemplo, el mejor lugar donde instalar los sensores de apertura [4].



Ilustración 1. a) Escalón de hormigón tras ser sometido a prueba de carga. Muestra una gran grieta [4].

Ilustración 2. b) Tinte penetrante para mostrar grietas que no son visibles a simple vista [4].

Ilustración 3. c) Salida del DIC como mapa de deformación local que muestra grietas no visibles para el ojo humano [4].

En la actualidad, DIC ha dejado de ser una curiosidad universitaria para ser aceptada industrialmente y, empresas de relevancia, como SIEMENS, están desarrollando soluciones que incorporan técnicas de seguimiento y registro de imágenes [Ilustración 4]. Tras toda imagen en color, siempre hay datos de campo completo precisos, fiables y cuantitativos, que se ajustan a resultados de análisis de elementos finitos. Comparado con las mediciones puntuales con sensores, DIC proporciona muchos más datos con un tiempo de instrumentación limitado. Estos resultados identifican las propiedades mecánicas de

materiales nuevos. Además, aumentan la precisión y fiabilidad de los modelos de simulación y aceleran las pruebas de validación, de modo que los ciclos de desarrollo son más rápidos y flexibles. [5].



Ilustración 4. El departamento de Energía Eólica de la Universidad Técnica de Dinamarca colabora con Siemens con el fin de utilizar las soluciones de DIC para el innovador proyecto ReliaBlade [5].

Dentro de los estudios de correlación digital de imágenes existen dos métodos bien diferenciados: el 2D y el 3D. Debemos destacar que DIC 2D no tiene percepción espacial, al contrario que DIC 3D, que sí la tiene. En el proyecto realizado solo se utilizará el 2D, ya que su objetivo es corregir las imágenes que se encuentren en una perspectiva bidimensional que pueda inducir error al programa de correlación de imágenes (ver [Ilustración 5](#)) [6].

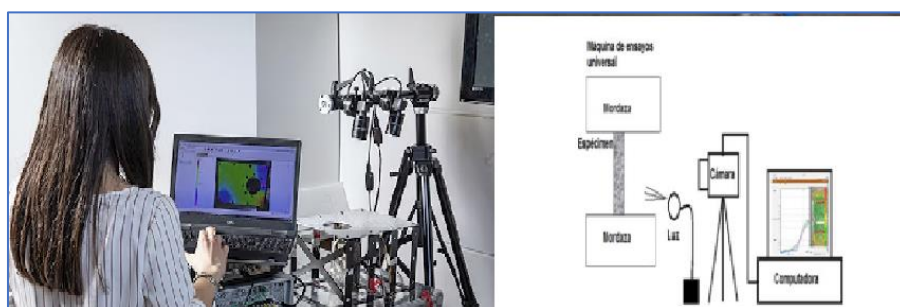


Ilustración 5. Configuración para realizar la correlación digital de imágenes 2D (Análisis de deformaciones unitarias en juntas soldadas utilizando correlación digital de imágenes, 2017) [6].

Otro de los pilares en los que se basa este trabajo, y no menos importante, es el desarrollo de algoritmos sofisticados para medir ángulos de rotación y cambios de

perspectiva, así como subsanar errores producidos en el momento de la captación de las pruebas. El objetivo es proporcionar herramientas precisas para la evaluación de estructuras y materiales; y su aplicación en posteriores ensayos experimentales. Esto permitiría el desarrollo de modelos que demuestren la aplicabilidad y eficacia de estos métodos en escenarios del mundo real [7].

Todo lo anteriormente expresado, conduce a los primeros planteamientos que se deben resolver, a fin de alcanzar los objetivos que el presente Trabajo Fin de Grado plantea.

El problema de posibles errores en los ensayos realizados en el laboratorio no es baladí para una ingeniería que, como la mecánica experimental, busca la exactitud y la precisión, pero también el desarrollo de técnicas que puedan solucionar fallas, si se producen.

Así pues, para un caso donde el objeto se encontrará perfectamente centrado y ortogonal a la visión de la cámara, cualquier línea de la imagen que fuera medida en el objeto y en píxeles, nos permitiría conocer la relación píxeles unidades/reales. Sin embargo, si, por el contrario, el objeto no se encuentra perfectamente alineado con respecto a la cámara, esta relación píxeles/realidad se perderá en función del grado y la dirección del desalineado. Por lo que, para un estudio como, por ejemplo, el de correlación de imágenes, que se basa en esta relación para hacer todos los cálculos, perdería bastante precisión. Por el simple hecho de no estar alineado, como se puede ver en la [Ilustración 6](#) [8].

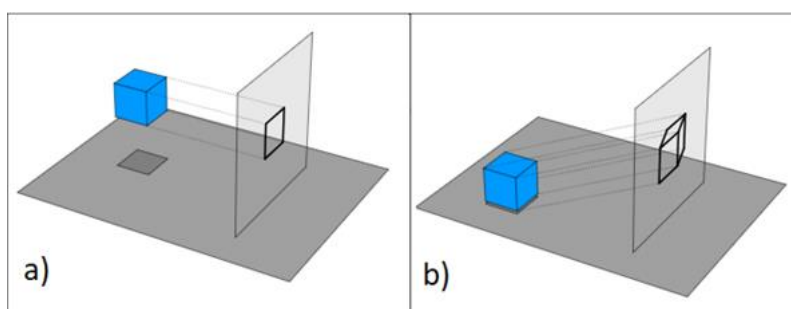


Ilustración 6. a) representa a un objeto perfectamente alineado. b) representa a un objeto desalineado [8].

Además, las imágenes en 2D no son capaces de representar de forma real el tamaño de dos objetos que se encuentran en un espacio 3D a diferentes distancias. Esto implica que, a la hora de capturar una imagen de una probeta que se encuentre inclinada con respecto a la lente, una parte del material se encontrará a más distancia que otra. Lo que significa que, al

hacer un estudio, como el de correlación de imágenes, el programa detectará que existe menor movimiento de los elementos en las zonas más alejadas de la cámara y mayor en las más cercanas. Lo cierto es que este movimiento depende completamente de la fuerza de la aplicación y no de la perspectiva de la imagen, por lo que degenera nuevamente en un error (Ilustración 7) [9].

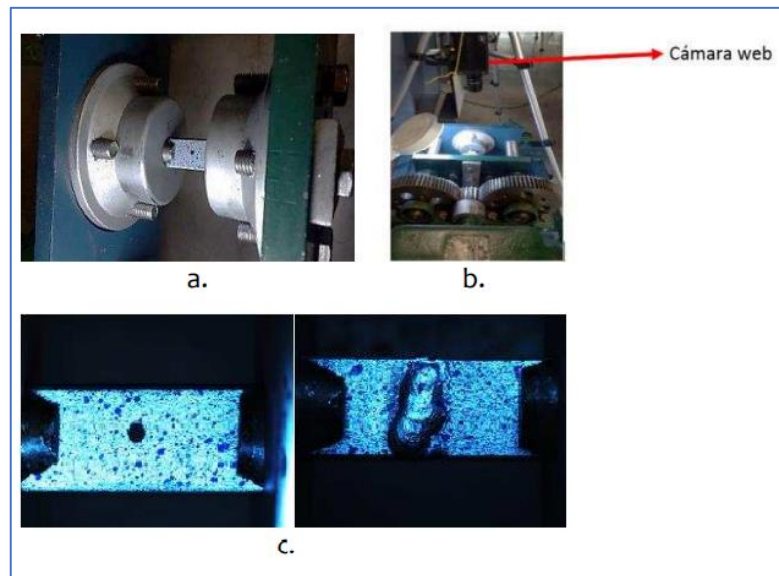


Ilustración 7. a) Probeta barrenada preparada con el moteado, b) Dispositivo que aplica tensión con la probeta montada y la cámara [10].

2. OBJETIVO

El objetivo principal de este trabajo de fin de grado (TFG) es analizar las posibilidades que ofrecen las diferentes herramientas de computación para la transformación perspectiva de imágenes, con el objetivo de utilizarlas en técnicas de campo completo en mecánica experimental. Se propone su uso en la mejora de las técnicas ópticas de campo completo, como son DIC 2D, para lograr ensayos más fiables y cómodos.

Para ello se recurre a proponer una herramienta de transformación perspectiva, que permita obtener imágenes corregidas, en las que las probetas se vean perpendiculares y se asegure una relación de magnificación homogénea.

Como objetivos secundarios se establecen:

- Analizar las diferentes herramientas utilizadas para el estudio de perspectiva y correlación de imágenes mediante revisión bibliográfica.
- Evaluar los requerimientos y requisitos experimentales de técnicas ópticas comúnmente utilizadas, como DIC 2D.
- Investigar sobre las herramientas accesibles, y preferiblemente de software libre, que puedan ser útiles para modificar las imágenes y optimizar la perpendicularidad.
- Implementar una herramienta para mejorar la técnica de transformación perspectiva mediante el desarrollo de programas informáticos anteriormente mencionados.
- Ejecutar ensayos dirigidos al estudio y constatación de la efectiva aplicación de los objetivos antes mencionados.

3. FUNDAMENTOS TEORICOS

1.1 LA VISIÓN ARTIFICIAL

Para poder alcanzar los objetivos marcados en este TFG, se debe comenzar por el análisis de lo más evidente: el proceso que posibilita el milagro de la visión humana y su posterior aplicación a la captación de imágenes de forma artificial.

Ambos procesos son parecidos, pero muy diferente, pero no cabe duda de que serían imposibles de comprender sin antes analizar el elemento que los posibilita: La Luz.

3.1.1 MODELO FÍSICO DE LA LUZ

Desde el punto de vista del procesado de imagen, se debe considerar la luz como una onda. Según este modelo, las características de un rayo de luz vienen determinadas por dos propiedades: su amplitud y su longitud de onda. Sin embargo, este modelo explicativo no es ni el primero, ni tampoco el que más se aproxima a las observaciones [11].

La primera teoría sobre la naturaleza de la luz es debida a Euclides (330 a. de C) que consideraba que la luz era una especie de rayo lanzado por el ojo hacia la cosa vista [11].

No fue hasta el siglo XVII, cuando se realizaron los mayores progresos de la mano de Newton, que hizo notables avances en la teoría del color y la dispersión. Newton era defensor de una teoría corpuscular, según la cual la luz estaba formada por un flujo de partículas proyectadas por el cuerpo luminoso. Al mismo tiempo, otros científicos, como Hooke y Huygens, defendían una teoría ondulatoria que explicaba mejor ciertos hechos, como que dos haces luminosos se cruzasen sin perturbarse. Debido al peso de la autoridad de Newton, la teoría corpuscular se impuso durante doscientos años [11]

Sin embargo, hacia 1900 se constató un nuevo fenómeno, denominado efecto fotoeléctrico, que proporcionó evidencias experimentales de que la luz tenía carácter corpuscular en su interacción con la materia. Esto llevó de nuevo al replanteamiento de la

naturaleza de la luz. Albert Einstein, Louis de Broglie y otros, construyeron una nueva teoría que llamaron teoría onda-corpúsculo. Propugnaba que la luz la forman unas partículas, los fotones, cada una con una ecuación de ondas. Así, cuando la luz interactúa con la materia, como en el efecto fotoeléctrico, se invoca a un modelo corpuscular para explicar tal interacción, mientras que, para explicar fenómenos relativos a su propagación, como en la difracción de los rayos X, se recurre a un modelo ondulatorio [11]. Este modelo dual, onda-corpúsculo, permite explicar todos los fenómenos observados hasta la fecha.

3.1.2 MODELO FISIOLÓGICO

Hay experimentos que demuestran que, en algunas ocasiones, el sistema de percepción humano confunde elementos que son iguales, y en otras no encuentra diferencias entre elementos que le parecen idénticos, pero no lo son. Esto se debe a que el sistema visual humano impone ciertas limitaciones [11].

El ojo humano es un órgano que captura la luz y la transforma en un impulso neuronal que transmite al cerebro para su procesamiento. La luz, tras atravesar una lente llamada cristalino, incide en la retina, que está situada en la parte anterior del ojo. Allí, actúa sobre unas células fotosensibles (los conos y los bastones) que forman parte de la retina. Estas células en presencia de luz generan impulsos neuronales que son enviados al cerebro mediante el nervio óptico. Luego se procesa la información que recibe y genera sensaciones. Es lo que se conoce como el proceso de percepción visual [11].

Aunque las ondas luminosas constituyen una parte muy pequeña del conjunto de ondas electromagnéticas, son especialmente interesantes porque tienen la particularidad de que son captadas por los ojos y procesadas en el cerebro. El ojo humano es capaz de distinguir radiaciones de longitudes de onda comprendidas entre 400 y 700 nanómetros (1 nanómetro = 10^{-9} metros), como indica la [Ilustración 8](#) [11].

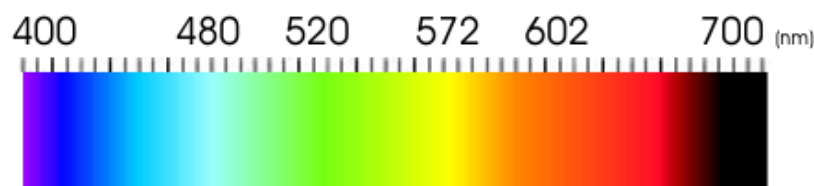


Ilustración 8. Espectro Electromagnético visible por el ojo humano [11].

3.2 REPRESENTACIÓN ARTIFICIAL DE LA REALIDAD. LA VISIÓN ARTIFICIAL

La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real para producir información que pueda tratar una máquina. Es una aplicación de la teórica clásica de matrices a dimensiones considerablemente grandes. Su objetivo principal es la deducción automática de la estructura y propiedades de un mundo tridimensional a partir de una o varias imágenes bidimensionales [12].

En comparación con la visión humana, ofrece una mejor evaluación de las magnitudes físicas y un mayor desempeño en las tareas rutinarias. [12].

El concepto surgió en la década de los sesenta y su principal objetivo era poder conectar una cámara de vídeo a un ordenador. Concretamente, la experimentación comenzó en 1959, cuando un grupo de neurofisiólogos le mostraron a un gato una serie de imágenes con la finalidad de comprender cómo las entendía y procesaba el animal. Descubrieron que sus primeras respuestas iban dirigidas a los bordes o líneas sólidas, con lo que concluyeron que el procesamiento de las imágenes empezaba con formas simples, como los bordes [13].

Al mismo tiempo, comenzó a desarrollarse la primera tecnología de escaneo artificial de imágenes, que permitió a los ordenadores adquirirlas y digitalizarlas. En 1963, las computadoras fueron capaces de transformar imágenes bidimensionales en formas tridimensionales [13].

En la década de 1960 apareció la Inteligencia Artificial, como un campo de estudio académico, y esta comenzó a usarse para resolver el problema de la visión artificial. En el año 2000, el estudio se focalizaba en el reconocimiento de objetos; y para el año 2001 se empezaron a crear aplicaciones de reconocimiento facial [13].

Como se ha indicado, lo que se pretende es extraer información del mundo físico a partir de imágenes, utilizando un ordenador. Se trata de un objetivo ambicioso y complejo que actualmente encuentra múltiples aplicaciones, como son: [14]

- **Sensores de visión.** Son los más básicos y muy fáciles de integrar en un sistema automatizado. Se emplean para detectar el paso de un elemento. Estos van a ser los utilizados para la realización de los ensayos del presente trabajo [14].

- **Cámaras inteligentes y sistemas de visión integrados.** Son cámaras que cuentan con sistemas de cálculo mucho más complejos. Su capacidad para procesar información y la posibilidad de trabajar en remoto las convierten en la solución presente en la mayor parte de la industria [14].
- **Sistemas de visión avanzados.** Tanto su *software* como su *hardware* están en un nivel superior de desarrollo y eso influye positivamente en la cantidad de datos que pueden acumular y procesar [14].

3.2.1 ELEMENTOS PARA UNA CORRECTA CAPTACIÓN DE IMÁGENES

En un principio, la calidad de la imagen estaba relacionada con las lentes y la calidad del sensor. Con el avance de la tecnología, la calidad se ha centrado en la minimización de la relación señal ruido, SNR, y en la arquitectura de las cámaras. Muchas pruebas han dado medidas sobre la calidad de la imagen. La mayoría de las métricas están relacionadas con la calidad de las ópticas, la resolución de la cámara o la relación señal ruido del elemento sensor. En general, las imágenes con altas prestaciones ópticas y bajo ruido son clasificadas como imágenes de gran calidad. Las prestaciones del sistema de visión dependen de las cámaras, lentes, iluminación, filtros ópticos, etc. Solo al final se puede plantear la calidad de la imagen [15].

El éxito de un sistema de visión depende del uso de los siguientes componentes clave, tal y como se observa en la [Ilustración 10](#):

3.2.1.1 ILUMINACIÓN

Los sistemas de Visión Artificial trabajan, normalmente, con iluminaciones estructuradas. El espacio captado por las cámaras está aislado del resto. No hay contaminación lumínica, no hay sombras inesperadas, no existen objetos que obstaculicen a los escenarios que se desean capturar. La iluminación se diseña para que sea capaz de resaltar los objetos de interés. La iluminación depende básicamente de la naturaleza de las superficies donde se va a reflejar y del tipo de fuente luminosa. Son fundamentales el conjunto de artefactos que producen radiación electromagnética para que incidan sobre los objetos a visualizar. Se pueden citar algunos elementos como lámparas, pantallas fotográficas, filtros de luz, láseres,[15].

3.2.1.2 CAPTACIÓN

Son los transductores que convierten la radiación reflejada luminosa en señales eléctricas. Fundamentalmente se habla de las cámaras CCD, no sólo en el espectro visible, sino que van desde la radiación gamma hasta la radiofrecuencia o microondas, dando paso a sensores de ultrasonidos, sonar, radar, telescopía, ... [15].

La capacidad de la cámara para capturar una imagen correctamente iluminada del objeto inspeccionado, depende del sensor de imagen dentro de la cámara. Estos sensores generalmente usan un dispositivo de carga acoplada (CCD) o un semiconductor complementario de óxido metálico (CMOS) para convertir la luz (fotones) en señales eléctricas (electrones). Básicamente, el trabajo del sensor de imagen es capturar la luz y convertirla en un ruido de imagen digital, sensibilidad y rango dinámico. La imagen es una colección de píxeles. La luz baja produce píxeles oscuros, mientras que la luz brillante crea píxeles más brillantes. Es importante asegurar que la cámara tenga la resolución del sensor correcta para la aplicación. Cuanto mayor sea la resolución, más detalles tendrá una imagen, y más precisas serán las mediciones. El tamaño de la pieza, las tolerancias de inspección y otros parámetros dictarán la resolución requerida [16].

Los dos sensores tienen el mismo fundamento, el efecto fotoeléctrico. Es decir, que acumulan carga eléctrica para cada píxel mediante fotositos. Estos producirán una corriente eléctrica de la que dependerá de intensidad de luz que se perciba. En la [Ilustración 9](#), podemos observar el esquema de ambos sensores [17].

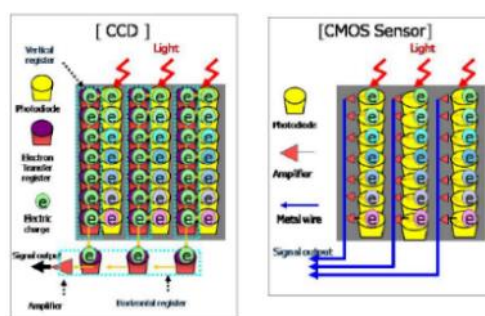


Ilustración 9. Esquema CCD y CMOS [17].

Los sensores CCD son más sensibles a la luz y trabaja con los colores primarios (RGB), a su vez es más sensible que los sensores CMOS y registra mayores sombreados en

las imágenes. Estos sensores, al ser tan sensibles, necesitan un tiempo de calentamiento, mientras que los sensores CMOS no lo requieren [17].

Los sensores CMOS rinden mucho mejor tanto en velocidad como en consumo, sin embargo, es menos sensible a la luz [17].

Por lo tanto, los sensores CCD crean imágenes de mejor calidad y con poco ruido, y los sensores CMOS son más rápidos tomando imágenes, por lo que las cámaras que necesiten que tomen unas secuencias de imágenes de forma rápida, están dispuestas con las sensaciones CMOS [18].

3.2.1.3 ADQUISICIÓN

La señal eléctrica procedente de las cámaras forma la señal de vídeo. Hay una tendencia creciente a que su naturaleza sea de tipo digital, pero todavía existen algunas señales de vídeo de carácter analógico (CCIR, PAL, RS170, NTSC,). Para ser tratadas hay que muestrearlas y cuantificarlas. Ambas tareas son realizadas por las tarjetas de adquisición. También se las llama frame grabbers. Se depositan en el bus de expansión del ordenador. Hay para buses desde PCI hasta VMP. Actualmente, también se emplean las tecnologías de USB, FireWire, multiplataforma por conexión USB, basadas en microcontrolador con interfaz Ethernet, Coax Express... [15].

3.2.1.4 PROCESAMIENTO

Suele ser a través de un ordenador o un clúster de computadoras, dependiendo de las necesidades de los algoritmos de Visión Artificial. Parten de una representación digital de las imágenes y procesan esta información hasta alcanzar otro tipo de información de más alto nivel. La transformación dependerá de la algoritmia [15].

Todas las imágenes adquiridas por medios ópticos, electrópticos o eléctricos sufren, en cierta medida, los efectos de la degeneración que se manifiesta en forma de ruido, pérdida de definición y de fidelidad de la imagen. La degradación es provocada por el ruido de los sensores de captura, imprecisiones en el enfoque de la cámara, movimiento de esta o de perturbaciones aleatorias. Los mecanismos que tratan de contrarrestar estos efectos se incluyen dentro de la etapa de procesamiento y reciben el nombre de operaciones de restauración. Se pretende reparar en la imagen los desperfectos como: deformaciones, ruido, poco o mucho contraste o brillo, etc. Son los algoritmos de procesado los que permiten

modificar la imagen a fin de eliminar los defectos mencionados. Se pretende mejorar el resultado final de la imagen capturada [12].

Las aplicaciones de los algoritmos de transformación geométrica más frecuente son [12]:

A. ALGORITMOS DE TRASLACIÓN O DESPLAZAMIENTO

Consiste en sustituir cada píxel por el correspondiente a coordenadas más el desplazamiento en cada dirección k y l . Si los desplazamientos son enteros para ambas direcciones, el proceso consiste en sustituir cada píxel (i, j) por su píxel $(i+k, j+l)$, mientras que si tienen parte decimal es necesario realizar una interpolación [12].

Resumiendo, la función de translación sería:

$$IMB(i, j) = IMA(i + k, j + l) \quad \text{Ec.1}$$

En las aplicaciones prácticas, el proceso de translación o desplazamiento se utiliza cuando se quiere posicionar cierto objeto detectado en un punto determinado para realizarle posteriores procesos como, por ejemplo: el uso de funciones lógicas AND, OR, XOR, etc.; con máscaras para obtener o eliminar ciertas partes de este, procesos de unión o fusión con otras imágenes, etc. [12].

B. ALGORITMOS DE ROTACIÓN DE GIRO

Los algoritmos de giro son generalmente los más complejos y, por lo tanto, los que tienen un tiempo de procesamiento más costoso. Debido a esto, sólo se utilizan cuando es posible obtener una posición de giro que simplifique más posteriores procesos [12].

Dado un punto $IMB(i, j)$ y se rota θ grados, las coordenadas i' y j' del nuevo punto serán:

$$\begin{bmatrix} i' \\ j' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} * \begin{bmatrix} i \\ j \end{bmatrix}$$

Por lo tanto, la versión girada de la imagen principal será:

$$IMG(i',j') = IMB(i * \cos \theta - j * \sin \theta, i * \sin \theta + j * \cos \theta) = IMA(i,j) \quad \text{Ec.2}$$

Después será necesaria realizar una interpolación.

Por ejemplo, si se tiene una imagen formada por múltiples rectángulos y cuadrados situados de forma perpendicular entre ellos y la imagen está girada, es muy útil alinearlos respecto a los ejes x e y de la imagen. De esta forma, los procesos posteriores de segmentación y análisis sobre estos rectángulos o líneas serán más rápidos y eficaces [12].

C. ALGORITMO DE ESCALADO Y DE ZOOM

Los algoritmos de escalado permiten reducir o aumentar la imagen, así como realizar el zoom de ciertas partes de la imagen.

Dado un factor de escalado α para las coordenadas x y otro β para las coordenadas y, obtendríamos:

$$IMB(k,l) = IMB(\alpha \cdot i, \beta \cdot j) = IMA(i,j) \quad \text{Ec.3}$$

Requiriendo una interpolación cuando i/α o j/β no son enteros.

El uso de las funciones de escalado no se circunscribe únicamente a la etapa de preprocesado, sino que pueden incluirse en pasos de otras etapas. Por ejemplo, existen procesos posteriores (erosionados, reducción de la resolución, etc.) que no necesitan tanta cantidad de información, reduciéndose la imagen para que esos procesos sean más rápidos ya que tratan con una imagen de tamaño reducido [12].

D. ALGORITMO DE ESPEJO

Algoritmos que realizan un espejo respecto de la horizontal, vertical o diagonal. Si la imagen es de dimensiones $M \times N$:

$$IMB(i,j) = IMA(N - i, j) \text{ (Espejo vertical)} \quad \text{Ec.4}$$

$$IMB(i,j) = IMA(i, M - j) \text{ (Espejo horizontal)} \quad \text{Ec.5}$$

$$IMB(i,j) = IMA(N - i, M - j) \text{ (Espejo diagonal)} \quad \text{Ec.6}$$

Los algoritmos de espejos se pueden realizar de múltiples formas, dependiendo de los ejes elegidos para realizarlo [12].

3.2.1.5 SUBSISTEMAS DE PERIFÉRICOS: conjunto de elementos receptores de la información de alto nivel. Puede ser un monitor de altas prestaciones gráficas, un automatismo, una impresora sacando las características [15].

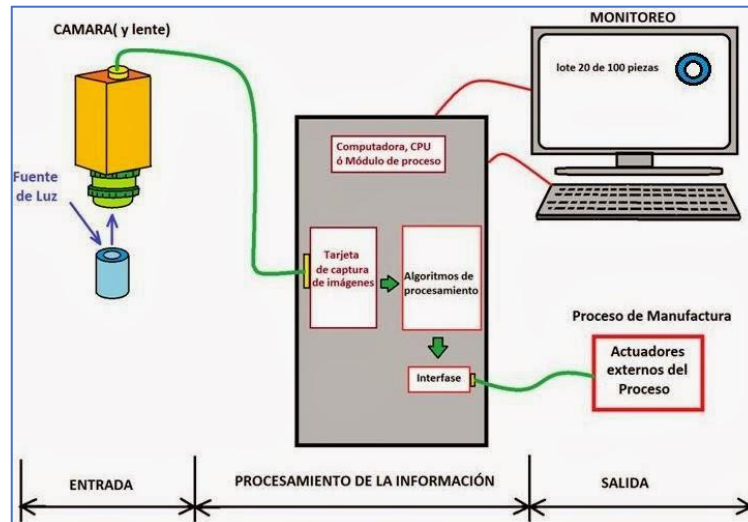


Ilustración 10: Componentes de un sistema de Visión Artificial [19].

3.2.2 ETAPAS DE UN SISTEMA DE VISIÓN ARTIFICIAL

Como se ha visto, el ser humano captura la luz y la información a través de los ojos. Esta circula y se transmite, a través del nervio óptico, hasta el cerebro, donde se procesa. Existen razones para creer que el primer paso de este procesado consiste en encontrar elementos más simples en los que descomponer la imagen (como segmentos y arcos) [11].

Después, el cerebro interpreta la escena, y, por último, actúa en consecuencia. La visión artificial, en un intento de reproducir este comportamiento, define las siguientes fases principales (ver [Ilustración 11](#)) [20]:

1. Adquisición y digitalización: es el proceso de capturar una imagen y pasarla a algún formato digital. Se utiliza una cámara para capturar la escena y después se envía a una unidad donde pueda ser procesada.

2. Preprocesamiento: es el proceso que se encarga de eliminar información que no es de interés para el problema analizado.

3. Segmentación: en este punto se reconocen y se extraen cada uno de los objetos presentes en la imagen. Para ello se utilizan técnicas como la detección de bordes o la umbralización de regiones.

4. Extracción de características: se estudian las “características” apropiadas para la identificación de los objetos deseados.

5. Identificación de objetos: el último paso consiste en la utilización de un modelo de toma de decisión para decidir a qué categoría pertenece cada objeto.

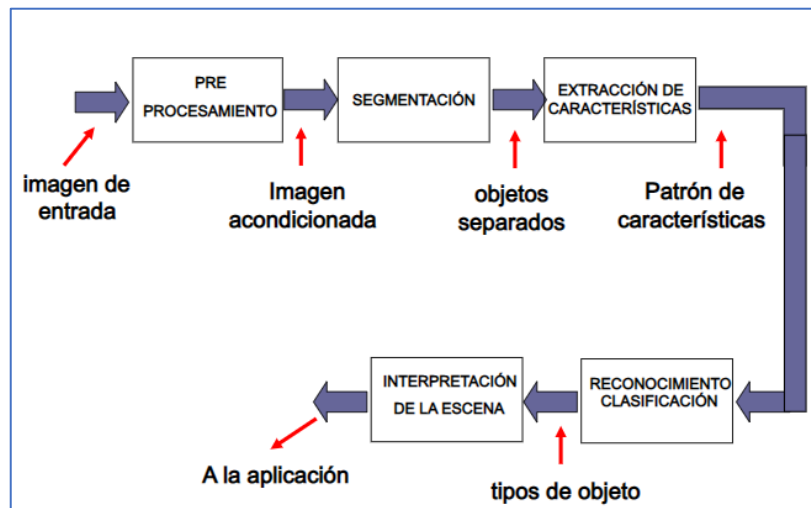


Ilustración 11. Diagrama de bloques de las etapas típicas en un sistema de visión artificial [21].

3.2.3 REPRESENTACIONES DE IMÁGENES

Según la disposición de los píxeles, las imágenes pueden ser:

- a) Unidimensionales (las obtenidas mediante cámaras lineales, asociadas a un vector de sensores).
- b) Bidimensionales (las obtenidas mediante cámaras matriciales, asociadas a una matriz de sensores). Son las más interesantes para el presente estudio.

3.2.3.1 LAS IMÁGENES BIDIMENSIONALES

Es importante diferenciar una imagen real de una digital. La primera es una imagen continua con un número infinito de puntos. Sin embargo, la segunda es una imagen discreta

con un número finito de puntos. Cada uno de los puntos de una imagen digital recibe el nombre de píxel. Cada píxel se identifica por la fila y la columna a la que pertenece [22].

Una imagen bidimensional es una función que, a cada par de coordenadas (x, y) asocia un valor relativo a alguna propiedad del punto que representa (por ejemplo, su brillo o su matiz) [11].

- Una imagen acromática, sin información de color, en la que a cada punto se le asocia información relativa al brillo, se puede representar como una superficie (Ilustración 12), en la cual la altura de cada punto indica su nivel de brillo [11].
- Una imagen en color RGB se puede representar asociando a cada punto una terna de valores que indica la intensidad de tres linternas (una roja, otra verde y otra azul) [11].
- Una imagen de color de espectro completo se puede representar asociando a cada punto un diagrama espectral de emisión de color [11].

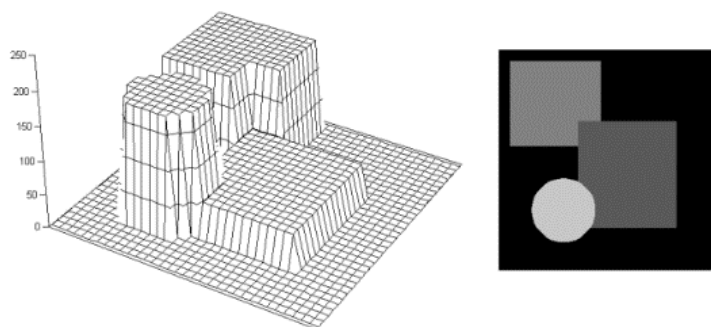


Ilustración 12. En la figura 2D la componente z puede estar representada por la intensidad del brillo de cada superficie [11].

Al proceso de obtención de imágenes digitales se le denomina digitalización. Consiste en la descomposición de la imagen real en una matriz discreta de puntos de un determinado tamaño, donde cada uno tiene un valor proporcional a su nivel de color. Por tanto, puede decirse que una imagen digital se puede asimilar a una matriz de n filas y m columnas. A cada celda de la matriz se le denomina píxel (picture x element) y esta representa una superficie que es función de su tamaño $(\Delta x, \Delta y)$. A cada píxel le corresponde uno o más valores digitales (número o valor digitales “Nd”) (Ilustración 13 e Ilustración 14) [23].

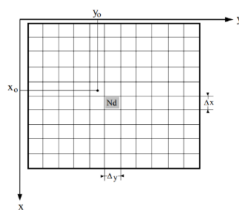


Ilustración 13. Definición de una imagen digital [23].

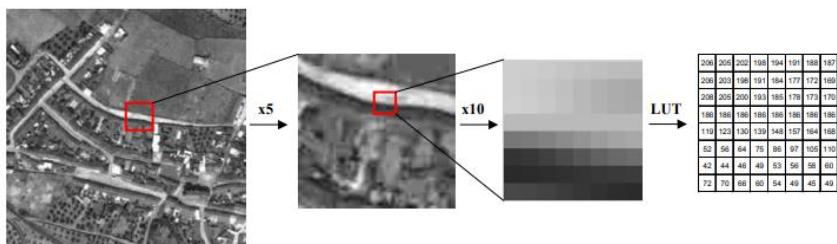


Ilustración 14. Imagen digital. Representación visual y representación numérica [23]

La utilización de las imágenes digitales lleva consigo una serie de ventajas que están íntimamente ligados a las características de la Fotogrametría Digital frente a las técnicas clásicas de tipo analógico.

Las ventajas más importantes son: [23]

1. Por su soporte de almacenamiento, carecen de los problemas derivados de la estabilidad dimensional que afectan a las imágenes analógicas cuando se modifican las condiciones medioambientales de su almacenamiento.
2. No requieren manipulación directa a la hora de ser utilizadas, por lo que se eliminan los problemas ligados al deterioro que dicha manipulación conlleva. Las imágenes digitales pueden ser modificadas para mejorar su visualización (ajuste de niveles de brillo y contraste) mediante la aplicación de técnicas de análisis de imágenes. Por otro lado, también es posible modificar la imagen para poner de manifiesto algún elemento de interés en la misma (por ejemplo, mediante la aplicación de un filtro de detección de bordes).
3. Pueden ser transmitidas de una forma sencilla, rápida y segura a través de las redes informáticas, además es posible que distintos operadores trabajen de una forma

simultánea o secuencia] sobre el mismo fotograma, lo que facilita sensiblemente los procesos de edición y revisión.

4. Los productos derivados del tratamiento de las imágenes digitales son obtenidos directamente en formato digital, por lo que son directamente integrables en sistemas tipo CAD o SIG.

3.3 CORRELACIÓN DIGITAL DE IMÁGENES

La correlación digital de imágenes (DIC por sus siglas en inglés), tanto en superficies 2D como 3D, es una técnica que puede resultar ideal para el estudio de la propagación de grietas, vibración, impacto y la deformación de materiales, entre otros, en aplicaciones del mundo real, ya que tiene el potencial de convertirse en una solución barata, simple pero precisa [4].

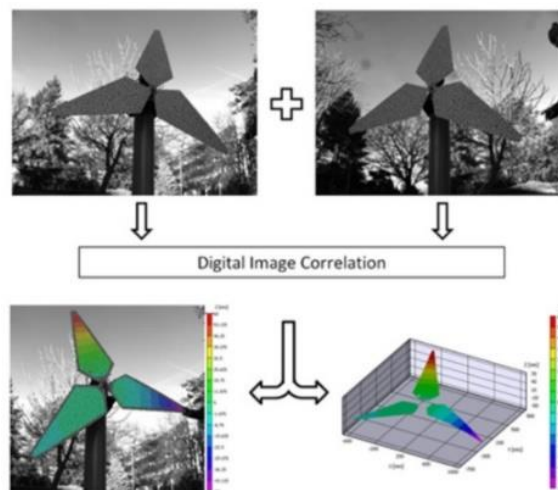


Ilustración 15. Uso de la correlación de imágenes digitales en el montaje experimental para las deformaciones de las palas del rotor [24].

Hoy en día, se ha convertido en uno de los métodos ópticos de campo completo más utilizados para mediciones de movimiento y deformación. En su versión más simple, utilizando una sola cámara, DIC se utiliza para mediciones bidimensionales en el plano (2D-DIC). De igual modo, se pueden realizar mediciones fotogramétricas tridimensionales (3D-DIC) utilizando dos cámaras en configuración estéreo. Además de la buena precisión de medición del método, DIC también ofrece otras características atractivas que incluyen: Configuración experimental relativamente simple, preparación de muestras simple o nula y bajos requisitos para el entorno de medición, como se puede ver en la [Ilustración 15](#) [25].

Las imágenes se pueden obtener de una gran cantidad de fuentes, como CCD o CMOS, convencionales o cámaras digitales comerciales, vídeo de alta velocidad, macroscopios y microscopios, etc. El proceso de correlación DIC no se limita a imágenes ópticas y también se puede aplicar a otros conjuntos de datos, como mapas de rugosidad de superficies [4].

Se trata de una técnica óptica de formas de tipo full field measurement (de campo completo) sin contacto, y desplazamientos a partir de los cuales pueden calcularse las deformaciones unitarias. Así pues, tiene como objetivo la localización, de forma automática, de un mapa de desplazamientos en la superficie del objeto [4]. Para simplificar el problema, supongamos que partimos de que se conoce, de forma aproximada, la ubicación del objeto dentro de la imagen y, por tanto, es factible hacer una preselección del área de búsqueda. Esta área preseleccionada se conoce como región de interés (Region of interest (ROI) en inglés) y dentro de ella vamos a localizar unas determinadas agrupaciones de píxeles, que llamaremos facetas (facet o subsets), y que definen un objeto concreto. La idea básica de DIC es comparar dos imágenes digitales adquiridas en diferentes estados (por ejemplo, una antes de la deformación y la otra después), dividiendo la imagen en facetas y luego haciendo coincidir, matemáticamente, las facetas entre las dos imágenes (según los niveles de intensidad) para determinar la nueva ubicación de cada una de ellas en la segunda imagen. A partir de ahí, se puede obtener el mapa de deformación de campo completo, como se puede apreciar en la [Ilustración 16](#) [25].

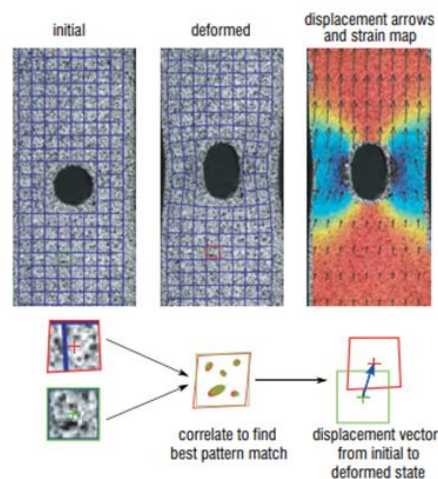


Ilustración 16. La imagen se divide en facetas para luego buscar una nueva posición tras la deformación [27].

Cuando el área de búsqueda es demasiado grande en las dos direcciones (x e y), se utiliza lo que se denomina correlación bidimensional. En cualquier método de correlación que se utilice, el primer paso es comparar, dentro del área de búsqueda, los componentes que tengan un tamaño similar, procurando buscar fuera de esta área los puntos que se corresponden con los que hay dentro del área que se representa. Puesto que una imagen fotográfica está grabada en dos dimensiones, la correlación de imágenes es esencialmente un trabajo en dos dimensiones [23].

DIC presenta numerosas ventajas con respecto a las técnicas tradicionales, entre las que se puede destacar, su capacidad de obtener más información sobre la deformación en un mismo ensayo, la simplicidad de montaje e implementación y la minimización de errores. Estos suelen ocurrir, en los ensayos realizados por técnicas de contacto, debido a los deslizamientos de extensómetros, galgas extensiométricas u otro tipo de sensor de deformación [28].

La medición del desplazamiento y la deformación son tareas importantes en la ingeniería en general y, en mayor medida, en la experimental. Sin embargo, medir estos parámetros, fuera del laboratorio, requiere una elección difícil entre técnicas convencionales, ya que se debe equilibrar la precisión, la simplicidad y el costo. El auge y desarrollo de la captura de la imagen digital y la proliferación de herramientas, técnicas y procesos basados en la visión artificial deben ser aprovechadas para obtener información de gran fiabilidad y la utilización en los experimentales [4].

Por otro lado, se han desarrollado técnicas de software para obtener resoluciones de subpíxeles y permitir la ejecución eficiente de los algoritmos. Estos permiten realizar mediciones de alta resolución, de modo que, con la fotografía digital disponible comercialmente, la deformación de la superficie se puede medir hasta una parte por millón del campo de visión [4].

Como se ha indicado, la implementación de los programas de correlación es bastante básica. Se numeran las posiciones de cada punto y guarda su ubicación (x_i), de esta forma se tendrá la posición inicial. Ahora simplemente comparará las imágenes posteriores en las que se le está realizando algún tipo de ensayo y donde esas posiciones serán diferentes (x_f), obteniendo así una variación del desplazamiento (ver **¡Error! No se encuentra el origen de la referencia.**) [29].

Para el caso 2D:

$$\Delta x = x_f - x_i \text{ o } \Delta y = y_f - y_i \quad \text{Ec.7}$$

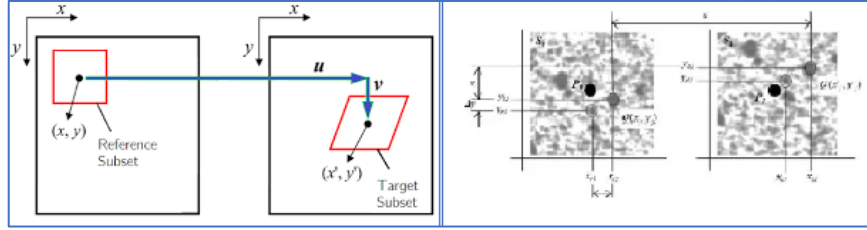


Ilustración 17. A la derecha representación del desplazamiento de un punto [30]. A la izquierda Correlación de dos imágenes, S_r y S_d , son las ROI de la imagen de referencia y con una deformación de una superficie respectivamente [10].

Todo lo expresado ha propiciado que el método DIC sea extremadamente popular entre la comunidad de mecánica experimental, y tanto 2D-DIC como 3D-DIC se utilizan cada vez más en una amplia gama de aplicaciones [21].

3.3.1 CORRELACIÓN DE IMÁGENES DIGITAL 2D

A mediados del siglo pasado, investigadores en inteligencia artificial y robótica desarrollaron algoritmos y metodologías de estéreo visión en paralelo con aplicaciones de fotogrametría. El objeto de interés, de esas primeras investigaciones, se basó en sectores como: el reconocimiento de caracteres, microscopía, medicina y radiología y fotogrametría. Por entonces, no existían aplicaciones de esta tecnología para la ingeniería. Pasado un tiempo, y habiendo alcanzado un mayor desarrollo la digitalización de imágenes, algunos de los campos experimentales de la mecánica del sólido se centraron en investigar posibles aplicaciones de la tecnología láser, dando lugar al desarrollo de la holografía, la interferometría de Moiré y, la interferometría holográfica, entre otros [31].

Uno de los primeros documentos en el que se propuso la utilización de imágenes digitales para obtener las deformaciones fue publicado por Peter y Ranson en 1982 [32]. Posteriormente, se realizaron numerosas investigaciones en relación con el algoritmo de optimización del método. A esta técnica para el cálculo de las deformaciones se le denomina actualmente correlación de imágenes digitales (DIC) [31].

Siguiendo los criterios que se emplean en la correlación digital 2D, se puede clasificar en dos grupos:

- CC: Se basa en la correlación cruzada. Presenta varias variantes, como es la correlación cruzada simple (CC), la normalizada (NCC), y la normalizada cero (ZNCC) [29].
- SSD: Se basa en las diferencias de sumas cuadradas. También dentro de este grupo hay variantes como la normalizada (ZSSD), y la normalizada cero (ZNSSD) [29].

Durante la realización del ensayo se toman imágenes sucesivas que tendrán unos puntos con sus ubicaciones. El desplazamiento que sufre se realiza mediante una transformación lineal de primer orden: [3]

$$\tilde{y}_{cur_j} = y_{ref_j} + v_{rc} + \frac{\delta v}{\delta x_{rc}}(x_{ref_i} - x_{ref_c}) + \frac{\delta v}{\delta y_{rc}}(y_{ref_j} - y_{ref_c}) \quad \text{Ec.8}$$

$$p = \left\{ u \ v \ \frac{\delta u}{\delta x} \ \frac{\delta u}{\delta y} \ \frac{\delta v}{\delta x} \ \frac{\delta v}{\delta y} \right\}^T \quad \text{Ec.9}$$

x_{ref_i} e y_{ref_j} : Son las coordenadas “x” e “y” en el que esta, ubicado un punto dentro de la imagen de referencia [3].

x_{ref_c} e y_{ref_c} : Sitúan las coordenadas “x” e “y” del centro de la imagen de referencia [3].

x_{cur_i} e y_{cur_j} : Son las coordenadas “x” e “y” en el que está ubicado un punto dentro de la imagen realizada en ese instante [3].

i, j : Se utilizan para señalar en que ubicación se sitúan los puntos del centro de esta [3].

S : Engloba la pertenencia de todos los puntos de un subconjunto [3].

Subíndice rc: Define que la metodología realizada es una transformación del sistema de referencia al de la imagen actual [3].

P: Es el vector de desplazamiento o deformación en este caso es de deformación [3].

Dado que se puede dar la ocasión de que el sistema de referencia pueda tener algún tipo de deformación, se tendrá en cuenta haciendo una pequeña modificación:[3]

- Siendo *rr* la deformación (posición final menos la posición inicial del sistema de referencia) que sufre la imagen de referencia [3].

Al final lo que se busca es encontrar el valor del vector de deformaciones en el sistema actual considerando que el de referencia tiene un valor igual a cero, por lo tanto x_{ref_i} e y_{ref_j} serán muy parecidos a $\tilde{x}_{cur_i}$ e $\tilde{y}_{cur_j}$ [3].

$$\tilde{x}_{cur_i} = x_{ref_i} + u_{rc} + \frac{\delta u}{\delta x_{rr}}(x_{ref_i} - x_{ref_c}) + \frac{\delta u}{\delta x_{rc}}(y_{ref_i} - y_{ref_c}) \quad \text{Ec.10}$$

$$\tilde{y}_{cur_j} = y_{ref_j} + v_{rc} + \frac{\delta v}{\delta x_{rr}}(x_{ref_i} - x_{ref_c}) + \frac{\delta v}{\delta y_{rc}}(y_{ref_j} - y_{ref_c}) \quad \text{Ec.11}$$

Será necesario conseguir la similitud entre el subconjunto tanto actual como de referencia. Para ello, se comparan los valores de la escala de grises en todos los puntos actuales y de referencia. Por ello, el criterio de correlación cruzada normalizada buscará la obtención de un valor próximo a 1, mientras que, si se sigue un criterio de mínimos cuadrados, se buscará un valor próximo a 0 para poder asegurar una buena coincidencia [3].

3.4 FUNDAMENTOS PARA LA PROGRAMACIÓN

A continuación, se procede a analizar el lenguaje de programación que se ha utilizado para implementar las herramientas de cambio de perspectiva y de estudio de giro, necesarias para la realización de los ensayos que son uno de los objetivos finales del presente trabajo.

3.4.1 PYTHON

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en él. Son ejecutadas directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina [33].



Ilustración 18. Imagen del logotipo del lenguaje de programación Python [33].

Python fue creado por Guido van Rossum a principios de los años 90. Su nombre se inspira 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 [34]. En la [Ilustración 18](#) se puede ver su conocido logotipo.

Sus principales características son:

- **Lenguaje interpretado:** Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente un ordenador (lenguajes compilados) [34].

La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo, los lenguajes interpretados son más flexibles y ágiles [34].

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semiinterpretado [34].

- **Multiplataforma:** El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que, si no utilizamos librerías específicas de cada plataforma, el programa podrá correr en todos estos sistemas sin grandes cambios [34].

- **Cercano al lenguaje natural:** Python es un lenguaje que todo el mundo debería conocer. Su sintaxis simple, clara y sencilla; el tipado dinámico, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otras cosas, hacen que desarrollar una aplicación en Python sea sencillo y rápido [34].

3.4.2 BIBLIOTECAS

Python tiene una extensa biblioteca estándar, comúnmente citada como una de sus mayores fortalezas. Proporcionando herramientas adecuadas para muchas tareas. Esto es deliberado y ha sido descrito como una filosofía de Python de "todo incluido" ("batteries included" en inglés). Para aplicaciones orientadas a Internet se admiten una gran cantidad de formatos y protocolos estándar (como MIME y HTTP). De igual modo, se incluyen módulos para crear interfaces gráficas de usuario, conectarse a bases de datos relacionales, puede realizar cálculos con decimales de precisión arbitraria, manipular expresiones regulares y realizar pruebas unitarias. Para las pruebas de software, la biblioteca estándar proporciona los módulos unittest y doctest [35].

Algunas partes de la biblioteca estándar están cubiertas por especificaciones (por ejemplo, la implementación WSGI wsgiref sigue la PEP 333), pero la mayoría de los módulos no lo están. Se especifican mediante su código, documentación interna y conjunto de pruebas (si se proporciona). Sin embargo, debido a que la mayor parte de la biblioteca estándar es código Python multiplataforma, solo hay algunos módulos que deben ser modificados o completamente reescritos por implementaciones alternativas [35].

La biblioteca estándar no es esencial para ejecutar Python o integrarlo en una aplicación. Por ejemplo, Blender 2.49 omite la mayor parte de la biblioteca estándar [35].

El Índice de Paquetes de Python, que es su repositorio oficial de software de terceros, contiene más de 25.000 "paquetes", que cubren una amplia gama de funcionalidades, incluyendo: [30]

- Interfaz gráfica de usuario, marco de trabajo web, multimedia, bases de datos, redes y comunicaciones.
- Marcos de pruebas, herramientas de documentación, administración de sistemas.
- Cómputo científico, procesamiento de texto, procesamiento de imágenes.

En la actualidad, el desarrollo de algoritmos y programas informáticos ha tomado un papel relevante en el avance de la ingeniería mecánica y la experimentación. En este trabajo, se ha optado por utilizar Python, para desarrollar una serie de programas que permitan realizar un cambio de perspectiva de una imagen.

3.5 OPEN CV

Dentro de las citadas bibliotecas y herramientas de Python, existen muchas que facilitan el procesamiento y análisis de imágenes. Una de las bibliotecas más destacadas, en este campo, es OpenCV (Open Source Computer Vision Library), que proporciona una amplia gama de funcionalidades y algoritmos para el tratamiento y análisis de imágenes y visión artificial.



Ilustración 19. Imagen del logotipo de la librería para Python OpenCV [38].

“OpenCV es una librería de visión artificial, cuyo logotipo aparece en la Ilustración 19. Lo que la caracteriza es que permite que medios informáticos puedan «ver», adquirir, procesar e interpretar imágenes del mundo real. Algo parecido a lo que logran los ojos humanos, en conjunto con el cerebro, para comprender lo que se está viendo.

El reconocimiento de personas, animales y objetos en general es fundamental en muchas de las ramas de la inteligencia artificial y aprendizaje de modelos” [37].

OpenCV [Open Source Computer Vision) es una biblioteca de visión artificial de código abierto [39]. La biblioteca está escrita en C. y C++ y se ejecuta en Linux, Windows y Mac OS X. Hay un desarrollo activo en interfaces para Python, Ruby, Matlab y otros lenguajes [39].

OpenCV fue diseñado para la eficiencia a la hora de programar y tiene un claro enfoque hacia aplicaciones en tiempo real. Está escrito en C optimizado y puede aprovechar el multinúcleo de los procesadores [39].

Uno de los objetivos de OpenCV es proporcionar una infraestructura de visión artificial fácil de usar. La biblioteca contiene más de 500 funciones que abarcan muchas áreas de la visión, inspección de productos, imágenes médicas, seguridad, interfaz de usuario, calibración de cámaras, estéreo visión y robótica. Debido a que la visión artificial y el aprendizaje automático a menudo van de la mano, también contiene una biblioteca de aprendizaje automático (MLL) completa y de uso general [39].

3.5.1 FUNCIONAMIENTO DE OPENCV

OpenCV es una biblioteca de computación visual que se puede usar para desarrollar aplicaciones de visión artificial en Python. Esto significa que podemos aprovechar las herramientas de procesamiento de imágenes que proporciona OpenCV para desarrollar aplicaciones que puedan analizar imágenes, detectar objetos, realizar el seguimiento de objetos, identificar patrones, etc. Todo esto se consigue a través de herramientas básicas para el procesamiento de imágenes, que permiten cargar y mostrar una imagen, cambiar su tamaño y recortar partes de ella. Estas operaciones se pueden realizar utilizando la librería de procesamiento de imágenes OpenCV. Utilizando funciones sencillas. Por ejemplo, para realizar un cambio de tamaño en la imagen se utiliza el código que aparece en la [Ilustración 20](#) [40].

```
# Dimensiones a las que se quiere reescalar
width = 300
height = 200

# Redimensionar la imagen
img = cv2.resize(img, (width, height))
```

Ilustración 20. Código de modificación de tamaño de imagen en OpenCV [40].

Las herramientas básicas son:

- Cargar y mostrar imagen
- Cambiar tamaño de imagen
- Recortar parte de una imagen
- Reconocimiento de Objetos
- Detección de Objetos
- Etiquetado de Objetos

3.5.2 FUNDAMENTOS TÉCNICOS

El presente apartado se centrará fundamentalmente en dos conceptos: los parámetros y técnicas necesarias para la calibración de la cámara, y la detección de bordes.

En relación con el primer fundamento, se debe tener presente que para este estudio se hace necesaria una aproximación a la geometría de las transformaciones proyectivas del plano. Esto se debe a que, cuando se capturan imágenes con una cámara, se producen distorsiones, de tal modo que, algunas propiedades geométricas se conservan, como la colinealidad (una línea recta se proyecta en una recta). Pero otros no, por ejemplo, el paralelismo: en general, las líneas paralelas no se presentan como tales en la imagen. Por este motivo, la geometría proyectiva se convierte en una herramienta de gran utilidad porque aporta traslaciones en el proceso de adquisición de la imagen, al mismo tiempo que da una representación matemática apropiada para los cálculos [41].

En cuanto al segundo elemento, debemos destacar, que la detección de bordes se refiere a un conjunto de técnicas matemáticas para detectar bordes o curvas en una imagen digital cuando el brillo de la imagen cambia bruscamente o, más formalmente, tiene discontinuidades. La detección de pasos es el problema de identificar discontinuidades en señales unidimensionales, mientras que la detección de cambios es el problema de encontrar discontinuidades en la señal a través del tiempo. En el procesamiento de imágenes, la visión artificial y la visión por ordenador, la detección de bordes es una técnica fundamental, especialmente en los campos de la identificación y la extracción de características [42].

3.5.2.1 NOTACIÓN DE LOS PARÁMETROS DE LA CÁMARA

Los parámetros de una cámara se dividen en 2 tipos [43]:

a) Los Parámetros Extrínsecos. Estos identifican la orientación y posición de la cámara con respecto a las coordenadas mundiales. Son:

- *Vector de Traslación t* , definido en el espacio describiendo las posiciones relativas de los orígenes de los dos marcos de referencia, el de la cámara y las mundiales.
- *Matriz de Rotación R* , trayendo los correspondientes ejes de los dos marcos uno sobre el otro, con la condición de ser una matriz ortogonal.

En la [Ilustración 21](#) se muestran los parámetros extrínsecos de una cámara en un modelo en proyección perspectiva en donde se identifican a (X^w, Y^w, Z^w) como los ejes en coordenadas mundiales, (X^c, Y^c, Z^c) como los ejes en coordenadas de la cámara y (x, y) las coordenadas en el plano de la imagen. Se aprecia que w_0 denota la transformación de (R, t)

b) Los Parámetros Intrínsecos. Estos caracterizan las propiedades ópticas, geométricas y digitales de la visión de la cámara que son necesarias para unir las coordenadas en píxeles de un punto imagen con las correspondientes coordenadas en el marco de la cámara. Estos parámetros se muestran en la Ilustración 21 y son [43]:

- *Longitud Focal f* que es la distancia del plano imagen al centro de la lente.
- *Coordenadas del centro de imagen o punto principal (O_x, O_y)* que intersecta el eje óptico de la cámara y el plano imagen.
- *Tamaño efectivo del píxel en dirección horizontal y vertical S_x, S_y* (en milímetros), que dan el aspecto de una escena tomada según el escalamiento y los elementos receptores de la cámara. También es conocida como la razón de aspecto α , que es igual al tamaño del píxel en dirección vertical entre el tamaño del píxel en dirección horizontal.
- *Coficiente de distorsión radial $K1$* , que se presenta debido a que, por la naturaleza de la lente o del proceso de adquisición de una imagen, al captar objetos formados por líneas rectas, estas aparecen en la imagen como líneas curvas. Es decir, representa el desplazamiento radial dependiendo de la calidad de la lente del cámara usado y la distancia del punto en el espacio al centro de imagen. La distorsión es un fenómeno no deseable y más en modelos geométricos.

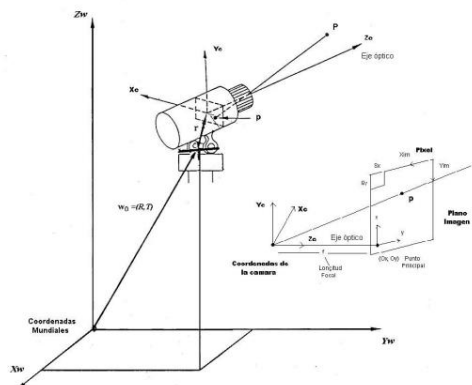


Ilustración 21. Modelo de cámara en proyección perspectiva [43].

Así pues, un punto 2D se representa por $m = [u, v]^T$. Un punto 3D se representa por $M = [X, Y, Z]^T$. Usamos $\hat{m}$ y $\hat{M}$ para representar al vector aumentado agregando 1, como último elemento: $\hat{m} = [u, v, 1]^T$ y $\hat{M} = [X, Y, Z, 1]^T$. La relación entre un punto 3D M y su proyección de imagen m será: [45]

$$s\hat{m} = A[R \ t]\hat{M} \quad \text{Ec.12}$$

Donde s es un factor de escala, $[R \ t]$ son los llamados parámetros extrínsecos (definen la orientación externa de la cámara), R el vector de rotación y t el vector de traslación que relaciona el sistema de coordenadas mundial con el sistema de coordenadas de la cámara, y A , llamada matriz intrínseca de la cámara (define parámetros de la cámara tales como la distancia focal, el punto principal y el centro óptico) será: [44]

$$A = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Ec.13}$$

- Donde (u_0, v_0) son las coordenadas del punto principal de la imagen. Un punto que llamaremos principal.
- α y β los factores de escala de los ejes u y v de la imagen.
- γ el parámetro que describe la asimetría de los ejes [44].

3.5.2.2 HOMOGRAFIA ENTRE EL PLANO MODELO Y LA IMAGEN

Las transformaciones que van a permitir las proyecciones homográficas son de gran importancia para la realización de los diferentes ensayos que se deben realizar en este TFG. Como se ha indicado, las cámaras utilizadas en los ensayos tienen las limitaciones propias de la observación en un plano, por lo que las proyecciones van a ser una herramienta fundamental para obtener una correspondencia que posibilite la simulación de varios puntos de observación simultáneos [45].

Una homografía se define como una proyectividad biyectiva $f(x) = x'$ entre dos espacios n-dimensionales, es decir, una función que asigna a cada elemento x un elemento x' tal que, para algún l , perteneciente a R , se cumple $l x' = A \cdot x$, siendo A una matriz invertible. En particular, asigna a cada elemento otro de su misma especie, a un punto corresponde otro punto, a una recta otra recta, a un plano otro plano, etc. Y es reversible y univoca, de forma que a un objeto origen corresponde un sólo objeto imagen y viceversa [45].

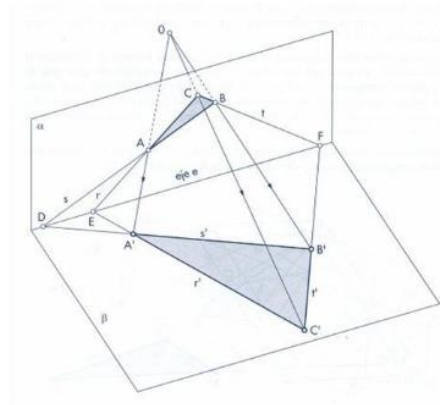


Ilustración 22. Proyección homográfica de dos figuras planas [45].

Así pues, la homografía es una rotación proyectiva que establece una correspondencia entre dos formas geométricas, de modo que, a un elemento, punto o recta, de una de ellas le corresponde otro elemento de la misma especie, punto o recta, de la otra (ver [Ilustración 22](#)) [45].

Asumiendo que el plano modelo está en $Z = 0$, se debe expresar la columna de rotación de la siguiente manera, partiendo de la ecuación del apartado anterior [44]:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} =$$

$$= A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

De manera que $\hat{M} = [X \ Y \ 1]^T$ y $\hat{m} = [u \ v \ 1]$, por lo que un punto modelo $\hat{M}$ tiene una relación homográfica H con el punto $\hat{m}$, donde H es: [44]

$$\widetilde{s\mathbf{m}} = H\widetilde{\mathbf{M}} \text{ con } H = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \quad \text{Ec.14}$$

Por último, se calcula H de la siguiente manera $H = [h_1 \ h_2 \ h_3]$ y de sustituirse en la ecuación anterior [44].

$$\begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = \lambda A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \quad \text{Ec.15}$$

Siendo λ un valor escalar arbitrario y sabiendo que r_1 y r_2 son ortonormales las ecuaciones resultantes serán las siguientes [44]:

$$h_1 A^{-T} A^{-1} h_2 = 0 \quad \text{Ec.16}$$

$$h_1 A^{-T} A^{-1} h_1 = h_1 A^{-T} A^{-1} h_1 \quad \text{Ec.17}$$

Pudiendo de esta forma despejar la matriz A gracias a la combinación de estas tres ecuaciones [44].

3.5.2.3 LA CALIBRACIÓN DE LA CÁMARA

Calibrar la cámara significa encontrar sus parámetros intrínsecos, extrínsecos y posibles distorsiones. También va a ser un paso necesario, en la visión artificial, para extraer información de las imágenes 2D, en otras palabras, las magnitudes contenidas en ellas.

Las ecuaciones obtenidas son de gran importancia, ya que van a ser la base para realizar los algoritmos que, mediante diferentes manipulaciones, van a permitir los cambios en las imágenes necesarios para los diferentes ensayos proyectados.

Ya habiendo resuelto el concepto de homografía de un punto en la imagen, solo faltaría resolver el problema de calibración de la cámara: [44]

Considerando todo lo anterior, y siendo B la calibración, esta se resolvería de la siguiente manera [44]:

$$B = A^{-T} A^{-1} = \begin{bmatrix} B_{11} & B_{21} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2 \beta} & \frac{v_0 \gamma - u_0 \beta}{\alpha^2 \beta} \\ -\frac{\gamma}{\alpha^2 \beta} & -\frac{\gamma}{\alpha^2 \beta} + \frac{1}{\beta^2} & -\frac{\gamma(v_0 \gamma - u_0 \beta)}{\alpha^2 \beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0 \gamma - u_0 \beta}{\alpha^2 \beta} & -\frac{\gamma(v_0 \gamma - u_0 \beta)}{\alpha^2 \beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0 \gamma - u_0 \beta)^2}{\alpha^2 \beta^2} - \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} \quad \text{Ec.18}$$

Nótese que B es simétrica, y se define por un vector 6D

$$b = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T \quad \text{Ec.19}$$

Definimos cada columna de la matriz por i siendo un vector H de la forma $h_i = [h_{i1} \ h_{i2} \ h_{i3}]^T$. Se obtiene: [36]

Siendo:[44]

$$h_i^T B h_j = v_{ij}^T b \quad \text{Ec.20}$$

$$v_{ij} = [h_{i1} h_{j1}, h_{i1} h_{j2} + h_{i2} h_{j1}, h_{i2} h_{j2}, h_{i3} h_{j1} + h_{i1} h_{j3}, h_{i3} h_{j2} + h_{i2} h_{j3}, h_{i3} h_{j3}]^T \quad \text{Ec.21}$$

Conociendo las dos ecuaciones fundamentales (Ec.12 y (Ec.14), de una homografía dada, pueden ser reescritas como 2 ecuaciones homogéneas en b :[44]

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \quad \text{Ec.22}$$

Si n imágenes del plano modelo son observadas, y guardamos las n ecuaciones Ec.22 correspondientes resulta:[44]

$$Vb = 0 \quad \text{Ec.23}$$

Donde V es una matriz de $2n \times 6$. Si $n \geq 3$,

En general habrá una solución única b definida por el factor de escala.

- Si $n = 2$, se impondrá la restricción $\gamma = 0$, es decir, $[0 \ 1 \ 0 \ 0 \ 0 \ 0]b = 0$, que se agrega como una ecuación adicional (Ec.20).
- Si $n = 1$, solo se podrán resolver dos parámetros intrínsecos de la cámara, por ejemplo, α y β , asumiendo que u_0 y v_0 son conocidos (por ejemplo, en el centro de la imagen) siendo $\gamma = 0$

La solución de (Ec.20) es conocida como el vector propio de $V^T V$ asociado con el valor propio más pequeño (equivalentemente, el vector singular derecho de V asociado con el valor singular más pequeño). Una vez que se conoce b , se puede calcular toda la matriz intrínseca de la cámara A [44].

Una vez que A es conocida, los parámetros extrínsecos para cada imagen pueden ser calculados. A partir de Ec.12 se deduce: [44]

$$r_1 = \lambda A^{-1} h_1 \quad \text{Ec.24}$$

$$r_2 = \lambda A^{-1} h_2 \quad \text{Ec.25}$$

$$r_3 = r_1 \times r_2 \quad \text{Ec.26}$$

$$t = \lambda A^{-1} h_3 \quad \text{Ec.27}$$

Siendo

$$\lambda = 1/\|A^{-1} h_1\| = 1/\|A^{-1} h_2\| \quad \text{Ec.28}$$

3.5.2.4 DETECCIÓN DE BORDES

Cuando hablamos de bordes en una imagen, hablamos de los píxeles donde hay un cambio de intensidad. Los bordes de una imagen se pueden definir como transiciones entre dos regiones de niveles de gris significativamente distintos. Estos suministran una valiosa información sobre las fronteras de los objetos que puede ser utilizada para la segmentación de la imagen, reconocimiento de objetos, visión estéreo, etc. [46].

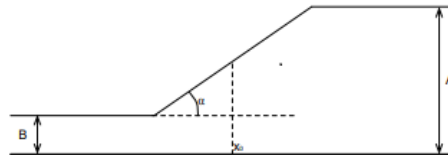


Ilustración 23. Modelo unidimensional y continuo de un borde ideal [46].

En la [Ilustración 23](#) se muestra un modelo unidimensional y continuo de un borde.

En las imágenes reales los bordes nunca se ajustan totalmente al modelo anterior.

Las causas para ello son diversas, destacándose las siguientes:

- Las imágenes son discretas.
- Están afectadas por ruido diverso.
- El origen de los bordes puede ser muy diverso: bordes de oclusión, superficies de diferente orientación, distintas propiedades reflectantes, distinta textura, efectos de la iluminación (sombras y/o reflejos), etc.

La importancia que tiene la determinación de los contornos de una figura dentro de una imagen radica en que muchos algoritmos requieren que se demarquen los objetos que están contenidos en una imagen. Por lo tanto, el primer paso del proceso de cambio de perspectiva es el procesamiento de la imagen, para que esta solo muestre los bordes. Es a través de la segmentación de una imagen que se logra su discretización en dos niveles de intensidad de píxel, 1 o 0, en función de un umbral del nivel de gris n [46].

Los algoritmos de segmentación se basan en dos propiedades básicas de los valores del nivel de gris: discontinuidad o similitud entre los niveles de gris de píxeles vecinos. Este es

un paso imprescindible en el entendimiento de una imagen. Hay dos formas de generar la segmentación de una imagen: [47]

- Procesamiento del borde, donde se identifican las discontinuidades en las imágenes. [47]
- Procesamiento de una región, donde se buscan las regiones homogéneas de la imagen. [47]

La detección de bordes fue desarrollada para realizarse sobre imágenes con escala de grises. Los bordes se localizan donde ocurre un cambio abrupto de brillo y por eso normalmente se considera como una derivada en 2D (es decir, tasa de cambio) (ver Ilustración 24) [47].



Ilustración 24. Una imagen en escala de grises de una iglesia (izquierda), la magnitud del gradiente de borde de Sobel (centro izquierda), una versión umbralizada de la magnitud del gradiente (centro derecha) y las orientaciones de Sobel correspondientes (derecha) [47]

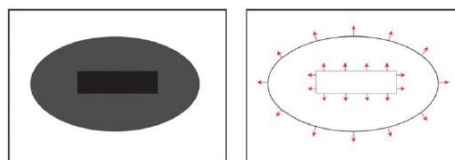


Ilustración 25. La imagen de la izquierda es el ejemplo donde se aplica el estudio del borde y la de la derecha es un ejemplo de simulación de gradiente en los bordes y su orientación [47].

La magnitud del gradiente es proporcional al tamaño de los vectores mostrados en la Ilustración 25. La transición de gris a blanco tiene mayor magnitud que de gris a negro [42].

En este punto, merece la pena señalar que las derivadas solo se pueden utilizar en funciones continuas y que las imágenes que se procesan están en un dominio discreto. Para aproximar las derivadas (tanto la primera como la segunda derivada), se utilizan las diferencias entre píxeles en la imagen (Ilustración 26) [47].



Ilustración 26. Una imagen simple en escala de grises (izquierda), junto con la magnitud de intensidad (centro izquierda), la función de primera derivada (centro derecha) y la función de segunda derivada (derecha) [47].

La mayoría de las técnicas para detectar bordes emplean operadores locales basados en distintas aproximaciones discretas de la primera y segunda derivada de los niveles de grises de la imagen, si bien existen otras posibilidades para ello, como el empleo de patrones de bordes ideales. A continuación, se describirán algunas de las mismas.

A. PRIMERA DERIVADA PARA DETECTAR BORDES

Los detectores de bordes de primera derivada calculan dos derivadas parciales y luego combinan estas para determinar los valores de gradiente y orientación para cada punto de borde. Los gradientes resultantes deben ser suprimidos por no-máximo y aplicar umbral. Es importante tener en cuenta que hay una gran cantidad de detectores de bordes de primera derivada que se pueden utilizar para calcular las derivadas parciales, pero los resultados de estos no varían significativamente, por lo que, como ejemplos, solo se tienen en consideración Roberts y los detectores de bordes de brújula, Sobel y Precito [42].

B. DETECTOR DE BORDES ROBERTS

El 'Roberts Edge Detector' es un algoritmo utilizado en visión artificial para la detección de bordes en imágenes. Este método calcula la diferencia diagonal entre píxeles adyacentes para resaltar áreas con cambios bruscos de intensidad, típicamente bordes y contornos de objetos [47].

Utiliza dos operadores de derivada parcial, δ_1 y δ_2 , aplicados de la siguiente manera: [42]

$$\delta_1(i, j) = f(i, j) - f(i + 1, j + 1) \quad \text{Ec.29}$$

$$\delta_2(i, j) = f(i + 1, j) - f(i, j + 1) \quad \text{Ec.30}$$

Donde $f(i, j)$ representa la intensidad del píxel en la posición (i, j) . Estos operadores se representan matemáticamente y se convolucionan a través de la imagen. El gradiente total se calcula como la raíz cuadrada de la suma de los cuadrados (RMS) de las dos derivadas parciales o, a veces, como la suma de las diferencias absolutas. La orientación del borde se determina mediante la función arco tangente inversa [47].

La ventaja del detector de Roberts es su simplicidad y rapidez, adecuada para aplicaciones en tiempo real, pero es sensible al ruido y produce bordes más finos, y asume que los bordes ocurren entre dos píxeles adyacentes, lo cual no es ideal para imágenes con bordes más suaves o gruesos (Ilustración 27) [42].

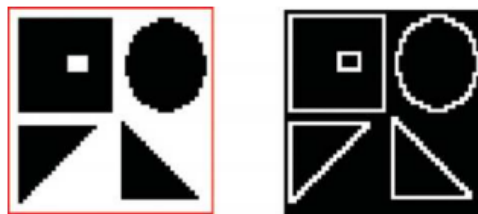


Ilustración 27. Detector de bordes sin aplicar (imagen de la izquierda) detector aplicado (imagen de la derecha) [42].

Los detectores de bordes de segunda derivada se centran en la tasa de cambio de la intensidad de la imagen. El Laplaciano es uno de los filtros de segunda derivada más comunes (Ver Ilustración 28), con dos aproximaciones discretas comunes que son: [47]

$$h^1(i, j) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ y } h^2(i, j) = \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad \text{Ec. 31}$$

El píxel central en estas matrices tiene un peso significativo, lo que puede introducir problemas en presencia de ruido, por lo que a menudo se realiza un suavizado previo al filtrado Laplaciano [47].

Mientras que estos detectores pueden determinar la magnitud y ubicación del gradiente, no sirven para determinar la orientación del gradiente. Para la magnitud, se debe determinar la pendiente de la función de segunda derivada en los puntos de cruce por cero. A menudo se utiliza la magnitud del gradiente de la función de primera derivada, debido a la complejidad de trabajar directamente con la segunda derivada [47].

Los operadores de segunda derivada son excelentes para determinar la ubicación precisa de los bordes. Ejemplos de implementaciones prácticas incluyen el Laplaciano de Gaussian y el detector de bordes de Canny, que son robustos frente al ruido y ofrecen una localización precisa de bordes [47].

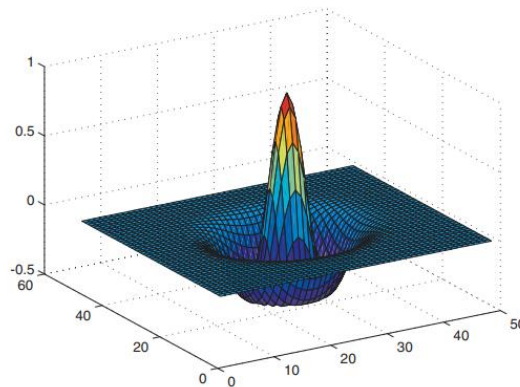


Ilustración 28. Representación de densidad de cambio que se utiliza para aplicar el filtro de segunda derivada (La técnica utilizada para hallar esta representación es la del filtro Laplaciano Gaussiano) [47].

C. DETECTOR DE BORDES CANNY

El filtro de Canny es un destacado algoritmo de detección de bordes utilizado en visión artificial, que utiliza, tanto filtros de primera derivada, como de segunda. Se caracteriza por: la detección precisa de los bordes, la correcta localización de estos y la garantía de una única respuesta por borde real detectado [47].

El proceso del filtro de Canny implica varios pasos, como se aprecia en la Ilustración 29. Primero, la imagen se suaviza mediante la convolución con un filtro Gaussiano para reducir el ruido. Luego, se estima la orientación del borde utilizando la primera derivada de la imagen suavizada. Los bordes se detectan y se afinan suprimiendo valores no máximos y buscando los cruces por cero en la derivada secundaria con respecto a la orientación del borde. El gradiente del borde se calcula con base en la primera derivada de la imagen suavizada [47].

Una característica significativa del filtro de Canny es el uso de la umbralización con histéresis para diferenciar los verdaderos bordes de las variaciones de intensidad debido al ruido. Finalmente, el detector puede procesar la imagen a múltiples escalas y combinar los bordes detectados para una representación más completa y precisa [42]. Este filtro es

especialmente útil en la detección de bordes en condiciones donde la presencia de ruido puede interferir con la interpretación de las características de la imagen, lo que lo convierte en una herramienta valiosa para la implementación en proyectos de visión artificial donde la precisión es crucial [47].



Ilustración 29. Detección de bordes de Canny. Las tres imágenes de Canny utilizan el mismo filtro de suavizado, pero diferentes umbrales de gradiente bajo y alto (7 y 236 – superior derecho, 7 y 100 – inferior izquierdo, 1 y 1 – inferior derecho). La imagen inferior derecha muestra todos los bordes posibles y las otras dos muestran algunos de los efectos de los umbrales [42].

El uso de Python y la biblioteca OpenCV ha facilitado la implementación de un sistema de cambio de perspectiva flexible y eficiente, que puede ser adaptado a diferentes condiciones experimentales y configuraciones de cámara. Además, gracias a la naturaleza de código abierto de estas herramientas, el trabajo desarrollado en este TFG podrá ser compartido y utilizado por otros investigadores y profesionales.

4. METODOLOGÍA

En base a los objetivos y a los fundamentos descritos anteriormente, se propone un procedimiento para la corrección de imágenes de manera que puedan modificarse para que simulen que se han capturado perpendicularmente. Esto permitirá a los algoritmos de correlación digital de Imágenes 2D funcionar de manera óptima, aumentando la robustez de la experimentación.

Este procedimiento se basará en la calibración del sistema óptico, por el cual se podrá determinar en qué orientación se debe disponer la cámara con respecto al plano de la probeta ensayada. A partir de esta información, en las siguientes imágenes se podrán corregir los errores de orientación sobre cada una de las imágenes, permitiendo que el sistema de DIC obtenga unos resultados adecuados, sin aberraciones por falta de perpendicularidad.

Los pasos que seguir se desarrollan mediante los siguientes apartados:

- **PRIMER APARTADO:** Metodología seguida para el desarrollo de los algoritmos de procesamiento de imagen para la calibración y corrección de imágenes basados en los fundamentos previamente investigados.
- **SEGUNDO APARTADO:** explicará la metodología experimental seguida para la evaluación y validación del trabajo desarrollado.

4.1 ALGORITMOS DE PROCESAMIENTO DE IMAGEN

Los algoritmos de procesamiento de imagen modifican las características geométricas de las mismas, siendo utilizados en la reconstrucción de imágenes deformadas, el giro y ajuste de estas, o la deformación intencionada de ciertos rasgos para análisis posteriores. Los más utilizados son los de escalado, translación, giro y espejo.

Todos los algoritmos correspondientes a transformaciones geométricas se basan en realizar una nueva distribución de los píxeles según el objetivo perseguido.

En esta parte se van a introducir los algoritmos desarrollados a lo largo de la realización del trabajo, son tres y se también nombran las metodologías en Python utilizadas:

- Método que determina el ángulo girado por la probeta. Gracias al uso de las técnicas de visión artificial, este algoritmo nos permitirá determinar ángulos espaciales con respecto a la cámara.
- Método de detección de perspectiva. Este algoritmo nos permite, mediante técnicas de procesamiento de imagen, determinar la perspectiva de la cámara sobre el objeto.
- Método de aplicación de cambio de perspectiva. Este algoritmo permite, a partir de la obtención de los parámetros de cambio del algoritmo anterior, cambiar la perspectiva de una imagen elegida. Es de gran interés para el estudio, ya que, el objetivo último de este trabajo es desarrollar una herramienta que nos permita estudiar una imagen que esté completamente desalineada y buscando la máxima perpendicularidad.

Cabe aclarar que, a lo largo del desarrollo de los algoritmos, se han utilizado diferentes bibliotecas, todas ellas en sus últimas versiones y que se enlistan en las siguientes:

- **Numpy (Versión 1.26.3):** Se trata de una biblioteca fundamental para la computación científica en Python, conocida por su capacidad para manejar arrays y matrices multidimensionales. Ofrece un amplio rango de herramientas matemáticas y se integra fácilmente con lenguajes como C y Fortran, siendo esencial en campos como la ciencia de datos y la ingeniería. (Un array es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneo, es decir, todos ellos del mismo tipo y relacionados).
- **cv2 (OpenCV-Python, Versión 4.9.0.80):** Esta biblioteca es un envoltorio de Python para las capacidades de OpenCV en el procesamiento de imágenes y visión por computadora. Proporciona herramientas para el análisis y manipulación de imágenes y videos, incluyendo detección y reconocimiento de objetos.
- **Math:** Parte de la librería estándar de Python, ofrece funciones matemáticas básicas y avanzadas. Su versión varía según la versión de Python utilizada, y es ampliamente usada en aplicaciones que requieren cálculos matemáticos.

Las características y desarrollo del código creado con Python, que ha permitido el desarrollo de los algoritmos, se encuentran en su totalidad en el apartado de anexos de este TFG.

4.1.1 MÉTODO PARA DETERMINAR EL ÁNGULO DE GIRO

Este primer algoritmo permite hacer una primera aproximación al campo de estudio seleccionado, ya que, a lo largo de todo el proyecto, se estudian, de una u otra manera, los ángulos de giro, resulta conveniente tener una herramienta que permita determinarlos.

Esta clase de algoritmos necesitan una serie de entradas específicas, la experiencia demuestra que los vértices de los rectángulos son fáciles de identificar por los algoritmos de visión artificial. Por todo lo cual, se necesita un input con una imagen que contenga una cuadrícula para la realización de ese proceso. Esto se debe al alto número de vértices que ofrece dicha figura, lo que va a permitir una correcta aproximación del ángulo medido. El proceso sería el siguiente, tal y como se ve en la [Ilustración 30](#):

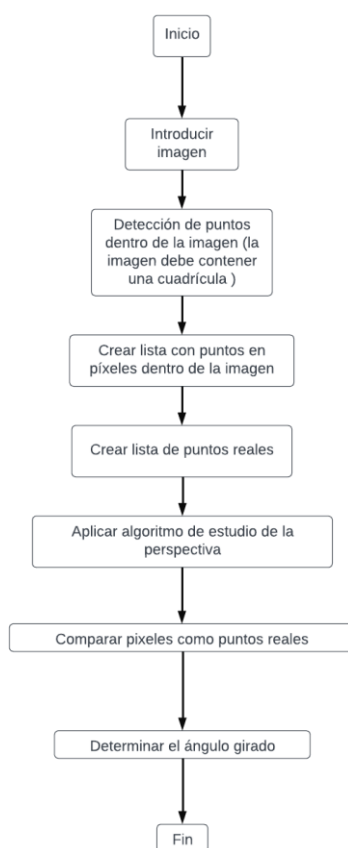


Ilustración 30. Diagrama de flujo que explica el proceso que determina el ángulo de giro [Elaboración propia].

Parte 1. Configuración inicial:

- Importa las bibliotecas **numpy** y **cv2** (OpenCV).
- Define el tamaño del patrón de calibración, la dimensión real de los cuadrados en la cuadrícula de ajedrez y la ruta de la imagen a calibrar.

Parte 2. Generación de Coordenadas del Patrón:

- Se introduce el tamaño del patrón, es decir el número de esquinas total de alto y ancho que tiene la cuadrícula de ajedrez.

Parte 3. Inicialización de Listas:

- A partir de los arrays implementados en la propia versión de Python, se definen las listas de coordenadas tridimensionales y coordenadas en píxeles de los puntos en el patrón de calibración (cuadrícula de ajedrez). La [Ilustración 31](#) muestra tanto el código de generación como el resultado que se muestra en pantalla. El primer pasó será generar una matriz que se encuentre vacía donde se introducirán los puntos seleccionados ([Ilustración 32](#)).

```
# Definir el tamaño del patrón de calibración (numero de puntos)
patron_x = 8
patron_y = 13

# Definir la dimensión real de los cuadrados en la cuadrícula de ajedrez
square_size = 0.5 # en cm
puntos_reales=[]
puntos_img=[]

# Crear una lista de coordenadas de los puntos del patrón de calibración
patron_coord = np.zeros((patron_x*patron_y, 3), np.float32)
patron_coord[:, :2] = np.mgrid[0:patron_x, 0:patron_y].T.reshape(-1, 2) * square_size

gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Encontrar los puntos del patrón de calibración en la imagen
ret, corners = cv2.findChessboardCorners(gris, (patron_x, patron_y), flags=cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_NORMALIZE_IMAGE)

# Si se encontraron los puntos, agregarlos a las listas
if ret == True:
    puntos_reales.append(patron_coord)
    puntos_img.append(corners)

height, width = img.shape[:2]
print(puntos_reales)

# Calibrar la cámara a partir de los puntos encontrados
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(puntos_reales, puntos_img, gris.shape[:1], None, None)
```

Ilustración 31. Imagen del script del algoritmo donde se muestra la definición de la lista de coordenadas tridimensionales, de los píxeles de la imagen y como se rellenan ambas listas [Elaboración propia].

puntos_reales [array([[0. , 0. , 0.],	puntos_img: [array([[398.4018 , 204.61147]],
[0.5, 0. , 0.],	
[1. , 0. , 0.],	[[445.42795, 203.48376]],
[1.5, 0. , 0.],	
[2. , 0. , 0.],	[[493.13443, 201.9147]],
[2.5, 0. , 0.],	
[3. , 0. , 0.],	[[541.55676, 200.72716]],
[3.5, 0. , 0.],	
[0. , 0.5, 0.],	[[590.5044 , 199.28545]],
[0.5, 0.5, 0.],	
[1. , 0.5, 0.],	[[640.0214 , 198.1902]],
[1.5, 0.5, 0.],	
[2. , 0.5, 0.],	[[690.4974 , 196.31487]],
[2.5, 0.5, 0.],	
[3. , 0.5, 0.],	[[741.2325 , 195.63025]],
[3.5, 0.5, 0.],	
[0. , 1. , 0.],	[[398.403 , 257.0474]],
imagen a)	imagen b)

Ilustración 32. En la imagen a) se muestra la lista de puntos reales y en la b) la lista de los píxeles de la imagen donde se han detectado los puntos [Elaboración propia].

Parte 4. Procesamiento de la Imagen:

- Lee y carga la imagen especificada.
- La convierte a escala de grises.

Parte 5. Detección de Puntos del Patrón:

- Utiliza el algoritmo de OpenCV para encontrar los puntos del patrón de calibración en la imagen.

Parte 6. Almacenamiento de Puntos Detectados:

- Si se encuentran puntos, se añaden a las listas **puntos_reales** y **puntos_img**.

Parte 7. Calibración de la Cámara:

- Utiliza los puntos detectados y las coordenadas reales para calibrar la cámara. Devuelve la matriz de la cámara (**mtx**), los coeficientes de distorsión (**dist**), y las matrices de rotación y traslación (**rvecs** y **tvecs**).

Parte 8. Impresión de Resultados:

- Imprime la matriz de la cámara y los coeficientes de distorsión.

Parte 9. Cálculo de Ángulos de Rotación:

- Calcula el ángulo de rotación del eje Y a partir de las matrices de rotación y traslación.

Parte 10. Impresión de Ángulos:

- Imprime el ángulo de rotación pasado a grados.

4.1.2 MÉTODO PARA LA DETERMINACIÓN DE LA PERSPECTIVA

La figura geométrica en la que se basa este algoritmo es la de un cuadrado, al igual que sucedía con el anterior, sin embargo, en este caso no es necesaria la cuadrícula. Se debe a que solo nos interesa un área de píxeles de la imagen. Por lo que la función de la geometría utilizada debe ser más simple para y así facilitar al algoritmo la detección de sus cuatro vértices.

Al igual que pasaba con el algoritmo anterior, a continuación, se presenta el funcionamiento mediante un diagrama de bloques ([Ilustración 33](#)):

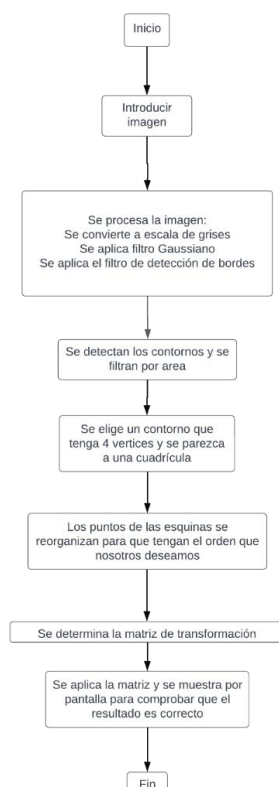


Ilustración 33. Diagrama de flujo que explica el proceso que determina el ángulo de giro [Elaboración propia].

Parte 1. Importación de bibliotecas

- Se importan tres bibliotecas: **cv2** para el procesamiento de imágenes, **numpy** para operaciones numéricas y **math** para funciones matemáticas.

Parte 2. Cargar y mostrar la imagen

- El programa carga una imagen desde una ruta específica y la muestra en una ventana llamada "Bordes". La ventana puede ser redimensionada.

Parte 3. Procesamiento de la imagen

- La imagen se convierte a escala de grises.
- Se aplica un filtro Gaussiano para reducir el ruido.
- Luego se utiliza el operador Canny para detectar los bordes en la imagen.

Parte 4. Encontrar contornos

- Se encuentran los contornos en la imagen de bordes.

Parte 5. Filtrar contornos por área

- Se filtran los contornos basados en su área, manteniendo solo aquellos cuya área sea mayor que un umbral específico.

Parte 6. Encontrar el contorno de la cuadrícula

- El programa busca el contorno que representa la cuadrícula. Esto se hace verificando si tiene al menos 4 vértices y si puede ser aproximado a un cuadrilátero.

Parte 7. Ordenar los puntos de interés

- Los puntos del contorno de la cuadrícula se reorganizan para asegurarse de que estén en el orden correcto (por ejemplo, esquina superior izquierda, esquina superior derecha, etc.).

Parte 8. Crear una máscara y marcar puntos

- Se crea una máscara negra del mismo tamaño que la imagen original. Luego, se marcan los puntos de interés con círculos rojos.

Parte 9. Combinar la imagen original y la máscara

- La imagen original se combina con la máscara para resaltar los puntos de interés.

Parte 10. Calcular la matriz de transformación

- Se calcula una matriz de transformación perspectiva para corregir la perspectiva de la imagen.

Parte 11. Aplicar la transformación

- La transformación perspectiva se aplica a la imagen.

Parte 12. Guardar y mostrar la imagen final

- La imagen final se guarda en una ruta específica y se muestra en una ventana llamada "Bordes".

4.1.3 MÉTODO DE APLICACIÓN DE PERSPECTIVA

Este algoritmo retiene la solución obtenida en el apartado anterior en forma de matriz de transformación de perspectiva y la aplica a cualquier imagen elegida.

Igual que las anteriores, se procede a su representación mediante un diagrama de bloques ([Ilustración 34](#)):

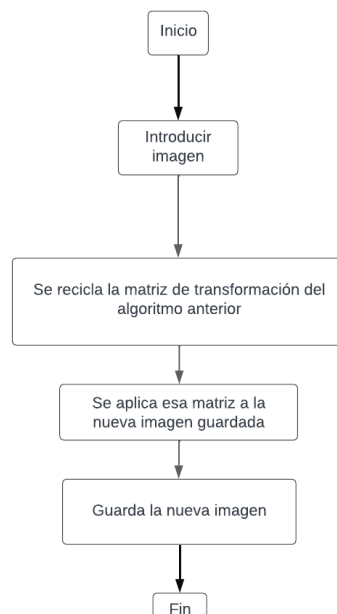


*Ilustración 34. Diagrama de flujo que explica el proceso donde se aplica una matriz de transformación ya guardada.
[Elaboración propia].*

Este algoritmo es el que permitirá aplicar la corrección angular en función de la información obtenida anteriormente:

Parte 1. Cargar imagen

- El programa carga una imagen desde una ubicación específica en el ordenador.

Parte 2. Definir la matriz de transformación

- Define una matriz de transformación llamada "M" que será aplicada a la imagen.

Parte 3. Aplicar la transformación

- Aplica la transformación perspectiva definida por la matriz "M" a la imagen.

Parte 4. Guardar la imagen

- Guarda la imagen transformada en un archivo en una ubicación específica

4.2 METODOLOGÍA EXPERIMENTAL

En las páginas que siguen se relatará toda la parte experimental del proyecto, que culminará con los distintos ensayos realizados en pro de determinar el buen funcionamiento de los algoritmos previamente explicados. Hay cuatro ensayos o metodologías de ensayos distintas. Las cuales son:

- Ensayo de alineación: en el que se estudiará la capacidad de los programas para estudiar la ortogonalidad de las probetas con respecto de la cámara.
- Ensayo de estudio de giro: en el que como su propio nombre indica se estudiará el giro de las probetas con nuestros algoritmos.
- Ensayo de giro con correlación de imágenes: donde se verá la capacidad de DIC para poder estudiar el desplazamiento que sufre una probeta que rota sobre un eje.
- Ensayo de giro y desplazamiento con correlación de imágenes y cambio de perspectiva.

Se comenzará por la descripción del diseño y ensamblaje de las piezas necesarias para la construcción del montaje experimental utilizado en el desarrollo de los ensayos.

4.2.1 SOPORTE DE LA CÁMARA Y DE LOS PATRONES DE SPECKLES

A la hora de diseñar el montaje experimental, y tras analizar las necesidades específicas de los ensayos, se considera que debía de tener las siguientes características:

- La cámara y las probetas deben estar alineadas.
- Las probetas debían de tener la posibilidad de desplazarse, siempre respetando la premisa del punto anterior.
- Los desplazamientos debían de estar completamente controlados, pudiéndose hacer en todo momento mediciones manuales de estos.
- El montaje experimental debía poder hacer estudios de distintas probetas, por lo que se debía diseñar una manera de cambiarlas de manera sencilla.

Con todas estas condiciones en mente, se construyó un montaje experimental se ajustaba a todas las exigencias:

- Se dispuso una estructura rígida base que pudiera soportar tanto la cámara como las probetas y que permitiera alinear estas dos.
- La placa de soporte de las probetas se colocó sobre un goniómetro de manera que esta pudiera controlar el giro sobre el eje vertical y, a su vez, este se dispuso sobre una plataforma de desplazamiento lineal de manera que se pudiera controlar el movimiento a lo largo del eje vertical.
- Se han utilizado diferentes probetas a lo largo del proyecto: una cuadrícula, un rectángulo, y un speckle, siendo esta última la probeta más utilizada en el estudio de correlación de imágenes (DIC), ya que se compone de puntos aleatorios fáciles de seguir por un programa debido precisamente a esa aleatoriedad.

A continuación, se procede a relatar la utilización específica de las diferentes piezas y probetas utilizadas en el proyecto, acompañando dicha descripción de las imágenes capturadas duran su construcción y montaje, a fin de facilitar la comprensión de las acciones realizadas.



Ilustración 35. Esquema montaje experimental de pruebas [Elaboración propia].

El apoyo de la cámara se ha colocado en la zona o poste derecho visto desde la [Ilustración 35](#), se diferencia claramente cuál no es el perfil Bosch de la cámara y cuál sí, porque este último es más corto. El apoyo de la cámara lo forman tres piezas de Nailon y una metálica que se han mecanizado, precisamente para ser el soporte de estas cámaras.

El soporte de la cámara está dividido en tres partes, una pieza inferior que une el perfil Bosch de la estructura con el resto del apoyo; una pieza intermedia que una a las dos restantes y la pieza base principal de la cámara que une a esta última con el resto del soporte. Estas tres piezas se muestran en la [Ilustración 36](#).



Ilustración 36. Pieza superior (arriba en la imagen), pieza intermedia (en medio de la imagen), pieza base (abajo en la imagen) [Elaboración propia].

El apoyo de la probeta está colocado en el poste derecho visto desde la [Ilustración 35](#). Está formado por dos piezas de Nailon mecanizadas, una plataforma de desplazamiento

lineal, una plataforma de desplazamiento angular, una placa ferromagnética, una pieza metálica de acero mecanizada que sirve de unión entre la placa y la etapa angular. El apoyo de la etapa angular está formado por dos piezas de Nailon, también mecanizadas y unidas entre sí (Ilustración 37).

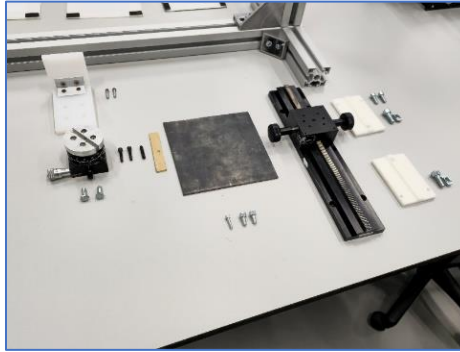


Ilustración 37. Imagen que muestra el apoyo completo la probeta [Elaboración propia].

La Ilustración 38 muestra las piezas que unen el goniómetro (o plataforma de desplazamiento angular) de la probeta, con la plataforma de desplazamiento lineal. Las piezas tienen una unión no permanente entre ellas, para asegurar la rigidez del conjunto y la precisión. La placa de acero se mantiene unida al conjunto por un tornillo prisionero y una tablilla de madera.



Ilustración 38. Imagen que muestra tanto la base de la placa como el goniómetro [Elaboración propia].

A. PLACAS DE LAS PROBETAS

Se han de configurar tres placas de metacrilato cortadas con láser que sirven de soporte para las cartulinas. Una es para la probeta plana con patrón de Specle y son necesarias otras dos placas para otras dos tarjetas, una de cuadrícula de ajedrez y otra de un cuadrado.

A las tres placas se les deben unir imanes mediante adhesivo. Dado que la placa de acero es ferromagnética, permite cambiar las distintas tarjetas con facilidad y así realizar los experimentos de forma dinámica.

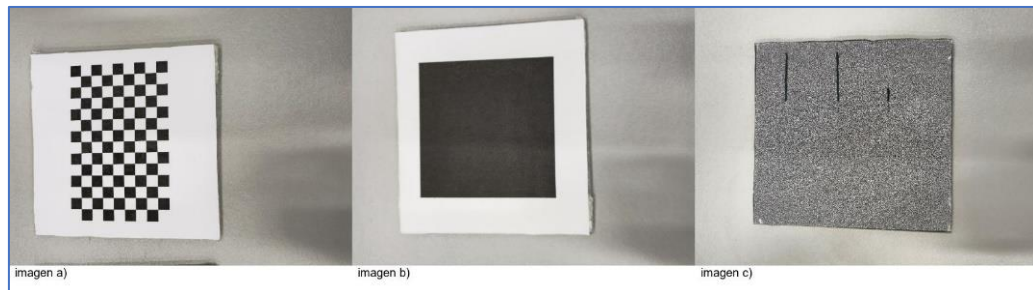


Ilustración 39. La imagen a) muestra la placa con cuadrícula de ajedrez, La imagen b) muestra la placa rectángula, la imagen c) muestra la placa con el Speckle [Elaboración propia].

En la [Ilustración 39](#) podemos ver todos los patrones que usamos para los experimentos:

- Imagen a): muestra la tarjeta cuadrícula que permite realizar el estudio del ángulo de la placa de acero.
- Imagen b): muestra la tarjeta rectángula que permite realizar el cambio de perspectiva.
- Imagen c): permite hacer el estudio de correlación de imágenes 2D.

4.2.2 ENSAYOS DE EVALUACIÓN DEL ALGORITMO DE CÁLCULO DE ORIENTACIÓN MEDIANTE CUADRICULA

Se procede a la explicación del método utilizado para la ejecución de los ensayos, así como de los pasos necesarios para la ejecución de las interfaces elaboradas para facilitar la realización de estos.

4.2.2.1 ENSAYO 1: CÁLCULO DEL ÁNGULO DE ALINEACIÓN

El primer ensayo busca comprobar la perpendicularidad, es decir, el ángulo de alineación, mediante una tarjeta de calibración y sin necesidad de emplear Correlación Digital de Imágenes.

Así pues, se hace un estudio del ángulo de giro de la probeta. Todos los experimentos se realizan colocando la probeta sobre un goniómetro, de tal manera que siempre se conozca

cuál es el ángulo de giro en el que se encuentra. Dado que el trabajo trata, en gran medida, del estudio de estos ángulos, se debe desarrollar un algoritmo que estudie el giro de forma digital. Tanto este primer ensayo, como el segundo, determinarán si el primer programa desarrollado trabaja correctamente.

Se van a ejecutar una serie de mediciones sobre el ángulo 0 del eje vertical de la probeta (la probeta se mantendrá ortogonal a la cámara y el programa debe dar por solución que, en efecto, el ángulo no se altera). El procedimiento debe ser el siguiente: se tomarán 5 imágenes de la probeta sobre la que se ha colocado una cuadrícula de rectángulos negros (14 filas por 9 columnas. [Ilustración 40](#)). No se realizará ningún cambio sobre el ángulo del goniómetro, comprobando que este sea 0 y, tras realizar esta comprobación, se realizarán las 5 fotografías.

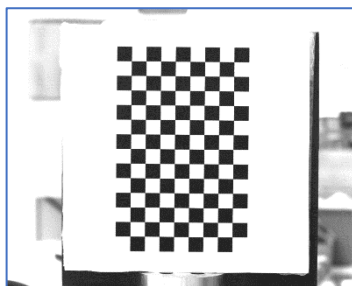


Ilustración 40. Fotografía del ángulo 0 de la cuadrícula [elaboración propia].

Tras la captación, se introducen en el programa la realización del estudio del ángulo, arrojando los valores calculados.

Dado que a lo largo del trabajo se han tomado muchas imágenes, es necesario desarrollar interfaces gráficas con el fin de facilitar y aligerar el estudio de algunos valores.

La primera interfaz gráfica desarrollada tendrá por misión el cálculo del ángulo ([Ilustración 41](#)), su funcionamiento es simple:

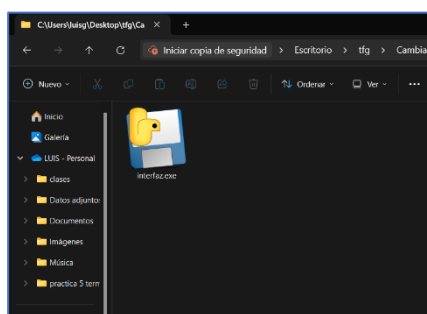


Ilustración 41. Captura que muestra el .exe de la interfaz [elaboración propia].

- Se abre la interfaz (Ilustración 42):

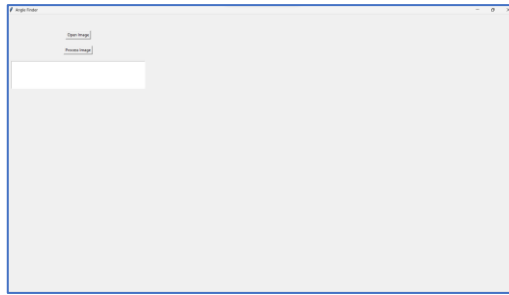


Ilustración 42. Ventana de inicio de la interfaz [elaboración propia].

- Al pulsar el botón “Open Image” se puede seleccionar la imagen cuyo ángulo se desea estudiar (es necesario que la imagen contenga una cuadrícula (Ilustración 43):

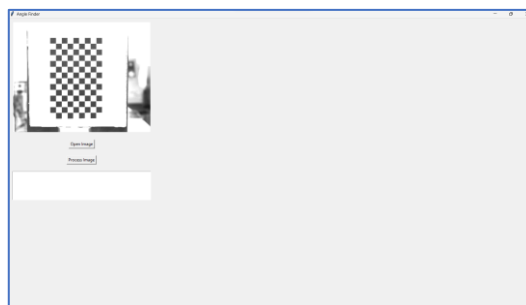


Ilustración 43. Captura que muestra la imagen ya cargada [elaboración propia].

- El botón “Process Image” posibilita que el programa estudie el ángulo de la cuadrícula, tal y como se observa en la [Ilustración 44](#):

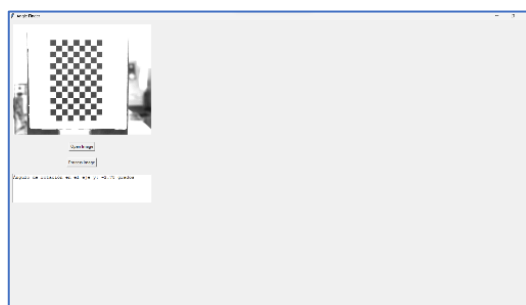


Ilustración 44. Ventana final con el ángulo de la cuadrícula ya obtenido [elaboración propia].

- En caso de desear obtener más ángulos, solo será necesario abrir una imagen nueva (Ilustración 45):

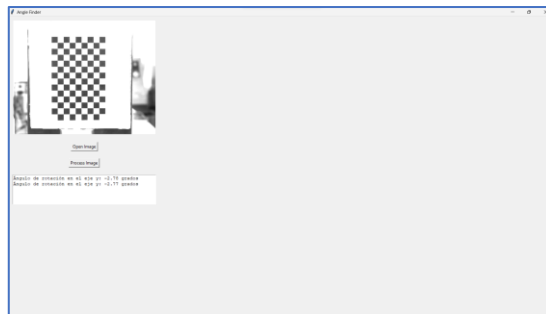


Ilustración 45. Captura con varios resultados para diferentes imágenes [elaboración propia].

La interfaz muestra siempre, más abajo, el último ángulo calculado.

4.2.2.2 ENSAYO 2: CÁLCULO DEL ÁNGULO GIRADO

El ensayo de análisis de giro será bastante similar al ensayo anterior. Sin embargo, se busca analizar si el programa es capaz de reconocer correctamente los ángulos. Para este caso, sí habrá variación en el ángulo girado, concretamente, se tomarán 5 ángulos de giro diferentes. De la misma forma que antes, se representarán sobre una gráfica los ángulos aportados por el programa.

El proceso de ensayo será muy similar. Sin embargo, también se utilizará un nuevo programa, esencial para el proyecto, y que genera un algoritmo específico para la realización del trabajo. Está basado, en la mayoría de sus funciones, en la utilización de la biblioteca OpenCV. Este segundo programa no da por solución un ángulo de giro, sino que, utiliza una técnica de cambio de perspectiva a partir de una imagen en la que aparece un rectángulo. Es decir, el programa detectará una figura rectangular dentro de una imagen y, mediante técnicas de procesamiento de imágenes, generará una nueva imagen resultado que se verá desde una perspectiva ortogonal.

- El estudio comienza utilizando el programa del ensayo anterior. Se selecciona la imagen y se obtiene el resultado de varios ángulos. Se deben captar tres imágenes en una posición de la cuadrícula en concreto, y comprobar cuánto error existe para cada ángulo medido.

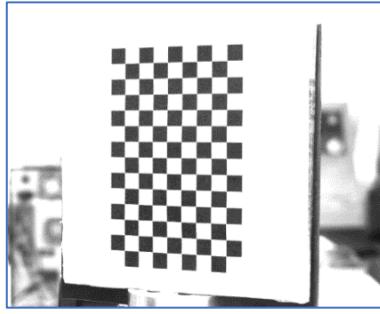


Ilustración 46. Captura con varios resultados para diferentes imágenes [elaboración propia].

- Para el segundo ensayo se utiliza la misma interfaz que en el primero, sin embargo, ahora, en vez de introducir imágenes con un solo grado, las imágenes tendrán diferentes ángulos (ver [Ilustración 47](#)).

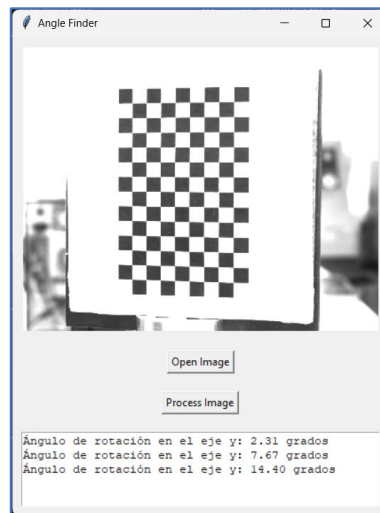


Ilustración 47. Captura de la interfaz 1 utilizada para el cálculo de varios ángulos (elaboración propia).

4.2.3 ENSAYOS PARA VALIDACIÓN DE LA CORRECCIÓN DE LA PERPENDICULARIDAD

En este caso, se llevará a cabo una metodología cuyo objetivo final sea realizar una corrección de la imagen capturada, de manera que la probeta se mostrará como si realmente estuviera perpendicular. Para ello, se han planteado dos ensayos. En el primero, se evalúa si el algoritmo de corrección de perpendicularidad introduce algún desplazamiento no deseado. En el segundo se valorará si el algoritmo propuesto permite una mejor medida de los desplazamientos verticales inducidos cuando la probeta está inclinada. En ambos casos, la corrección de la orientación recurrirá a una tarjeta de calibración de forma cuadrada y a la medida de desplazamientos con DIC.

4.2.3.1 ENSAYO 3: EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERPENDICULARIDAD CON DIC

Se trata de comprobar el cálculo de desplazamientos que haría DIC si la probeta plana con patrón de Speckle no apunta directamente a la cámara, sino que más bien, estará inclinada tantos grados como se deseen. Concretamente se estudiarán a 5, 10, 15 y 20 grados sobre una imagen de referencia totalmente alineada o grados 0. Tras este estudio previo, donde claramente DIC determinará que ha existido un desplazamiento, se hará el mismo estudio, pero ahora, se le aplicará a cada imagen el programa de cambio de perspectiva, incluida la imagen completamente alineada, y analizaremos cuáles son las diferencias de desplazamientos que encuentra DIC.

Primer Paso: Evaluación de aberración por perspectiva

Primero se aplica la correlación de imágenes sin cambio de perspectiva, abriendo Ncorr y procediendo como se indica en el ANEXO 1.6 relativo a dicho programa.

En primera instancia, se hará un estudio de desplazamiento solo para imágenes en las que no se ha corregido la perspectiva para poder cuantificar la corrección.

- Toma de imágenes de probetas no perpendiculares. Primero se toman las fotografías a la probeta plana con speckle en sus distintas posiciones angulares ([Ilustración 48](#)).

•

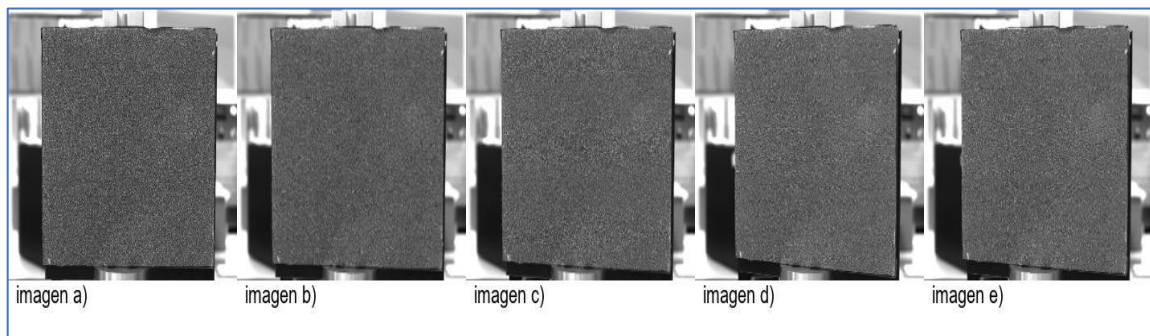


Ilustración 48. a) Ángulo a 0°, b) Ángulo 5°, c) Ángulo 10°, d) Ángulo 2°, e) Ángulo 30° [elaboración propia].

- Tras la toma de imágenes se procede a usar DIC para estudiar los desplazamientos (ver [Ilustración 49](#)).

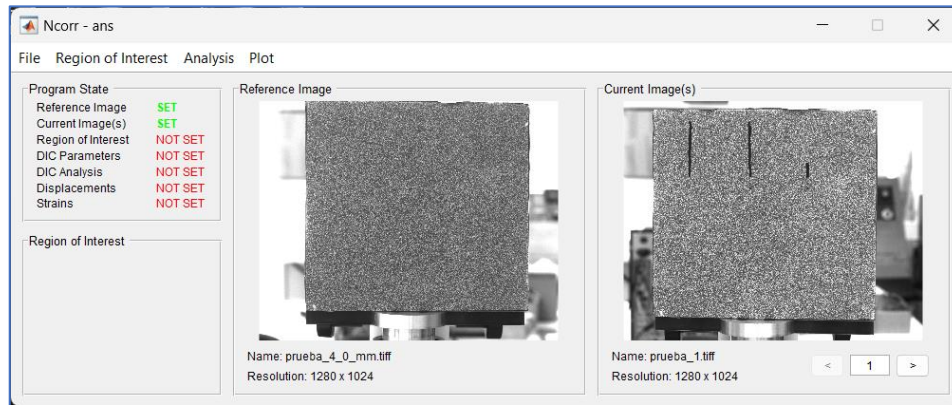


Ilustración 49. Estudio de desplazamiento con DIC sin cambio de perspectiva [elaboración propia].

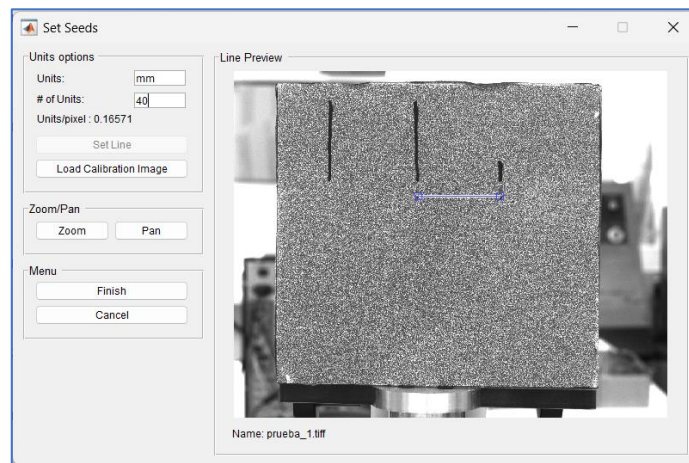


Ilustración 50. Imagen que muestra una captura de la ventana de Ncorr donde se permite aplicar la relación mm/píxeles [elaboración propia]

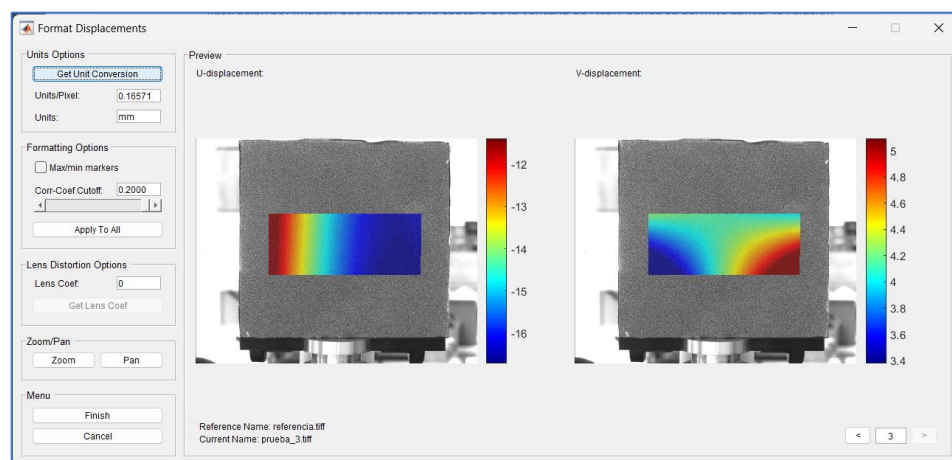


Ilustración 51. Imagen donde se ve el mapa de calor de los desplazamientos de la placa [elaboración propia].

Segundo paso: Corrección de imágenes

En él se realiza el estudio de deformaciones, sin embargo, ahora se aplicará el algoritmo de cambio de perspectiva, mediante la utilización de la segunda interfaz desarrollada.

a) Comenzamos utilizando la interfaz gráfica para cambio de perspectiva.

Al igual que con los ensayos anteriores, también se desarrolla una interfaz para el usuario, de tal manera que el proceso de toma de imágenes, y su procesamiento, sea más intuitivo.

Los pasos que se deben seguir son los siguientes:

- Al abrir interfaz, aparecerá la siguiente ventana, como indica la [Ilustración 52](#):

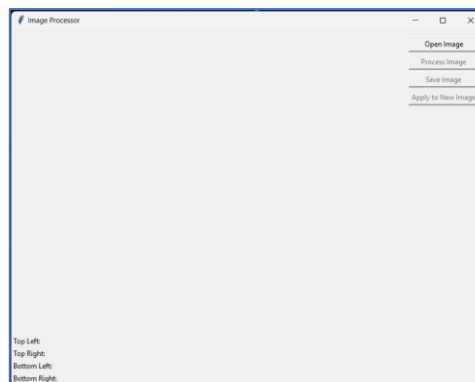


Ilustración 52. Ventana de la GUI de ensayo 3-4 sin cargar imagen [elaboración propia].

En dicha ventana aparecen una serie de comandos, arriba a la derecha, entre los cuales constan abrir imagen (Open Image), Process Image (Procesamiento de imagen), Save Image (Guardar imagen) y Apply to New Image (Aplicar a la nueva imagen). Abajo a la izquierda se encuentran las posiciones de los puntos del cuadrado, que se cargarán cuando el programa las encuentre y aparecerán en píxeles (x, y). Lo que ocurrirá una vez que el programa las procese.

- El próximo paso será cargar la imagen:

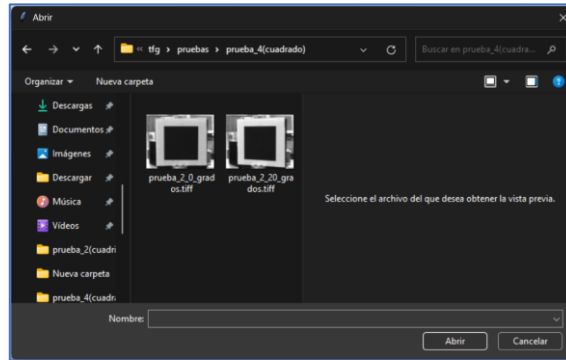


Ilustración 53. Ventana de explorador del sistema operativo que se abre para buscar la imagen tras pulsar el botón Open Image [elaboración propia].

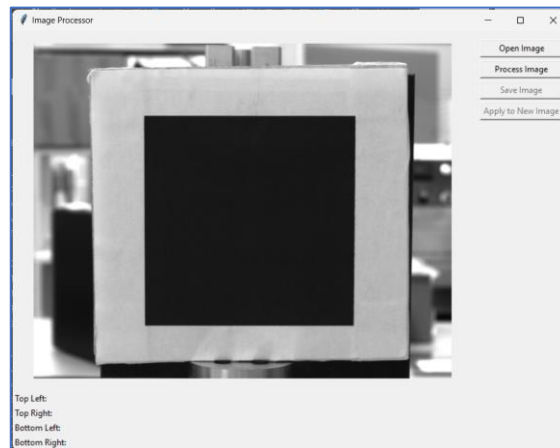


Ilustración 54. Ventana del GUI con la imagen cargada [elaboración propia].

- Como se puede comprobar, se ha habilitado la opción de Process Image tras haberla cargado. Se procede a pulsar el botón.

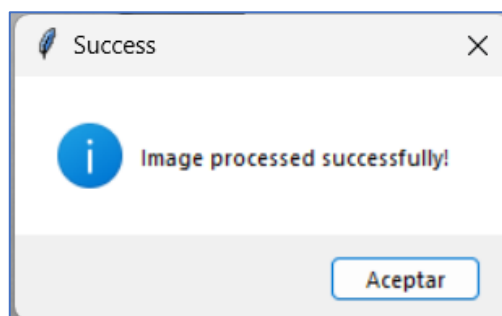


Ilustración 55. Mensaje que sale si la imagen ha sido procesada correctamente [elaboración propia].

Tras procesar la imagen, la ventana estará lista para proceder con el procesamiento de la imagen o imágenes que tengan el mismo patrón de grados. Es decir, se pueden utilizar las imágenes con la probeta que tengan el mismo grado de giro y la imagen con el cuadrado que se ha procesado. Esto permite estudiar el desplazamiento de la probeta, si lo hubiera, con correlación de imágenes ([Ilustración 56](#)).

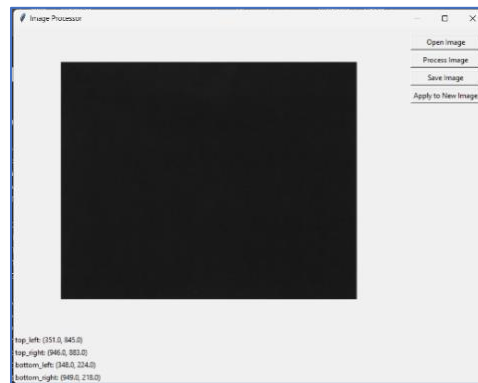


Ilustración 56. Ventana final tras procesamiento y con la posición de los puntos y el cambio de perspectiva ya cargado (elaboración propia).

- Ya se ha cargado el cambio de perspectiva del cuadrado de la imagen que abrimos previamente, que no presenta ángulo de giro. Esto supone que solamente se ha producido un cambio de perspectiva, un centrado, sin cambiar su ángulo. Es decir que, si se aplica a un patrón speckle, que tampoco tenga ángulo de giro, con la opción Apply to New Image de la interfaz, quedará de la siguiente manera, tal y como aparece en la [Ilustración 57](#)):

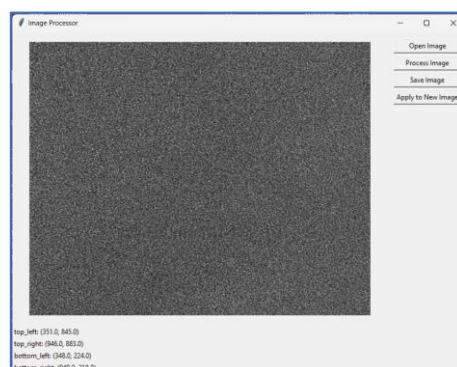


Ilustración 57. Ventana de la interfaz con el speckle procesado. [elaboración propia].

- En todo momento se podrán guardar las imágenes procesadas utilizando la opción de Save Image.

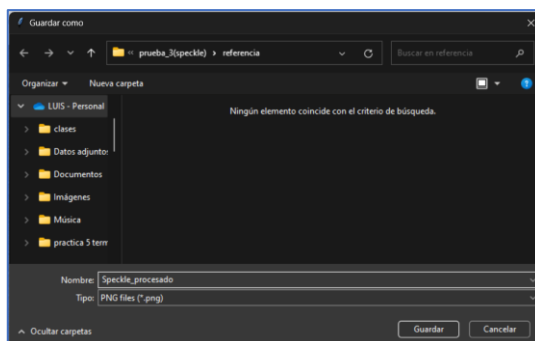


Ilustración 58. Ventana del explorador de archivos, para guardar imagen [elaboración propia].

b) A continuación realizamos la comparativa de imágenes pre-procesado y pos-procesado

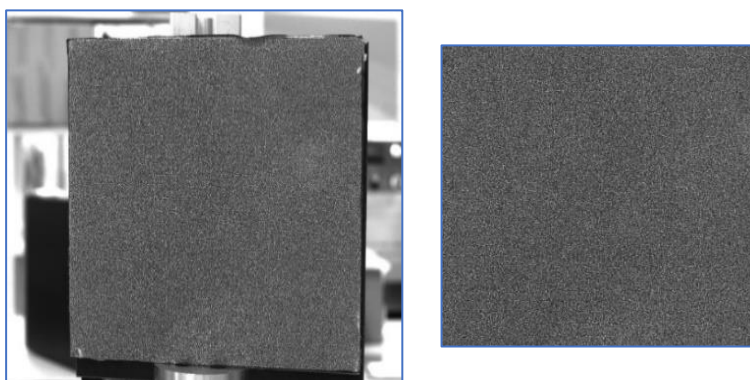


Ilustración 59. La imagen de la izquierda es la de la placa a 5° y la de la derecha es la misma imagen de la placa de 5° pos-procesado [elaboración propia].

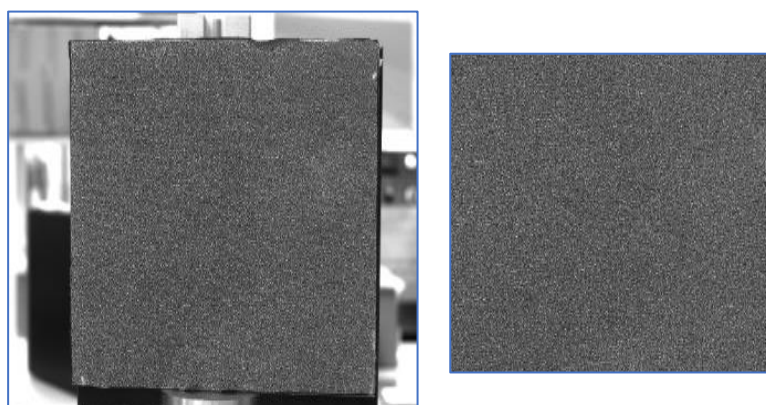


Ilustración 60. La imagen de la izquierda es la de la placa a 10° y la de la derecha es la misma imagen de la placa de 20° pos-procesado [elaboración propia].

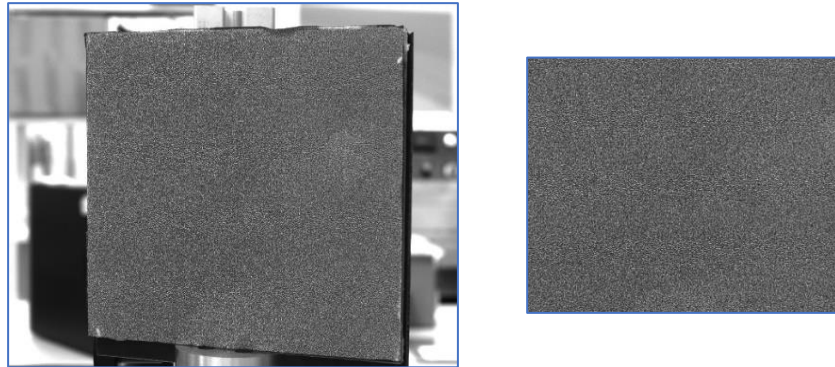


Ilustración 61. La imagen de la izquierda es la de la placa a 15° y la de la derecha es la misma imagen de la placa de 5° pos-procesado [elaboración propia].

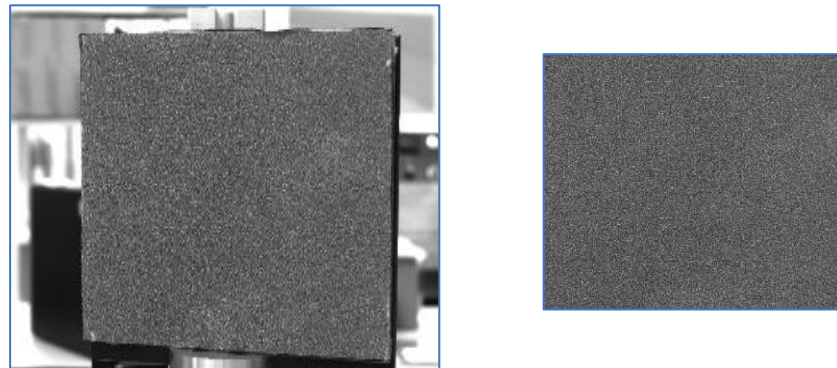


Ilustración 62. La imagen de la izquierda es la de la placa a 20° y la de la derecha es la misma imagen de la placa de 10° pos-procesado [elaboración propia].

Tras haber hecho el procesado a las imágenes, se procederá de la misma manera con el programa de Ncorr. Pero ahora con las nuevas imágenes obtenidas para poder comparar si ha habido un cambio en el estudio de deformaciones con la nueva perspectiva.

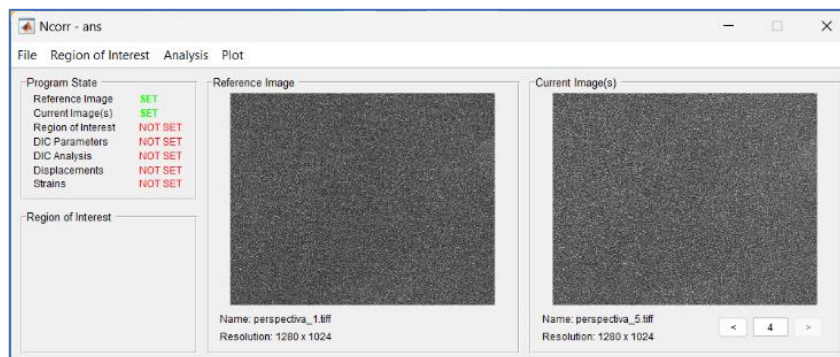


Ilustración 63. GUI de Ncorr con imágenes pos-procesado [elaboración propia].

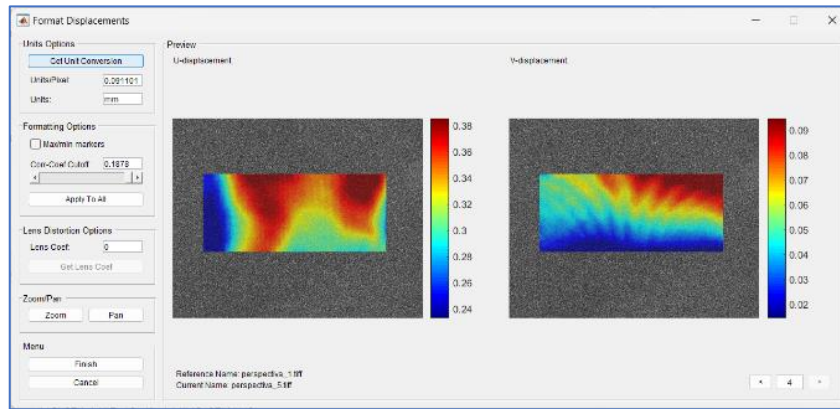


Ilustración 64. GUI de Ncorr de mapa de calor para el estudio de desplazamiento, con imágenes pos-procesado y pos-calibrado de la relación píxel/mm [elaboración propia].

4.2.3.2 ENSAYO 4. EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA ANTE DESPLAZAMIENTOS VERTICALES

En este caso, el ensayo consistirá en realizar una combinación de todos los anteriores, e implementar una comparativa entre tres estudios diferentes:

- A. Estudio de desplazamiento sin ángulo de giro y sin procesamiento de imagen.
- B. Estudio de desplazamiento con ángulo de giro y sin procesamiento de imagen.
- C. Estudio de desplazamiento con ángulo de giro y con procesamiento de imagen.

A. Desplazamientos sin ángulo de giro y sin procesamiento

Se toman imágenes de la probeta, como normalmente se haría para su estudio con correlación imágenes.

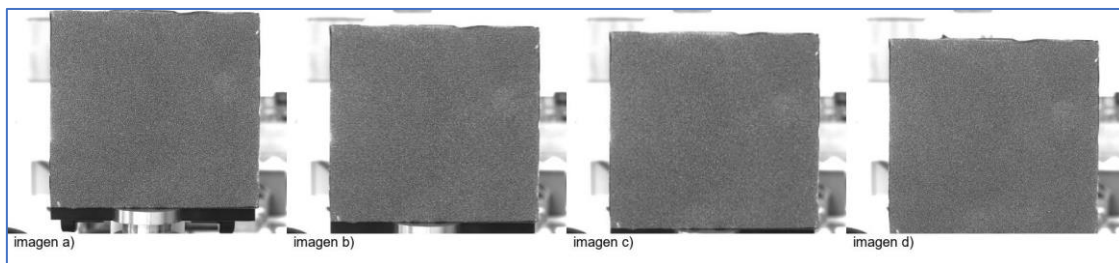


Ilustración 65. Imagen a) Imagen de referencia probeta caso sin giro., Imagen b) Desplazamiento de 5 mm la probeta caso sin giro., Imagen c) Desplazamiento de 10 mm de la probeta caso sin giro., Imagen d) Desplazamiento de 15 mm de la probeta caso sin giro [elaboración propia].

De la misma manera que en los ensayos anteriores, se genera un estudio de desplazamientos con DIC.

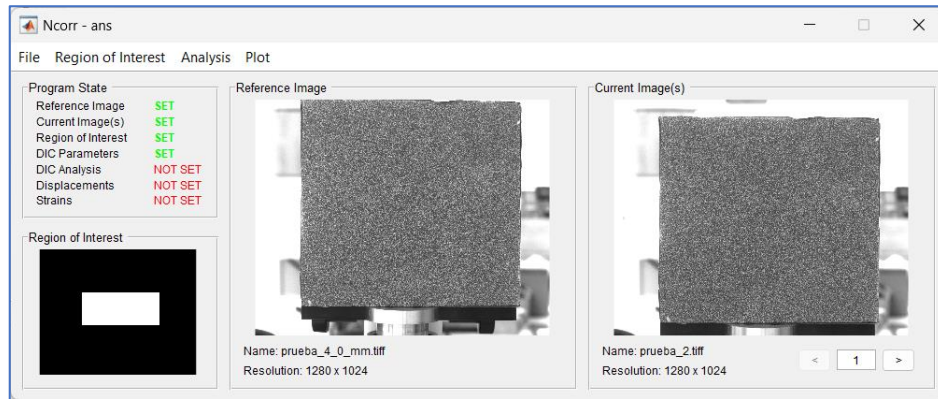


Ilustración 66. Imagen del GUI de Ncorr en donde se ve las imágenes utilizadas para el estudio de desplazamiento sin giro [elaboración propia].

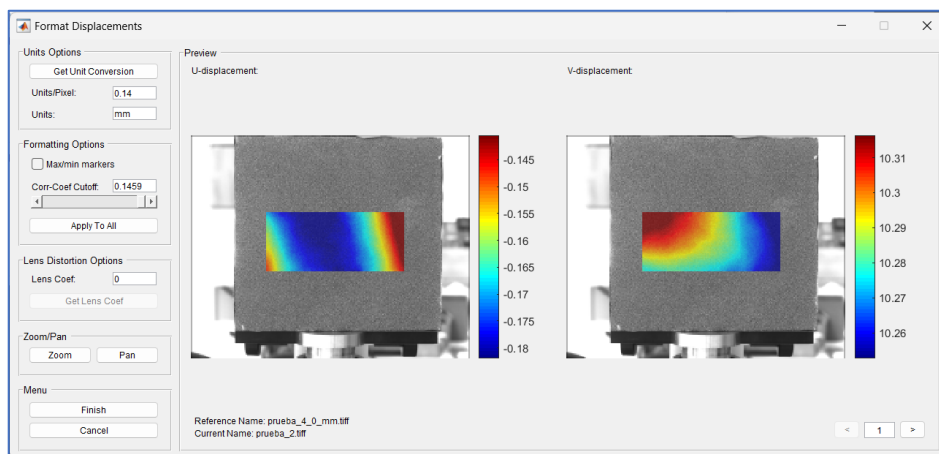


Ilustración 67. Imagen del GUI de Ncorr para desplazamientos en donde se ve el mapa de calor de desplazamientos probeta plana con patrón de Speckle [elaboración propia].

B. Desplazamientos con ángulo de giros y sin procesamiento

Es similar al primero. Se toman las imágenes de la probeta plana con patrón de speckle en distintas posiciones a lo largo del eje longitudinal, pero en este caso, con una variación angular que será de 10, 20 y 30 grados, medidos por el goniómetro.

- **10 GRADOS**

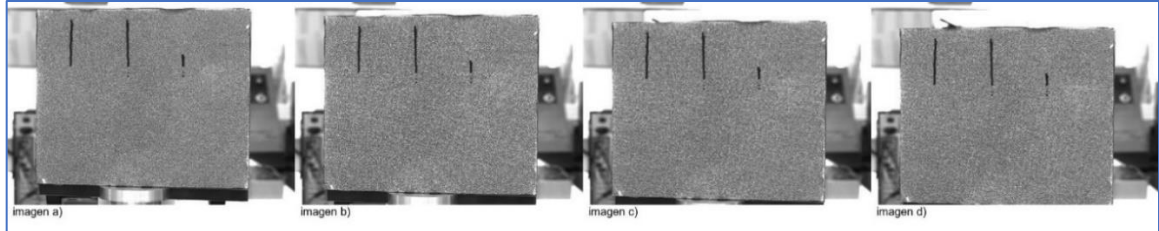


Ilustración 68. Imagen a) Referencia sin desplazamiento a 10 grados., Imagen b) Imagen de desplazamiento de 5 mm a 10 grados, Imagen c) Imagen de desplazamiento de 10 mm a 10 grados., Imagen d) Imagen de desplazamiento de 15 mm a 10 grados [elaboración propia].

- **20 GRADOS**

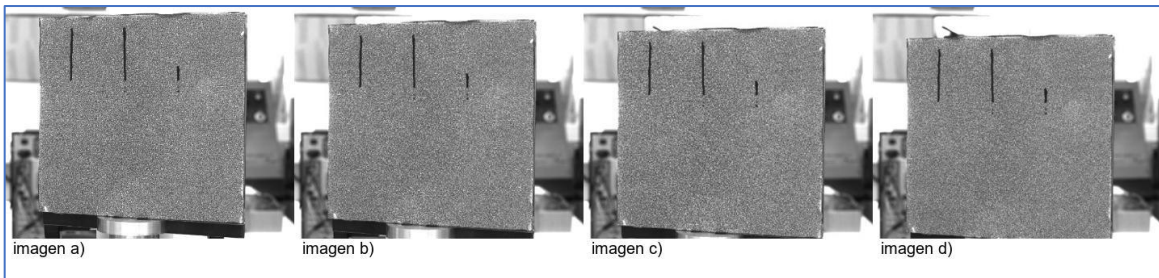


Ilustración 69. Imagen a) Referencia sin desplazamiento a 20 grados., Imagen b) Imagen de desplazamiento de 5 mm a 20 grados, Imagen c) Imagen de desplazamiento de 10 mm a 20 grados., Imagen d) Imagen de desplazamiento de 15 mm a 20 grados [elaboración propia].

- **30 GRADOS**

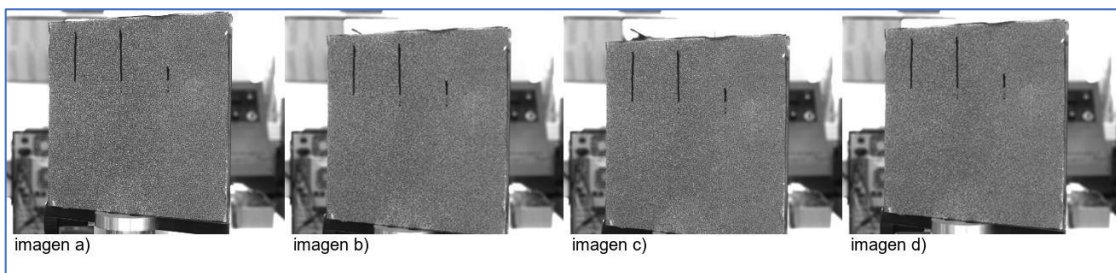


Ilustración 70. Imagen a) Referencia sin desplazamiento a 30 grados., Imagen b) Imagen de desplazamiento de 5 mm a 30 grados, Imagen c) Imagen de desplazamiento de 10 mm a 30 grados., Imagen d) Imagen de desplazamiento de 15 mm a 30 grados [elaboración propia].

De la misma manera que en los ensayos anteriores, debe generarse un estudio de desplazamientos con DIC.

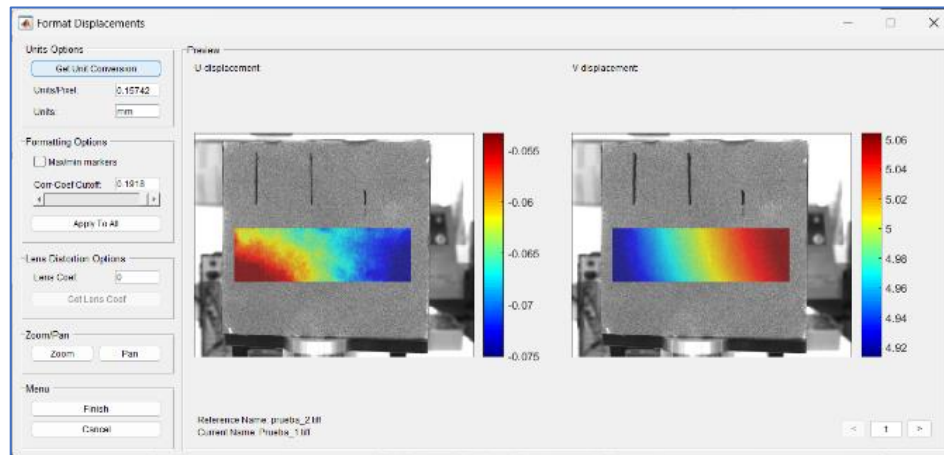


Ilustración 71. Imagen del GUI de DIC para desplazamientos en donde se ve el mapa de calor de las imágenes con desplazamientos y giro de la probeta plana con patrón de Speckle [elaboración propia].

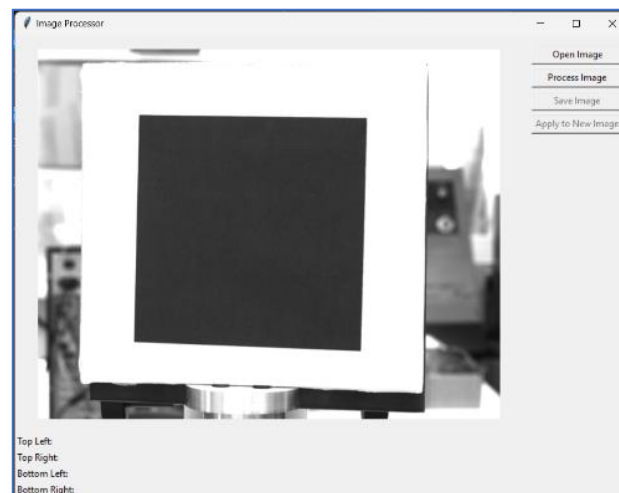


Ilustración 72. Imagen del GUI de la interfaz gráfica donde muestra la cuadrícula [elaboración propia].

C. Desplazamientos con ángulo de giro y con procesamiento

Ya solo queda aplicar el algoritmo de procesamiento de imagen una vez más, en este caso nuevamente con la interfaz del ensayo 3, como sigue:

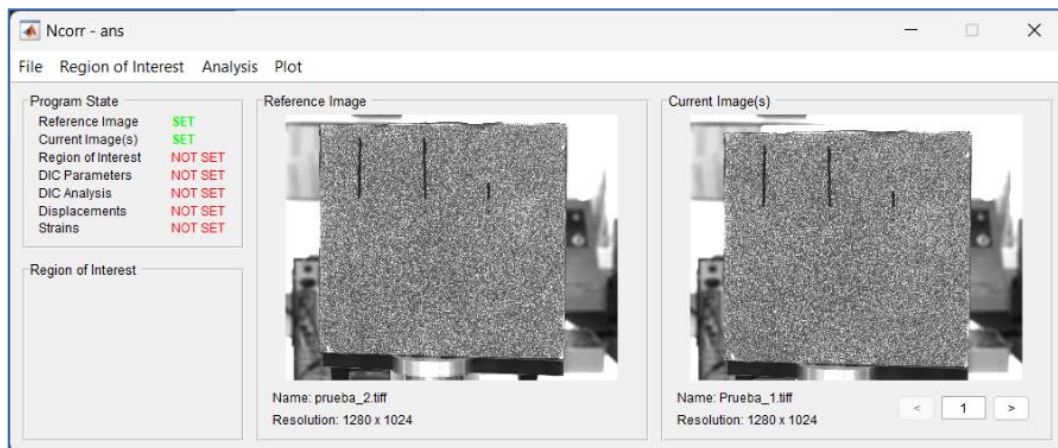


Ilustración 73. Imagen de la GUI ya con la imagen de la probeta plana con patrón de Speckle cargada con un ángulo de giro de 10° [elaboración propia].

Tras cargar la imagen, se procesa lo que dará la referencia de los puntos de las esquinas del cuadrado en píxeles. De tal manera que podamos comprobar que el procesamiento se ha completado correctamente. También cargará el cambio de perspectiva en la interfaz. Se debe cargar las imágenes de las probetas y procesarlas, pulsando el botón Apply to New Image. Esto aplica un cambio de perspectiva a todos los desplazamientos. Si, por el contrario, a cada desplazamiento le es introducido un cambio de perspectiva diferente, es decir, también procesáramos una imagen del cuadrado donde se desplace en la misma dirección y magnitud que el speckle, y posteriormente le aplicáramos el cambio de perspectiva, estaríamos perdiendo el movimiento del speckle, lo que haría que DIC no pudiera estudiar ningún desplazamiento.

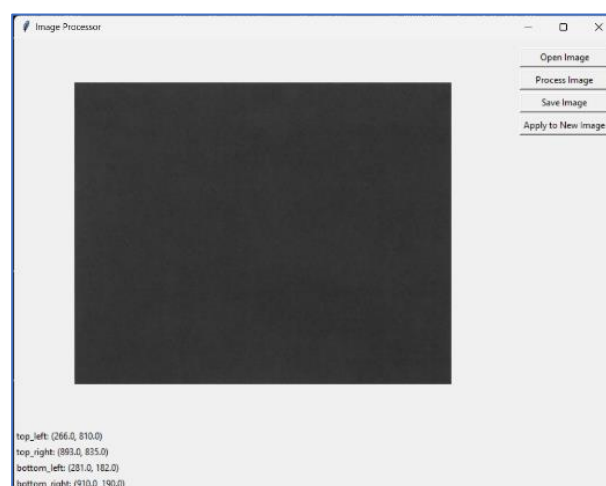


Ilustración 74. Imagen del GUI de la interfaz gráfica de los ensayos con cambio de perspectiva con la imagen cuadrado a 10° ya procesada [elaboración propia].

- **IMÁGENES CON PROBETAS PROCESADAS a 10 GRADOS**

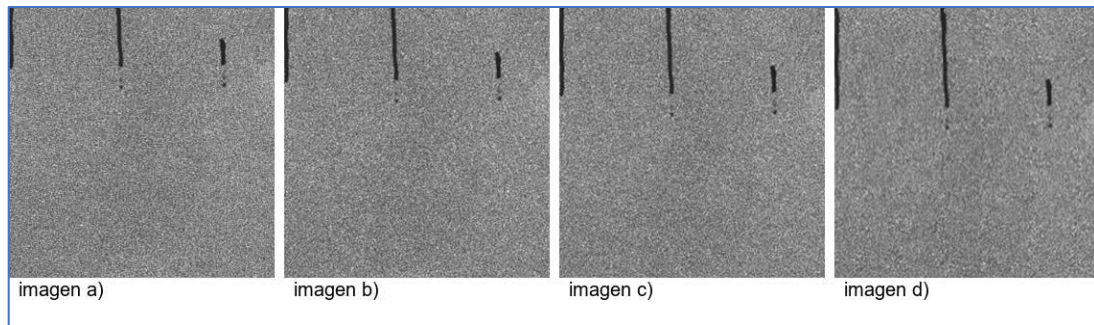


Ilustración 75. a) Imagen de referencia de Speckle a 10° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 10°, c) Imagen con Speckle, desplazamiento de 10 mm a 10°, d) Imagen con Speckle, desplazamiento de 15 mm a 10° [elaboración propia]

- **IMÁGENES CON PROBETAS PROCESADAS a 20 GRADOS**

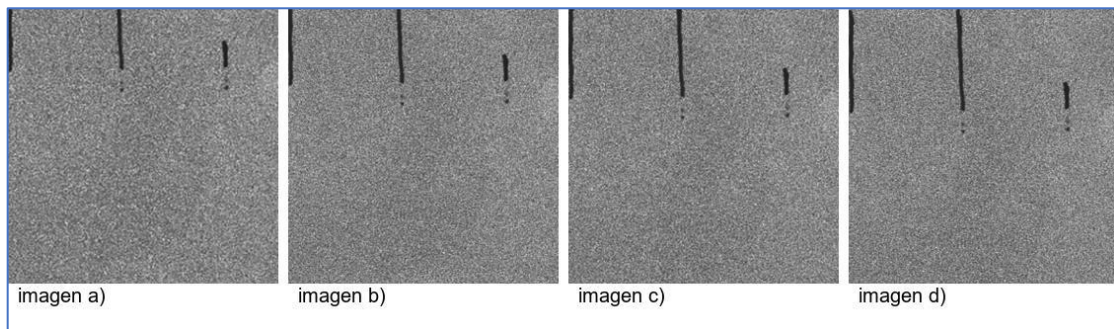


Ilustración 76. a) Imagen de referencia de Speckle a 20° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 20°, c) Imagen con Speckle, desplazamiento de 10 mm a 20°, d) Imagen con Speckle, desplazamiento de 15 mm a 20° [elaboración propia].

- **IMÁGENES CON PROBETAS PROCESADAS a 30 GRADOS**

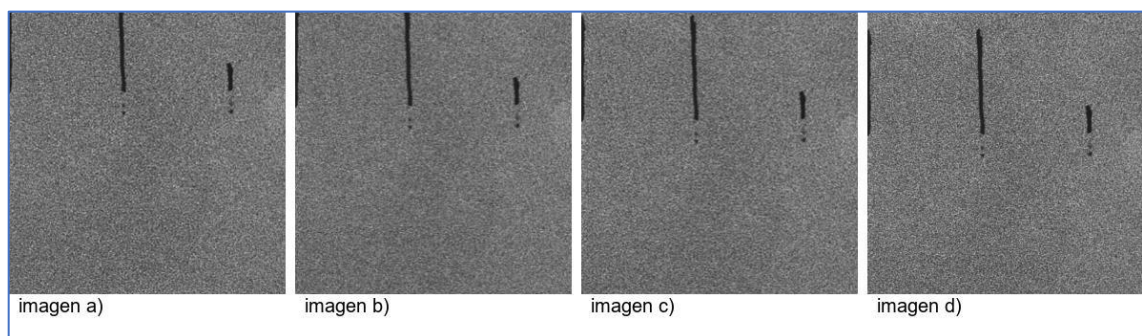


Ilustración 77. a) Imagen de referencia de Speckle a 30° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 30°, c) Imagen con Speckle, desplazamiento de 10 mm a 30°, d) Imagen con Speckle, desplazamiento de 15 mm a 30° [elaboración propia].

Por último, quedaría aplicar correlación de imágenes.

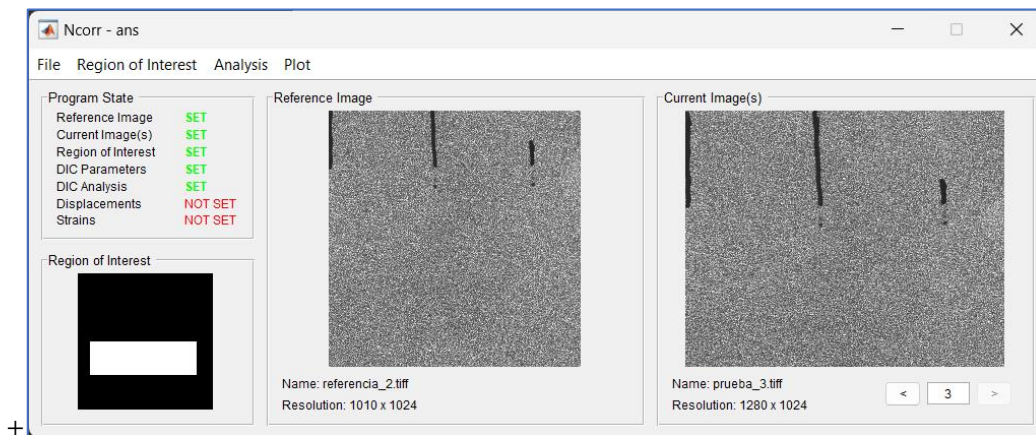


Ilustración 78. Imagen del GUI de Ncorr en donde se ve las imágenes utilizadas para el estudio de desplazamiento con giro angular y cambio de perspectiva para un giro de 10° [elaboración propia].

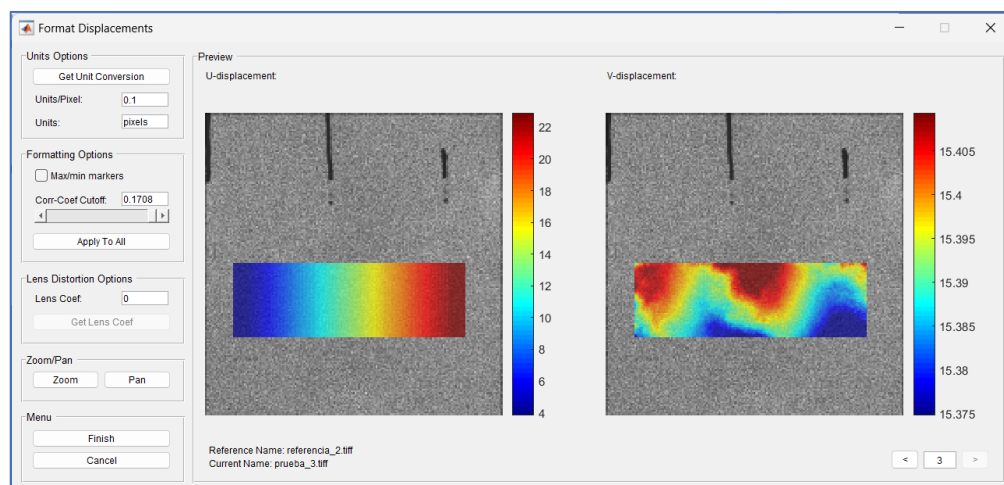


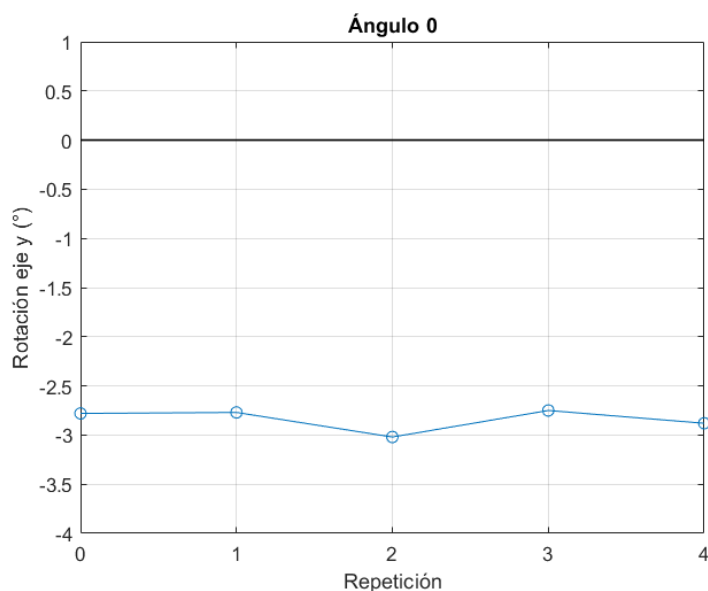
Ilustración 79. Imagen del GUI de Ncorr para desplazamientos en donde se ve el mapa de calor de las imágenes con desplazamientos, giro de 10° y procesamiento de la imagen [elaboración propia].

5. RESULTADOS

5.1 ENSAYO 1. CÁLCULO DEL ÁNGULO DE ALINEACIÓN

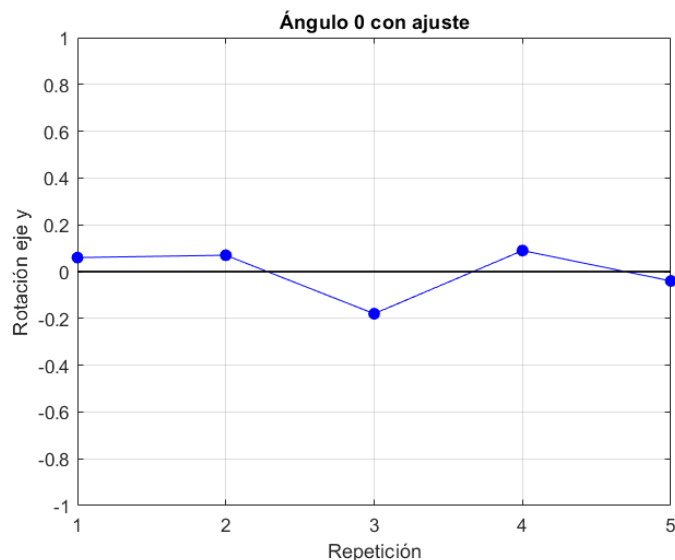
Como se vio en el ensayo 1, se trataba de determinar la pericia del algoritmo para determinar los ángulos de giro de la placa. los resultados arrojados en el mismo corresponden a la siguiente gráfica:

GRÁFICO 1. GRÁFICA QUE MUESTRA LAS MEDICIONES DEL ÁNGULO 0 SIN AJUSTAR EL ÁNGULO



El eje vertical se corresponde con los valores angulares expresados en grados sexagesimales, y el eje horizontal expresa el número de repetición correspondiente. Como se puede ver, hay poca dispersión y los valores se alejan del ángulo 0 en 3 grados aproximadamente. Se va a considerar, a partir de aquí, que el error producido se debe a la calibración del montaje experimental, y que el valor de la media de estos 5 valores será el ángulo que debemos ajustar y restar a cada uno de los valores anteriores para que nuestro resultado sea 0.

GRÁFICO 2. GRÁFICA QUE MUESTRA LAS MEDICIONES DEL ÁNGULO 0 CON ÁNGULO AJUSTADO



Y siendo la tabla y la media que la forma la siguiente:

TABLA 1. RESULTADOS ANGULARES DE LA ORTOGONALIDAD DE LA PLACA

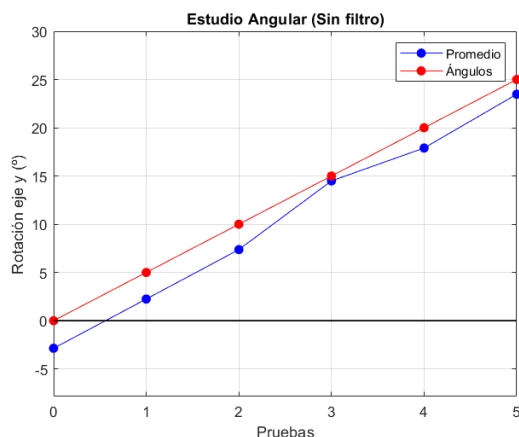
FILTRADA		MEDIA (ÁNGULO MEDIDO)	MEDIA (TRAS AJUSTAR EL ÁNGULO)
PRUEBA	ROTACIÓN EJE Y	-2,84	-8,88E-17
1	0.06		
2	0.07		
3	-0.18		
4	0.09		
5	-0.04		

Siendo los valores de la tabla los ya ajustados por la media.

Como se ha indicado, el objetivo del ensayo es comprobar la precisión del algoritmo desarrollado para determinar los ángulos de giro. Los resultados muestran poca dispersión y los valores se alejan del ángulo 0 en aproximadamente 3 grados, lo cual se considera un error en la calibración del montaje experimental.

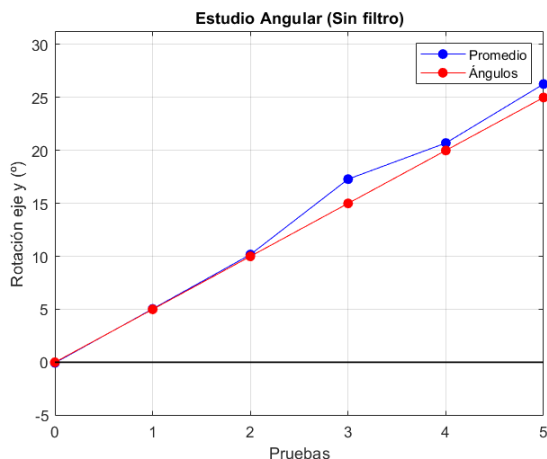
5.2 ENSAYO 2. CÁLCULO DEL ÁNGULO GIRADO

GRÁFICO 3. GRÁFICO QUE MUESTRAS LAS MEDICIONES MEDIAS DE DISTINTOS ÁNGULOS SIN EL AJUSTE



El eje vertical corresponde con los valores angulares expresados en grados sexagesimales, y el eje horizontal expresa la prueba realizada. La línea roja representa el valor teórico que la prueba debería haber dado como resultado, y la azul, el valor experimental. Los valores experimentales se tomaron a través de 5 ensayos en los que la orientación iba aumentando de 0° a 25° en saltos de 5° . Como se puede ver, hay poca dispersión y los valores se alejan del ángulo en 3 grados aproximadamente, igual que sucedía con el estudio cero del ángulo, por lo que aplicaremos el mismo ajuste. Así pues, se procede de la misma manera, y se ajustarán los datos como se ve en la gráfica siguiente:

GRÁFICO 4. GRÁFICO QUE MUESTRAS LAS MEDICIONES MEDIAS DE DISTINTOS ÁNGULOS CON EL AJUSTE



En este caso, se ha procedido con los ensayos de manera diferente. Dado que buscábamos estudiar varios ángulos, se han tomado tres imágenes diferentes a la misma posición de la probeta y, tras obtener el ángulo dado por el programa, se ha realizado la media de estos 3 datos. Todo queda registrado en la siguiente tabla:

TABLA 2. RESULTADOS ANGULARES EMITIDOS POR EL PROGRAMA

PRUEBA	PRIMERA (°)	SEGUNDA (°)	TERCERA (°)	PROMEDIO (°)	AJUSTADO CON (2,84°)	ÁNGULO REAL (°)
0	-2,78	-2,77	-3,02	-2,86	-0,06	0
1	2,31	2,29	2,12	2,24	5,04	5
2	7,67	7,22	7,21	7,37	10,17	10
3	14,4	14,56	14,51	14,49	17,29	15
4	17,59	18,16	17,94	17,90	20,70	20
5	23,87	22,3	24,23	23,47	26,27	25

Los datos muestran consistencia, con una media que se ajustaba a los valores reales esperados.

5.3 ENSAYO 3. EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERPENDICULARIDAD CON DIC

En este ensayo se comprueba la fiabilidad de la correlación de imágenes para detectar aberración de las imágenes giradas sin desplazamiento vertical, para ello necesitamos una imagen de referencia que no esté girada que sería la siguiente.

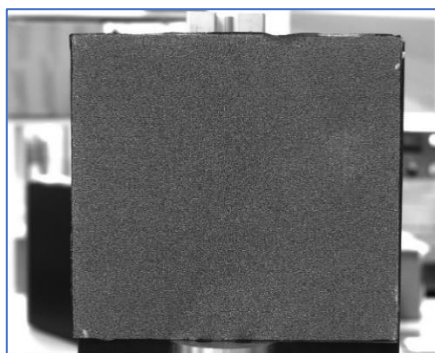


Ilustración 80. Imagen de referencia a 0°

Las ilustraciones que siguen muestran las probetas planas con el patrón de speckle giradas en diferentes ángulos (5°, 10°, 15° y 20°). Estas imágenes son las directamente capturadas por la cámara y no cuentan con ninguna técnica de procesamiento de imágenes, simplemente se les ha realizado el estudio de desplazamientos, previendo que, efectivamente, cuanto más giro haya mayor será el desplazamiento registrado.

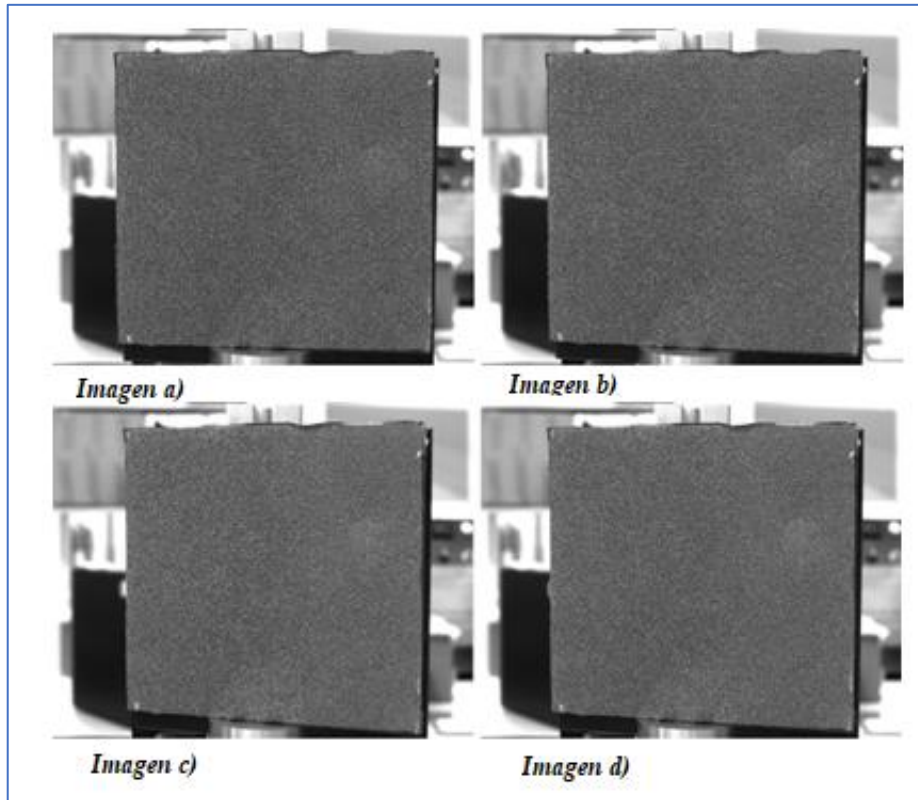


Ilustración 81. a) probeta plana con patrón de Speckle sin desplazamiento con 5 grados de giro, b) probeta plana con patrón de Speckle sin desplazamiento con 10 grados de giro, c) probeta plana con patrón de Speckle sin desplazamiento con 15 grados de giro, d) probeta plana con patrón de Speckle sin desplazamiento con 20 grados de giro [elaboración propia].

Las ilustraciones que siguen muestran los mapas de desplazamientos resultantes de procesar con 2D DIC las imágenes anteriores. Es decir, son los desplazamientos que el programa detecta con referencia a una imagen donde considerábamos que la probeta tenía giro y desplazamiento nulo.

Las imágenes obtenidas son una representación del mundo real; sin embargo, los sistemas digitales generan estas imágenes a través de píxeles, por lo que para poder hacer medidas de la realidad será necesario encontrar una relación de $\frac{\text{píxeles}}{\text{unidades reales}}$. Para

solucionar este problema, DIC permite, a partir de una imagen de referencia, expresar esta relación. En la Ilustración 50 se puede observar una ventana del programa de DIC donde se puede dibujar una línea sobre la imagen, si se conoce el tamaño de esa línea en la realidad, el propio programa calcula la relación. Como se puede observar, sobre la probeta hay una serie de líneas dibujadas que nos sirven como referencias del mundo real, concretamente entre línea y línea vertical hay 40 mm.

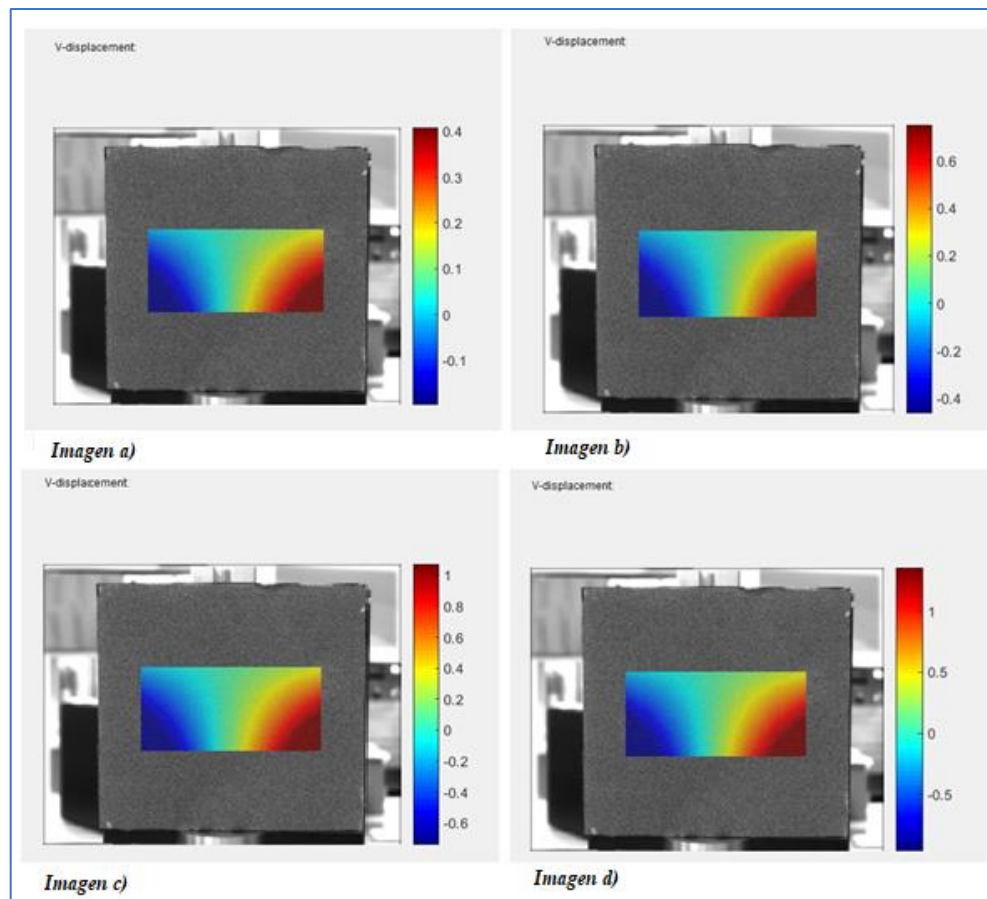


Ilustración 82. [Mapa de desplazamiento de la placa] Imagen a) con 5 grados de giro, imagen b) con 10 grados de giro, imagen c) con 15 grados de giro, imagen d) con 20 grados de giro [elaboración propia].

El método utilizado para cambiar la perspectiva de la imagen permite reducir los errores para casos donde la cámara no puede mirar de manera directa la cara de la probeta que se quiere estudiar. Para todos los casos de las imágenes anteriores, donde se giraron las probetas, los mapas de desplazamientos mostraron que existía movimiento en el eje vertical, aunque este nunca existió. Es aquí donde el cambio de perspectiva cobra sentido, ya que

nuestros algoritmos permiten simular una posición donde la cámara mira a las probetas de forma completamente perpendicular.

Las ilustraciones siguientes corresponde al mapa de desplazamiento de la misma placa, y con los mismos giros, que el de las imágenes anteriores. Sin embargo, en esta ocasión sí hemos utilizado la técnica de procesamiento de imágenes en la que cambiamos la perspectiva.

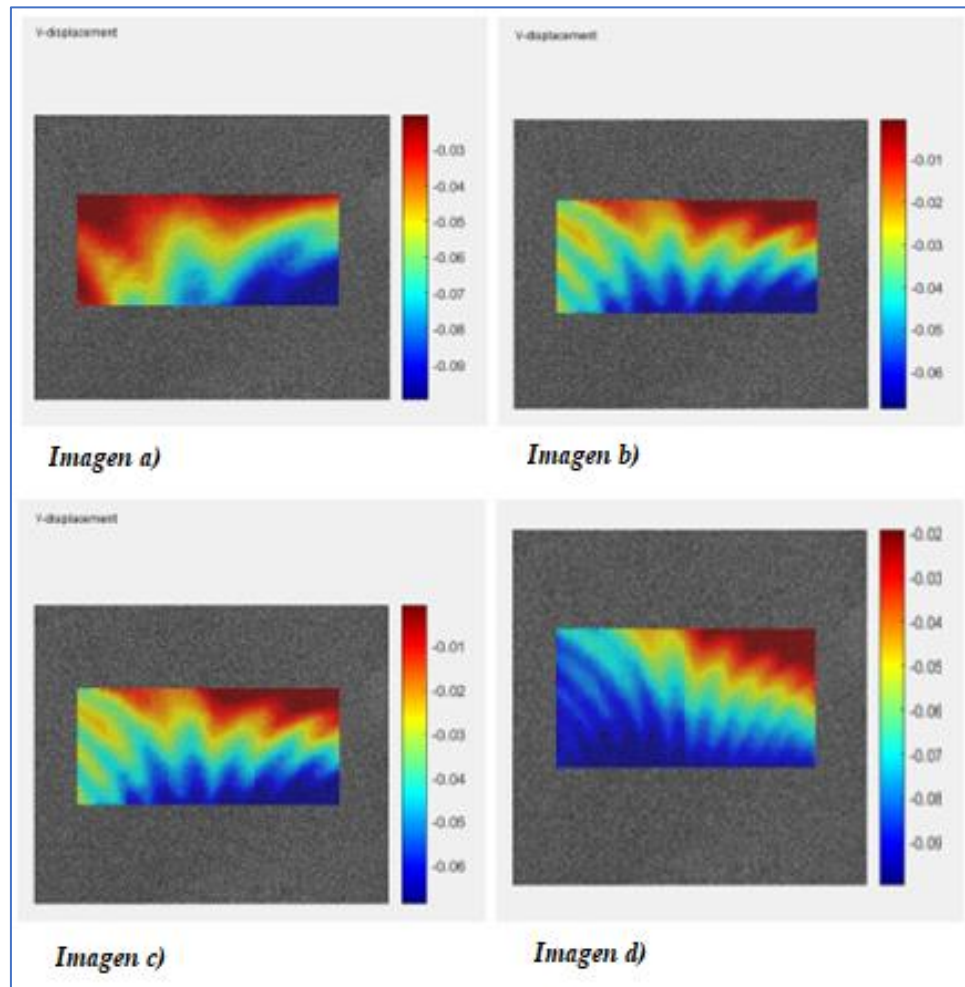


Ilustración 83. [Mapa de desplazamiento de la placa con procesamiento de imágenes] Imagen a) con 5 grados de giro, imagen b) con 10 grados de giro, imagen c) con 15 grados de giro, imagen c) con 20 grados de giro [elaboración propia].

Como se puede comprobar por las imágenes resultantes, el rango del campo de desplazamientos se reduce en gran medida, entre un 88% para el caso de la probeta con un giro de 5°; y un 95% para el caso de la probeta con 20°. Asimismo, también se puede comprobar que surgen formas como de abanicos en los mapas de desplazamientos, cuando

lo que cabría esperar es que fueran constantes. Esto sucede porque la precisión de nuestro sistema no es absoluta y debido a esto nuestro programa sigue encontrando desplazamientos, pero muy pequeños, en su mayoría cercanos a las centésimas de milímetro.

Para determinar la mejora de los datos en las 4 series de estudio de desplazamientos, se va a comparar tanto la media, como el RMS (Root Mean Square). En el estudio puede verse claramente, cómo la media de cantidad de desplazamientos aumenta conforme aumenta el ángulo. Esto se debe a que la precisión, a la hora de dibujar el área de estudio dentro del programa de DIC, no es absoluta y, aunque se intentó guardar la simetría a la hora de generar estas áreas, no se pudo mantener por completo, lo que degeneró en este aumento progresivo. Aun así, la media es muy próxima a 0, ya que los valores máximos y mínimos se anulan, pero no se corrigen por completo, por los motivos indicados. El RMS, se comporta de la misma manera.

También debe incrementarse para el caso de imágenes sin cambio de perspectiva, ya que contabiliza la magnitud promedio, o determina el dato que es más probable que se repita. Es decir, al igual que la media, este dato debería ser próximo a 0, para casos donde no hubiera desviación en los desplazamientos de los extremos de la placa. Pero al igual que antes, la imposibilidad para mantener el área en el centro perfecto de la imagen degeneró en este pequeño error. Por el contrario, para las imágenes ya procesadas, la media debería ser nula o mucho más próxima a 0 que en las imágenes anteriores, al igual que el RMS. Ya que el programa no ha podido detectar un giro en estas imágenes, como sí los hizo en las anteriores.

TABLA 3. RESULTADOS DE MEDIA Y RMS CON Y SIN TRANSFORMACIÓN

	SIN TRANSFORMACIÓN		CON TRANSFORMACIÓN	
	RMS	MEDIA	RMS	MEDIA
1	0,13104	-0,055542	0,050392	-0,046164
2	0,29304	-0,17233	0,032955	-0,039496
3	0,46446	-0,30208	0,061236	-0,057399
4	0,60668	-0,4057	0,05058	-0,047107

5.4 ENSAYO 4. EVALUACIÓN DEL ALGORITMO DE CORRECCIÓN DE PERSPECTIVA ANTE DESPLAZAMIENTOS VERTICALES

Para el ensayo 4, estableceremos resultados individuales para cada uno de los procesos y también una comparativa común de la media, el RMS y la desviación estándar de cada estudio. Empezamos con los mapas de calor de todos los procesos.

- **RESULTADOS PARA 5 mm DE DESPLAZAMIENTO.**

En las ilustraciones siguientes se muestran las condiciones sin giro ni procesamiento de imágenes para un desplazamiento de 5 mm, los casos siguientes se compararán con estas imágenes.

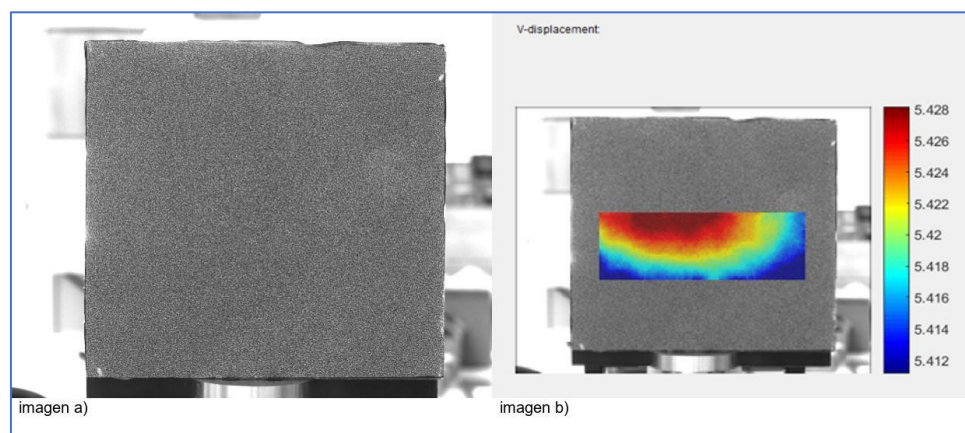


Ilustración 84. Imagen a) Placa sin giro desplazada 5 mm, imagen b) mapa de calor correspondiente [elaboración propia].

A continuación, se ve la probeta plana con patrón de Speckle girada una serie de ángulos y todas ellas desplazadas 5 mm como indica el título de este apartado.

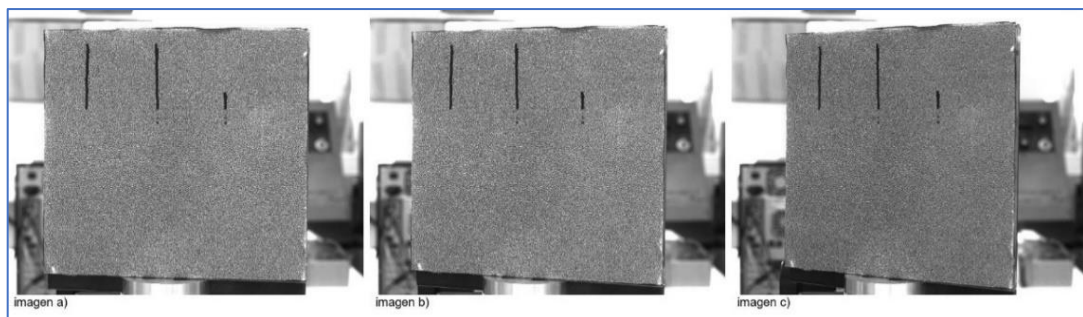


Ilustración 85. [Placa desplazada 5 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia]

Las ilustraciones siguientes muestran el mapa de desplazamientos de las 3 imágenes anteriores estudiado por DIC.

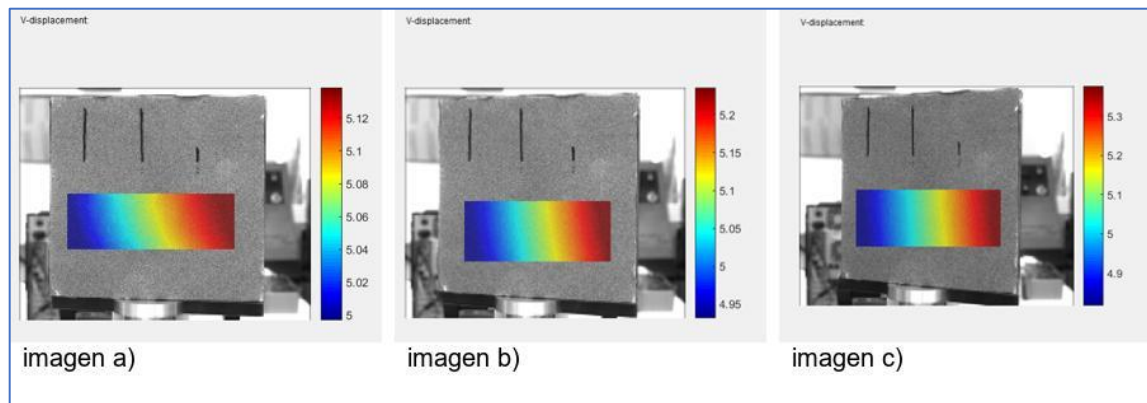


Ilustración 86. [Mapa de desplazamiento de la placa desplazada 5 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

Por último, solo queda comparar las imágenes que muestran el desplazamiento sin cambio de perspectiva de la *Ilustración 86* con las siguientes, que sí cuentan con cambio de perspectiva. En las imágenes anteriores se podía ver claramente un desplazamiento mayor en la zona más cercana a la cámara y uno menor conforme se alejaba de la misma. Este es, precisamente, el error o aberración en el desplazamiento por culpa de la perspectiva al que hemos venido haciendo referencia. El objetivo último de este proyecto es solucionar la aparición de estas aberraciones a la hora de la realización de los ensayos.

A continuación, se puede ver claramente cómo hay un cambio en la franja de desplazamientos, concretamente un 85% para el caso de la franja de la imagen con 10°, de giro 92% para la de 20° y 95% para el caso de 30°.

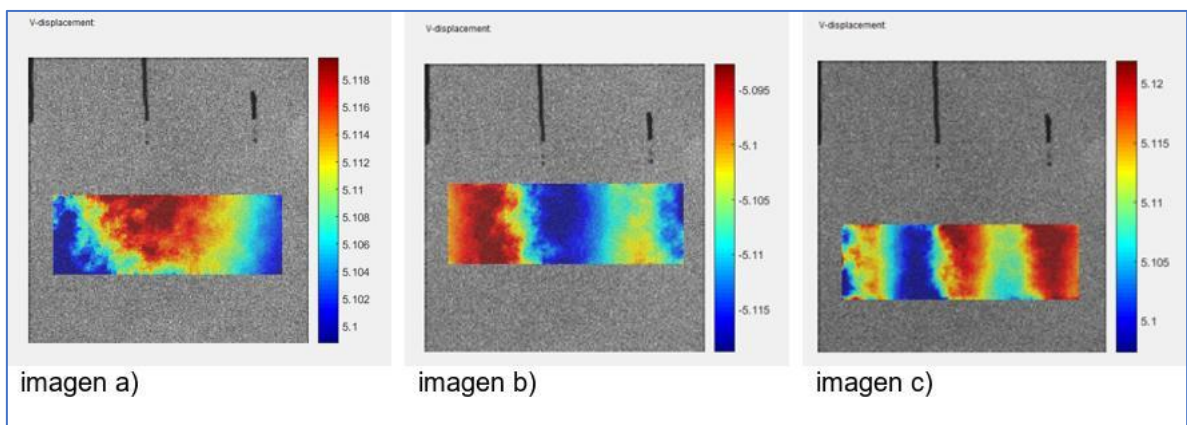


Ilustración 87. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 5 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

• **RESULTADOS PARA 10 mm DE DESPLAZAMIENTO**

En las ilustraciones siguientes se muestran las condiciones sin giro para un desplazamiento de 10 mm, los casos siguientes se compararán con estas imágenes.

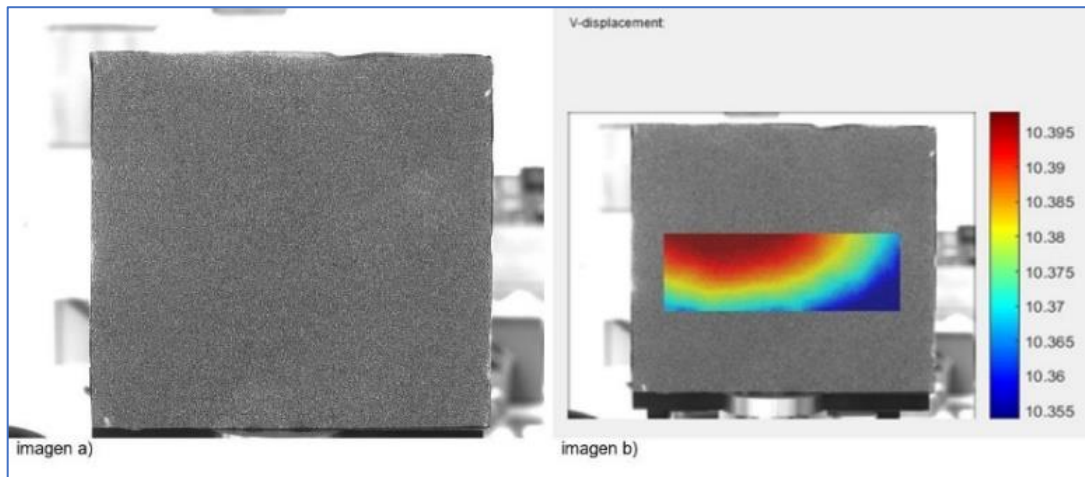


Ilustración 88. Imagen a) Placa sin giro desplazada 10 mm, imagen b) mapa de calor correspondiente [elaboración propia].

En estas ilustraciones se ver la probeta plana con patrón de Speckle girada una serie de ángulos y todas ellas desplazadas 10 mm, como indica el título de este apartado.

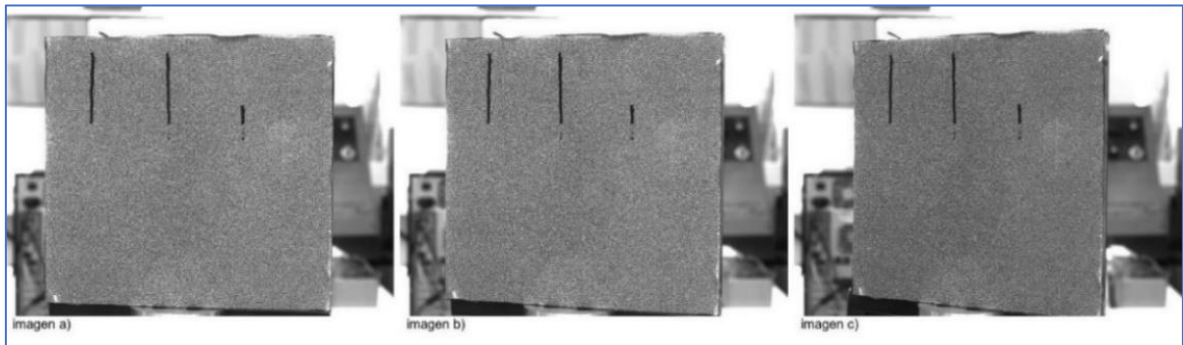


Ilustración 89. [Placa desplazada 10 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

Las ilustraciones siguientes, muestran el mapa de desplazamientos de las tres imágenes anteriores estudiado por DIC.

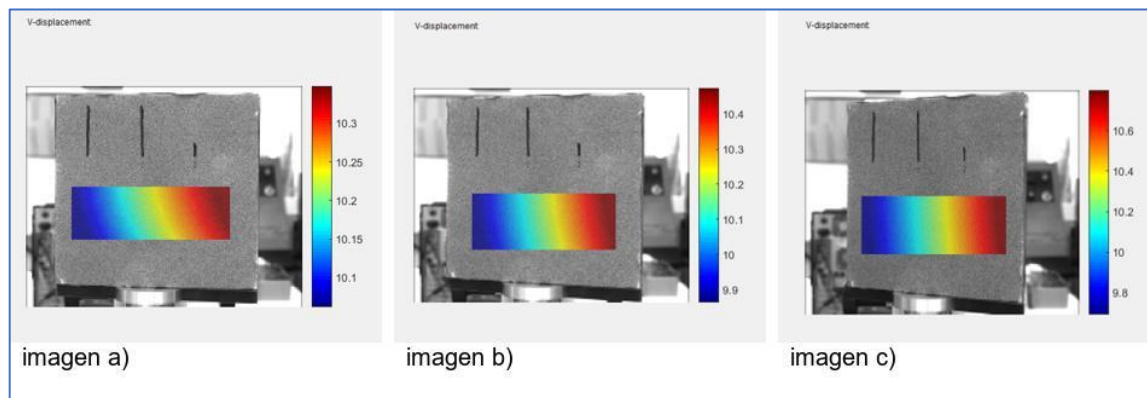


Ilustración 90. [Mapa de desplazamiento de la placa desplazada 10 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

Ahora queda comparar las imágenes que muestran el desplazamiento sin cambio de perspectiva, que representaban las imágenes de la *Ilustración 90* con las siguientes que si cuentan con cambio de perspectiva.

Aquí se observa cómo hay un cambio en la franja de desplazamientos, concretamente un 85% para el caso de la franja de la imagen con 10° de giro, 93% para la de 20 °y 97% para el caso de 30°.

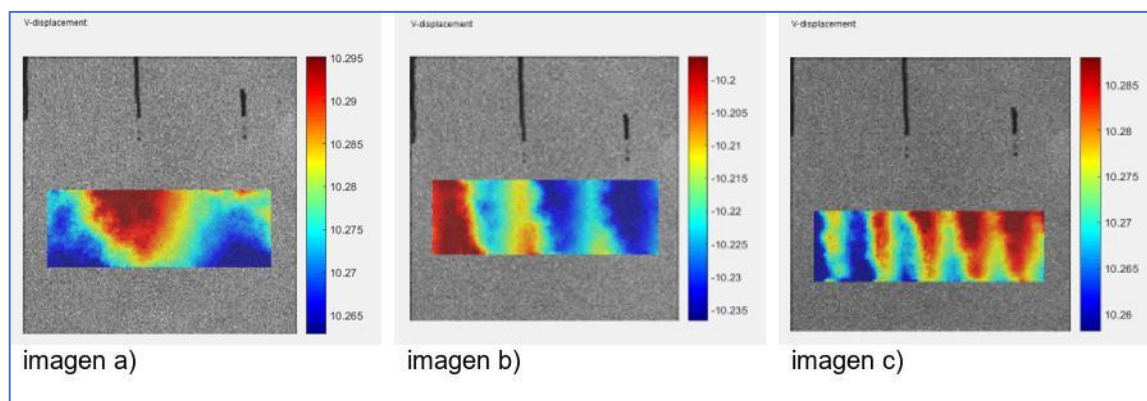


Ilustración 91. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 10 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

- **RESULTADOS PARA 15 mm DE DESPLAZAMIENTO.**

A continuación, se muestran las imágenes sin giro para un desplazamiento de 15 mm, los casos siguientes se compararán con estas imágenes.

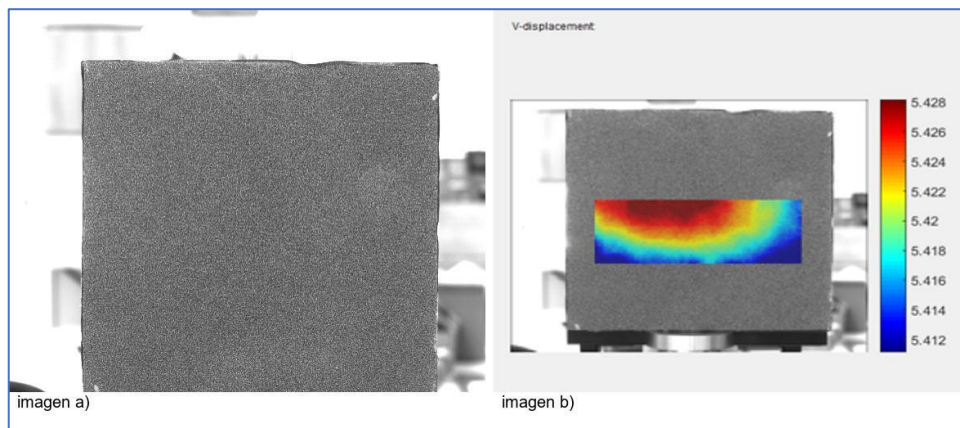


Ilustración 92. Imagen a) Placa sin giro desplazada 15 mm, imagen b) mapa de calor correspondiente [elaboración propia].

En las ilustraciones siguientes, se ve la probeta plana con patrón de Speckle girada una serie de ángulos y todas ellas desplazadas 15 mm como indica el título de este apartado.

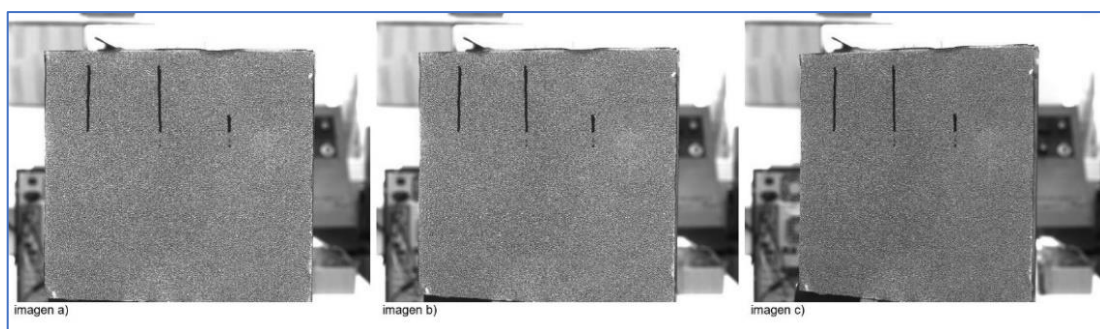


Ilustración 93. [Placa desplazada 15 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

En las siguientes imágenes se muestran los mapas de desplazamientos de las tres imágenes anteriores estudiado por DIC.

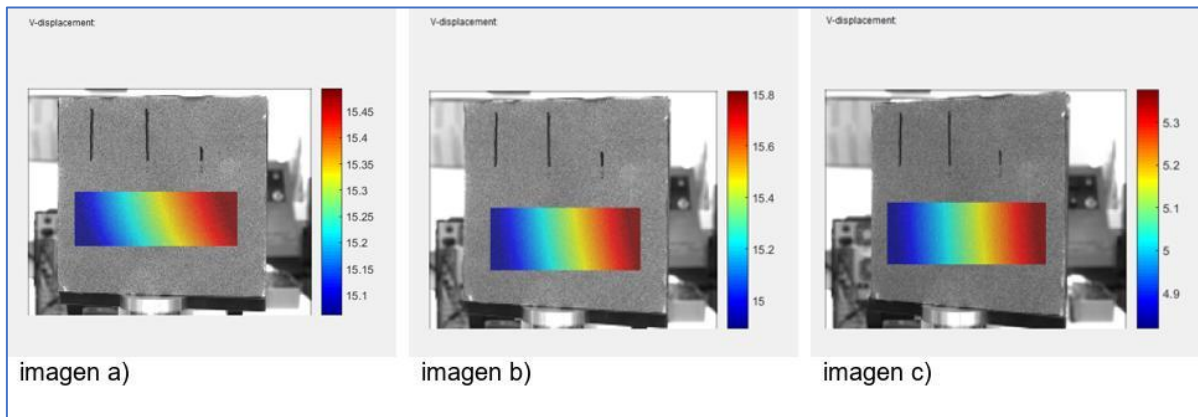


Ilustración 94. [Mapa de desplazamiento de la placa desplazada 15 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

Por último, falta comparar las imágenes que muestran el desplazamiento sin cambio de perspectiva, que representaban las imágenes de la *Ilustración 94*, con las siguientes que sí cuentan con cambio de perspectiva.

Las imágenes siguientes se puede ver claramente como hay un cambio en la franja de desplazamientos concretamente un 90% para el caso de la franja de la imagen con 10°, de giro 95% para la de 20° y 94% para el caso de 30°.

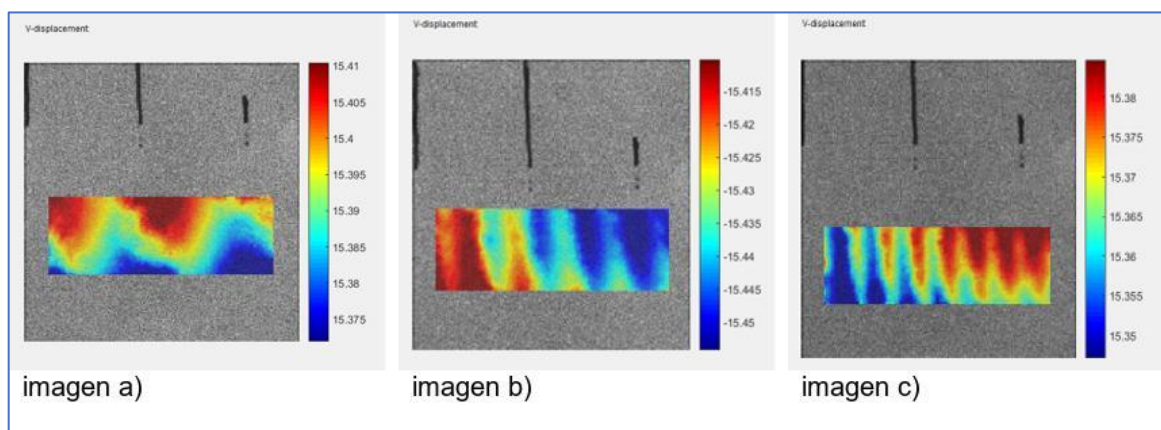


Ilustración 95. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 15 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].

TABLA 4. RESULTADOS RMS, MEDIA Y DESVIACION ESTANDAR DE LOS DESPLAZAMIENTOS PARA ESTUDIO SIN CAMBIO ANGULAR

DESPLAZAMIENTO SIN GIRO			
DES (mm)	RMS	MEDIA	DESVIACION ESTANDAR
5	0,0051403	5,42060	0,0051408
10	0,013075	10,37970	0,013076
15	0,034883	15,39140	0,034886

La contracción DES se corresponden a desplazamiento

Estas son las medidas de los desplazamientos dados por DIC, sin ángulo de giro ni cambio de perspectiva.

Como se ve los valores son los esperables, el RMS es próximo a 0, los valores de las distancias medias son los estudiados, y la desviación estándar indica que los valores no se alejan en gran medida de la media. Estos valores confirman que el programa de DIC funciona de manera coherente.

5.4.1 COMPARATIVA DEL RMS, MEDIA Y DESVIACIÓN ESTANDAR CON ÁNGULO DE GIRO Y DESPLAZAMIENTO SIN Y CON CAMBIO DE PERSPECTIVA.

TABLA 5. RESULTADOS CON Y SIN CORRECCIÓN DEL ÁNGULO DE GIRO A 5mm, 10mm y 15mm en 10°, 20° y 30°

DES	(°)	SIN CORRECCIÓN			CON CORRECCIÓN		
		RSM	MEDIA	D/E	RMS	MEDIA	D/E
5mm	10	0,045806	5,07240	0,04581	0,0065261	5,11020	0,006266
	20	0,09791	5,08520	0,09792	0,0084577	5,10660	0,0084583
	30	0,1685	5,09810	0,16852	0,0073379	5,11190	0,0073385
10mm	10	0,09283	10,21030	0,092389	0,010078	10,27820	0,010079
	20	0,19688	10,17390	0,1969	0,012303	10,22020	0,012303
	30	0,33295	10,25670	0,33299	0,0085398	10,27610	0,0085404
15mm	10	0,123876	15,28740	0,1387	0,011522	15,39270	0,011523
	20	0,29657	15,36050	0,2966	0,013875	15,43520	0,013876
	30	0,49807	15,32280	0,48812	0,011457	15,36960	0,011458

La contracción DES se corresponden a desplazamiento y las siglas D/E se corresponden con Desviación Estándar

La tabla 5 muestra de forma comparativa los valores de RMS, media y desviación estándar de todas las pruebas hechas en el ensayo 4. Se divide en dos columnas en función de si los datos corresponden a las pruebas realizados con la corrección de la aberración o sin ella. Igualmente, se compartimenta en tres tablas que diferencian los desplazamientos de 5mm, 10mm y 15mm al igual que sucedía con la tabla 4. Finalmente, en cada una de estas tablas se consideran si fueron tomadas las imágenes a 10°, 20° o 30° de giro con respecto la cámara.

Los resultados obtenidos para los desplazamientos con corrección se alejan de los que no presentan corrección. Esto se debe a que al estudiar las correlación de imágenes, esta

no considera el ángulo de giro de la probeta plana, por lo que genera desplazamiento en los extremos, debido a la perspectiva de la imagen. Esto se refleja en los resultados de las tablas, y aunque la media es correcta, y equivale al valor real del desplazamiento, porque los extremos se anulan entre sí, podemos ver tanto en el RMS, como en la desviación estándar, un incremento en ambos valores. Esto indica que la magnitud eficaz se aleja del promedio, que es lo que indica el valor de RMS. Los resultados se deben a la misma razón que sucedía en los ensayos 3, la falta de precisión del usuario para encontrar el centro perfecto de la imagen a la hora de generar el área de estudio. Para el caso de la desviación estándar, sin embargo, el hecho de que se aleje de 0 si se debe directamente al giro de la probeta plana, ya que este nos indica si existen muchos valores que se alejan del promedio. Al producirse desplazamientos ficticios por culpa de la aberración de la perspectiva de la imagen, la D/E se incrementa. Cosa que no sucede con la imagen que tiene corrección de este tipo de aberración.

Se han combinado diferentes métodos de estudio para comparar los distintos tipos de imágenes en cada caso. Las imágenes procesadas ofrecen resultados más precisos y veraces a los obtenidos antes sin la corrección del giro.

6. CONCLUSIONES

En este trabajo se han analizado satisfactoriamente las diferentes herramientas de computación para la transformación perspectiva de imágenes, y también se consigue el segundo propósito de mejorar las técnicas ópticas de campo completo, concretamente DIC 2D. El segundo objetivo se ha llevado a buen fin gracias a la creación de una herramienta de transformación de perspectiva, consistente de tres algoritmos y sus correspondientes interfaces, que han posibilitado unos ensayos más fiables y cómodos. De hecho, los datos arrojados para nuestra técnica de procesamiento de imágenes han mejorado los rangos de desplazamientos en 85%, para los casos donde las probetas fueron giradas 10°, y superaron el 90% de mejora para todos los casos donde las probetas superaron este valor de 10°, llegando a una mejora del 97% para el caso de un desplazamiento de 10 mm y giro de 30°.

Estos objetivos se han cumplido debido a una correcta combinación entre Python y OpenCV, logrando maximizar las ventajas de la combinación de este lenguaje de programación, de código abierto, y una de sus bibliotecas fundamentadas en la visión artificial, que ha permitido la creación de algoritmos destinados a corregir aberraciones que se puedan producir en los ensayos de deformación y desplazamiento de materiales.

En la siguiente lista se resume los diferentes objetivos alcanzados en este proyecto:

- Se ha demostrado la eficacia del algoritmo desarrollado para el análisis de imágenes y la detección de desplazamientos y ángulos.
- Se han creado Interfaces de Usuario: la implementación de una interfaz de usuario potente y completa facilita el uso del programa.
- Los ensayos han validado la precisión del algoritmo, particularmente en condiciones de procesamiento de imágenes, lo que sugiere su aplicabilidad en contextos prácticos donde se requiere precisión en la medición de desplazamientos y ángulos.

7. BIBLIOGRAFÍA

- [1] UNIVERSIDADESAPP. Especialidad en ingeniería mecánica experimental [Consulta: febrero de 2024]. Disponible en línea: <https://universidades.app/carreras/especialidades/ingenieria-y-ciencias-aplicadas/ingenieria-mecanica-y-electrica/especialidad-en-ingenieria-mecanica-experimental>
- [2] OTERO GÓMEZ, J.J. 2002. Modelo interpolativo punto-superficie para calibrado de cámaras en Visión Artificial. Tesis doctoral. Universidad Politécnica de Madrid. Madrid. DOI:10.20868/UPM.thesis.601
- [3] BLABER, J. DIC Algorithms Ncorr v1.2. [Consulta: diciembre de 2023] Disponible en línea: http://www.ncorr.com/index.php/dic-algorithms#_ENREF_1
- [4] MCCORMIC, N & LORD, J, 2010. “ *Digital imagen correlation*”. Ed. Material today. Vol.13, nº12, pag. 52-54. Londres. Reino Unido. DOI: 10.1016/S1369-7021(10)70235-2
- [5] SIEMENS.com. Correlación de Imágenes Digitales. [Consulta: febrero de 2024] Disponible en línea: <https://plm.sw.siemens.com/es-ES/simcenter/simulation-test/structural-dynamics-testing/digital-image-correlation/>
- [6] PALACIOS PINEDA, L. M. y RESÉNDIZ HERNÁNDEZ, J. E. 2017. “Análisis de deformaciones unitarias en juntas soldadas utilizando correlación digital de imágenes”. Revista de ingeniería mecánica. ECORFAN-Spain, . Vol. 1, num. 1, pag 10-19. España.
- [7] PERPETUALENIGMA.es, 2014. Understanding Camera Calibration. [Consulta: enero de 2024] Disponible en línea: <https://prateekvjoshi.com/2014/05/31/understanding-camera-calibration/>
- [8] PRIMERARESPUESTA.es, 2019. Diferencia entre proyecciones ortogonales y axonométricas. [Consulta: enero de 2024] Disponible en línea: <https://www.primerarespuesta.es/es/diferencia-entre-proyecciones-ortogonales-y-axonomicas>
- [9] VIGGIANI, G., & Hall, S. A. “Full-field measurements, a new tool for laboratory experimental geomechanics”, Fourth symposium on deformation characteristics of geomaterials, vol. 1, pp. 3-26, 2008. Atlanta. USA.

[10] VALADEZ VALDERRAMA D.A. , ARROYO QUINTANA, R. y PARRA MICHEL, J.R. 2015. Análisis de la deformación de soldadura en probetas mediante la técnica correlación digital de imágenes. Revista Innovación y desarrollo tecnológico. Vol. 7. Núm. 3. Pág. 134-142 Ed. Universidad De La Salle Bajío, Escuela de Ingenierías. México.

[11] VÉLEZ SERRANO, J.F., 2003. Visión Por Computador. España: Dykinson, S.L. CÁMARAS DE LUZ VISIBLE, Ed. Dykinson, S.L. Madrid. España.

[12] GRUPO DE INVESTIGACIÓN EDMANS, 2006. Técnicas y algoritmos básicos de visión artificial. 2006. Universidad de la Rioja. Logroño. España.

[13] IBM.com. ¿Qué es la visión artificial? El uso de machine learning y redes neuronales para enseñar a los ordenadores a ver defectos y problemas antes de que afecten a las operaciones. 2022 [Consulta: enero 2024] Disponible en línea: <http://www.ibm.com/es-es/topics/computer-vision>

[14] UNIR, UNIVERSIDAD EN INTERNET. La visión artificial: una revolución en la Industria 4.0. 2021. [Consulta: enero de 2024] Disponible en línea: <https://www.unir.net/ingenieria/revista/vision-artificial/>

[15] PLATERO DUEÑAS C. Apuntes de Visión Artificial. Dpto. Electrónica, Automática e Informática Industrial. Universidad Politécnica de Madrid. España .

[16] COGNEX.com, 2018. Introducción a la visión artificial. Una guía para la automatización de procesos y mejorar la calidad. Disponible en línea: https://bcnvision.es/uploads/videotutoriales/uploads/guias%20por%20sectores/introduccion%20a%20la%20vision%20artificial_compressed.pdf

[17] PARENTESIS.com, MEDINA, J. 2012. ¿Sensor CCD o CMOS? ¿Qué significa todo esto? [Consulta: noviembre 2023] Disponible en línea: https://parentesis.com/tutoriales/Sensor_CCD_o_CMOS_Que_significa_todo_esto

[18] INFOIMON.com RUIZ, T. 2017. Sensores y cámaras CMOS. [Consulta: noviembre 2022] Disponible en: <https://infaimon.com/enciclopedia-de-la-vision/sensores-y-camaras-cmos/>

[19] COPAROMAN. 2014. Sistema de visión artificial. [Consulta: enero de 2024] Disponible en línea: <https://coparoman.blogspot.com/2014/01/sistema-de-vision-artificial.html>

- [20] MOLLEDA MERÉ, J. 2008. Técnicas de visión por computador para la reconstrucción en tiempo real de la forma 3D de productos laminados. Tesis doctoral. Universidad de Oviedo. 2008. Asturias. España.
- [21] GALINDO, A. 2015. Etapas y Componentes de un Sistema de Visión Artificial. [Consulta: enero de 2024] Disponible en línea: [Etapas y Componentes de un Sistema de Visión Artificial - ppt video online descargar \(slideplayer.es\)](#)
- [22] FERNÁNDEZ GARCÍA, N.L. 2016. Introducción a la Visión Artificial. Visión Artificial Avanzada. Escuela Politécnica Superior Universidad de Córdoba, pp. 1–434. 2016. Córdoba España.
- [23] PÉREZ ÁLVAREZ, J.A. 2001. Apuntes de Fotometría. Departamento de Topografía. Universidad de Mérida. Extremadura. España.
- [24] INSTITUTE OF TURBOMACHINERY AND FLUID DYNAMICS. 2024. Digital Image Correlation (DIC). 2024. [Consulta: febrero de 2024] Disponible en línea: <https://www.tfd.uni-hannover.de/en/research/our-services/measurement-technology/digital-image-correlation>
- [25] HIJAZI. A, FRIEDL. A, AND KÄHLER. C.J. “*Influence of Camera’s Optical Axis Non-perpendicularity on Measurement Accuracy of Two-dimensional Digital Image Correlation*”. Jordan Journal of Mechanical and Industrial Engineering. Vol. 5, N° 4. Pag. 373-382. 2011. Alemania.
- [26] POULAIN, X; KOHLMAN, L; BINIENDA, W; ROBERTS, G; GOLDBERG, R. and BENZERGA, A. 2013. "Determination of the intrinsic behavior of polymers using digital image correlation combined with video-monitored testing", International Journal of Solids and Structures, vol. 50, no. 11-12, pp. 1869-1878, DOI:10.1016/j.ijsolstr.2013.01.041
- [27] ANALITICAL TECHNOLOGIES. 2020. DIGITAL IMAGE CORRELATION (DIC). [Consulta: enero de 2024] Disponible en línea: <https://analytical-online.com/digital-image-correlation-dic.html>
- [28] LÓPEZ-ALBA, E.; FELIPE SESÉ L; VASCO-OLMO, J.M; DÍAZ-GARRIDO, FA. 2013. “Aplicaciones industriales de técnicas ópticas de campo completo para la medida de tensiones y deformaciones en elementos de máquinas”. DYNA, vol. 80, núm. 181, pp. 98-108 Universidad Nacional de Colombia. Medellín. Colombia.

[29] FELIPE SESÉ, L.A., Integración de Proyección de Franjas y Correlación digital de imágenes 2D para la medida de deformaciones y desplazamientos. Tesis doctoral. Universidad Politécnica de Jaén. 2014. Jaén. España.

[30] CASTILLO GONZÁLEZ, A., PEÑARANDA DOMÍNGUEZ, A.E. Y DÍAZ RODRÍGUEZ, G. 2020. “Implementación de la Técnica de Correlación de Imágenes Digitales con Software Libre”. AIBI, Revista de investigación, administración e ingeniería. Volumen 8, Número 3, Pág. 25-32. Universidad Santo Tomás, Cúcuta. 2020. Colombia.

[31] INTERACTIVE.org, GÓMEZ, M. Historias de la imagen digital [Consulta: enero de 2024] disponible en línea: <https://interartive.org/2017/04/historias-de-la-imagen-digital-marisa-gomez>

[32] PETER WH, AND RANSON WF. “Digital imaging techniques in experimental stress analysis”. Optical Engineering; Vol 21 (3): 427-31 (1982). DOI:10.1117/12.7972925

[33] SANTANDER UNIVERSIDADES. 2021. Python: qué es y por qué deberías aprender a utilizarlo. [Consulta: 25 de diciembre de 2023]. Disponible en línea: <https://www.becas-santander.com/es/blog/python-que-es.html>

[34] SCI.es. GONZÁLEZ DUQUE, R., 2011. Python para todos. Libro gratuito en línea.

[35] VAN ROSSUM, G., “Python Programming Language”. *USENIX annual technical conference*. pp. 1-36. 2007. Santa Clara. California. USA.

[36] OPENCV.ORG. 2023. Página web de la Compañía. [Consulta: 10 junio de 2023] Disponible en línea: <https://opencv.org/>

[37] DAMIANDELUCA.com.ar. ¿Qué es OpenCV? 2019. [Consulta: septiembre de 2023] disponible en <https://damiandeluca.com.ar/que-es-opencv>

[38] SOURCEFORCE.net. Summary. OpenCV. Open Source Vision Computer Library. [Consulta: septiembre 2023] disponible en línea: <http://SourceForge.net/projects/opencvlibrary>

[39] BRADSKI, G Y KAEHLER, A. 2008. Learning OpenCV: Computer vision with the OpenCV library. Ed. O'Reilly Media. Inc. California. USA DOI:[10.4236/jcc.2022.1010004](https://doi.org/10.4236/jcc.2022.1010004)

[40] IMAGINAFORMACIÓN.com. Tutorial de Procesamiento de imágenes con OpenCV en Python. 2023. [Consulta: febrero de 2024] Disponible en línea: <https://imaginaformacion.com/tutoriales/opencv-en-python>

[41] DIBUJOTECNIC.COM. Geometría proyectiva. homología y afinidad. 2023. [Consulta: febrero 2024] Disponible en línea: <https://dibujotecni.com/geometria-metrica-y-proyectiva/geometria-proyectiva-homologia-y-afinidad/>

[42] DATASMART.net. Lo que necesitas saber sobre la detección de bordes. 2024. [Consulta: febrero de 2024] Disponible en línea: <https://www.datasmarts.net/lo-que-necesitas-saber-sobre-la-deteccion-de-bordes/>

[43] DRON, L. 1993. Dynamic Camera self-calibration from controlled motion”, In Proc. IEEE. Conference on Computer Vision and Pattern Recognition, pp 501--506, New York, NY, USA. DOI:[10.1109/CVPR.1993.341083](https://doi.org/10.1109/CVPR.1993.341083)

[44] ZHENGYOU, Z.. 1998 “A Flexible New Technique for Camera Calibration Transactions on pattern analysis and machine intelligence”. 2000, vol. 22, no 11, p. 1330-1334. Institute of Electrical and Electronics Engineers. Trinity College. Irlanda. DOI: [10.1109/VR.2008.4480745](https://doi.org/10.1109/VR.2008.4480745)

[45] IRLES MAS, F. 2009. Transformaciones homográficas de modelos tridimensionales. Análisis proyectivo para el desarrollo de una aplicación infográfica. Tesis doctoral. Elche. Universidad Miguel Hernández . 2009. Elche. Identificador de cita: <https://hdl.handle.net/11000/1751>

[46] GONZÁLEZ JIMÉNEZ, J. 1999. Visión por Computador. Ed. Paraninfo, Madrid. España.

[47] DAWSON-HOWE, K. 2014. A practical introduction to computer vision with opencv. Ed. John Wiley & Sons, Trinity College. Dublin. Irlanda.

Listado de Gráficos

Gráfico 1. Gráfica que muestra las mediciones del ángulo 0 sin ajustar el ángulo.....	85
Gráfico 2. Gráfica que muestra las mediciones del ángulo 0 con ángulo ajustado.....	86
Gráfico 3. Gráfico que muestras las mediciones medias de distintos ángulos sin el ajuste.....	87
Gráfico 4. Gráfico que muestras las mediciones medias de distintos ángulos con el ajuste.....	87

Tabla De Ilustraciones

<u>Ilustración 1. a) Escalón de hormigón tras ser sometido a prueba de carga. Muestra una gran grieta [4].</u>	
<u>Ilustración 2. b) Tinte penetrante para mostrar grietas que no son visibles a simple vista [4]. Ilustración 3. c) Salida del DIC como mapa de deformación local que muestra grietas no visibles para el ojo humano [4].</u>	7
<u>Ilustración 4. El departamento de Energía Eólica de la Universidad Técnica de Dinamarca colabora con Siemens con el fin de utilizar las soluciones de DIC para el innovador proyecto ReliaBlade [5].</u>	8
<u>Ilustración 5. Configuración para realizar la correlación digital de imágenes 2D (Análisis de deformaciones unitarias en</u>	8
<u>Ilustración 6. a) representa a un objeto perfectamente alineado. b) representa a un objeto desalineado [8].</u>	9
<u>Ilustración 7. a) Probeta barrenada preparada con el moteado, b) Dispositivo que aplica tensión con la probeta montada y la cámara [10].</u>	10
<u>Ilustración 8. Espectro Electromagnético visible por el ojo humano [11].</u>	13
<u>Ilustración 9. Esquema CCD y CMOS [17].</u>	16
<u>Ilustración 10: Componentes de un sistema de Visión Artificial [19].</u>	20
<u>Ilustración 11. Diagrama de bloques de las etapas típicas en un sistema de visión artificial [21].</u>	21
<u>Ilustración 12. En la figura 2D la componente z puede estar representada por la intensidad del brillo de cada superficie [11].</u>	22
<u>Ilustración 13. Definición de una imagen digital [23].</u>	23
<u>Ilustración 14. Imagen digital. Representación visual y representación numérica [23]</u>	23
<u>Ilustración 15. Uso de la correlación de imágenes digitales en el montaje experimental para las deformaciones de las palas del rotor [24].</u>	24
<u>Ilustración 16. La imagen se divide en facetas para luego buscar una nueva posición tras la deformación [27].</u>	25
<u>Ilustración 17. A la derecha representación del desplazamiento de un punto [30]. A la izquierda Correlación de dos imágenes, S_r y S_d, son las ROI de la imagen de referencia y con una deformación de una superficie respectivamente [10].</u>	27
<u>Ilustración 18. Imagen del logotipo del lenguaje de programación Python [33].</u>	30

<u>Ilustración 19. Imagen del logotipo de la librería para Python OpenCV [38].</u>	32
<u>Ilustración 20. Código de modificación de tamaño de imagen en OpenCV [40].</u>	33
<u>Ilustración 21. Modelo de cámara en proyección perspectiva [43].</u>	35
<u>Ilustración 22. Proyección homográfica de dos figuras planas [45].</u>	37
<u>Ilustración 23. Modelo unidimensional y continuo de un borde ideal [46].</u>	41
<u>Ilustración 24. Una imagen en escala de grises de una iglesia (izquierda), la magnitud del gradiente de borde de Sobel (centro izquierda), una versión umbralizada de la magnitud del gradiente (centro derecha) y las orientaciones de Sobel correspondientes (derecha) [47].</u>	42
<u>Ilustración 25. La imagen de la izquierda es el ejemplo donde se aplica el estudio del borde y la de la derecha es un ejemplo de simulación de gradiente en los bordes y su orientación [47].</u>	42
<u>Ilustración 26. Una imagen simple en escala de grises (izquierda), junto con la magnitud de intensidad (centro izquierda), la función de primera derivada (centro derecha) y la función de segunda derivada (derecha) [47].</u>	43
<u>Ilustración 27. Detector de bordes sin aplicar (imagen de la izquierda) detector aplicado (imagen de la derecha) [42].</u>	44
<u>Ilustración 28. Representación de densidad de cambio que se utiliza para aplicar el filtro de segunda derivada (La técnica utilizada para hallar esta representación es la del filtro Laplaciano Gaussiano) [47].</u>	45
<u>Ilustración 29. Detección de bordes de Canny. Las tres imágenes de Canny utilizan el mismo filtro de suavizado, pero diferentes umbrales de gradiente bajo y alto (7 y 236 – superior derecho, 7 y 100 – inferior izquierdo, 1 y 1 – inferior derecho). La imagen inferior derecha muestra todos los bordes posibles y las otras dos muestran algunos de los efectos de los umbrales [42].</u>	46
<u>Ilustración 30. Diagrama de flujo que explica el proceso que determina el ángulo de giro [Elaboración propia].</u>	49
<u>Ilustración 31. Imagen del script del algoritmo donde se muestra la definición de la lista de coordenadas tridimensionales, de los píxeles de la imagen y como se rellenan ambas listas [Elaboración propia].</u>	50
<u>Ilustración 32. En la imagen a) se muestra la lista de puntos reales y en la b) la lista de los píxeles de la imagen donde se han detectado los puntos [Elaboración propia].</u>	51

<u>Ilustración 33. Diagrama de flujo que explica el proceso que determina el ángulo de giro [Elaboración propia].</u>	52
<u>Ilustración 34. Diagrama de flujo que explica el proceso donde se aplica una matriz de transformación ya guardada. [Elaboración propia].</u>	54
<u>Ilustración 35. Esquema montaje experimental de pruebas [Elaboración propia].</u>	57
<u>Ilustración 36. Pieza superior (arriba en la imagen), pieza intermedia (en medio de la imagen), pieza base (abajo en la imagen) [Elaboración propia].</u>	57
<u>Ilustración 37. Imagen que muestra el apoyo completo la probeta [Elaboración propia].</u>	58
<u>Ilustración 38. Imagen que muestra tanto la base de la placa como el goniómetro [Elaboración propia].</u>	58
<u>Ilustración 39. La imagen a) muestra la placa con cuadrícula de ajedrez, La imagen b) muestra la placa rectángulo, la imagen c) muestra la placa con el Speckle [Elaboración propia].</u>	59
<u>Ilustración 40. Fotografía del ángulo 0 de la cuadrícula [elaboración propia].</u>	60
<u>Ilustración 41. Captura que muestra el .exe de la interfaz [elaboración propia].</u>	60
<u>Ilustración 42. Ventana de inicio de la interfaz [elaboración propia].</u>	61
<u>Ilustración 43. Captura que muestra la imagen ya cargada [elaboración propia].</u>	61
<u>Ilustración 44. Ventana final con el ángulo de la cuadrícula ya obtenido [elaboración propia].</u>	61
<u>Ilustración 45. Captura con varios resultados para diferentes imágenes [elaboración propia].</u>	62
<u>Ilustración 46. Captura con varios resultados para diferentes imágenes [elaboración propia].</u>	63
<u>Ilustración 47. Captura de la interfaz 1 utilizada para el cálculo de varios ángulos (elaboración propia).</u>	63
<u>Ilustración 48. a) Ángulo a 0°, b) Ángulo 5°, c) Ángulo 10°, d) Angulo 2°, e) Angulo 30° [elaboración propia].</u>	64
<u>Ilustración 49. Estudio de desplazamiento con DIC sin cambio de perspectiva [elaboración propia].</u>	65
<u>Ilustración 50. Imagen que muestra una captura de la ventana de Ncorr donde se permite aplicar la relación mm/píxeles [elaboración propia]</u>	65
<u>Ilustración 51. Imagen donde se ve el mapa de calor de los desplazamientos de la placa [elaboración propia].</u>	65
<u>Ilustración 52. Ventana de la GUI de ensayo 3-4 sin cargar imagen [elaboración propia].</u>	66

<i><u>Ilustración 53. Ventana de explorador del sistema operativo que se abre para buscar la imagen tras pulsar el botón Open Image [elaboración propia].</u></i>	67
<i><u>Ilustración 54. Ventana del GUI con la imagen cargada [elaboración propia].</u></i>	67
<i><u>Ilustración 55. Mensaje que sale si la imagen ha sido procesada correctamente [elaboración propia].</u></i>	67
<i><u>Ilustración 56. Ventana final tras procesamiento y con la posición de los puntos y el cambio de perspectiva ya cargado (elaboración propia).</u></i>	68
<i><u>Ilustración 57. Ventana de la interfaz con el speckle procesado. [elaboración propia].</u></i>	68
<i><u>Ilustración 58. Ventana del explorador de archivos, para guardar imagen [elaboración propia].</u></i>	69
<i><u>Ilustración 59. La imagen de la izquierda es la de la placa a 5° y la de la derecha es la misma imagen de la placa de 5° pos-procesado [elaboración propia].</u></i>	69
<i><u>Ilustración 60. La imagen de la izquierda es la de la placa a 10° y la de la derecha es la misma imagen de la placa de 20° pos-procesado [elaboración propia].</u></i>	69
<i><u>Ilustración 61. La imagen de la izquierda es la de la placa a 15° y la de la derecha es la misma imagen de la placa de 5° pos-procesado [elaboración propia].</u></i>	70
<i><u>Ilustración 62. La imagen de la izquierda es la de la placa a 20° y la de la derecha es la misma imagen de la placa de 10° pos-procesado [elaboración propia].</u></i>	70
<i><u>Ilustración 63. GUI de Ncorr con imágenes pos-procesado [elaboración propia].</u></i>	70
<i><u>Ilustración 64. GUI de Ncorr de mapa de calor para el estudio de desplazamiento, con imágenes pos-procesado y pos-calibrado de la relación píxel/mm [elaboración propia].</u></i>	71
<i><u>Ilustración 65. Imagen a) Imagen de referencia probeta caso sin giro., Imagen b) Desplazamiento de 5 mm la probeta caso sin giro., Imagen c) Desplazamiento de 10 mm de la probeta caso sin giro., Imagen d) Desplazamiento de 15 mm de la probeta caso sin giro [elaboración propia].</u></i>	71
<i><u>Ilustración 66. Imagen del GUI de Ncorr en donde se ve las imágenes utilizadas para el estudio de desplazamiento sin giro [elaboración propia].</u></i>	72
<i><u>Ilustración 67. Imagen del GUI de Ncorr para desplazamientos en donde se ve el mapa de calor de desplazamientos probeta plana con patrón de Speckle [elaboración propia].</u></i>	72

<i><u>Ilustración 68. Imagen a) Referencia sin desplazamiento a 10 grados., Imagen b) Imagen de desplazamiento de 5 mm a 10 grados, Imagen c) Imagen de desplazamiento de 10 mm a 10 grados., Imagen d) Imagen de desplazamiento de 15 mm a 10 grados [elaboración propia].</u></i>	73
<i><u>Ilustración 69. Imagen a) Referencia sin desplazamiento a 20 grados., Imagen b) Imagen de desplazamiento de 5 mm a 20 grados, Imagen c) Imagen de desplazamiento de 10 mm a 20 grados., Imagen d) Imagen de desplazamiento de 15 mm a 20 grados [elaboración propia].</u></i>	73
<i><u>Ilustración 70. Imagen a) Referencia sin desplazamiento a 30 grados., Imagen b) Imagen de desplazamiento de 5 mm a 30 grados, Imagen c) Imagen de desplazamiento de 10 mm a 30 grados., Imagen d) Imagen de desplazamiento de 15 mm a 30 grados [elaboración propia].</u></i>	73
<i><u>Ilustración 71. Imagen del GUI de DIC para desplazamientos en donde se ve el mapa de calor de las imágenes con desplazamientos y giro de la probeta plana con patrón de Speckle [elaboración propia].</u></i>	74
<i><u>Ilustración 72. Imagen del GUI de la interfaz gráfica donde muestra la cuadrícula [elaboración propia].</u></i>	74
<i><u>Ilustración 73. Imagen de la GUI ya con la imagen de la probeta plana con patrón de Speckle cargada con un ángulo de giro de 10° [elaboración propia].</u></i>	75
<i><u>Ilustración 74. Imagen del GUI de la interfaz gráfica de los ensayos con cambio de perspectiva con la imagen cuadrado a 10° ya procesada [elaboración propia].</u></i>	75
<i><u>Ilustración 75. a) Imagen de referencia de Speckle a 10° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 10°, c) Imagen con Speckle, desplazamiento de 10 mm a 10°, d) Imagen con Speckle, desplazamiento de 15 mm a 10° [elaboración propia]</u></i>	76
<i><u>Ilustración 76. a) Imagen de referencia de Speckle a 20° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 20°, c) Imagen con Speckle, desplazamiento de 10 mm a 20°, d) Imagen con Speckle, desplazamiento de 15 mm a 20° [elaboración propia].</u></i>	76
<i><u>Ilustración 77. a) Imagen de referencia de Speckle a 30° con cambio de perspectiva, b) Imagen con Speckle, desplazamiento de 5 mm a 30°, c) Imagen con Speckle, desplazamiento de 10 mm a 30°, d) Imagen con Speckle, desplazamiento de 15 mm a 30° [elaboración propia].</u></i>	76
<i><u>Ilustración 78. Imagen del GUI de Ncorr en donde se ve las imágenes utilizadas para el estudio de desplazamiento con giro angular y cambio de perspectiva para un giro de 10° [elaboración propia].</u></i>	77

<u>Ilustración 79. Imagen del GUI de Ncorr para desplazamientos en donde se ve el mapa de calor de las imágenes con desplazamientos, giro de 10° y procesamiento de la imagen [elaboración propia].</u>	77
<u>Ilustración 80. Imagen de referencia a 0°</u>	81
<u>Ilustración 81. a) probeta plana con patrón de Speckle sin desplazamiento con 5 grados de giro, b) probeta plana con patrón de Speckle sin desplazamiento con 10 grados de giro, c) probeta plana con patrón de Speckle sin desplazamiento con 15 grados de giro, d) probeta plana con patrón de Speckle sin desplazamiento con 20 grados de giro [elaboración propia].</u>	82
<u>Ilustración 82. [Mapa de desplazamiento de la placa] Imagen a) con 5 grados de giro, imagen b) con 10 grados de giro, imagen c) con 15 grados de giro, imagen d) con 20 grados de giro [elaboración propia].</u>	83
<u>Ilustración 83. [Mapa de desplazamiento de la placa con procesamiento de imágenes] Imagen a) con 5 grados de giro, imagen b) con 10 grados de giro, imagen c) con 15 grados de giro, imagen c) con 20 grados de giro [elaboración propia].</u>	84
<u>Ilustración 84. Imagen a) Placa sin giro desplazada 5 mm, imagen b) mapa de calor correspondiente [elaboración propia].</u>	86
<u>Ilustración 85. [Placa desplazada 5 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia]</u>	86
<u>Ilustración 86. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 5 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].</u>	¡Error! Marcador no definido.
<u>Ilustración 87. Imagen a) Placa sin giro desplazada 10 mm, imagen b) mapa de calor correspondiente [elaboración propia].</u>	88
<u>Ilustración 88. [Placa desplazada 10 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].</u>	88
<u>Ilustración 89. [Mapa de desplazamiento de la placa desplazada 10 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia].</u>	89

Ilustración 90. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 10 mm y girada]
imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia]. **¡Error! Marcador no**
definido.

Ilustración 96. [Placa desplazada 15 mm y girada] imagen a) 10 grados, imagen b) 20 grados, imagen c) 30
grados [elaboración propia]. **¡Error! Marcador no definido.**

Ilustración 97. [Mapa de desplazamiento de la placa desplazada 15 mm y girada] imagen a) 10 grados,
imagen b) 20 grados, imagen c) 30 grados [elaboración propia]. **¡Error! Marcador no definido.**

Ilustración 98. [Mapa de desplazamiento con cambio de perspectiva de la placa desplazada 15 mm y girada]
imagen a) 10 grados, imagen b) 20 grados, imagen c) 30 grados [elaboración propia]. **¡Error! Marcador no**
definido.

ANEXOS

1. ANEXOS

1.1. PROGRAMA QUE GENERA UNA CUADRICULA EN PYTHON

```
from PIL import Image, ImageDraw

# Definir las dimensiones de la cuadrícula

filas = 9

columnas = 6

ancho_celda = 50

alto_celda = 50

# Calcular el tamaño de la imagen

ancho_imagen = columnas * ancho_celda

alto_imagen = filas * alto_celda

# Crear una nueva imagen en modo RGB con fondo blanco

imagen = Image.new("RGB", (ancho_imagen, alto_imagen), "white")

# Crear un objeto para dibujar en la imagen

dibujar = ImageDraw.Draw(imagen)

# Dibujar las celdas de la cuadrícula

for i in range(filas):

    for j in range(columnas):

        x1 = j * ancho_celda

        y1 = i * alto_celda

        x2 = x1 + ancho_celda

        y2 = y1 + alto_celda

        # Alternar el color de las celdas

        if (i + j) % 2 == 0:

            dibujar.rectangle([x1, y1, x2, y2], fill="black")

        else:
```

```
dibujar.rectangle([x1, y1, x2, y2], fill="white")
```

```
# Mostrar la imagen
```

```
imagen.show()
```

```
# Especificar la ruta para guardar la imagen
```

```
ruta_de_la_imagen = r"C:\Users\luisg\Desktop\tfg\Carpeta de resultados\cuadrícula.jpg"
```

```
# Guardar la imagen en la ruta especificada
```

```
imagen.save(ruta_de_la_imagen)
```

1.2. ALGORITMO PARA EXTRAER ANGULOS DE LA CUADRICULA EN PYTHON

```
import numpy as np
import cv2

# Definición de variables para el patrón de calibración
patron_x = 8 # Número de puntos en dirección X
patron_y = 13 # Número de puntos en dirección Y
square_size = 0.5 # Dimensión de los cuadrados en cm

# Inicialización de listas para almacenar puntos en el mundo real y en la imagen
puntos_reales = []
puntos_img = []

# Creación de la matriz de coordenadas del patrón de calibración
patron_coord = np.zeros((patron_x * patron_y, 3), np.float32)
patron_coord[:, :2] = np.mgrid[0:patron_x, 0:patron_y].T.reshape(-1, 2) * square_size

# Ruta al archivo de imagen
fname = r"C:\Users\luisg\Desktop\tfg\pruebas\prueba_2(cuadricula)\25_grados_3.tiff"

# Lectura de la imagen
img = cv2.imread(fname)

# Conversión de la imagen a escala de grises
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Búsqueda de los puntos del patrón de calibración en la imagen
ret, corners = cv2.findChessboardCorners(gris, (patron_x, patron_y),
                                         flags=cv2.CALIB_CB_ADAPTIVE_THRESH +
cv2.CALIB_CB_NORMALIZE_IMAGE)
```

```

# Si se encontraron los puntos, se agregan a las listas correspondientes
if ret == True:
    puntos_reales.append(patron_coord)
    puntos_img.append(corners)

# Calibración de la cámara utilizando los puntos encontrados
ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(puntos_reales, puntos_img, gris.shape[:,-1],
None, None)

# Impresión de los resultados de la calibración
print('Matriz de la cámara:\n', mtx)
print('Coeficientes de distorsión:\n', dist)

# Cálculo de la matriz de extrínsecos
rmat, _ = cv2.Rodrigues(rvecs[0])
extrinsics = np.concatenate((rmat, tvecs[0]), axis=1)

print(extrinsics)

# Cálculo de ángulos de rotación en los tres ejes del espacio (ejes x, y, z)
theta_x = np.arctan2(extrinsics[2, 1], extrinsics[2, 2])
theta_y = np.arctan2(-extrinsics[2, 0], np.sqrt(extrinsics[2, 1]**2 + extrinsics[2, 2]**2))
theta_z = np.arctan2(extrinsics[1, 0], extrinsics[0, 0])

# Conversión de ángulos de radianes a grados
theta_x_deg = np.rad2deg(theta_x)
theta_y_deg = np.rad2deg(theta_y)
theta_z_deg = np.rad2deg(theta_z)

# Impresión de los ángulos de rotación
print("Ángulo de rotación en el eje x: {:.2f} grados".format(theta_x_deg))

```

```
print("Ángulo de rotación en el eje y: {:.2f} grados".format(theta_y_deg))  
print("Ángulo de rotación en el eje z: {:.2f} grados".format(theta_z_deg))
```

1.3. PROGRAMA QUE DETECTA UN RECTÁNGULO NEGRO CON UN AREA MÍNIMA, DETERMINA LA POSICIÓN DE SUS VERTICES EN PÍXELES DENTRO DE LA IMAGEN Y APLICA EL CAMBIO DE PERSPECTIVA AL MISMO.

```
import cv2

import numpy as np

import math

# Cargar la imagen

image =
cv2.imread(r"C:\Users\luisg\Desktop\tfg\pruebas\prueba_4(speckle)\Referencia\Referencia_2
0g.tiff")

# Crear y mostrar ventana con la imagen original

cv2.namedWindow("Bordes", cv2.WINDOW_NORMAL)

cv2.resizeWindow("Bordes", 600, 500)

cv2.imshow("Bordes", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Convertir a escala de grises y aplicar filtro Gaussiano

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

gray = cv2.GaussianBlur(gray, (5, 5), 0)

# Detectar bordes con Canny y mostrar resultado

edges = cv2.Canny(gray, 100, 200)

cv2.namedWindow("Bordes", cv2.WINDOW_NORMAL)

cv2.resizeWindow("Bordes", 600, 500)

cv2.imshow("Bordes", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

```

# Encontrar contornos en los bordes
contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Filtrar contornos por área
filtered_contours = []
min_area_threshold = 1000 # Ajustar según necesidad
for contour in contours:
    area = cv2.contourArea(contour)
    if area > min_area_threshold:
        filtered_contours.append(contour)

# Buscar contorno de la cuadrícula
grid_contour = None
for contour in filtered_contours:
    if len(contour) >= 4:
        perimeter = cv2.arcLength(contour, True)
        approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)
        if len(approx) == 4:
            grid_contour = approx
            break

print("Estos son los grid:", grid_contour)

# Ordenar los puntos de interés del contorno de la cuadrícula
points = grid_contour.reshape(4, 2)
points = np.float32(points)
points = points[np.argsort(points[:, 1])]
print(points)

# Obtener puntos superiores e inferiores
top_left = points[2][0], points[2][1]

```

```

bottom_left = points[1][0], points[1][1]
top_right = points[3][0], points[3][1]
bottom_right = points[0][0], points[0][1]

# Crear máscara y dibujar círculos rojos en los puntos de interés
mask = np.zeros_like(image)
for punto in points:
    x, y = punto.astype(int)
    cv2.circle(mask, (x, y), 5, (0, 0, 255), -1)

# Combinar imagen original y máscara, y mostrar resultado
result = cv2.add(image, mask)
cv2.namedWindow("Bordes", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Bordes", 600, 500)
cv2.imshow('Bordes', result)
cv2.waitKey(0)

# Convertir los puntos a numpy array y calcular matriz de transformación
points = np.array(points, dtype=np.float32)
h, w, _ = image.shape
print(h, w)
pts1 = np.float32([[bottom_left], [bottom_right], [top_left], [top_right]])
pts2 = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
M = cv2.getPerspectiveTransform(pts1, pts2)
print("Esta es la matriz de transformación:")
print(M)

# Aplicar la transformación a la imagen y dividirla en canales de color
result = cv2.warpPerspective(image, M, (image.shape[1], image.shape[0]))
b, g, r = cv2.split(result)

```

```
# Aplicar suavizado gaussiano al canal verde y unir canales
```

```
g = cv2.GaussianBlur(g, (5, 5), 0)
```

```
img_smoothed = cv2.merge((b, g, r))
```

```
result = img_smoothed
```

```
# Aplicar filtro de eliminación de ruido y guardar imagen resultante
```

```
resultado_final = cv2.fastNlMeansDenoisingColored(result, None, 10, 10, 7, 21)
```

```
ruta =
```

```
r"C:\Users\luisg\Desktop\tfg\pruebas\prueba_4(speckle)\Referencia\Prueba_nueva.tiff"
```

```
cv2.imwrite(ruta, resultado_final)
```

```
cv2.namedWindow("Bordes", cv2.WINDOW_NORMAL)
```

```
cv2.resizeWindow("Bordes", 500, 500)
```

```
cv2.imshow("Bordes", resultado_final)
```

```
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

1.4. INTERFAZ DE USUARIO QUE DETERMINA EL ANGULO DE GIRO A PARTIR DE UNA CUADRÍCULA Y LO MUESTRA POR PANTALLA.

```
import tkinter as tk
from tkinter import filedialog, Text
import cv2
from PIL import Image, ImageTk
import numpy as np

class AngleFinderGUI:
    def __init__(self, root):
        # Inicialización de la interfaz gráfica

        self.root = root

        self.root.title("Angle Finder")

        # Creación de widgets

        self.image_label = tk.Label(self.root)

        self.image_label.grid(row=0, column=0, padx=10, pady=10)

        self.open_btn = tk.Button(self.root, text="Open Image", command=self.open_image)

        self.open_btn.grid(row=1, column=0, padx=10, pady=10)

        self.process_btn = tk.Button(self.root, text="Process Image",
command=self.process_image)

        self.process_btn.grid(row=2, column=0, padx=10, pady=10)

        self.console = Text(self.root, height=5, width=50)

        self.console.grid(row=3, column=0, padx=10, pady=10)

    def open_image(self):
        # Función para abrir una imagen

        file_path = filedialog.askopenfilename()
```

```

if file_path:

    self.image_path = file_path

    self.image = cv2.imread(file_path)

    self.display_image(self.image)

def display_image(self, image):

    # Función para mostrar una imagen en la interfaz

    image = self.resize_image(image, 400)

    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    pil_image = Image.fromarray(image_rgb)

    photo = ImageTk.PhotoImage(pil_image)

    self.image_label.config(image=photo)

    self.image_label.image = photo

def resize_image(self, image, width_target):

    # Función para redimensionar una imagen

    aspect_ratio = image.shape[1] / float(image.shape[0])

    new_width = width_target

    new_height = int(new_width / aspect_ratio)

    resized_image = cv2.resize(image, (new_width, new_height))

    return resized_image

def calcular_angulo(self, ruta_imagen):

    # Función para calcular el ángulo de rotación de una imagen

    patron_x = 8

    patron_y = 13

    square_size = 0.5

    patron_coord = np.zeros((patron_x * patron_y, 3), np.float32)

    patron_coord[:, :2] = np.mgrid[0:patron_x, 0:patron_y].T.reshape(-1, 2) * square_size

    img = cv2.imread(ruta_imagen)

```

```

gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

ret, corners = cv2.findChessboardCorners(gris, (patron_x, patron_y),
flags=cv2.CALIB_CB_ADAPTIVE_THRESH + cv2.CALIB_CB_NORMALIZE_IMAGE)

if ret == True:
    puntos_reales = [patron_coord]
    puntos_img = [corners]
else:
    return "No se pudo detectar el patrón en la imagen."

_, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(puntos_reales, puntos_img, gris.shape[:-1], None, None)

rmat, _ = cv2.Rodrigues(rvecs[0])
extrinsics = np.concatenate((rmat, tvecs[0]), axis=1)

theta_y = np.arctan2(-extrinsics[2, 0], np.sqrt(extrinsics[2, 1]**2 + extrinsics[2, 2]**2))
theta_y_deg = np.rad2deg(theta_y)

return f"Ángulo de rotación en el eje y: {theta_y_deg:.2f} grados"

def process_image(self):
    # Función para procesar la imagen y mostrar el resultado
    resultado = self.calcular_angulo(self.image_path)
    self.console.insert(tk.END, resultado + "\n")
    self.display_image(self.image)

# Creación de la instancia de Tkinter y ejecución de la GUI
root = tk.Tk()
gui = AngleFinderGUI(root)
root.mainloop()

```

1.5. INTERFAZ GRÁFICA QUE APLICA EL ALGORITMO DE CAMBIO DE PERSPECTIVA A UNA IMAGEN Y PERMITE GUARDAR UN CAMBIO DE PERSPECTIVA REALIZADO A UNA IMAGEN ANTERIOR PARA APLICARLO A LAS POSTERIORES IMÁGENES QUE SE PROCESAN.

```
import cv2

import tkinter as tk

from tkinter import filedialog, messagebox

from PIL import Image, ImageTk

import numpy as np

import math


class ImageProcessor(tk.Tk):

    def __init__(self):
        super().__init__()

        # Configuración de la ventana

        self.title("Image Processor")

        self.geometry("800x600")

        # Define el grid de la ventana principal

        self.grid_rowconfigure(0, weight=1)

        self.grid_columnconfigure(0, weight=1)

        self.grid_columnconfigure(1, weight=0) # Columna para botones no debe expandirse
        tanto como la imagen

        # Label para mostrar la imagen

        self.image_label = tk.Label(self)

        self.image_label.grid(row=0, column=0, sticky="nsew", padx=10, pady=10)
```

```

# Frame para los botones

buttons_frame = tk.Frame(self)

buttons_frame.grid(row=0, column=1, sticky="ns", padx=10, pady=10)

# Botones dentro del Frame, alineados verticalmente usando pack

self.open_btn = tk.Button(buttons_frame, text="Open Image",
command=self.open_image)

self.open_btn.pack(fill="x", pady=2)

self.process_btn = tk.Button(buttons_frame, text="Process Image",
command=self.process_image, state="disabled")

self.process_btn.pack(fill="x", pady=2)

self.save_btn = tk.Button(buttons_frame, text="Save Image", command=self.save_image,
state="disabled")

self.save_btn.pack(fill="x", pady=2)

self.apply_btn = tk.Button(buttons_frame, text="Apply to New Image",
command=self.apply_to_new_image, state="disabled")

self.apply_btn.pack(fill="x", pady=2)

# Atributo para almacenar la matriz de transformación

self.transformation_matrix = None

# Variables para las imágenes

self.image = None

self.processed_image = None

# Etiquetas de coordenadas debajo del frame de la imagen

self.coords_frame = tk.Frame(self)

self.coords_frame.grid(row=1, column=0, sticky="ew")

self.grid_rowconfigure(1, weight=0) # Esta fila no debe expandirse

```

```

self.coords_labels = {
    "top_left": tk.Label(self.coords_frame, text="Top Left: "),
    "top_right": tk.Label(self.coords_frame, text="Top Right: "),
    "bottom_left": tk.Label(self.coords_frame, text="Bottom Left: "),
    "bottom_right": tk.Label(self.coords_frame, text="Bottom Right: ")
}

for i, label in enumerate(self.coords_labels.values()):
    label.grid(row=i, column=0, sticky="w")

```

```

def open_image(self):

```

```

    file_path = filedialog.askopenfilename()

    if file_path:
        self.image = cv2.imread(file_path)

        if self.image is not None:
            # Convertir de BGR a RGB
            image_rgb = cv2.cvtColor(self.image, cv2.COLOR_BGR2RGB)

            # Convertir la imagen de OpenCV a formato PIL
            pil_image = Image.fromarray(image_rgb)

            # Determinar el tamaño máximo para la imagen
            max_width = 600
            max_height = 400

            # Mantener la relación de aspecto
            aspect_ratio = pil_image.width / pil_image.height

            if pil_image.width > max_width or pil_image.height > max_height:
                if pil_image.width > pil_image.height:

```

```

        pil_image = pil_image.resize((max_width, int(max_width / aspect_ratio)))
    else:
        pil_image = pil_image.resize((int(max_height * aspect_ratio), max_height))

    # Convertir la imagen PIL a formato Tkinter
    self.tk_image = ImageTk.PhotoImage(pil_image)

    # Mostrar la imagen en el label
    self.image_label.config(image=self.tk_image)
    self.process_btn["state"] = "normal"

def process_image(self):
    if self.image is not None:

        gray = cv2.cvtColor(self.image, cv2.COLOR_BGR2GRAY)
        gray = cv2.GaussianBlur(gray, (5, 5), 0)

        edges = cv2.Canny(gray, 100, 200)

        contours, _ = cv2.findContours(edges, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
        filtered_contours = []
        min_area_threshold = 1000

        for contour in contours:
            area = cv2.contourArea(contour)
            if area > min_area_threshold:
                filtered_contours.append(contour)

        grid_contour = None
        for contour in filtered_contours:

```

```

if len(contour) >= 4:
    perimeter = cv2.arcLength(contour, True)
    approx = cv2.approxPolyDP(contour, 0.02 * perimeter, True)
    if len(approx) == 4:
        grid_contour = approx
        break

points = grid_contour.reshape(4, 2)
points = np.float32(points)
points = points[np.argsort(points[:, 1])]

top_left = points[2][0], points[2][1]
bottom_left = points[0][0], points[0][1]
top_right = points[3][0], points[3][1]
bottom_right = points[1][0], points[1][1]

self.update_coords_labels([bottom_left, bottom_right, top_left, top_right])

mask = np.zeros_like(self.image)

for punto in points:
    x, y = punto.astype(int)
    cv2.circle(mask, (x, y), 5, (0, 0, 255), -1)

self.processed_image = cv2.add(self.image, mask)
points = np.array(points, dtype=np.float32)

h, w, _ = self.image.shape
pts1 = np.float32([[bottom_left], [bottom_right], [top_left], [top_right]])
pts2 = np.float32([[0, 0], [w, 0], [0, h], [w, h]])
M = cv2.getPerspectiveTransform(pts1, pts2)

```

```

self.transformation_matrix = M # Guarda la matriz de transformación

self.apply_btn["state"] = "normal" # Habilita el botón para aplicar a nuevas imágenes


result = cv2.warpPerspective(self.image, M, (self.image.shape[1], self.image.shape[0]))
b, g, r = cv2.split(result)
g = cv2.GaussianBlur(g, (5, 5), 0)
img_smoothed = cv2.merge((b, g, r))
result = img_smoothed

self.processed_image = cv2.fastNlMeansDenoisingColored(result, None, 10, 10, 7, 21)


self.save_btn["state"] = "normal"
messagebox.showinfo("Success", "Image processed successfully!")

# Suponemos que 'resultado_final' es la imagen procesada al final de tu código
# Convertir de BGR a RGB
image_rgb = cv2.cvtColor(result, cv2.COLOR_BGR2RGB)


# Convertir la imagen de OpenCV a formato PIL
pil_image = Image.fromarray(image_rgb)


# Determinar el tamaño máximo para la imagen
max_width = 600
max_height = 400


# Mantener la relación de aspecto
aspect_ratio = pil_image.width / pil_image.height


if pil_image.width > max_width or pil_image.height > max_height:
    if pil_image.width / pil_image.height > aspect_ratio:
        # La imagen es más ancha que alta
        new_width = max_width

```

```

        new_height = int(new_width / aspect_ratio)
    else:
        # La imagen es más alta que ancha
        new_height = max_height
        new_width = int(new_height * aspect_ratio)

    pil_image = pil_image.resize((new_width, new_height))

    # Convertir la imagen PIL a formato Tkinter
    self.tk_image = ImageTk.PhotoImage(pil_image)

    # Mostrar la imagen en el label
    self.image_label.config(image=self.tk_image)
    self.save_btn["state"] = "normal"

def apply_to_new_image(self):
    file_path = filedialog.askopenfilename()

    if file_path and self.transformation_matrix is not None:
        # Cargar la nueva imagen
        new_image = cv2.imread(file_path)

        # Aplicar la transformación de perspectiva
        transformed_image = cv2.warpPerspective(new_image, self.transformation_matrix,
        (new_image.shape[1], new_image.shape[0]))

        # Actualiza la imagen procesada
        self.processed_image = transformed_image

        # Convertir de BGR a RGB
        image_rgb = cv2.cvtColor(transformed_image, cv2.COLOR_BGR2RGB)

```

```

# Convertir la imagen de OpenCV a formato PIL
pil_image = Image.fromarray(image_rgb)

# Determinar el tamaño máximo para la imagen
max_width, max_height = 600, 400

# Mantener la relación de aspecto
aspect_ratio = pil_image.width / pil_image.height
if pil_image.width > max_width or pil_image.height > max_height:
    if pil_image.width > pil_image.height:
        pil_image = pil_image.resize((max_width, int(max_width / aspect_ratio)))
    else:
        pil_image = pil_image.resize((int(max_height * aspect_ratio), max_height))

# Convertir la imagen PIL a formato Tkinter
self.tk_image = ImageTk.PhotoImage(pil_image)

# Mostrar la imagen en el label
self.image_label.config(image=self.tk_image)

def update_coords_labels(self, points):
    # Actualiza las etiquetas con las coordenadas de los puntos
    labels_order = ["bottom_left", "bottom_right", "top_left", "top_right"]
    for label, point in zip(labels_order, points):
        coords_text = f"{label}: ({point[0]}, {point[1]})"
        self.coords_labels[label].config(text=coords_text)

def save_image(self):
    file_path = filedialog.asksaveasfilename(defaultextension=".png", filetypes=[("PNG files",
    "*.png"), ("JPEG files", "*.jpg"), ("All files", "*.*")])
    if file_path and self.processed_image is not None:

```

```
# Guarda la imagen procesada o transformada  
cv2.imwrite(file_path, self.processed_image)  
messagebox.showinfo("Success", "Image saved successfully!")
```

```
if __name__ == "__main__":  
    app = ImageProcessor()  
    app.mainloop()
```

1.6 INSTALACIÓN DE NCORR Y PROCESO PARA OBTENER LOS DESPLAZAMIENTOS

Para realizar la correlación digital de imágenes 2D se ha empleado Ncorr. Se trata de un programa de Matlab de código abierto. Su ejecución se lleva a cabo dentro del entorno de Matlab. Su funcionamiento es relativamente intuitivo, y los algoritmos de cálculo intensivo que emplea se optimizan mediante el uso de C++/MEX. [38]

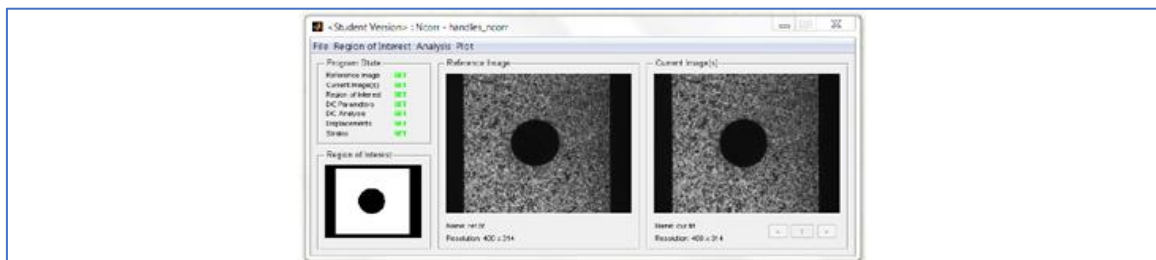


Ilustración 1. Captura tomada de la GUI de NCORR.

1. Correlación Digital 2D:

- **Instalación del módulo Ncorr en Matlab, e inicio del programa.**

El primer paso por realizar sería instalar el módulo del programa en Matlab, Tras ello se abre Ncorr directamente desde la ventana de comandos de Matlab escribiendo “ncorr”.

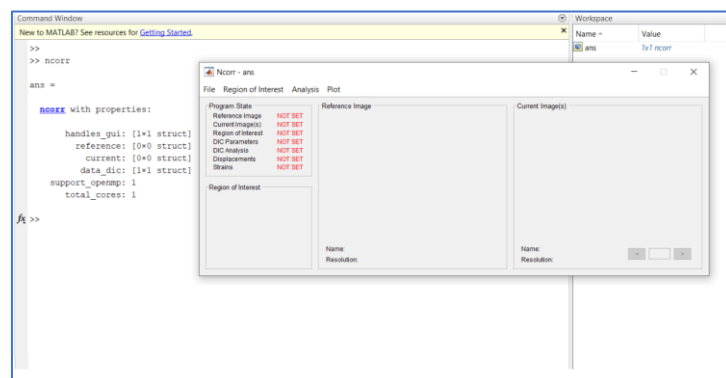


Ilustración 2: GUI de inicio de Ncorr [elaboración propia].

2. Pasos en Ncorr:

- **Cargar Imagen de Referencia:** Imagen tomada en la posición inicial antes de cualquier desplazamiento

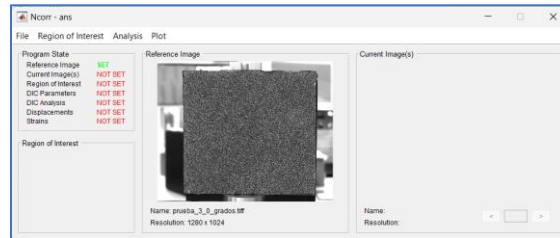


Ilustración 3: Captura del GUI con la imagen de referencia [elaboración propia].

- **Cargar Resto de Imágenes:** Utilizar el método “load lazy” para evitar el consumo excesivo de RAM.

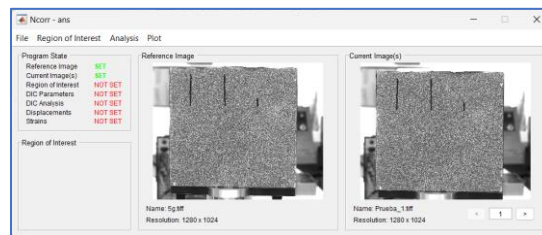


Ilustración 4: Captura del GUI con la imagen de referencia y el resto de las imágenes [elaboración propia].

- **Determinación de la Región de Interés (ROI):** Seleccionar la región de interés. Se utilizará el segundo método especificado en el manual de Ncorr que básicamente consiste en dibujar marcando mediante un polígono la región de interés. Para ello se accede a “Region of interest”, después “Set Reference ROI” y por último “Draw ROI”.

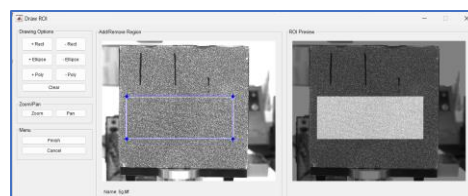


Ilustración 5: Creación del ROI [elaboración propia].

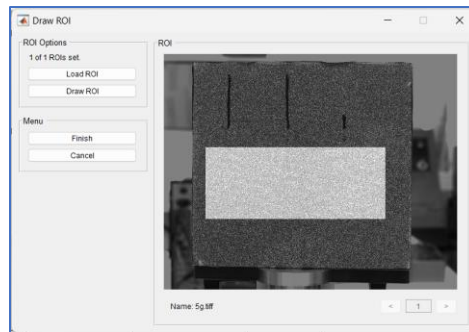


Ilustración 6: Resultado del ROI ya dibujado [elaboración propia].

- **Selección de Parámetros DIC:**

Para ello, se debe acceder a “analysis” y seleccionamos “set DIC parameters”

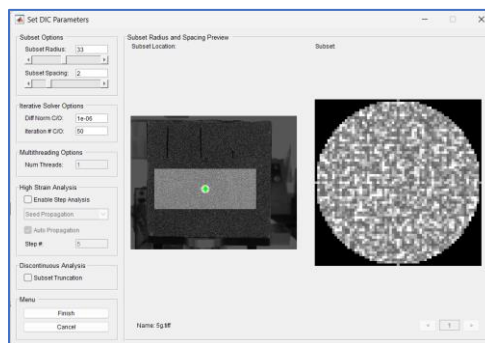


Ilustración 7: Selección de parámetros [elaboración propia].

En este paso se elige el tamaño de la faceta (subset radius). En general, se selecciona un tamaño de faceta (aprox 17-40 pixeles) que permita, al menos, la visualización de tres o cuatro puntos (aprox 5-7 pixles) del speckle. También se debe elegir el “subset spacing” o espaciado entre facetas, con lo que se indicará lo solapadas que se encuentra una faceta con respecto a su contigua (*Ilustración 7*) Por lo general, se mantiene el tamaño que viene prefijado, y debe establecerse espaciado de 5.

Después de la selección de parámetros, se accede a “analysis” y se selecciona “perform DIC analysis”. Aparecerá la región de interés y sobre ella se ubica una única semilla (*Ilustración 8*), dado que el ensayo no busca cálculos de tracción, ni de rotura él speckle. no sé dividirá en ningún momento.

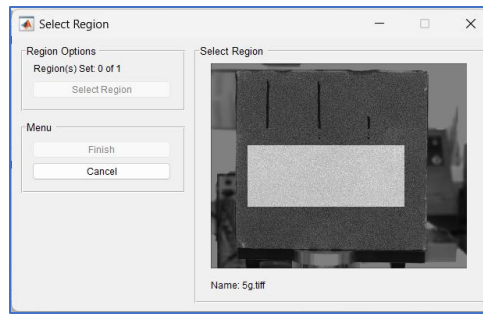


Ilustración 8: Selección de la semilla [elaboración propia].

La semilla colocada será seguida por el software en cada una de las imágenes, de modo que la correlación digital se podrá realizar correctamente. En ese momento aparecerá una ventana auxiliar en la que se puede ir visualizando la semilla en cada una de las imágenes, así como el coeficiente de correlación (un buen coeficiente de correlación presentará valores próximos a cero). También podremos ver el número de iteraciones que, por defecto, se ha fijado que como máximo sean 50; por lo que resultará beneficioso que el número de iteraciones sea bajo (lejano a 50).

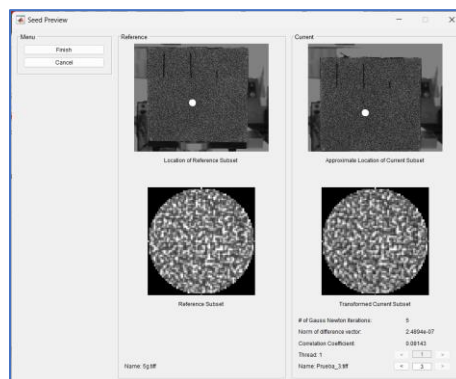


Ilustración 9: Ventana auxiliar [elaboración propia].

Las imágenes del ensayo se encuentran dentro de la normalidad para poder realizar un correcto estudio de las marcas del speckle, el número de iteraciones es bajo (entre 5-6), el coeficiente de correlación es próximo a cero (entre 0,2-0,08) y el software ha sido capaz de seguir la semilla.

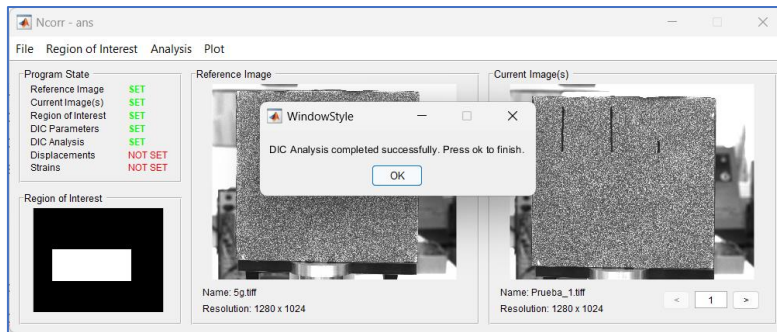


Ilustración 10: Ensayo realizado con éxito [elaboración propia].

- **Cálculo de los desplazamientos:**

Para realizar el cálculo de los desplazamientos se debe acceder a “analysis” y seleccionar “format displacements”. Debe establecerse la relación mm/px, para lo cual, debe pulsarse sobre “get unit conversion”. Así se cargará la imagen de referencia y al pulsar sobre “set line”, se dibujará una recta sobre una dimensión conocida de la probeta, en este caso se introducirá el ancho.

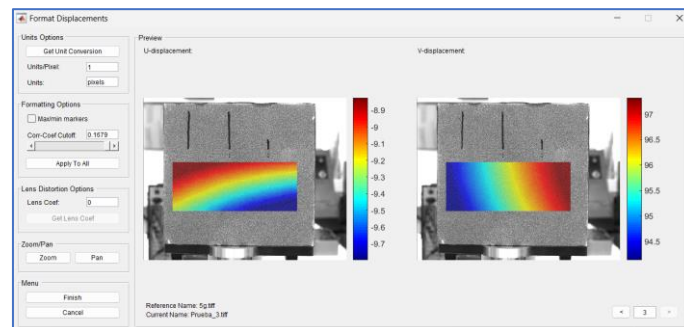


Ilustración 11: Imagen previa a introducir la relación [elaboración propia].

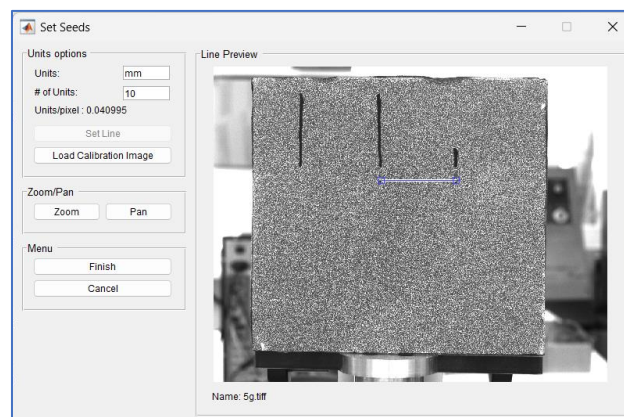


Ilustración 12: Imagen que enseña cómo se dibuja la línea para introducir la relación [elaboración propia].

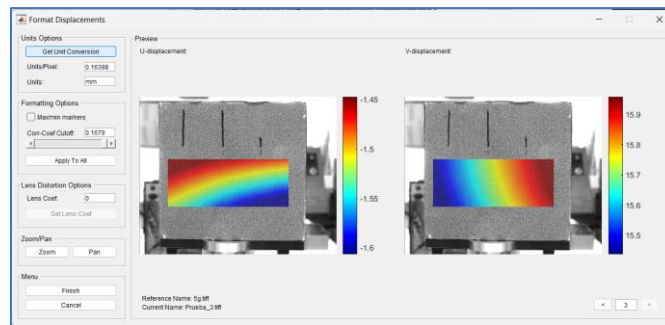
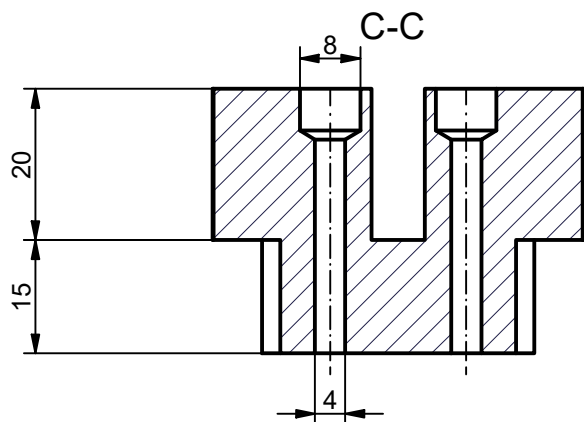
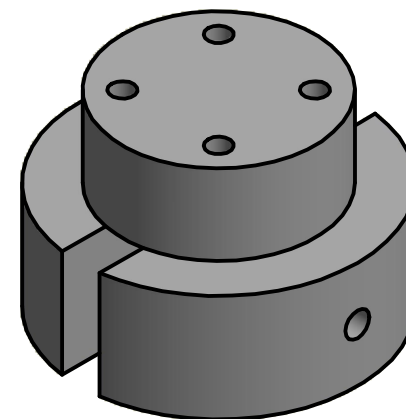
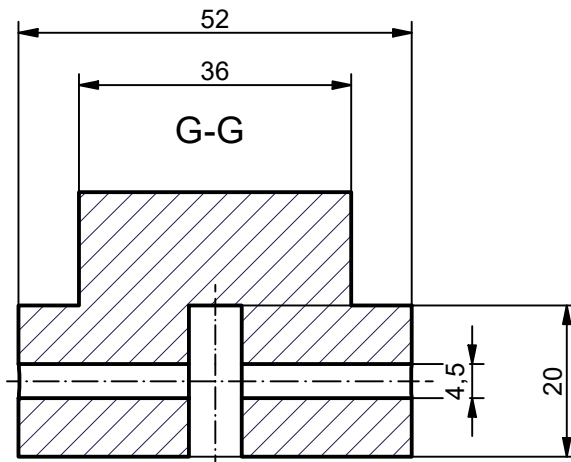
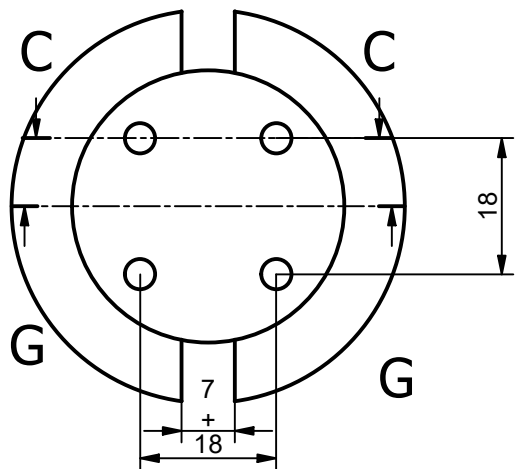


Ilustración 13: Imagen pos-calibrada [elaboración propia].

BIBLIOGRAFÍA DE Ncorr

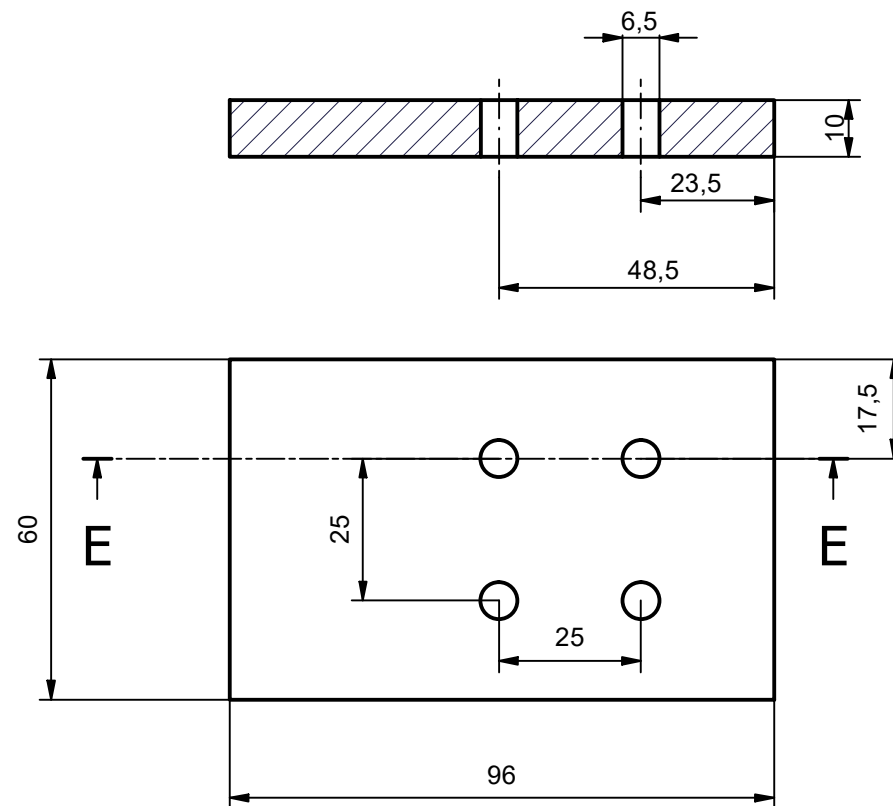
BLABER, J. Ncorr v1.2. Disponible en: <http://www.ncorr.com/>

1.7 PLANOS

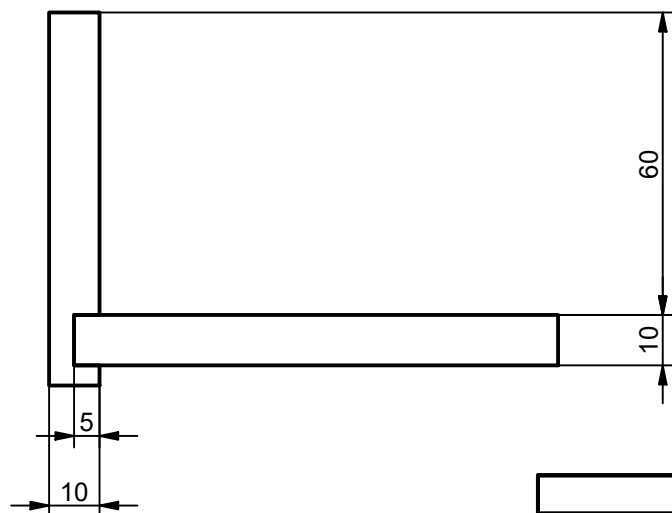
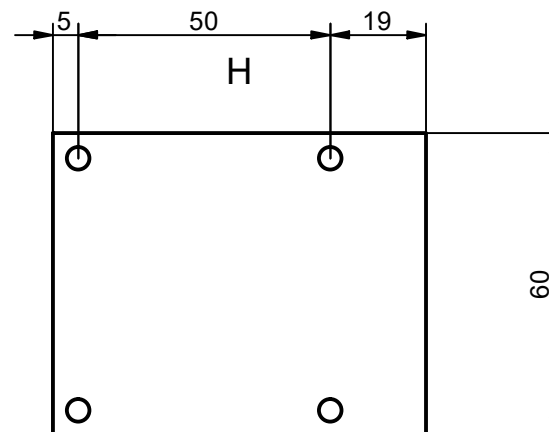
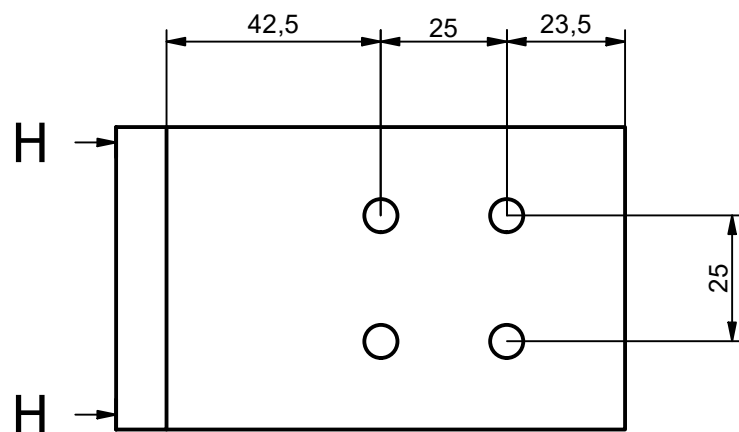


	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 1 / 7
				SUSTITUYE A
				SUSTITUIDO POR

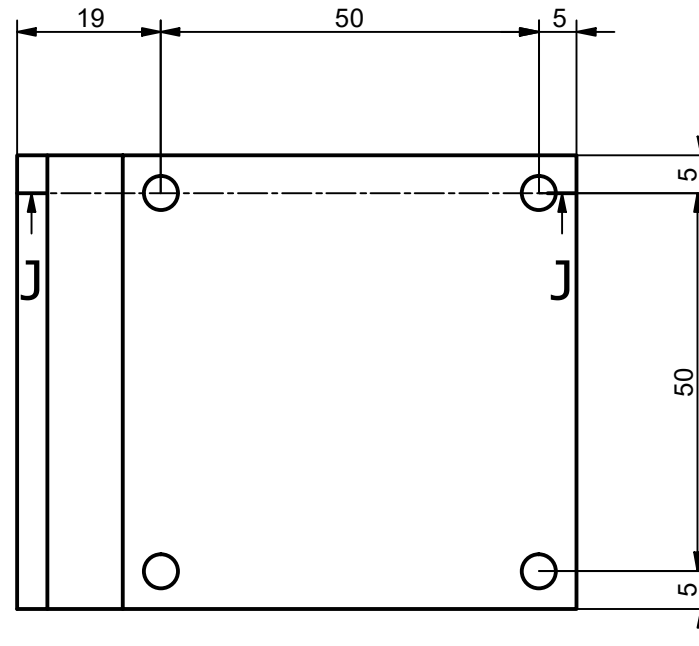
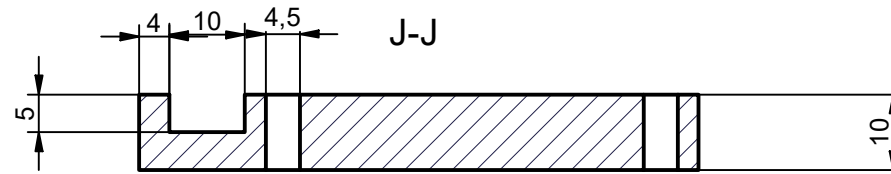
E-E



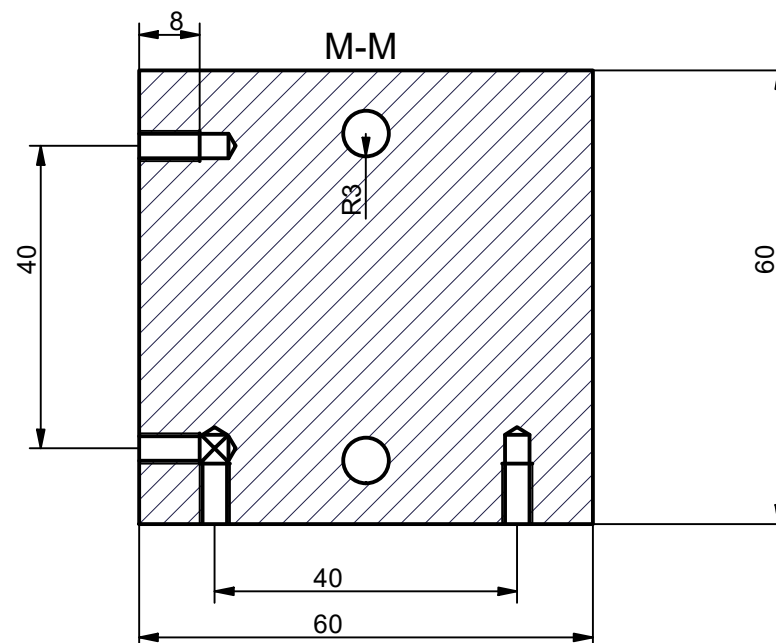
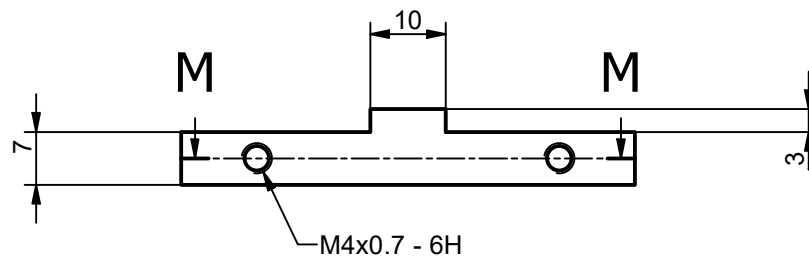
	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 2:3	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 2 / 7
				SUSTITUYE A
				SUSTITUIDO POR



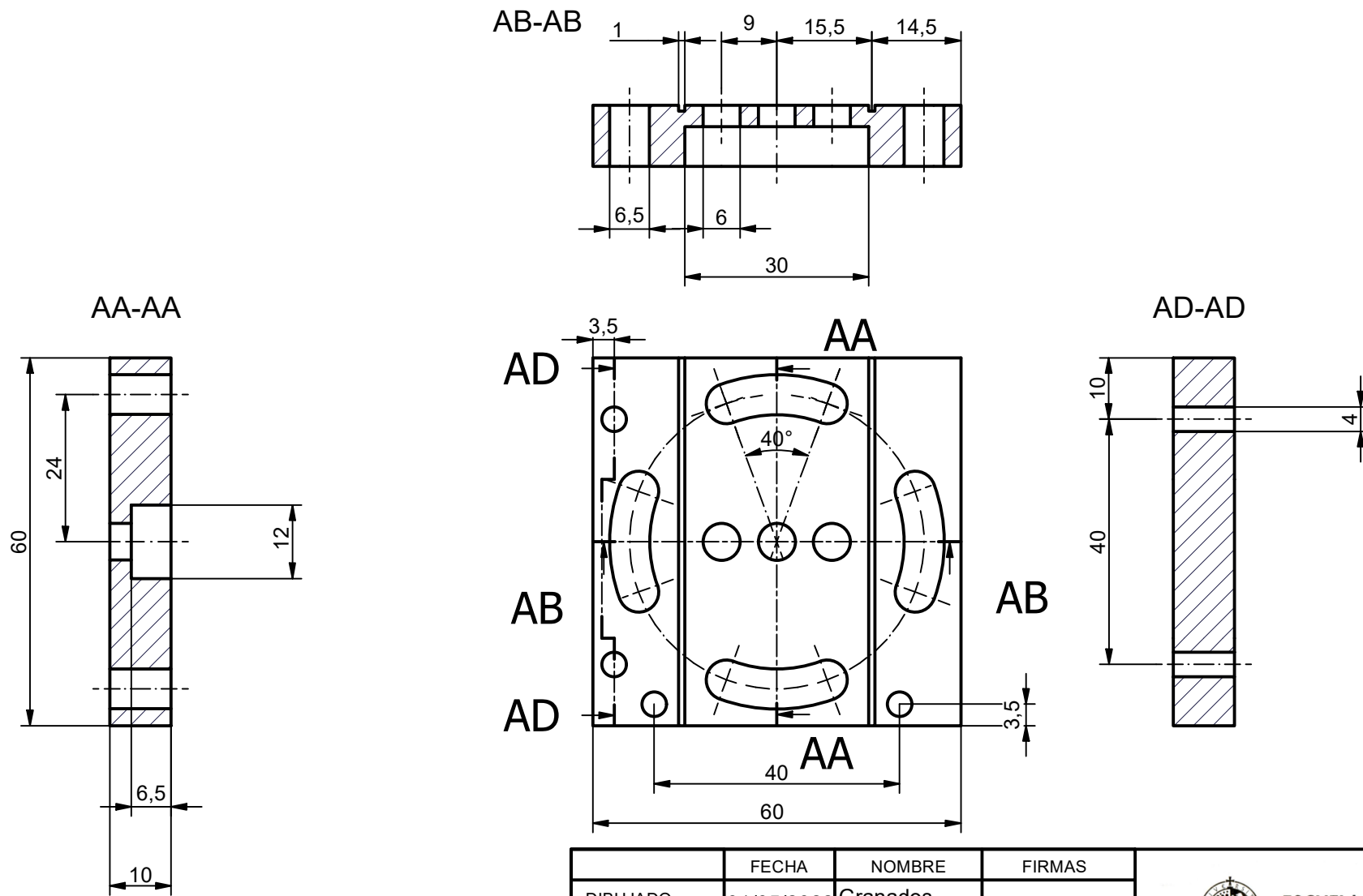
	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 3 / 7
				SUSTITUYE A
				SUSTITUIDO POR



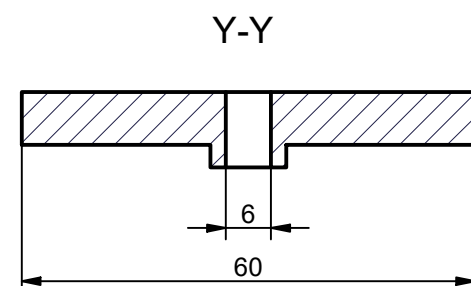
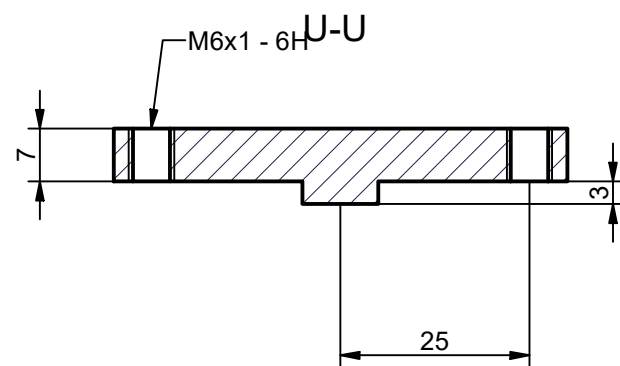
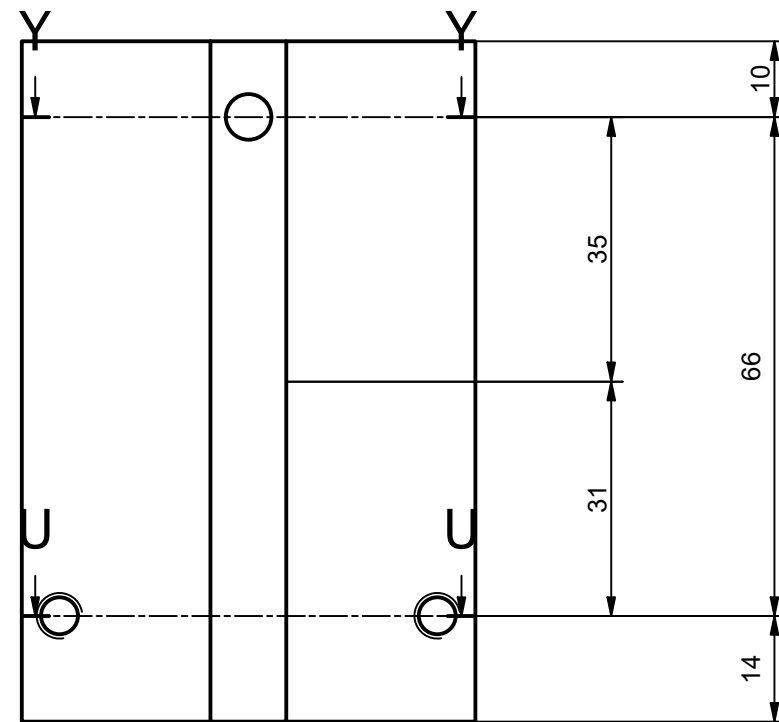
	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 4 / 7
				SUSTITUYE A
				SUSTITUIDO POR



	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 5 / 7
				SUSTITUYE A
				SUSTITUIDO POR



	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 6 / 7
				SUSTITUYE A
				SUSTITUIDO POR



	FECHA	NOMBRE	FIRMAS	 ESCUELA POLITÉCNICA SUPERIOR LINARES UNIVERSIDAD DE JAÉN
DIBUJADO	01/05/2022	Granados		
COMPROBADO		Fernández,		
C. CALIDAD		Luis		
ESCALA 1:1	DESIGNACIÓN TRABAJO DE FIN DE GRADO			Nº DE PLANO 7 / 7
				SUSTITUYE A
				SUSTITUIDO POR