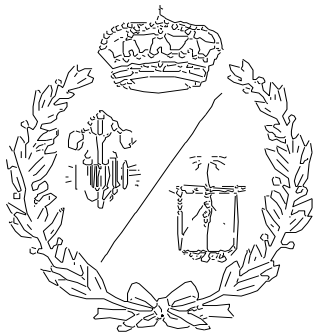
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

METROLOGÍA DE PIEZAS A TRAVÉS DE IMÁGENES DE ESCÁNER 2D

(Metrology of Parts Through 2D Scanner Images)

Para acceder al Título de

GRADUADA EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Helena Ruiz Fernández

Septiembre - 2025

Agradecimientos

En estos cuatro años de carrera he tenido la suerte de estar rodeada de personas que han marcado una huella muy positiva en mi vida.

En primer lugar, quiero agradecer a mi tutor, Iñigo Ugarte, por su dedicación, su orientación y por todos los consejos que me han guiado durante el desarrollo de este trabajo.

También quiero dar las gracias a mis amigos Ángela y Vicen, que en este último año me han aportado tanto en tan poco tiempo y me han acompañado en los momentos más importantes de esta última etapa de la carrera.

A mis compañeros de clase de estos cuatro años, Nacho, Rober, Juan y Kike, por todo lo compartido dentro y fuera del aula, y a Carlos, con quien, aunque solo he coincidido este último curso, he ganado una enorme confianza y amistad en muy poco tiempo.

Quiero dedicar unas palabras también a mis amigos y amigas del pueblo, con quienes he compartido mi vida desde siempre: algunos prácticamente desde que nacimos y otros desde el instituto, pero todos ellos han estado a mi lado de forma incondicional. Su apoyo constante y su amistad son una de mis mayores fortalezas.

Por último, eternamente agradecida a mi familia: a mis padres, por su apoyo incondicional y por acompañarme incluso en mis momentos más difíciles y a mi hermano, por su ayuda en los estudios, por compartir conmigo la experiencia de ser ingeniero y por entenderme en todos los aspectos relacionados con la carrera. Este logro no habría sido posible sin vosotros.

A todos, muchas gracias.

RESUMEN

El objetivo de este Trabajo de Fin de Grado es realizar una prueba de funcionamiento en el desarrollo de un algoritmo implementado en Python, que permita determinar las dimensiones de una pieza con una precisión de hasta 10 µm a partir de imágenes obtenidas mediante un escáner 2D.

En primer lugar, se incluye un apartado teórico que contextualiza históricamente la evolución de la metrología, el procesamiento de imágenes y las herramientas de software utilizadas en este ámbito.

A continuación, se presenta un análisis de los distintos tipos de escáneres, comenzando por los modelos 2D y posteriormente los 3D. Se realiza una comparativa entre ambas tecnologías, evaluando sus ventajas, limitaciones y aplicaciones, permitiendo justificar la elección del escáner 2D para el desarrollo del proyecto. Posteriormente, se detallan los parámetros técnicos que se deben considerar para su correcta selección, así como el modelo finalmente elegido. Además, se describen los posibles procesos de calibración que se pueden realizar con el objetivo de asegurar la precisión de las medidas y cuál de ellos se ha escogido.

Posteriormente, se justifica la elección de Python como lenguaje de programación frente a otras alternativas, destacando sus ventajas en términos de simplicidad, versatilidad y disponibilidad de herramientas para el procesamiento de imágenes. Asimismo, se incluye la descripción de las principales librerías empleadas en el desarrollo del código y su función dentro del proyecto.

Después, se trata el desarrollo del código implementado en Python, describiendo el planteamiento seguido para resolver el problema de medición de piezas a partir de imágenes. Se expone la estructura del programa, detallando la organización del flujo de trabajo desde la captura de la imagen, el preprocesamiento necesario, la detección de figuras geométricas y, finalmente, la extracción de las dimensiones con alta precisión.

Por último, en los capítulos finales se presentan los resultados obtenidos tras la ejecución del programa, incluyendo un análisis de la precisión y la fiabilidad de las medidas; el presupuesto estimado para la implementación del proyecto; y las conclusiones generales, donde se evalúa el cumplimiento de los objetivos planteados y se proponen posibles futuras mejoras.

ABSTRACT

The objective of this Final Degree Project is to carry out a functionality test, consisting of the development of an algorithm implemented in Python that allows determining the dimensions of a piece with a precision of 10 µm. To perform the measurements, the pieces are scanned using a 2D scanner.

The project begins with a theoretical section that provides a historical overview of the evolution of metrology, image processing techniques and the software tools commonly used in this field.

Next, an analysis of different types of scanners is presented, starting with 2D models and then 3D models. A comparison between both technologies is carried out, evaluating their advantages, limitations and applications, which justifies the choice of the 2D scanner for the development of the project. The technical parameters to consider for its proper selection are then detailed, along with the final model selected. Additionally, the possible calibration processes are described, indicating which one was chosen to ensure the precision of the measurements.

Subsequently, the choice of Python as the programming language is justified over other alternatives, highlighting its advantages in terms of simplicity, versatility and the availability of tools for image processing. A description of the main libraries used in the code and their role within the project is also included.

Afterwards, the development of the Python code is addressed, explaining the approach taken to solve the problem of measuring pieces from images. The structure of the program is outlined, detailing the organization of the workflow from image acquisition, necessary preprocessing, geometric shape detection to the extraction of dimensions with high precision.

Finally, the last chapters present the results obtained after executing the program, including an analysis of the precision and reliability of the measurements; the estimated budget for the implementation of the project; and the general conclusions, which assess the achievement of the proposed objectives and suggest possible future improvements.

Índice

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

La metrología se define como la "ciencia que tiene por objeto el estudio de los sistemas de pesas y medidas" [1] y fue actualizada en 2012 por el Vocabulario Internacional de Metrología como la "ciencia de las mediciones y sus aplicaciones" [2]. Aunque existen diversas definiciones, estas dos serán las adoptadas como referencia en este trabajo.

En las últimas décadas, los avances tecnológicos han impulsado el desarrollo de herramientas capaces de extraer y analizar información a partir de imágenes. Entre estas tecnologías destacan los escáneres 2D y 3D, que permiten capturar con alta precisión las dimensiones y características de piezas físicas. Sin embargo, también existen métodos de medición no ópticos, como el calibre, el pie de rey o los equipos de contacto, que obtienen las dimensiones al tocar directamente la superficie del objeto. Aunque estos instrumentos son ampliamente utilizados en entornos industriales, presentan limitaciones importantes: su resolución mínima suele estar en el orden de milímetros y al aumentar la exigencia de precisión, tanto la complejidad como el coste del equipo se incrementan considerablemente.

Ante estas limitaciones, es necesario realizar la búsqueda de métodos alternativos que permitan obtener mediciones precisas de forma más accesible y eficiente. En este proyecto se propone, a modo de prueba de funcionamiento, una solución basada en el análisis de imágenes digitales, que permite calcular las dimensiones de objetos físicos sin necesidad de contacto, aprovechando los recursos de procesamiento de imágenes disponibles actualmente.

Gracias al desarrollo de librerías especializadas y entornos de programación accesibles, hoy es posible implementar soluciones de medición automatizadas con relativa facilidad. Los escáneres capturan imágenes de las piezas, que posteriormente son procesadas mediante software para transformar la información visual en datos que pueden ser interpretados por algoritmos. Estos algoritmos permiten determinar, con gran precisión, las dimensiones de los objetos analizados.

Python se ha consolidado como uno de los lenguajes de programación más utilizados en este ámbito, debido a su simplicidad, versatilidad y la disponibilidad de librerías orientadas al procesamiento de imágenes, como por ejemplo OpenCV. Al tratarse de un lenguaje de código abierto, tiene la ventaja de no necesitar licencias comerciales, lo que facilita su uso tanto en entornos académicos como profesionales.

Para el desarrollo del proyecto se ha elegido el entorno de desarrollo integrado (IDE) Spyder, diseñado para programación científica y análisis de datos en Python.

1.1. Objetivos

El objetivo principal de este proyecto es desarrollar una solución capaz de obtener medidas dimensionales con una alta precisión (en el orden de los 10 µm) a partir de imágenes capturadas mediante un escáner 2D.

Para alcanzar este objetivo, en primer lugar se seleccionará un escáner que ofrezca las carac-

terísticas técnicas necesarias para lograr la precisión requerida. Además, será necesario establecer un proceso de calibración que permita ajustar el equipo correctamente en cada captura de imágenes, garantizando la fiabilidad de los resultados obtenidos.

A continuación, se llevará a cabo el desarrollo de un algoritmo de programación que automatice la obtención de medidas a partir de las imágenes de las piezas escaneadas. Este algoritmo deberá ser capaz de identificar correctamente las figuras geométricas presentes en las imágenes y calcular sus dimensiones con la precisión exigida.

En cuanto a los objetivos personales del proyecto:

- Familiarizarme con un nuevo lenguaje de programación, Python, y mejorar el empleo de librerías especializadas.
- Adquirir experiencia en la investigación autónoma de temas técnicos.
- Mejorar la capacidad de identificar los parámetros más relevantes de un dispositivo en función de las necesidades del análisis.
- Mejorar la capacidad de estudiar las características del entorno de trabajo que pueden afectar en condiciones reales.
- Mejorar la capacidad de toma de decisiones entre los diferentes métodos disponibles, evaluando sus resultados en condiciones reales.
- Consolidar y ampliar los conocimientos adquiridos en asignaturas como “Industrial Robotics and Computer Vision”.

2. MARCO HISTÓRICO

2.1. Metrología

Antes de la implantación del Sistema Métrico Decimal, las mediciones se realizaban utilizando partes del cuerpo humano como referencia. Se usaban los pies o los pasos para medir terrenos, el codo para objetos que se podían sostener a la altura de los brazos y medidas más pequeñas como la palma, el dedo o la pulgada para longitudes reducidas. En la figura 2.6 se representan estas unidades tradicionales: la palma, el palmo, el dedo y la pulgada.

Con el tiempo, surgió la necesidad de establecer relaciones entre estas unidades, dando lugar a las primeras equivalencias. Por ejemplo, una palma se formaba de cuatro dedos; un pie equivalía a cuatro palmas; un codo aproximadamente a un pie y medio y un paso se correspondía con dos pies y medio. Para simplificar el sistema se decidió tomar el pie como unidad principal, expresando el resto de las medidas en función de esta.

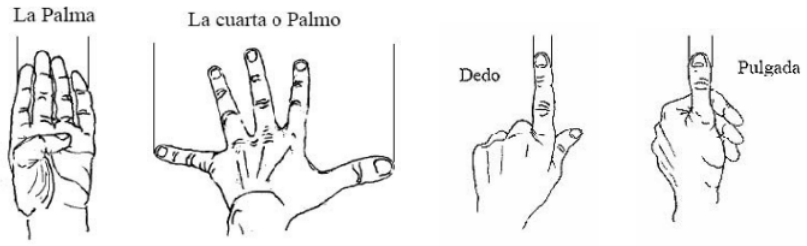


Figura 2.1: Primer sistema métrico. Fuente: [3]

Entre los siglos XV y XVIII, la metrología empezó a vincularse con el desarrollo científico. Los científicos de la época empezaron a exigir instrumentos capaces de proporcionar datos lo más exactos posibles, es decir, con el menor margen de error entre la medición obtenida y la realidad [3].

Durante el Renacimiento, el uso de la metrología estaba especialmente asociado a disciplinas como la astronomía y la geodesia (ciencia que estudia la forma y dimensiones de la Tierra). Sin embargo, figuras como Galileo introdujeron su aplicación para diferenciar entre propiedades medibles y no medibles de la materia. Más adelante, científicos como Descartes y Newton impulsaron su desarrollo debido a estudios que aplicaban conceptos metrológicos en fórmulas, cálculos y representaciones geométricas, en especial los triángulos, que permiten obtener mediciones más precisas.

Para poder comparar resultados de manera fiable, se recurrió a la creación de patrones físicos como forma de estandarizar las unidades. Estos patrones consistían en objetos que representaban medidas acordadas. Sin embargo, presentaban un problema importante: el desgaste con el uso, lo que alteraba su forma y, por tanto, alteraba la exactitud de las mediciones. Aunque se probaron diferentes materiales, todos presentaban algún tipo de variación con el tiempo.

En 1790, la Asamblea Nacional Francesa propuso el establecimiento de un sistema de medición uniforme, lo que dio lugar al nacimiento del sistema métrico decimal. La primera unidad definida fue el metro, establecido como la diez millonésima parte de la distancia entre el Polo Norte y el Ecuador siguiendo el meridiano de París. Jean-Baptiste-Joseph y Pierre Méchain fueron los científicos encargados de realizar las mediciones necesarias entre Dunkerque y Barcelona para establecer el patrón. Esta unidad fue transformada a un patrón físico, y aunque esta nueva unidad era más precisa, no eliminaba el problema del desgaste.

Posteriormente, en 1875, se firmó la Convención del Metro, que dio lugar a tres organismos clave en la metrología moderna: la Conferencia General de Pesas y Medidas (CGPM), el Buró Internacional de Pesas y Medidas (BIMP) y el Comité Internacional de Pesas y Medidas (CIMP). Su objetivo era perfeccionar y mantener un sistema internacional de unidades que permitiera realizar mediciones bajo las mismas condiciones en todo el mundo.

Gracias al trabajo de estos organismos, en 1899 se definió el kilogramo como la masa de un cilindro de platino e iridio conocido como el "Gran K", y el metro se representó mediante una barra del mismo material. Estos patrones fueron distribuidos internacionalmente y permitieron asentar las bases del actual Sistema Internacional de Unidades (SI).

No fue hasta 1960 que se estableció el Sistema Internacional de Unidades (SI), compuesto inicialmente por seis unidades base, las cuales se pueden observar en la tabla 2.1:

Tabla 2.1: Unidades base del Sistema Internacional de Unidades establecidas en 1960

Magnitud física	**Unidad**	**Símbolo**
Masa	Kilogramo	kg
Longitud	Metro	m
Tiempo	Segundo	s
Temperatura	Kelvin	K
Corriente eléctrica	Amperio	A
Intensidad luminosa	Candela	cd

En 1971 se añadió la séptima unidad base:

- Cantidad de sustancia: mol (mol).

La creación del SI supuso un avance fundamental en la metrología, ya que permitió unificar las mediciones a nivel internacional, facilitando el trabajo científico, industrial y técnico mediante un lenguaje común.

2.2. Escáneres

La idea de desarrollar un dispositivo capaz de digitalizar imágenes y textos surgió en el siglo XIX, como respuesta a la necesidad de automatizar la reproducción de documentos. A medida

que avanzaba la tecnología, empezaron a surgir distintos prototipos y sistemas que sentaron las bases de los escáneres actuales.

2.2.1. Teleautógrafo

Uno de los primeros avances significativos fue el teleautógrafo, inventado en 1888 por Elisha Gray. Este dispositivo permitía transmitir escritos y dibujos a través de líneas telegráficas, y aunque era muy rudimentario, se considera uno de los precursores del escáner moderno [4]. La figura 2.2 muestra cómo era este primer prototipo.

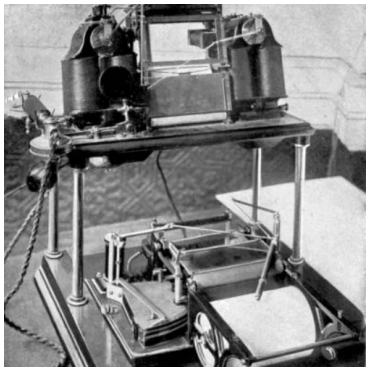


Figura 2.2: Teleautógrafo. Fuente: [4]

2.2.2. Cintas perforadas

Durante las décadas de 1920 a 1960, coincidiendo con la aparición de los primeros ordenadores, se produjo un gran avance en las tecnologías de escaneo de datos. En esta época, la información se almacenaba mediante cintas y tarjetas perforadas [5]. Estas consistían en tiras largas de papel en las que se hacían agujeros para codificar los datos, como se puede observar en la figura 2.3.

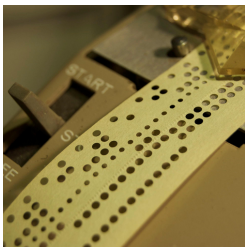


Figura 2.3: Cinta perforada. Fuente: [6]

2.2.3. Pantelégrafo

A finales del siglo XIX también se inventó el pantelégrafo, un dispositivo que utilizaba el telégrafo para enviar imágenes a distancia. Aunque era de gran tamaño y ofrecía una precisión limitada, supuso una evolución importante en la captura y transmisión electrónica de imágenes [7]. La figura 2.4 muestra este dispositivo.

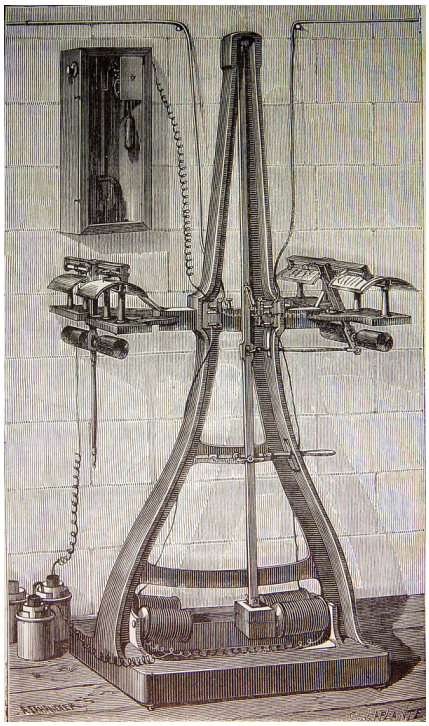


Figura 2.4: Pantelégrafo. Fuente: [7]

2.2.4. Escáner de Tambor

Entre las décadas 1950 y 1970, el avance de la informática y la electrónica permitió desarrollar dispositivos capaces de convertir imágenes analógicas en formato digital. En 1957, Russell A. Kirsch inventó uno de los primeros escáneres ópticos, que utilizaban un tambor giratorio junto a sensores fotoeléctricos, visible en la figura 2.5, para digitalizar las imágenes [8]. Este escáner permitió obtener la primera imagen digital de la historia: una fotografía del hijo de Kirsch, visible en la figura 2.6. Gracias a su alta resolución, este tipo de escáner se convirtió en una herramienta fundamental en la industria editorial y gráfica.

Figura 2.5: Escaner de Tambor. Fuente: [9]

Figura 2.6: Primera imagen digitalizada. Fuente: [8]

2.2.5. Escáner de código de barras

En la década de 1970 surgió el escáner de código de barras, desarrollado a raíz de la invención del propio código. El primer uso comercial documentado tuvo lugar en 1974, en un mercado de Ohio, lo que supuso una revolución en el ámbito logístico y comercial [10].

2.2.6. Escáner Plano

En los años 80, la empresa Microtek comercializó uno de los primeros escáneres planos destinados al entorno doméstico y de oficina [11]. Estos escáneres incorporaban sensores CCD (Charge-Coupled Device), que permitían capturar imágenes con buena precisión y color. Un ejemplo se muestra en la figura 2.7.

Figura 2.7: Escaner plano. Fuente: [12]

2.2.7. Escáneres modernos

Con el desarrollo de la tecnología digital, los escáneres han experimentado una evolución significativa, volviéndose más compactos, precisos y versátiles. Desde los años 2000, destacan tres avances principales: en primer lugar, los escáneres de alta resolución integrados en impresoras multifuncionales, que combinan las funciones de impresión, escaneado y copiado en un único dispositivo [13]; en segundo lugar, los escáneres 3D, diseñados para capturar con gran precisión objetos tridimensionales, han sido impulsados por el auge de la impresión 3D y la realidad virtual; y por último, los escáneres portátiles e inteligentes, que permiten digitalizar documentos mediante una fotografía obtenida desde dispositivos móviles. Estos últimos, gracias al uso de la inteligencia artificial, pueden mejorar automáticamente la calidad de los resultados obtenidos.

3. SELECCIÓN DEL ESCÁNER

Un escáner es un dispositivo cuya función principal consiste en convertir una imagen, documento u objeto físico en una representación digital. Para conseguirlo, utiliza sensores ópticos que capturan la luz reflejada sobre el objeto y transforman esta información en datos digitales.

3.1. Funcionamiento escáner 2D

En el caso de un escáner 2D, el proceso comienza cuando se introduce el documento, imagen u objeto que se desea digitalizar. Una vez iniciado el escaneo, una fuente de luz (que puede ser un LED, una lámpara fluorescente o de xenón) ilumina la superficie del elemento. Esta luz se refleja en las zonas más claras o sin tinta, mientras que las áreas con mayor saturación de color o tinta absorben parte de la luz emitida [14].

El sensor del escáner puede operar mediante dos tecnologías principales. Por un lado, el sistema CCD (Charge-Coupled Device), que emplea espejos y lentes para redirigir la luz hacia el sensor; por otro lado el sistema CIS (Contact Image Sensor), que capta la imagen de forma directa sin necesidad de elementos ópticos auxiliares. Aunque los sensores CIS permiten desarrollar escáneres más compactos, los CCD ofrecen una mejor calidad de imagen gracias a su mayor precisión en la captación de detalles y colores. Cabe destacar que, debido a su bajo coste y simplicidad de diseño, los sensores CIS se encuentran entre los más fabricados en la industria.

Una vez el sensor detecta la luz reflejada, convierte su intensidad en señales eléctricas. Estas señales se transforman en distintos niveles de voltaje, que se procesan mediante un convertidor analógico-digital (AD). El resultado son valores numéricos que representan los distintos píxeles, cada uno con su correspondiente nivel de brillo y color.

El software del escáner se encarga de organizar estos datos en una matriz de píxeles, lo que da como resultado la imagen digital final. En el caso de escaneos en blanco y negro, se emplea una escala de grises para representar los distintos niveles de luminosidad. Para imágenes de color, se utilizan filtros RGB (rojo, verde y azul), que permiten reconstruir los colores originales del objeto escaneado.

3.2. Funcionamiento escáner 3D

Un escáner 3D es un dispositivo capaz de capturar la geometría del mundo físico mediante el uso de tecnologías como láseres, luz estructurada o rayos X y generar representaciones digitales en forma de nubes de puntos densas o mallas poligonales. Su funcionamiento varía en función del alcance para el cual ha sido diseñado

3.2.1. Corto alcance

En el caso de los escáneres 3D de corto alcance, que operan a una distancias menores de un metro [15], se emplean tecnologías como la triangulación láser y la luz estructurada.

Escáneres 3D de triangulación láser

Los escáneres basados en triangulación láser proyectan una línea o punto láser sobre la superficie del objeto. La luz reflejada es captada por un sensor y mediante el análisis del ángulo de incidencia, el sistema calcula la distancia hasta el objeto aplicando principios de triangulación trigonométrica. Su funcionamiento se puede observar en la figura 3.1.

Con este método se logra obtener resultados de gran precisión y resolución, con la ventaja de generar poco ruido durante su funcionamiento. Sin embargo, presenta limitaciones como la necesidad de preparar la superficie, requerimientos específicos de iluminación, su difícil portabilidad y un diseño voluminoso.

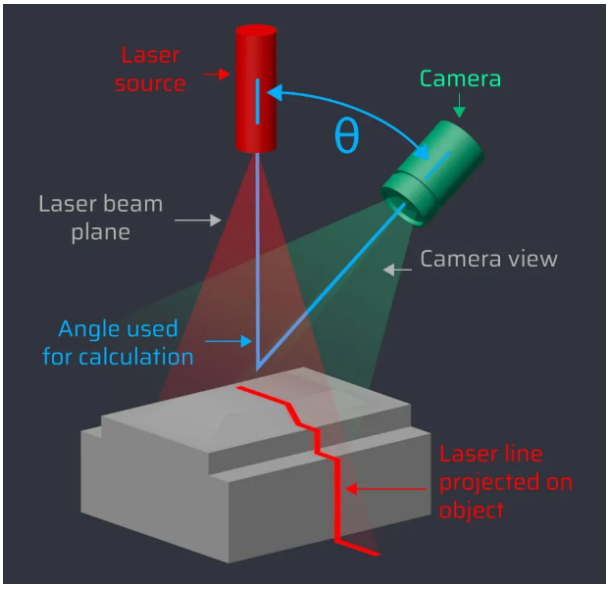


Figura 3.1: Escáner 3D triangulación láser. Fuente: [16]

Escáneres 3D de luz estructurada

Los escáneres de luz estructurada, también conocidos como de luz blanca o azul, proyectan patrones de líneas sobre el objeto. El sensor analiza las deformaciones de estos patrones y, mediante triangulación, determina la distancia al objeto. En la figura 3.2 se puede visualizar el funcionamiento de este tipo de escáner.

Esta tecnología ofrece mayor flexibilidad, ya que está disponible en múltiples formatos como escáneres portátiles, de brazo o de área. Además, no requiere iluminación específica y necesita menos preparación de la superficie. Sin embargo, ofrece una precisión y resolución menor y tiende a ser más ruidosa durante su uso.

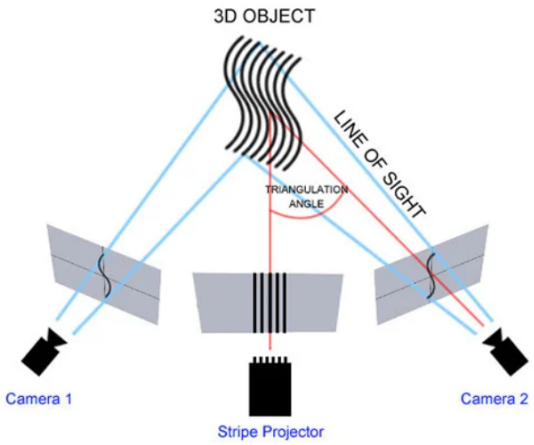


Figura 3.2: Escáner 3D luz estructurada. Fuente: [17]

3.2.2. Medio alcance

Cuando se trata de escaneos a mayor distancia, los escáneres de medio alcance, aquellos con una distancia focal de uno o más metros [15], emplean otras tecnologías como los escáneres láser de pulso y los de cambio de fase.

Escáneres láser de pulso 3D

Los escáneres láser de pulso, también conocidos como sistemas de tiempo de vuelo, calculan la distancia midiendo el tiempo que tarda un haz láser en alcanzar el objeto y reflejarse de vuelta al sensor. Al conocerse la velocidad de propagación del láser, es posible determinar la distancia recorrida. Para obtener una imagen completa, el sistema gira 360 grados alrededor del objeto. Esta técnica permite escanear objetos a distancias que van desde los 2 hasta los 1000 metros, aunque presenta desventajas como una menor precisión, adquisición de datos más lenta y mayor nivel de ruido.

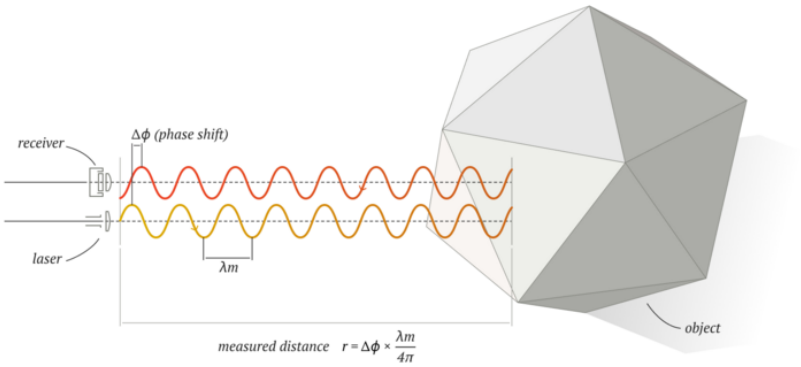


Figura 3.3: Escáner láser de pulso 3D. Fuente: [18]

Escáner láser de cambio de fase 3D

Los escáneres láser de cambio de fase también emplean el principio de tiempo de vuelo, pero añaden la modulación de la potencia del haz de láser. El sistema compara la fase del láser emitido con la fase del láser reflejado para calcular la distancia. Este método proporciona una mayor precisión y una adquisición de datos más rápida, además de operar con menor nivel de ruido. Sin embargo, su principal limitación es que solo se puede emplear en distancias medias, no pudiéndose adaptar para escaneos de largo alcance.

3.3. Comparativa entre escáner 2D y 3D

Los escáneres 2D emplean una iluminación uniforme durante todo el proceso de escaneo, utilizando LEDs o lámparas que proyectan luz sobre la superficie del objeto. Esta luz reflejada es captada por un sensor lineal que se desplaza a lo largo del área de trabajo, registrando la imagen línea por línea. El resultado es una imagen digital bidimensional de alta resolución, que puede alcanzar entre 600 y 4800 DPI. Además, esta tecnología ofrece una gran precisión en las dimensiones X e Y. Su funcionamiento es rápido, eficiente y más económico en comparación con otras tecnologías de escaneo.

Por otro lado, los escáneres 3D funcionan proyectando luz estructurada o un haz láser sobre el objeto y detectando las deformaciones del patrón reflejado mediante cámaras o sensores ópticos. A partir de estos datos se crea una nube de puntos que permite reconstruir un modelo tridimensional del objeto. Esta tecnología es ideal para capturar objetos voluminosos, con geometrías complejas, curvaturas o huecos, ya que no requiere contacto físico con la pieza y como resultado se obtiene una representación detallada en tres dimensiones. Sin embargo, el proceso es más complejo, requiere mayor procesamiento y suele implicar un coste más alto.

Teniendo en cuenta que el objetivo es medir con alta precisión detalles superficiales como ranuras sobre una superficie plana, la mejor opción es el escáner 2D. Este tipo de escáner proporciona una alta resolución y precisión en el plano, sin necesidad de capturar datos en el eje Z. Además, permite capturar imágenes de forma más rápida, sin distorsiones por profundidad y con un coste más reducido. Por estas razones, el escáner 2D es la opción más adecuada y eficiente para este tipo de trabajo.

3.4. Selección del escáner

Para seleccionar el escáner adecuado es necesario tener en cuenta tanto parámetros técnicos como aspectos prácticos relacionados con el uso específico del dispositivo.

3.4.1. Parámetros fundamentales

Resolución óptica

Entre los principales parámetros que se deben tener en cuenta se encuentra la resolución óptica, que indica la cantidad de puntos capturados por pulgada, medido en DPI (Dots Per Inch). Este aspecto es crucial para asegurar una correcta precisión. Para determinar la resolución óptica necesaria en función de la precisión deseada, se utiliza la relación de la ecuación 1.

$$DPI = \frac{25{,}4[mm]}{\text{precisión deseada}[mm]} \tag{1}$$

Si se desea alcanzar una precisión de 10 µm, es decir, 0,01 mm, la resolución necesaria se puede observar en la ecuación 2.

$$DPI = \frac{25{,}4}{0{,}01} = 2540DPI \tag{2}$$

Por tanto, para lograr capturar imágenes con una precisión de 10 µm, se requiere un escáner con una resolución óptica de al menos 2540 DPI. Los escáneres reales tienden a tener resoluciones de 2400 o 4800 DPI [19]. Por lo tanto, teniendo en cuenta la resolución mínima necesaria, se debe seleccionar un escáner con 4800 DPI.

Con un escáner de 4800 DPI se puede obtener una precisión de 5,29 µm, como se puede observar en la ecuación 3.

$$4800 = \frac{25{,}4}{\text{precisión deseada}[mm]} \quad \rightarrow \quad \text{precisión deseada} = 0{,}00529[mm] = 5{,}29[\mu m] \tag{3}$$

Tipo de sensor

El tipo de sensor es otro factor determinante. Los sensores CCD (Charge-Coupled Device) ofrecen una elevada sensibilidad a la luz, lo que se traduce en capturas con mayor detalle, mayor calidad de imagen, mejor reproducción de color y contraste, así como mayor profundidad de campo. Sin embargo, estos sensores suelen ser más caros y lentos. Son ideales para escanear superficies irregulares que no estén completamente en contacto con el cristal [20]. Por otra parte, los sensores CIS (Contact Image Sensor), basados en la tecnología CMOS, resultan más económicos y rápidos, con un diseño más compacto. No obstante, tienen menor sensibilidad y calidad de detalle y solo funcionan correctamente cuando el objeto se encuentra directamente en contacto con el cristal.

Teniendo en cuenta que se desea realizar mediciones de piezas planas, que van a estar en contacto completo con el cristal, un sensor CIS es adecuado.

Tamaño de escaneo

Otro parámetro relevante es el tamaño del área de escaneo. Este define las dimensiones máximas del objeto que puede ser capturado en una sola lectura. Las medidas estándar son A4 A4 (210x297 mm) y A3 (297x420mm). En este trabajo, teniendo en cuenta que las dimensiones no son excesivamente grandes, un escáner con formato A4 es suficiente. Además, es importante tener en cuenta que cuanto mayor sea el área de escaneo, mayor será también la cantidad de información capturada y por lo tanto los archivos serán de mayor tamaño y el tiempo de escaneo será mayor, especialmente si la resolución es alta también. Por esta razón, escoger un área de escaneo ajustada a las necesidades reales contribuye a mejorar el rendimiento del proceso.

Precisión dimensional

La precisión dimensional también resulta fundamental. Este parámetro indica la tolerancia al error en las mediciones. Para alcanzar los objetivos del proyecto, el escáner debe garantizar una precisión inferior a los 10 µm.

Compatibilidad con el software empleado

La compatibilidad con el software empleado representa otro criterio importante. El escáner debe ser capaz de exportar imágenes con alta resolución sin necesidad de ser comprimidas. Lo ideal es que genere archivos en formato TIFF (Tag Image File Format), ya sea como mapa de bits o en formato vectorial, permitiendo su utilización en programas como AutoCAD. Además, Python tiene librerías que permiten trabajar con este tipo de archivos, lo cual facilita el tratamiento posterior de las imágenes.

Velocidad de escaneo

Sobre la velocidad de escaneo, aunque no se considera un parámetro fundamental, una mayor rapidez de escaneo implica incrementar la eficiencia del proceso. Sin embargo, no consiste en un factor determinante en la elección del equipo.

Construcción física

Como último parámetro a tener en cuenta, es recomendable que el escáner tenga una construcción sólida y estable. Una base plana y sin vibraciones contribuye a escaneos consistentes y evita distorsiones en la digitalización.

3.4.2. Elección del escáner

Tras realizar un estudio de mercado enfocado en los parámetros técnicos descritos previamente, se han identificado tres modelos como posibles candidatos: Epson Perfection V19 [21], Canon CanoScan LiDE 400 [22] y Canon CanoScan LiDE 300 [23]. Estos escáneres han sido evaluados

en función de su resolución óptica, tipo de sensor, tamaño, compatibilidad con los diferentes formatos de salida, velocidad de escaneo y características físicas de soporte.

El modelo Epson Perfection V19 y el Canon CanoScan LiDE 400 disponen de una resolución óptica de 4800 DPI, mientras que el CanoScan LiDE 300 ofrece una resolución de 2400 DPI. Los tres modelos emplean sensores CIS, es decir, se consigue un óptimo resultado cuando el objeto se encuentra en contacto con el cristal. Respecto al tamaño del área de escaneo, los tres dispositivos permiten escanear documentos en formato A4.

Los tres escáneres soportan imágenes en JPG, PNG, TIFF y PDF. Respecto a la velocidad de escaneo varía entre los modelos de forma que el Epson Perfection V19 es el más lento, mientras que el Canon CanoScan LiDE 300 realiza el escaneo en 10 segundos y el LiDE 400 en aproximadamente 8 segundos. Acerca del soporte físico, el modelo Epson solo permite orientación horizontal, mientras que los modelos Epson permiten tanto horizontal como vertical, proporcionando mayor estabilidad.

Toda la información obtenida se resume en la tabla 3.1.

Tabla 3.1: Comparación de los escaneres

Parámetro	**Epson Perfection V19**	**CanoScan LiDE 300**	**CanoScan LiDE 400**
Resolución óptica (DPI)	4800	2400	4800
Sensor	CIS	CIS	CIS
Tamaño del escáner	A4	A4	A4
Formato de salida	JPG, PNG, TIFF y PDF	JPG, PNG, TIFF y PDF	JPG, PNG, TIFF y PDF
Velocidad	Lento	10 seg	8 seg
Soporte físico	Horizontal	Vertical y horizontal	Vertical y horizontal

Elección final

Considerando los parámetros fundamentales definidos y los tres modelos, el modelo **Canon CanoScan LiDE 400**, que se puede observar en la figura 3.4, es la opción más adecuada para este proyecto. En primer lugar, cumple con el requisito de resolución óptica, ya que dispone de 4800 DPI, permitiendo alcanzar una precisión de hasta 5,29 µm. Con esta resolución se supera la precisión mínima y se garantiza la fiabilidad de las medidas obtenidas.

En cuanto al sensor, utiliza la tecnología CIS, siendo compatible ya que las piezas van a estar en completo contacto con el cristal del escáner. Gracias a esta elección se logra reducir el coste sin alterar la calidad de los resultados.

El escáner permite tamaños hasta del formato A4, evitando archivos y tiempos de escaneo excesivamente grandes. Además, es compatible con formatos de archivo TIFF, facilitando el posterior tratamiento de las imágenes mediante las bibliotecas ya definidas de Python.

Este modelo también destaca por su velocidad de escaneo, siendo el más rápido entre los tres dispositivos, con un tiempo estimado de 8 segundos por escaneo. La rapidez de escaneo permite mejorar el rendimiento del proceso. Su soporte físico permite tanto una orientación horizontal como una vertical, permitiendo una mayor estabilidad durante el escaneo.

Así que, el Canon CanoScan LiDE 400 no solo cumple los requisitos técnicos fundamentales, sino que ofrece ventajas en términos de velocidad, formato de salida, construcción física y coste. Estas características lo convierte en la opción más eficiente y equilibrada para los requisitos del proyecto.

Figura 3.4: Escáner Canon CanoScan LiDE 400. Fuente: [22]

3.4.3. Calibración necesaria

Es necesario realizar una calibración porque un píxel en una imagen no tiene una unidad física establecida. Aunque el escáner establece un valor teórico de resolución en DPI, este se puede ver afectado por el procesamiento del software, variabilidad en la velocidad de escaneo, compresión de la imagen o los propios ajustes internos del dispositivo. Por lo tanto, para garantizar mediciones fiables, es imprescindible calibrar el sistema cada vez que se realice una captura de imagen.

Se han estudiado la fiabilidad de dos métodos para llevar a cabo la calibración. El primero consiste en escanear un patrón de referencia, como por ejemplo una hoja de cuadros blancos y negros de dimensiones conocidas e iguales. Al medir en la imagen escaneada cuántos píxeles tiene un cierto número de cuadros, se puede calcular cuánta es la longitud que equivaldría a un píxel. Sin embargo, este método tiene ciertas desventajas como que requiere de dos escaneos por separado (uno del patrón y otro del objeto a medir) y si en el proceso hay variaciones en la velocidad o en las condiciones de escaneo, se puede perder precisión, ya que la calibración no refleja el entorno del segundo escaneo.

La segunda opción consiste incluir, en el mismo escaneo del objeto a medir, otro objeto de dimensiones conocidas. De esta forma, se puede calcular la longitud que representa un píxel utilizando el objeto de referencia, y aplicar esa conversión en el resto de la imagen. Este método tiene como ventajas que se realiza en una única captura, se eliminan los errores asociados a los cambios entre los distintos escaneos y se obtiene una calibración fiable ya que ambos objetos han sido capturados bajo las mismas condiciones.

Como el proyecto requiere gran fiabilidad y precisión, se decidió combinar los dos métodos mencionados. En la misma captura de la pieza que se desea estudiar, se va a incluir una plantilla compuestas por cuadrados dispuestos en un patrón regular, tanto en el eje vertical como en el horizontal, cuyas dimensiones reales son conocidas. Esto permitirá obtener la relación píxel-milímetro en ambos ejes y además, verificar si las dimensiones sufren alteraciones a lo largo de los ejes, con el fin de aplicar correcciones ante las posibles distorsiones.

Actualmente, esta calibración se realiza mediante un marco diseñado en la aplicación Inkscape y posteriormente impreso, que sirve como patrón de referencia. No obstante, en un futuro, este marco impreso será reemplazado por una pieza rígida (fabricada en Rigid) similar a la que se puede observar en la figura 3.5, con la cuál se conservará el método de trabajo pero ofrecerá mayor durabilidad y se dispondrá de mayor espacio para colocar las piezas de estudio.

Figura 3.5: Futuro marco de calibración.

Al medir el número de píxeles que conforman los cuadrados, será posible calcular con gran exactitud la conversión a unidades físicas. Esta elección combina la principal ventaja del segundo método, asegurar la coherencia en las condiciones de escaneo, con la precisión y claridad que proporciona un patrón de referencia definido rigurosamente, como se establece en el primer método.

Creación del marco de calibración

Como se ha mencionado con anterioridad, el marco de calibración ha sido diseñado en la aplicación Inkscape, que consiste en un editor de gráficos vectoriales de código abierto. La figura 3.6 se corresponde al diseño final, cuya dimensión total es la del formato A4 (210 x 297 mm).

El diseño se organiza en diferentes secciones comenzando desde el borde exterior hacia el interior,

de la siguiente forma:

1. Márgenes exteriores: Se ha establecido un margen superior e inferior de 0,85 mm y un margen lateral de 0,5 mm, con la finalidad de asegurar la correcta impresión y el futuro escaneo sin perder información en los límites del marco.
2. Patrón cuadriculado: Se dispone de una serie de cuadrados en blanco y negro con una medida de 2 x 2 cm, localizados de manera alterna. Este patrón es la referencia de calibración, ya que se conocen sus dimensiones reales y permite establecer la relación entre píxeles y milímetros en la imagen escaneada.
3. Márgenes internos: Después del patrón, se han establecido nuevos márgenes de 0,5 mm en los cuatro lados, permitiendo delimitar el área del marco de calibración.
4. Zona de trabajo: En el interior se encuentra un rectángulo de 23 x 18 cm, simulando el espacio destinado a colocar las piezas que se desean estudiar.

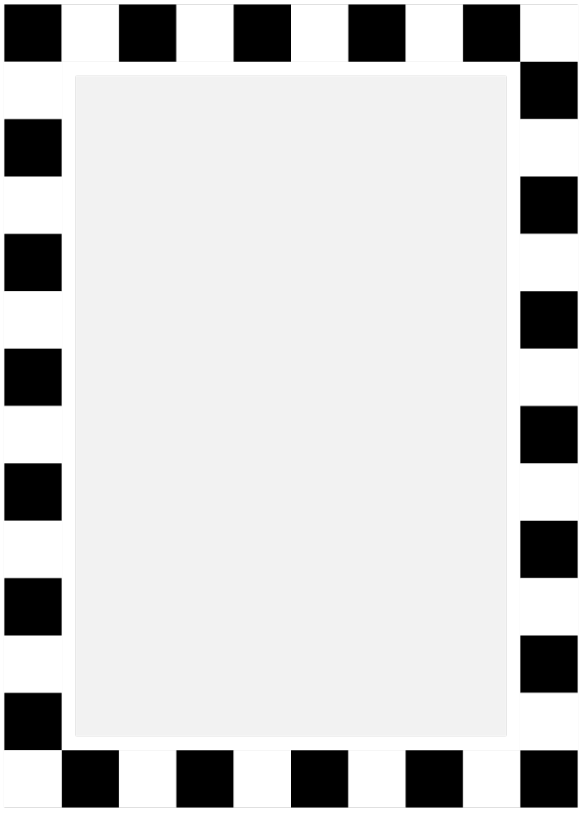


Figura 3.6: Actual marco de calibración.

4. SOFTWARE EMPLEADO

En esta sección se describe el software empleado para llevar a cabo el proyecto, incluyendo tanto la justificación de la elección del lenguaje de programación como las herramientas específicas que facilitan su implementación. En primer lugar, se detallan las razones por las que Python resulta el lenguaje idóneo para las necesidades del trabajo, teniendo en cuenta sus características, ventajas y las facilidades que proporciona para el procesamiento de imágenes y análisis de datos. Posteriormente, se detallan las librerías y módulos empleados, destacando las funciones que realizan dentro del programa, desde la manipulación y análisis de la captura hasta los métodos para visualizar los resultados, la interacción con el usuario y la gestión de datos.

4.1. Selección del lenguaje

Python es un lenguaje de programación desarrollado a principios de la década de 1990 por Guido van Rossum. Aunque comparte ciertas características con Perl,se distingue por una sintaxis más simple y clara, lo que favorece la legibilidad y el mantenimiento del código [24]. Se trata de un lenguaje interpretado o de script, lo que significa que su ejecución se realiza línea a línea a través de un intérprete, sin generar un archivo ejecutable independiente. Esto implica que, para poder ejecutarlo, es necesario disponer de dicho intérprete instalado. A cambio, este enfoque ofrece una mayor flexibilidad y facilita la depuración del código. Por el contrario, los lenguajes compilados convierten el código fuente en instrucciones de máquina mediante un compilador, generando un archivo ejecutable que el procesador puede ejecutar directamente.

Si bien los lenguajes compilados suelen ofrecer un mejor rendimiento en cuanto a velocidad de ejecución, los interpretados como Python destacan por su sencillez en el desarrollo, portabilidad y versatilidad. Además, Python optimiza sus ejecuciones mediante la generación de un pseudocódigo denominado “bytecode” en la primera ejecución, permitiendo que las siguientes se realicen de forma más eficiente. Otra característica importante es su tipificación dinámica: no es necesario declarar el tipo de dato de las variables, ya que este se asigna automáticamente cuando se ejecuta según el valor que contengan.

La elección de Python para este trabajo se debe a los siguientes motivos. En primer lugar, es uno de los lenguajes más utilizados en el ámbito profesional, especialmente para el análisis de datos, la inteligencia artificial y el procesamiento de imágenes. En segundo lugar, su sintaxis clara y concisa facilita el desarrollo de algoritmos complejos con estructuras simples y un número reducido de líneas de código. Finalmente, Python cuenta con un amplio ecosistema de librerías especializadas en procesamiento de imágenes y análisis geométrico, convirtiéndolo en una herramienta idónea para el desarrollo del proyecto [25].

4.2. Librerías empleadas

Para la implementación del programa desarrollado en este proyecto se han empleado diversas librerías de Python que aportan funciones específicas ya definidas, lo que ha permitido optimizar tanto el desarrollo como el rendimiento del software. Las principales librerías empleadas son: OpenCV, NumPy, Matplotlib, Tkinter, Pandas, OS, Scikit-image y Openpyxl.

4.2.1. OpenCV (cv2)

OpenCV es una librería ampliamente utilizada en el procesamiento y análisis de imágenes. Permite realizar operaciones como lectura y escritura de archivos gráficos, conversión entre formatos, el filtrado, la detección de bordes y el análisis geométrico. En este proyecto se utiliza para cargar las imágenes obtenidas del escáner y aplicar técnicas de preprocesamiento necesarias para llevar a cabo las mediciones. En la tabla 4.1, se pueden observar el nombre de las funciones que se emplean en el programa, incluyendo una breve descripción de su funcionalidad.

Tabla 4.1: Funciones empleadas de la librería OpenCV

Descripción	**Nombre funciones**
Conversión de color	cvtColor COLOR_BGR2GRAY COLOR_BGR2RGB
Umbralización	adaptiveThreshold threshold ADAPTIVE_THRESH_MEAN_C THRESH_BINARY_INV
Manejo de eventos de ratón y ventanas	EVENT_LBUTTONDOWN namedWindow WINDOW_NORMAL setMouseCallback imshow waitKey destroyAllWindows
Dibujar en imágenes	FONT_HERSHEY_SIMPLEX getTextSize putText
Cambiar el tamaño	resize
Leer imágenes	imread

4.2.2. NumPy

NumPy proporciona soporte para el manejo de arreglos multidimensionales y la realización de operaciones matemáticas de alto rendimiento de manera eficiente. En este proyecto es necesaria para procesar los datos de las imágenes en forma matricial y realizar cálculos numéricos de forma más rápida y precisa. En la tabla 4.2, se pueden observar el nombre de las funciones que se emplean en el programa, incluyendo una breve descripción de su funcionalidad.

Tabla 4.2: Funciones empleadas de la librería NumPy

Descripción	**Nombre de funciones**
Creación de arrays	array zeros_like
Operaciones matemáticas	sqrt abs pi degrees
Valor especial	nan
Búsqueda y filtrado	where

4.2.3. Matplotlib

Dentro de la librería Matplotlib, se utiliza el módulo pyplot, que permite generar gráficos en dos dimensiones y facilita la visualización de datos. En este proyecto se emplea para representar gráficamente los resultados, resaltar elementos detectados en las imágenes y crear salidas visuales claras para su análisis. En la tabla 4.3, se pueden observar el nombre de las funciones que se emplean en el programa, incluyendo una breve descripción de su funcionalidad.

Tabla 4.3: Funciones empleadas de la librería Matplotlib

Descripción	**Nombre de funciones**
Crear y configurar figuras	figure title axis close
Mostrar imágenes	imshow show
Escribir sobre las imágenes	text

4.2.4. Tkinter

Tkinter es un módulo estándar de Python para la creación de interfaces gráficas de usuario (GUI). Se emplea en este proyecto para desarrollar una interfaz que permite al usuario cargar imágenes, iniciar procesos y visualizar resultados de forma interactiva. En concreto, se ha diseñado una ventana que muestra la imagen escaneada y permite hacer clic sobre las regiones para obtener y mostrar en pantalla las medidas correspondientes. En la tabla 4.4, se pueden observar el nombre de las funciones que se emplean en el programa, incluyendo una breve descripción de su funcionalidad.

Tabla 4.4: Funciones empleadas del módulo Tkinter

Descripción	**Nombre de funciones**
Crear ventana de la interfaz	Tk()
Ocultar ventana de la interfaz	root.withdraw
Mostrar información al usuario	messagebox.showinfo

4.2.5. Pandas

Pandas es una librería orientada al manejo y análisis de datos estructurados. Facilita la organización de la información y la lectura o escritura en distintos formatos. En este proyecto se emplea para exportar los datos obtenidos a un archivo Excel y mostrarlos en formato de tabla. En el caso de Pandas solo se emplea una función, "DataFrame", para la creación de tablas de resultados.

4.2.6. OS

El módulo OS permite interactuar con el sistema operativo, gestionando rutas de archivos, accediendo a directorios y manipulando la estructura de almacenamiento de los datos empleados por el programa. Como en el caso anterior, solo se emplea una función "path.exists", para verificar si el archivo que se desea abrir existe o no.

4.2.7. Scikit-image

Dentro de la librería se encuentra un módulo denominado "skimage.measure" que se enfoca en el análisis de regiones y obtener las propiedades de las regiones detectadas. En la tabla 4.5, se pueden observar el nombre de las funciones que se emplean en el programa, incluyendo una breve descripción de su funcionalidad.

Tabla 4.5: Funciones empleadas del módulo Tkinter

Descripción	**Nombre de funciones**
Etiquetado de las regiones detectadas	label
Cálculo de las propiedades geométricas de las regiones	regionprops

4.2.8. Openpyxl

Dentro de la librería openpyxl, en el programa se emplean dos módulos. La librería en su conjunto se emplea para convertir los datos guardados en DataFrames de Pandas en datos de Excel, las funciones empleadas se pueden visualizar en la tabla 4.6. Los módulos empleados son: "util"s para manejar columnas y celdas y "styles" para el formato y el estilo de las celdas.

Tabla 4.6: Funciones empleadas de la librería Openpyxl

Descripción	**Nombre de funciones**
Convertir de DataFrames a filas que se pueden escribir en Excel	dataframe_to_rows
Convertir número de columna a letra	get_column_letter
Formato y estilo de las celdas	Font alignment border side patternFill
Crear y abrir libros	workbook load_workbook

5. PROGRAMA

En esta sección se presenta el programa completo desarrollado. En primer lugar, se define el objetio del programa y seguido se realiza la expliación detallada de sus funciones. Dentro de esta explicación se incluye también las decisiones técnicas adoptadas durante la implementación, justificando la elección de las herramientas, estructuras y métodos empleados.

5.1. Programa desarrollado

En este apartado se presenta de manera detallada el flujo de trabajo implementado en el programa, desde la lectura inicial del fichero de entrada hasta la generación de las diferentes salidas. El esquema del proceso completo se puede observar en la figura 5.1, sirviendo como guía visual para seguir la explicación.

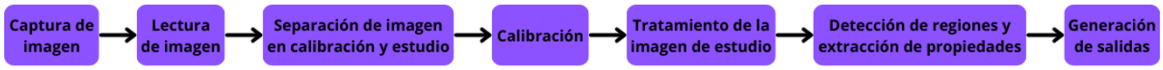


Figura 5.1: Diagrama general del proceso.

En el desarrollo de este proyecto se han tomado varias decisiones técnicas clave orientadas a garantizar la calidad y la fiabilidad de los resultados. La decisión principal consiste en priorizar la precisión en las medidas por encima de otras ventajas de procesamiento automático, evitando cualquier transformación que pueda distorsionar los resultados.

A continuación, se expone el objetivo principal del proceso y posteriormente, se explica paso a paso el funcionamiento del programa. En cada fase se explica de forma detallada las operaciones realizadas y las decisiones técnicas adoptadas durante el desarrollo.

5.1.1. Objetivo del programa

El programa desarrollado en este proyecto tiene como principal objetivo la medición de las dimensiones de elementos presentes en piezas físicas con una alta precisión, utilizando como partida una imagen digital de la pieza obtenida mediante un escáner 2D. Este método surge como respuesta a las limitaciones de los métodos de medición convencionales como el calibre o pie de rey, el micrómetro o los equipos de medición por contacto, ampliamente empleados en el ámbito industrial.

Aunque estos instrumentos ofrecen resultados fiables, presentan restricciones en cuanto a resolución y precisión. Por ejemplo, los calibres (figura 5.2) disponen de una precisión de 0,02 mm y una resolución de 0,01 mm, mientras que los micrómetros (figura 5.3) alcanzan una precisión de 2 µm y una resolución de 1 µm. En el caso de los sistemas de palpado continuo (figura 5.4), con las configuraciones más sencillas, el límite práctico se encuentra aproximadamente en los

6 mm, siendo 8 mm lo ideal para un funcionamiento estable. Sin embargo, con configuraciones más avanzadas se consigue alcanzar precisiones de en torno a los 4 µm.

Figura 5.2: Calibre o pie de rey. Fuente: [26]

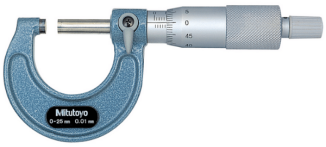


Figura 5.3: Micrómetro. Fuente: [27]

Figura 5.4: Palpado continuo. Fuente: [28]

Cuando se desea aumentar la resolución y alcanzar precisiones mayores, el coste, la complejidad y el tiempo de operación aumenta de forma significativa. Frente a estas alternativas, con el método propuesto se consigue alcanzar un margen de error inferior a los 10 µm, ofreciendo una solución más precisa, económica y sencilla de implementar.

5.1.2. Lectura y preprocesado de la imagen

El flujo de trabajo comienza solicitando al usuario tres datos: el nombre del archivo que contiene la captura del escaneo, la dimensión mínima de las regiones que se desean estudiar y el tipo de calibración que se está empleando, como se puede observar en la figura 5.5.

```
In [20]: %runfile C:/W/TFG/TFG_HELENA.py --wdir
Introducir el nombre del archivo de imagen (incluyendo extensión): P1.tiff
Introduce la anchura mínima del objeto (en milímetros): 20

********** MENÚ DE CALIBRACIÓN **********

1. Marco de calibración en papel
2. Marco de calibración en pieza

**********************************


Elige la opción que desee: 1
```

Figura 5.5: Preguntas realizadas al usuario por la terminal.

Antes de continuar con el preprocesado, es necesario exponer las decisiones técnicas adoptadas en la captura de la imagen, ya que de ellas depende la calidad del archivo que posteriormente se introduce en el programa. Estas decisiones afectan desde al fondo de la captura, el modo de salida en el escáner, el software empleado, la resolución del trabajo hasta la extensión del archivo.

En cuanto al fondo, se ha optado por utilizar uno de color negro tras comprobar que con un fondo blanco aparecían sombras y reflejos que dificultaban la detección de contornos. En la figura 5.6 se observa la captura con fondo negro y en la figura 5.7 la obtenida con un fondo blanco. En esta última se puede comprobar como hay menor contraste entre el objeto y el fondo, la iluminación genera sombras e introduce ruido. Mientras que con el fondo negro se obtiene una segmentación más clara y precisa. Se probó también a realizar una captura únicamente del fondo y otra con la pieza para restar ambas imágenes pero los resultados no fueron satisfactorios, por lo que se descartó esta opción. Para poder conseguir el fondo negro se utiliza una caja opaca lo suficientemente alta para evitar que la luz del escáner se refleje en las paredes y así evitar que capture figuras que generen ruido.

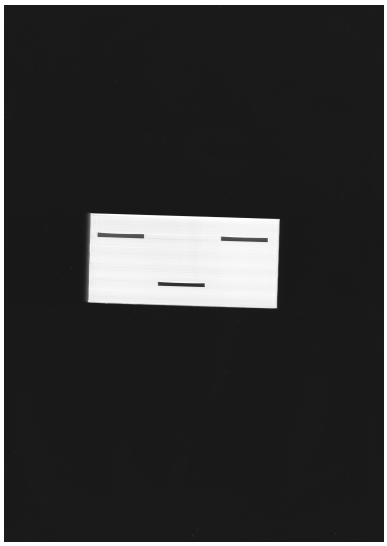


Figura 5.6: Escaneo con fondo negro.

Figura 5.7: Escaneo con fondo blanco.

Para realizar la captura de las imágenes se ha optado por utilizar el software *NAPS2*, en lugar

del programa original incluido con el escáner. Esta decisión se adoptó tras comprobar que, aunque el escáner dispone del software *IJ Scan Utility*, este presenta la siguiente limitación: para alcanzar la resolución máxima 4800 DPI restringe el tamaño de los escaneos, reduciendo el área máxima que se puede digitalizar, como se muestra en la figura 5.8.

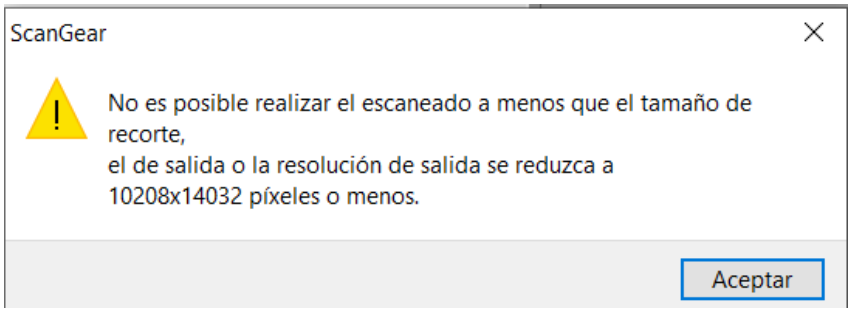


Figura 5.8: Limitación de dimensión de escaneo en el software original.

Con el fin de evitar esta restricción y poder trabaja con la máxima resolución a tamaño completo, se optó por emplear *NAPS2* como solución. En las figura 5.9 y 5.10 se muestran las principales diferencias entre las funciones que disponen cada uno de ellos.

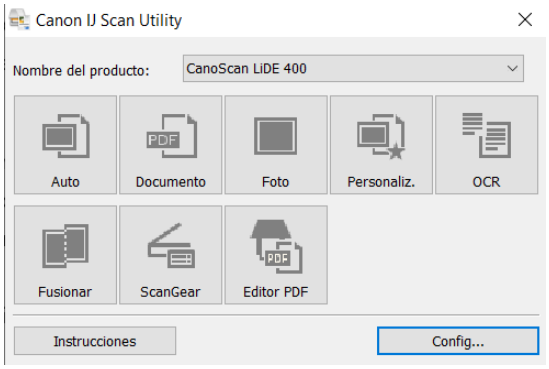


Figura 5.9: Software original "IJ Scan Utility".

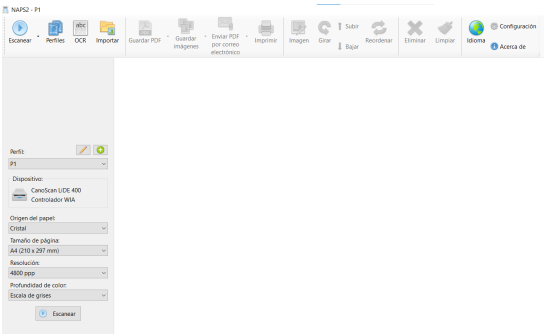


Figura 5.10: Software utilizado "NAPS2".

El escáner ofrece tres modos de salida: en color, escala de grises y en blanco y negro (binario). La salida en color se descartó por el elevado peso de los archivos y porque la información cromática no es relevante en este proyecto. Con el software original, la opción en blanco y negro resultaba demasiado sensible a la reflectancia, por lo que inicialmente se empleaba la escala de grises. Sin embargo, con *NAPS2*, con la salida en blanco y negro se obtienen mejores resultados, segmentando las piezas de manera más clara. En las figuras 5.11, 5.12 y 5.13 se puede comprobar la diferencia en el peso estimado para los archivos en función de su modo de salida. Además, en las figuras 5.14 y 5.15 se comparan las imágenes obtenidas en blanco y negro y en escala de grises al utilizar *NAPS2*. Se puede apreciar que la opción binaria se conserva la información necesaria para realizar la detección de las regiones, a diferencia de al emplear el software original.

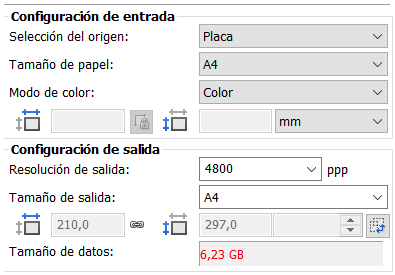


Figura 5.11: Peso archivo de salida a color.

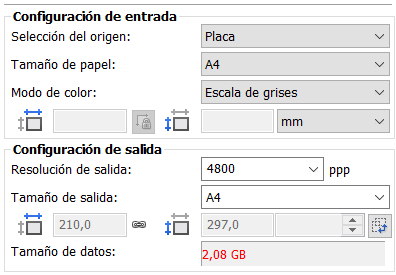


Figura 5.12: Peso archivo de salida en escala de grises.

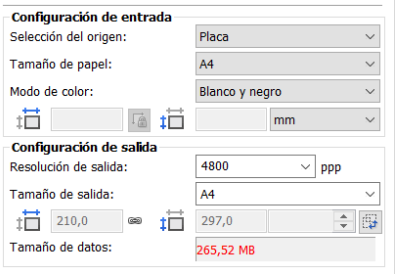


Figura 5.13: Peso archivo de salida en blanco y negro.

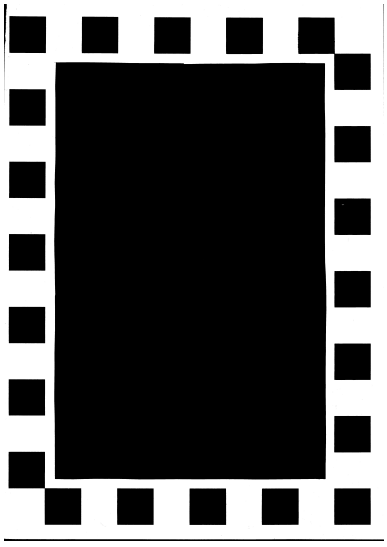


Figura 5.14: Escaneo con salida en blanco y negro utilizando *NAPS2*.

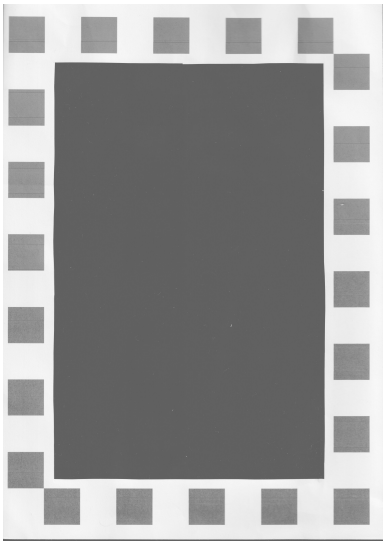


Figura 5.15: Escaneo con salida en escala de grises utilizando *NAPS2*.

Teniendo en cuenta que un incremento en la resolución del escaneo implica un aumento proporcional en el tamaño de los archivos creados, se decidió realizar todas las pruebas intermedias a una resolución de 1200 DPI. Con esta decisión se permite reducir el peso de los ficheros y el tiempo de escaneo, facilitando el desarrollo y los ajustes del programa. Una vez se finaliza la implementación del código, se realiza los escaneos a 4800 DPI, asegurando la precisión requerida.

Con estas decisiones técnicas, se procede a obtener las imágenes de entrada. En la figura 5.16 se muestra un ejemplo de una de las capturas. Una vez la imagen ha sido cargada, el siguiente paso consiste en separarla en dos zonas de trabajo. Por un lado, el patrón de calibración y por otro,

la pieza que se desea estudiar. Para realizar la separación se aprovecha la geometría del marco de calibración y la binarización Otsu. Este tipo de binarización permite encontrar un umbral óptimo que separa los píxeles en dos clases: figuras y fondo [29]. El tamaño de los cuadrados y su disposición en el marco proporcionan una referencia fiable para poder determinar el área de calibración dentro de la captura. Una vez se han identificado ambas zonas, se genera una imagen de calibración que contiene el patrón al completo y una imagen de estudio que solo contiene la pieza que se desea analizar.

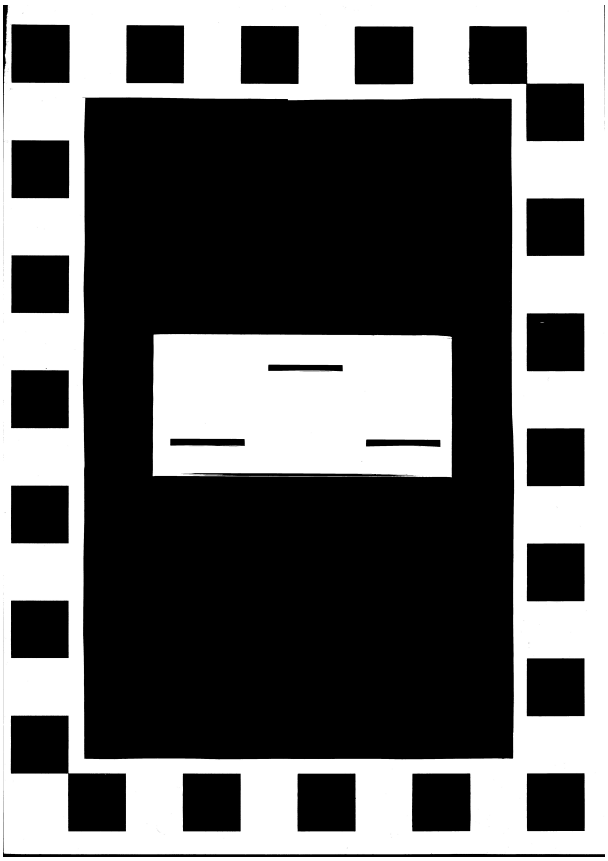


Figura 5.16: Captura de la pieza 1 junto con el marco de calibración.

En la delimitación del área de calibración se deja un margen adicional alrededor del patrón, en la figura 3.6 se puede observar que se corresponde con el espacio entre el patrón de cuadrados y el rectángulo central. Este margen actúa como zona de seguridad para posibles pequeñas variaciones en el escaneo (ligeros desplazamientos, irregularidades en el apoyo del documento o desviaciones angulares). Con esto se consigue garantizar que el patrón queda definido, sin recortes en sus bordes, dentro de los límites establecidos tanto en el eje vertical como en el horizontal.

Con esta separación se permite tratar de forma independiente la calibración y la pieza de estudio, evitando que las decisiones de una afecten a la otra y realizar los ajustes necesarios en la calibración para después realizar los mínimos cambios posibles en la pieza de estudio.

5.1.3. Calibración

La calibración se ha establecido como un paso fundamental para corregir variaciones que se producen en la propia captura, como pueden ser cambios locales de la iluminación, diferencias en la velocidad de escaneo o variaciones en el enfoque. Con este procedimiento se garantiza que las medidas obtenidas en píxeles se correspondan con las unidades físicas. Para ello, como se ha mencionado anteriormente, se decidió separar la zona de calibración de la de estudio. En la región de calibración se utiliza un marco con un patrón formado por cuadrados de dimensiones conocidas, dispuestos tanto en el eje horizontal como en el vertical. Gracias a este marco es posible comprobar que la relación entre las medidas se mantiene a lo largo de ambos ejes y así poder obtener un factor de corrección para las dimensiones.

Para ejecutar la calibración se necesitan los siguientes parámetros: la imagen que solo está formada por el marco de calibración (como se muestra en la figura 5.17), la relación de píxel-milímetro (se calcula como se muestra en la ecuación 3) y la medida real de los lados de los cuadrados que conforman al marco de la calibración.

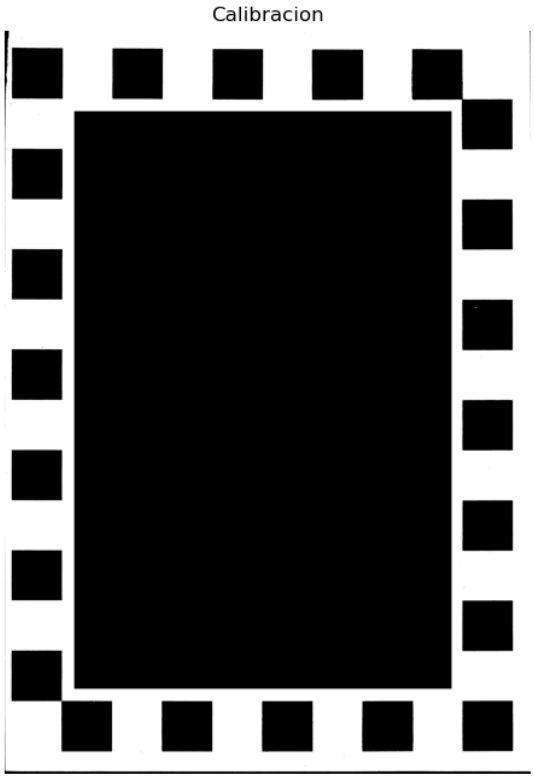


Figura 5.17: Marco de calibración separado por el programa.

En este proyecto, la calibración se realiza para conocer si se ha producido alguna distorsión de las distancias al realizar el escaneo, pero en un futuro se va a emplear para conocer a la

intensidad a la que se ha ejecutado este. Por esta razón, el propio código incluye el cálculo de este valor para posteriormente no tener que modificarlo. Gracias al menú que se visualiza en la figura 5.5, se puede conocer si es necesario calcular un factor de corrección debido a las distancias o calcular el umbral de intensidad.

En ambos casos, el primer paso consiste en la conversión de la imagen a escala de grises para poder binarizarla. Esta se realiza mediante un umbral fijo, es decir, convierte la imagen en una representación formada por dos niveles: los píxeles cuya intensidad supera el umbral se asigna a negro, mientras que aquellos con intensidad igual o inferior al umbral se asignan a blanco.

Normalmente la binarización se realiza de forma inversa, los valores superiores se asignan al blanco y los inferiores al negro. La motivación de invertir la binarización se debe a que permite diferenciar con mayor exactitud los bordes de los cuadrados del fondo, facilitando la segmentación y obtención de sus dimensiones. Además, para poder establecer el valor del umbral se han llevado a cabo pruebas experimentales y se ha obtenido un mejor resultado cuando el valor se correspondía a 127.

Una vez la imagen está binarizada, se realiza una limpieza morfológica para mejorar los resultados y además deja lista la imagen para poder extraer las regiones de interés: los cuadrados que conforman el marco de calibración. En la figura 5.18 se puede observar el resultado tras este proceso.

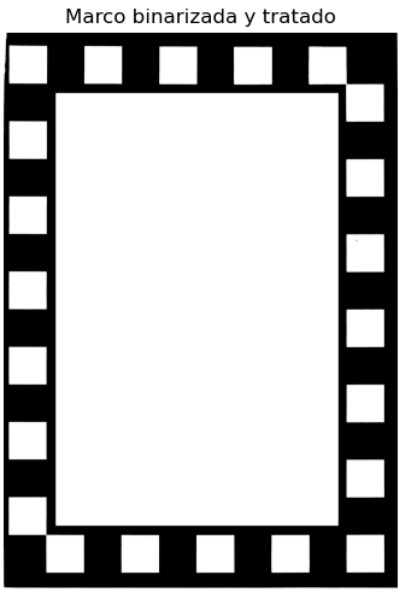


Figura 5.18: Marco de calibración binarizado y tratado.

El siguiente paso consiste en la detección de los cuadrados del marco y la obtención de sus parámetros. Teniendo en cuenta que también se detectan los rectángulos que delimitan los cuadrados, se ha realizado un descarte de regiones en función de las medidas. Para ello, se ha

establecido que la distancia de las regiones válidas se deben encontrar entre los 15 y los 25 mm. Estos valores han sido establecidos teniendo en cuenta que los cuadrados miden 20 x 20 mm y que se ha podido generar una distorsión en el escaneo. Finalmente, se almacenan las propiedades de las regiones que se han determinado como válidas para poder continuar con los cálculos.

A partir de este punto, la calibración continúa en función del método que se seleccionó en el menú. En el primer caso, se calcula el factor de corrección para los ejes horizontales y verticales por separado, siguiendo la ecuación 4 y los parámetros de la tabla 5.1.

$$k_v = \frac{Ancho_{\text{real}}}{Ancho_{\text{media obtenida}}} \qquad k_h = \frac{Alto_{\text{real}}}{Alto_{\text{media obtenida}}} \tag{4}$$

Tabla 5.1: Unidades base del Sistema Internacional de Unidades establecidas en 1960

Magnitud física	**Unidad**	**Simbolo**
k_v	Factor de corrección en el eje vertical	-
k_h	Factor de corrección en el eje horizontal	-
$Ancho_{\text{real}}$	Anchura real de los cuadrados	mm
$Alto_{\text{real}}$	Altura real de los cuadrados	mm
$Ancho_{\text{media obtenida}}$	Anchura media de los cuadrados obtenida por el programa	mm
$Alto_{\text{media obtenida}}$	Altura media de los cuadrados obtenida por el programa	mm

En el segundo caso (definido para un uso futuro con piezas fabricadas en Rigid), se determina el valor de la intensidad con el que se ha realizado la captura de la imagen. Para obtener este valor, se realiza el perfil de intensidad en el centro de las regiones, tanto en el eje horizontal como en el vertical.Analizando los perfiles se puede identificar, para cada nivel de intensidad, la distancia de los valles en unidades físicas y se establece como umbral aquel nivel en el que la distancia coincide con el tamaño real de los cuadrados.

Como ejemplo ilustrativo del procedimiento, se utilizó una pieza auxiliar fabricada en Rigid, mostrada en la figura 5.19. En este caso, se analizó el rectángulo interior de la pieza, cuyas dimensiones son 22.697 x 10.97 mm, obteniéndose un umbral de 145 a partir de los perfiles de intensidad que se muestran en las figuras 5.20 y 5.21. Sin embargo, esta pieza no se puede utilizar como pieza de calibración, ya que sus dimensiones son reducidas en comparación con el área de escaneo y solo permitiría determinar la corrección en una zona localizada. Por este motivo, su uso en este proyecto es con un fin explicativo, mostrando de manera práctica como en un futuro se calcularía el valor del umbral.

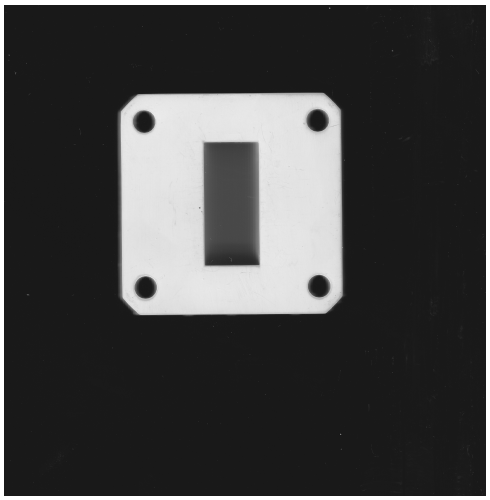


Figura 5.19: Pieza auxiliar.

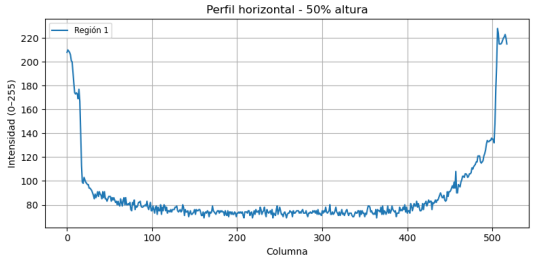


Figura 5.20: Perfil horizontal de la pieza auxiliar.

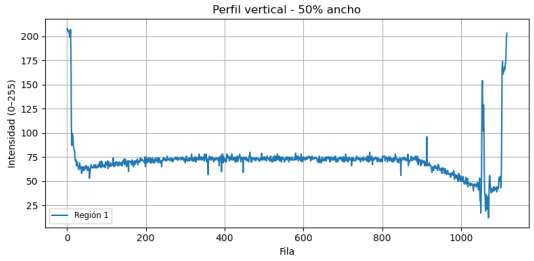


Figura 5.21: Perfil vertical de la pieza auxiliar.

Una vez se ha obtenido el factor de corrección o el nivel de umbral, ya se puede establecer que se ha realizado la calibración necesaria y se puede proceder a la etapa de tratamiento de la imagen de estudio.

5.1.4. Tratamiento de la pieza

Para llevar a cabo el tratamiento de la imagen, es necesario disponer de la imagen de estudio, como se puede visualizar en la figura 5.22, y conocer el valor de umbral con el que se realiza la binarización. Este umbral depende del material de la pieza, ya que las propiedades ópticas como el color y la reflectancia influyen en la intensidad de los píxeles. Gracias al proceso descrito en la etapa de calibración con la pieza auxiliar, se ha podido determinar que el valor para el Rigid se encuentra en torno a los 145. No se podrá obtener el valor automáticamente hasta que no se disponga de la pieza de calibración en el mismo material que la pieza de estudio. Por esta razón, cuando se selecciona la primera opción del menú de calibración, el umbral está definido manualmente a 145, mientras que cuando se selecciona la segunda opción este se define en función del valor obtenido en el proceso de calibración.

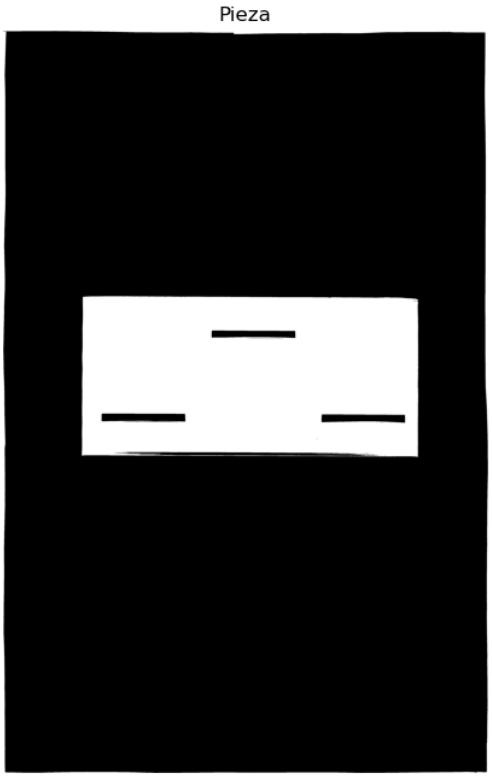


Figura 5.22: Pieza separada por el programa.

La principal decisión técnica tomada en esta etapa consiste en priorizar la máxima precisión posible en las mediciones frente a otras ventajas del proceso automático. Por este motivo, se ha decidido descartar cualquier tipo de tratamiento de la imagen que pueda alterar o distorsionar los contornos y las regiones que se desean analizar. Esto incluye evitar reescalados con interpolaciones agresivas, filtros de suavizados innecesarios, transformaciones geométricas o técnicas de ecualización de contraste [29], ya que todas ellas podrían modificar la relación espacial entre los píxeles. En su lugar, se ha mantenido la imagen lo más fiel posible a la captura original, aplicando las mínimas operaciones posibles y criterio de filtrado basado en las propiedades geométricas físicas de la pieza.

Al igual que ocurre en la calibración, para evitar errores derivados del formato de la captura, la imagen de estudio se convierte a escala de grises. Trabajar en esta escala simplifica el análisis, ya que cada píxel se representa por su intensidad luminosa y no por la información correspondiente al color, que no es necesaria para la segmentación. Se consigue reducir la complejidad de procesamiento y facilitar el manejo de los datos.

Una vez se ha obtenido la imagen en escala de grises, se procede a la binarización mediante un umbral fijo siguiendo el mismo criterio que el descrito en el apartado de la calibración.

La motivación de invertir la binarización también en el tratamiento de la pieza se debe a que permite que esta aparezca en negro y las regiones de interés se representen en blanco. Esta representación facilita el contraste con el fondo y la segmentación automática, simplificando la identificación y el análisis de las regiones en las etapas posteriores del tratamiento de la imagen. En la figura 5.23, se puede observar la pieza de estudio binarizada y como las regiones de interés se encuentran en blanco.

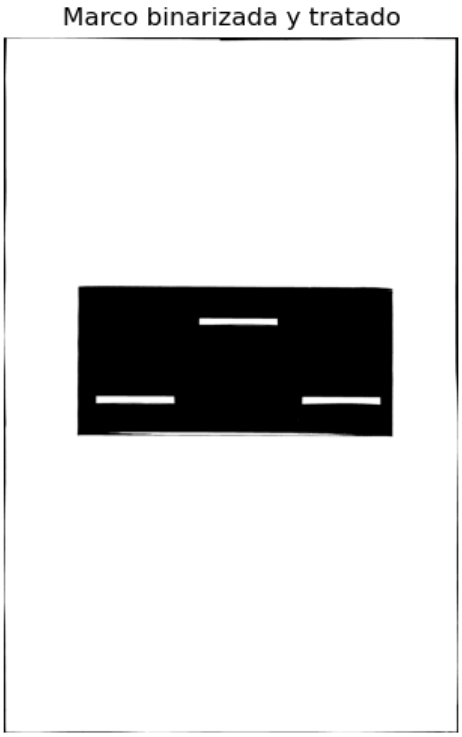


Figura 5.23: Pieza de estudio binarizada.

5.1.5. Detección de regiones y extracción de las propiedades

Para poder detectar las regiones que conforman la pieza y posteriormente, obtener sus propiedades es fundamental la imagen binarizada obtenida en el apartado anterior. En dicha imagen se encuentran diferenciadas las regiones de interés del fondo, así se evitan posibles fallos que podrían surgir si se trabajase con imágenes en escala de grises o en color.

El proceso comienza con la identificación de todas las regiones, definiéndose como regiones cualquier conjunto de píxeles conectados entre sí que comparten el mismo valor, en este caso, asignados a las áreas blancas. El recorrido de la imagen para la detección se realiza siguiendo un orden que establece el sistema, de arriba hacia abajo y de izquierda a derecha. El primer conjunto detectado se corresponde con el fondo de la captura y teniendo en cuenta que no aporta información útil para el estudio y solo sirve como una zona de separación entre la pieza

de estudio y el marco de calibración, se descarta del conjunto de regiones. Esto significa que el almacenamiento de las regiones válidas comienza a partir de la segunda región detectada.

Una vez se ha eliminado el fondo, se procede a una etapa vital como es la eliminación del ruido de fondo. En este contexto, el ruido se refiere a pequeñas agrupaciones de píxeles que se encuentran dentro del conjunto de regiones detectadas y que pueden haberse generado debido a imperfecciones en la superficie de la propia pieza, residuos generados durante su fabricación o suciedad en el escáner. Estos elementos son contabilizados como regiones y deben ser descartadas.

Para realizar este descarte, se emplea la medida introducida por la terminal que se ha mencionado en el apartado de la lectura y el procesamiento de imágenes. Esta medida se corresponde con la anchura mínima que debe tener la región para poder considerarse válida. Esta entrada se introduce en milímetros. Sin embargo, dado que el procesamiento de las imágenes se realiza en píxeles, se debe realizar una conversión precisa entre milímetros y píxeles, teniendo en cuenta la resolución del escáner. Además, se tiene en cuenta una tolerancia de 0,2 mm por si la medida introducida estuviese muy ajustada a la realidad. En la ecuación 5 se puede comprobar cómo se realiza la conversión si se establece que la anchura mínima debe ser 13 mm. Esta conversión asegura que la medida introducida por el usuario se aplica en el mismo dominio con el que se trabaja en el programa.

$$\text{Anchura} = 13\,[\text{mm}] \qquad \text{DPI} = 4800 \qquad \text{Tolerancia} = 0{,}2\,[\text{mm}]$$

$$\text{Relación pixel a milímetro} = \frac{25{,}4}{4800} = 0{,}00529167\,[\text{mm}] \tag{5}$$

$$\text{Anchura mínima en pixel} = \frac{(13 - 0{,}2)}{0{,}00529167} = 2418{,}89 \text{ pixeles}$$

El filtrado por la anchura se realiza a cada una de las regiones que se han detectado evaluando su dimensión horizontal. Si una región tiene una distancia menor que la especificada por el usuario, se descarta automáticamente. Además, como las regiones estudiadas en las diferentes piezas de las que se disponen nunca superan los 40 mm, se ha establecido que también se deben eliminar aquellas regiones en las que su anchura sean superiores a este valor. Con estos dos pasos además de eliminar el ruido de fondo, se consigue depurar la imagen de fragmentos del fondo o de zonas deterioradas de la pieza.

Tras realizar el filtrado, se consigue tener almacenado únicamente las regiones que se desean estudiar. Con esto ya definido, se procede a la fase de extracción de las propiedades. Entre las propiedades que se extraen se encuentran las dimensiones como la posición en el eje vertical y horizontal, la altura, anchura, el área, la orientación, el perímetro, el centroide o la excentricidad. Estos valores permiten establecer las características de las regiones.

Con este método, se consigue que solo se analicen aquellas regiones que estén bien definidas y

se consideren una región, consiguiendo que el proceso sea más eficiente al no trabajar con datos redundantes.

5.1.6. Generación de salidas

Una vez se ha finalizado la fase de filtrado y obtención de los parámetros, el sistema genera tres tipos de salidas complementarias que facilitan la interpretación, la verificación y el almacenamiento de la información. Mediante el menú que se muestra en la figura 5.24, se selecciona cual se desea generar. Este no se cierra hasta que no se elige la opción de salir para poder permitir la visualización de las diferentes salidas en una única ejecución del código.

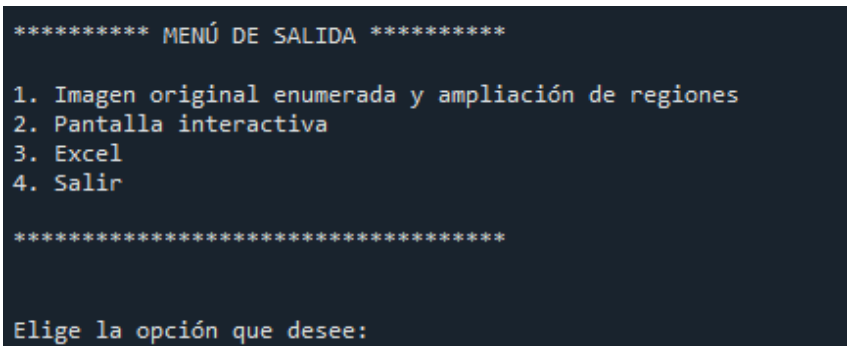


Figura 5.24: Menú de tipo de salida.

La primera salida consiste en una representación de la imagen original en la que cada región aparece identificada mediante un número colocado en su centroide. Además, se incluye una representación individual ampliada de cada región, permitiendo examinar el resultado de segmentación con mayor detalle y comprobar que el proceso se ha realizado de forma correcta. Este modelo de salida se puede visualizar en las figuras 5.25 y 5.26. Se recomienda que cuando se ejecuta esta opción, no se ejecute ninguna de las otras salidas y se termine el programa. Requiere muchas recursos y la generación de la imágenes se retrasa si se mantiene el código ejecutado.

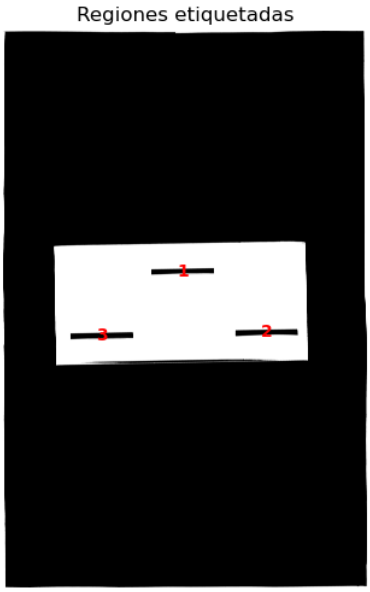


Figura 5.25: Salida 1: Enumeración de todas las regiones.

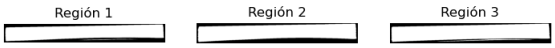


Figura 5.26: Salida 1: Ampliación de las regiones detectadas.

La segunda salida consiste en una interfaz interactiva en la que se visualiza la imagen original, con las regiones enumeradas como en la anterior salida, actuando como un mapa visual para poder acceder a la información. Realizando un clic sobre cualquiera de los números que sirve para identificar las regiones, se abre una ventana que muestra las propiedades geométricas y morfológicas correspondientes. Todas las medidas mostradas en la ventana se encuentran en unidades físicas, para poder obtenerlas se ha realizado la conversión inversa a la que se puede visualizar en la ecuación 5, es decir, se conocen los píxeles que forman la propiedad y se convierten a milímetros. Además, se tiene en cuenta el factor de corrección que se ha calculado en la etapa de calibración debido a la distorsión generada por el escaneo y el factor de corrección debido a la orientación de la pieza. A pesar de introducir las piezas lo más paralelas posibles a los ejes, siempre se va a cometer un fallo de precisión. Por esta razón, al extraer los parámetros también se tiene en cuenta la orientación y posteriormente se corrige.

Como decisión técnica, para ampliar las futuras aplicaciones del programa, se ha incluido la detección de elementos circulares mediante el análisis de su excentricidad (una circunferencia

perfecta tiene una excentricidad nula), permitiendo así diferenciar entre regiones rectangulares y circulares.

Se puede visualizar la pantalla interactiva en la figura 5.27 y los resultados de la región 1 en la figura 5.28. Esta salida resulta útil cuando se quiere conocer de manera rápida y precisa una región en concreto, evitando tener que revisar todas las regiones.

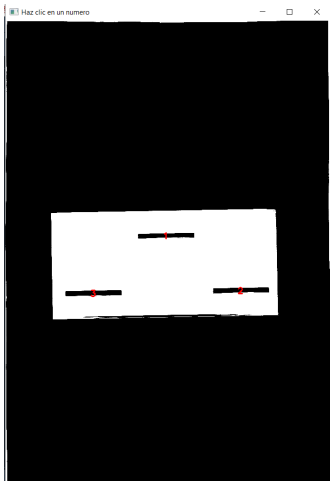


Figura 5.27: Salida 2: Pantalla interactiva.

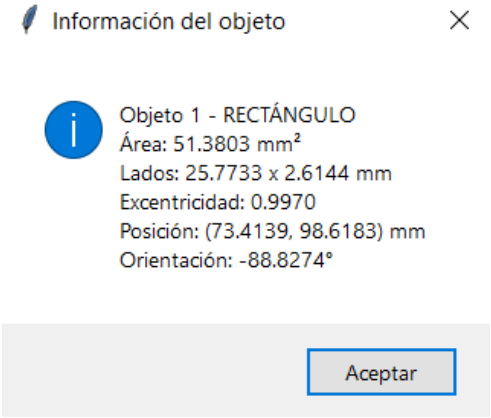


Figura 5.28: Salida 2: Resultados de la región 1.

Por último, la tercera salida se corresponde a la generación de un archivo formato Excel, denominado "Resultado_Pieza.xlsx". En este archivo se recopilan en formato de tabla todos los resultados obtenidos del análisis con la conversión y las correcciones mencionada en la salida anterior. Además, se realiza el cálculo de la desviación relativa del área para comprobar cuanto se desvian las piezas de las regiones ideales. Los parámetros empleados se puede visualizar en la tabla 5.2 y la ecuación empleada para realizar el cálculo de la desviación relativa del área se puede observar en la ecuación 6. Si la desviación relativa obtenida es positiva, eso significa que el área real es superior al área ideal y al contrario, si el resultado es negativo el área real es inferior al área ideal. Finalmente, se puede visualizar un ejemplo de tabla generada en la figura 5.29.

$$Desv = \frac{A_{real} - A_{ideal}}{A_{real}} \tag{6}$$

Tabla 5.2: Descripción de parámetros cálculo de la desviación relativa del área

Parámetro	**Significado**	**Unidad**
Desv	Desviación relativa del área de las regiones	%
A_{real}	Área calculada por los datos obtenidos de las propiedades	mm^2
A_{ideal}	Área obtenido de las propiedades	mm^2

Resultados de la pieza escaneada										
Objeto	Tipo	Área [mm²]	Radio [mm]	Ancho [mm]	Alto [mm]	Desviación Área [%]	Excentricidad	Orientación [°]	Centro X [mm]	Centro Y [mm]
1	Rectángulo	51.38026164		25.77325871	2.614363165	-23.74627528	0.997007632	-88.82736354	73.41390521	98.61834497
2	Rectángulo	56.17319813		25.71112423	2.865423064	-23.75360206	0.996393255	-88.90939484	107.7275934	123.7117261
3	Rectángulo	58.11501259		25.83638078	2.886286031	-22.06773617	0.996162952	-88.85593011	40.30082585	125.0184689

Figura 5.29: Salida 3: Tabla generada en el Excel.

Si este archivo ya existe, el sistema automáticamente añade una nueva hoja con los nuevos datos obtenidos, permitiendo mantener un registro de las mediciones tomadas. Este formato además de asegurar el almacenamiento ordenado y accesible, también facilita el empleo de la información recopilada en diferentes procesos posteriores como puede ser de control o de análisis estadístico.

En conjunto, la generación de las tres salidas proporciona un sistema de análisis completo, combinando la verificación visual, el análisis interactivo y el almacenamiento ordenado de la información. Gracias a esta metodología, el usuario puede validar la calidad del procesamiento y de la segmentación de forma inmediata y además, dispone de herramientas para verificar los datos obtenidos y conservar un registro de ellos. Así se garantiza la extracción de la información y además su gestión, trazabilidad y aprovechamiento para futuros procedimientos.

6. RESULTADOS

En esta sección se presentan los resultados obtenidos mediante el programa desarrollado. El objetivo principal es mostrar de forma clara y ordenada las dimensiones de las regiones internas que conforman las piezas de estudio. Para ello, se analizan dos piezas diferentes a las que se le ha aplicado la misma metodología de segmentación, filtrado y cálculo de medidas.

6.1. Primera pieza

La primera pieza se corresponde con la que se empleó en la sección anterior para presentar los distintos ejemplos ilustrativos. Sin embargo, en este apartado se incluye nuevamente con el fin de presentar los resultados de forma más ordenada.

Se trata de una pieza de geometría rectangular que contiene en su interior tres regiones de interés, también con forma rectangular. En la figura 6.1 se puede observar tanto la pieza de estudio como el marco de calibración utilizado en el proceso.

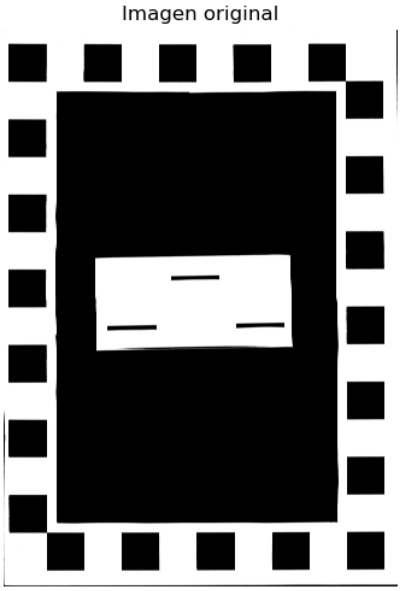


Figura 6.1: Pieza 1: captura de la pieza y el marco de calibración.

Como se ha mencionado con anterioridad, el primer paso del análisis consiste en separar el marco de calibración de la pieza de estudio. El marco de calibración queda guardado como se muestra en la figura 6.2 y la pieza como en la figura 6.3.

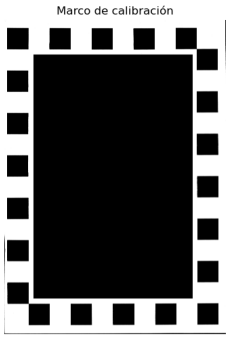


Figura 6.2: Pieza 1: marco de calibración.

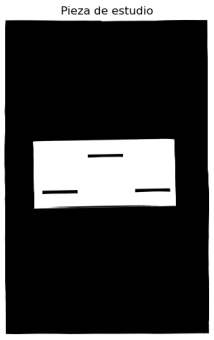


Figura 6.3: Pieza 1: pieza de estudio.

Una vez se ha aislado la pieza del marco y se ha realizado la calibración pertinente, en este caso se calculan los factores de correción ya que el marco se encuentra impreso, se procede a realizar la binarización de la imagen de estudio. Para ello se aplica un umbral de 145, obteniendo la figura 6.4 en blanco y negro en la que las regiones de interés quedan diferenciadas del fondo.

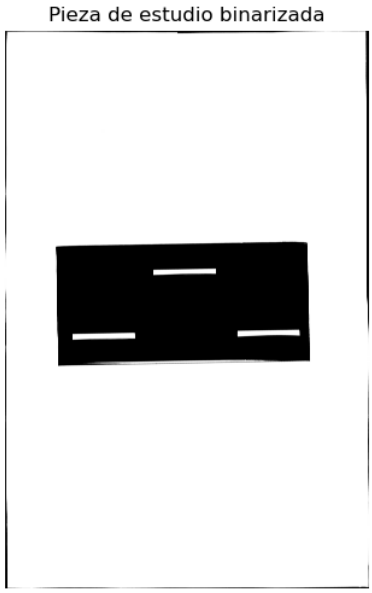


Figura 6.4: Pieza 1: binarización de la pieza de estudio.

A partir de la imagen binarizada, el programa extrae las regiones detectadas, se aplican los filtros dimensionales y calcula sus propiedades geométricas. Con la información ya recapitulada se generan tres tipos de salidas diferentes. La primera se corresponde con la generación de la figura 6.5, en la que se muestran las regiones de interés numeradas y la figura 6.6, mostrando una ampliación de dichas regiones. Con esta salida se consigue observar que la detección de las regiones era la deseada y además permite ver los defectos de impresión y como pueden alterar sus dimensiones y no ser una figura geométrica perfecta.

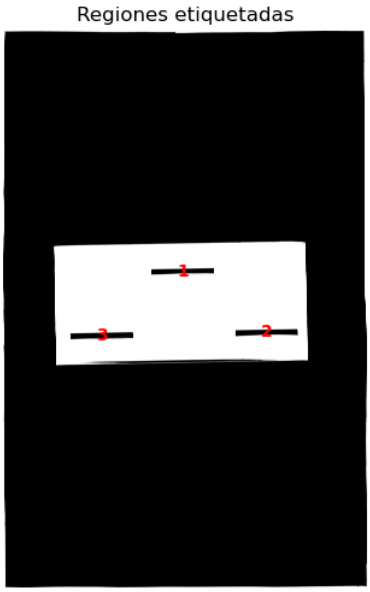


Figura 6.5: Pieza 1: salida que muestra las regiones de interés enumeradas.

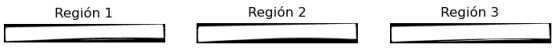


Figura 6.6: Pieza 1: salida que muestra la ampliación de las regiones de interés.

La segunda consiste en una ventana interactiva, que muestra la misma imagen que la salida anterior, pero que al hacer clic sobre el número asignado a cada una de las regiones, se muestras los resultados obtenidos de las mediciones. En la figura 6.7 se muestra la ventana interactiva y en la figura 6.8 los datos obtenidos de la pieza 1. Este tipo de salida permite consultar los datos de una región en concreto de manera más dinámica y ordenada. Permite conocer visualmente la región de la que se están visualizando las medidas mientras se lleva a cabo.

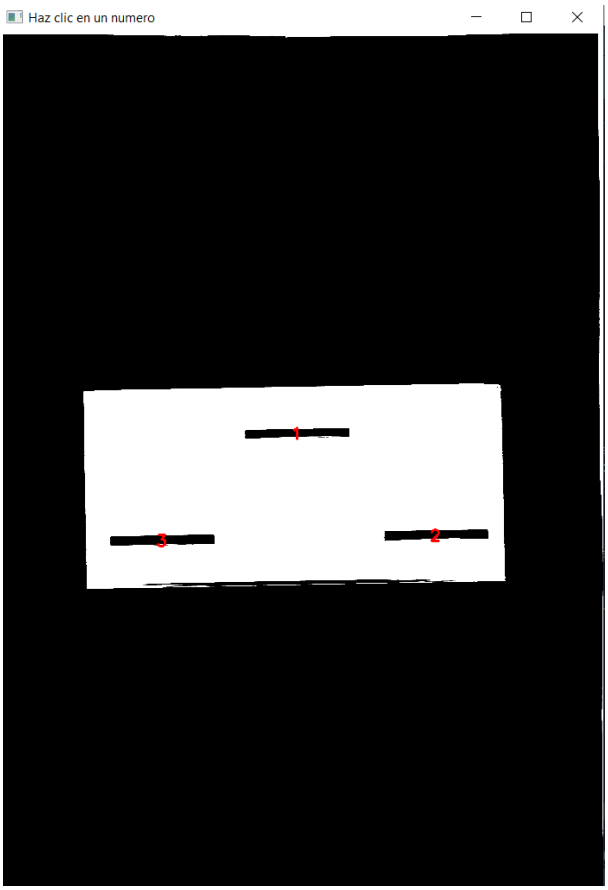


Figura 6.7: Pieza 1: salida que muestra la ventana interactiva.

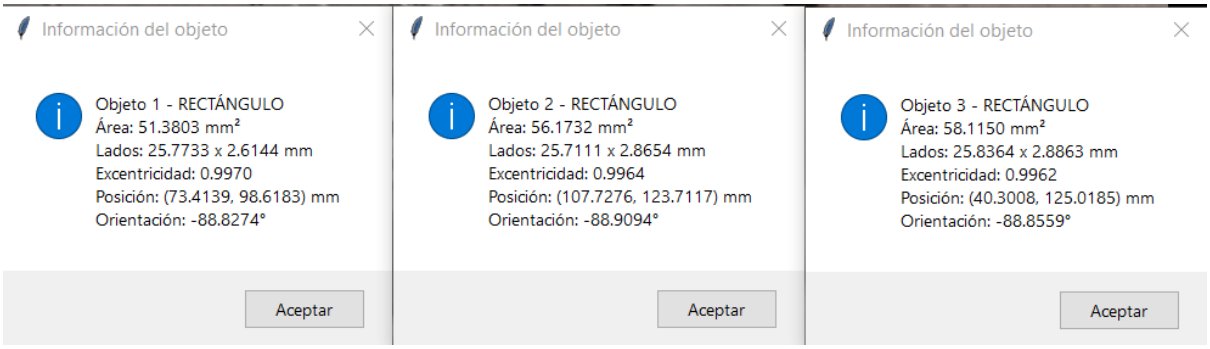


Figura 6.8: Pieza 1: resultados obtenidos de la ventana interactiva.

Finalmente, la tercera salida es una tabla generada automáticamente en Excel. Esta tabla recopila todos los parámetros calculados para cada región de interés y además incluye la desviación relativa del área de las regiones. Constituye una herramienta clara y práctica para interpretar los resultados y también permite almacenar la información de las diferentes capturas sin perder dichos datos.

Resultados de la pieza escaneada										
Objeto	Tipo	Área [mm²]	Radio [mm]	Ancho [mm]	Alto [mm]	Desviación Área [%]	Excentricidad	Orientación [°]	Centro X [mm]	Centro Y [mm]
1	Rectángulo	51.38026164		25.77325871	2.614363165	-23.74627528	0.997007632	-88.82736354	73.41390521	98.61834497
2	Rectángulo	56.17319813		25.71112423	2.865423064	-23.75360206	0.996393255	-88.90939484	107.7275934	123.7117261
3	Rectángulo	58.11501259		25.83638078	2.886286031	-22.06773617	0.996162952	-88.85593011	40.30082585	125.0184689

Figura 6.9: Pieza 1: tabla generada en Excel con los resultados obtenidos.

6.2. Segunda pieza

La segunda pieza analizada también presenta una geometría rectangular, aunque en este caso contiene seis regiones internas, disponiendo también de una forma rectangular. Al igual que en el caso anterior, en la figura 6.10 se puede observar la captura incial formada por la pieza de estudio y el marco de calibración.

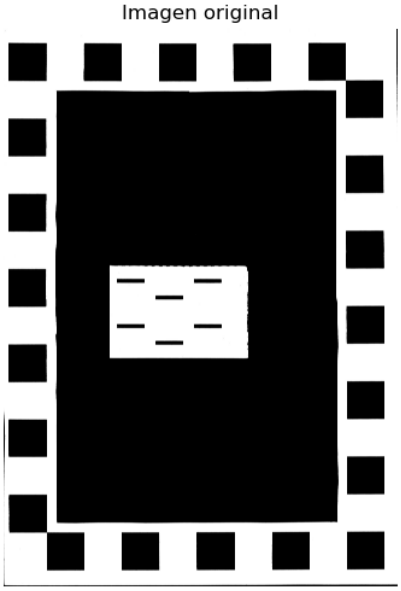


Figura 6.10: Pieza 2: captura de la pieza y el marco de calibración.

El procedimiento seguido para esta pieza es el mismo al descrito con la pieza anterior. En primer lugar, como se puede observar en las figuras 6.11 y 6.12, se separa el marco de calibración y la pieza de estudio.

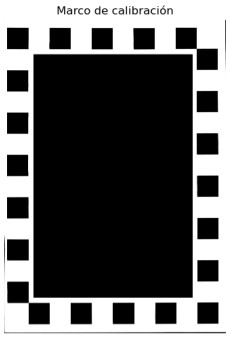


Figura 6.11: Pieza 2: marco de calibración.

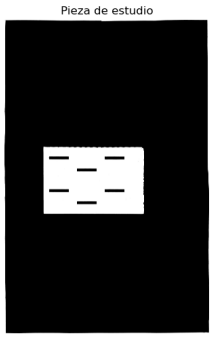


Figura 6.12: Pieza 2: pieza de estudio.

A continuación, se aplica el umbral de binarización en la pieza de estudio obteniendo una imagen

en la que se distinguen las seis regiones internas de la pieza. El resultado obtenido se puede visualizar en la figura 6.13.

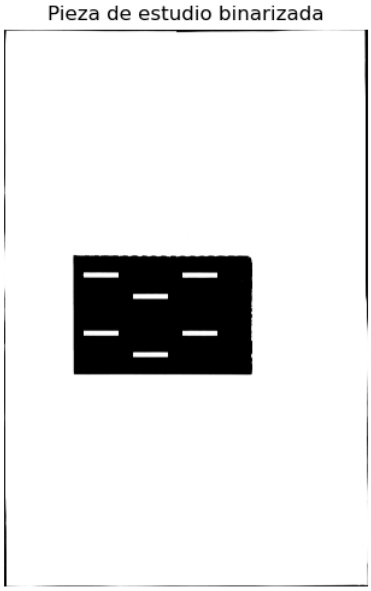


Figura 6.13: Pieza 2: binarización de la pieza de estudio.

El sistema extrae las propiedades geométricas de cada región y genera los resultados a través de las tres salidas diferentes. En la primera, se visualizan las regiones de interés enumeradas (figura 6.14) y una ampliación de dichas regiones (figura 6.15). Observando los resultados obtenidos, se puede establecer que todas las cavidades internas han sido correctamente identificadas.

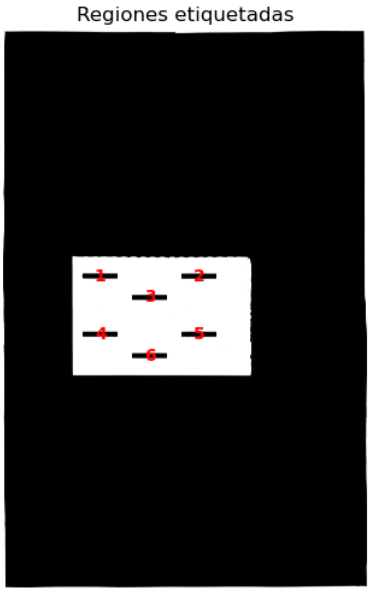


Figura 6.14: Pieza 2: salida que muestra las regiones de interés enumeradas.

Figura 6.15: Pieza 2: salida que muestra la ampliación de las regiones de interés.

La segunda salida consiste en la ventana interactiva, como se muestra en la figura 6.16, en la que se visualiza la pieza con las regiones detectadas enumeradas. Al hacer clic en la enumeración se consigue visualizar los resultados de cada una de las regiones, como se puede comprobar en la figura 6.15.

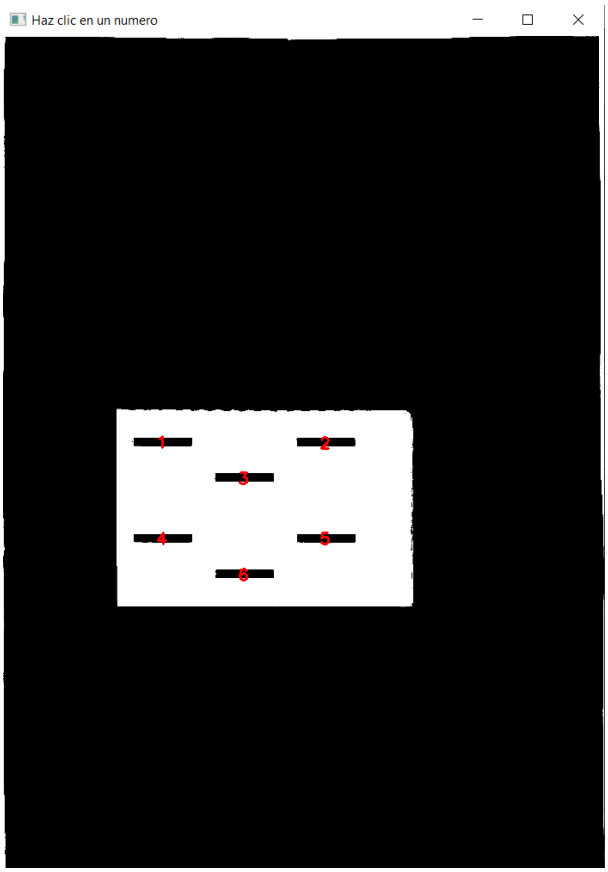


Figura 6.16: Pieza 2: salida que muestra la ventana interactiva.

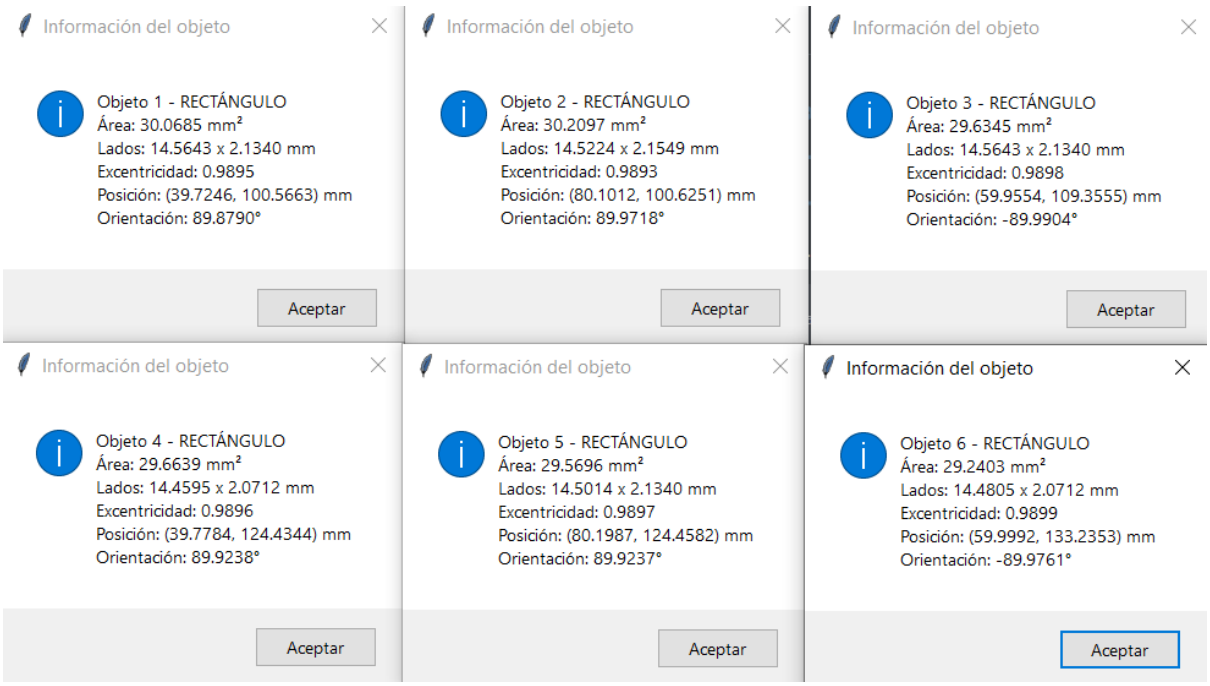


Figura 6.17: Pieza 2: resultados obtenidos de la ventana interactiva.

Finalmente, se genera la tercera salida, creando la tabla de resultados en formato Excel. En la figura 6.18 se pueden visualizar, de forma ordenado, los resultados obtenidos para la pieza

número dos.

Resultados de la pieza escaneada										
Objeto	Tipo	Área [mm²]	Radio [mm]	Ancho [mm]	Alto [mm]	Desviación Área [%]	Excentricidad	Orientación [°]	Centro X [mm]	Centro Y [mm]
1	Rectángulo	30.06852393		14.56426116	2.134010471	-3.255317194	0.989492449	89.87904527	39.72455986	100.5662685
2	Rectángulo	30.20969895		14.52238022	2.154936683	-3.467382969	0.989348062	89.97178405	80.10117325	100.6251345
3	Rectángulo	29.63447654		14.56429341	2.134015196	-4.652275491	0.989799508	-89.99038188	59.9554359	109.3554924
4	Rectángulo	29.66385146		14.45950173	2.071248241	-0.952833127	0.989597131	89.92379382	39.77836958	124.4344037
5	Rectángulo	29.56958864		14.50141332	2.134013336	-4.448431009	0.989738962	89.9237472	80.19868576	124.4582371
6	Rectángulo	29.24032642		14.48046908	2.071249892	-2.508424886	0.989909379	-89.97608441	59.99923472	133.2352835

Figura 6.18: Pieza 2: tabla generada en Excel con los resultados obtenidos.

7. PRESUPUESTO

Para poder llevar a cabo el desarrollo del presente proyecto, ha sido necesario tener en cuenta los costes asociados a la adquisición de los equipos, los recursos materiales y la mano de obra empleada. En este presupuesto se recopilan los principales elementos necesarios: la fabricación externa de piezas, medición externa del marco de calibración, el ordenador con el que se desarrolla el programa (en este caso el HP Pavilion Gaming 16-A0010NS), el escáner utilizado para la adquisición de imágenes y el coste estimado de la mano de obra de un ingeniero, calculado teniendo en cuenta el convenio de la categoría Ingenieros, Arquitectos y Licenciados del convenio siderometalúrgico de Cantabria.

Tabla 7.1: Presupuesto del proyecto

Descripción	**Unidad**	**Cantidad**	**Precio unidad**	**Precio total**
Fabricación externa de piezas	Unidades	4	100,00 €	400,00 €
Medición externa de piezas	Unidades	1	725,00 €	725,00 €
Ordenador	Unidades	1	1099,84 €	1099,84 €
Escáner	Unidades	1	99,00 €	99,00 €
Mano de obra	Horas	100	22,68 €/h	2268 €
			Total estimado	**4591,84€**

8. CONCLUSIONES

En esta sección se presentan las conclusiones derivadas del desarrollo del proyecto. El objetivo es establecer una reflexión global del trabajo realizado, evaluando el grado de cumplimiento de los objetivos planteados.

Además, se exponen las limitaciones detectadas durante el proceso de diseño e implementación, con el fin de identificar los factores que han condicionado los resultados obtenidos y restringido la aplicabilidad del programa.

Finalmente, se proponen posibles mejoras y desarrollos futuros con la finalidad de ampliar las funcionalidades del programa. De este modo, las conclusiones no solo sirven como cierre del presente trabajo, sino también como punto de partida para futuras investigaciones.

8.1. Reflexión global

El objetivo principal del proyecto consistía en una prueba de funcionamiento de un programa en Python capaz de detectar las regiones de interés de una pieza y proporcionar sus medidas con una precisión de 10 µm. Este propósito se ha cumplido, ya que se ha implementado una herramienta que a partir de la captura digital de la pieza, automáticamente se obtienen sus dimensiones. Además, teniendo en cuenta que el sistema trabaja a una resolución de 4800 DPI, se puede afirmar que se alcanzan las condiciones de precisión establecidas.

En relación a los objetivos específicos establecidos, el primero de ellos era familiarizarme con un nuevo lenguaje de programación, Python, así como con las librerías especializadas utilizadas a lo largo del desarrollo del proyecto. A pesar de no tener experiencia previa, he conseguido adquirir una base sólida del lenguaje y aprender a utilizar con soltura las diferentes funciones de las librerías utilizadas, demostrando el cumplimiento de este objetivo.

El segundo objetivo consistía en adquirir experiencia en la investigación autónoma de temas técnicos. Este objetivo se ha logrado gracias a la investigación de la evolución de la tecnología de los escáneres, los parámetros relevantes que se deben tener en cuenta a la hora de escoger un escáner, el tratamiento de imágenes y la exploración de alternativas en la ejecución del programa. Todo este trabajo de búsqueda y análisis se ha realizado de manera independiente, reforzando la capacidad de investigación técnica.

Este proceso de investigación también ha permitido mejorar la capacidad para identificar los parámetros más relevantes en función de las necesidades del análisis, estudiar las características del entorno de trabajo que pueden influir en condiciones reales y en la toma de decisiones entre diferentes métodos, evaluando los resultados obtenidos y seleccionando el más beneficioso en cada uno de los casos.

Finalmente, el proyecto ha servido para consolidar y ampliar los conocimientos adquiridos en asignaturas como “Industrial Robotics and Computer Vision”. En algunos aspectos se ha aplicado directamente lo aprendido en clase, como en la elección del tipo de binarización, mientras

que en otros casos se ha optado por ir más allá de lo trabajado durante el curso e innovar, como en el tratamiento posterior de las imágenes. A esto se le suma que el lenguaje trabajado en el proyecto es distinto al empleado en las prácticas de la asignatura, aumentando así el aprendizaje adquirido.

8.2. Limitaciones

En el desarrollo del programa han surgido una serie de limitaciones que condicionan su aplicabilidad y los resultados que pueden obtener.

En primer lugar, una limitación fundamental es el material de las piezas escaneadas. Cuando se emplean piezas metálicas, la luz del escáner genera reflejos y sombras debido a la alta reflectancia de su superficie, interfiriendo en el proceso de segmentación y pudiendo introducir errores en las mediciones. Por ello, el sistema resulta más fiable cuando se trabaja con piezas blancas, que reducen las sombras y los reflejos.

Otra limitación relacionada con el material empleado para las piezas es cuando se emplean dos piezas de diferentes materiales. Cuando se realiza el proceso con dos piezas del mismo material, la pieza de calibración se puede emplear para determinar la intensidad con la que se ha realizado el escaneo y a partir de este valor, procesar la pieza de estudio conociendo dicho valor. Sin embargo, cuando las piezas empleadas son de diferente material, la respuesta óptica es diferente y dificulta la calibración y el posterior tratamiento de las imágenes.

También existen limitaciones derivadas del software de escaneo. El software original del escáner restringe el tamaño máximo de área en el que se puede hacer la captura a altas resoluciones, obligando a buscar alternativas. Durante el proceso se probaron otras aplicaciones que a pesar de permitir captura en dimensiones mayores, incluyen una marca de agua que no permite trabajar con la captura. Finalmente, se optó por un software no oficial, que elimina la restricción del área de escaneo, pero genera una dependencia externa que puede provocar inestabilidad e incompatibilidad.

Otra limitación importante se encuentra asociada al peso de los archivos generados. Escanear a resoluciones de 4800 DPI implica grandes tiempos de captura y archivos de gran tamaño. Con esto se consigue una reducción de velocidad de ejecución del programa y no sea tan eficiente.

Por último, se debe considerar la limitación del área de estudio. Las piezas se pueden colocar para ser escaneadas dentro de un límite definido por el marco de calibración. Se consigue reducir la versatilidad del sistema para piezas de gran tamaño.

8.3. Posibles mejoras

A partir de las limitaciones detectadas, se pueden plantear una serie de mejoras que permitirán optimizar la precisión y la versatilidad del sistema.

La primera y la más relevante consiste en utilizar un marco de calibración fabricado en una

pieza rígida y con el mismo material que la pieza de estudio (en este caso, Rigid). Esta pieza además de facilitar la determinación del umbral de iluminación necesario para el procesamiento de las imágenes, también permitiría evaluar de forma más precisa las posibles distorsiones de las medidas, A partir de esta pieza, sería posible identificar como se alteran las dimensiones en las diferentes zonas de la superficie de escaneo, calcular los factores de corrección específicos y aplicarlos en las regiones de interés cuando se desee obtener sus medidas. De esta forma, se incrementaría la fiabilidad de las medidas obtenidas.

Dentro del programa desarrollado, además, se podría incluir el cálculo de la distancia entre las regiones de interés. Con esta información se conseguiría tener más localizados en el espacio las diferentes regiones.

Otra mejora consistiría en desarrollar un software específico para el escáner utilizado, incluyendo las funcionalidades del oficial, pero eliminando la restricción del área de captura a altas resoluciones. Permitiría escanear a pantalla completa sin recurrir a aplicaciones externas, asegurando mayor estabilidad y compatibilidad.

Además, se podrían llevar a cabo mejoras en el procesamiento de imagen. Incorporar métodos avanzados de filtrado o segmentación basada en técnicas de visión por computador o aprendizaje automático, como puede ser el “Deep Learning”.

Finalmente, se podría trabajar en la optimización del manejo de archivos. Reducir el peso de las imágenes escaneadas sin pérdida de información o incluir estrategias de procesamiento que ayuden a disminuir los tiempos de ejecución y mejorar la eficiencia del programa.

BIBLIOGRAFÍA

[1] Real Academia Española, *Metrología — Diccionario de la lengua española*, 2025. dirección: https://dle.rae.es/metrologia.

[2] Joint Committee for Guides in Metrology, *International Vocabulary of Metrology – Basic and general concepts and associated terms (VIM), 3rd edition*, DOI: 10.59161/JCGM200-2012, 2012. dirección: https://www.bipm.org/doi/10.59161/JCGM200-2012.

[3] Centro Español de Metrología, *Breve historia de la metrología*, 2021. dirección: https://www.cem.es/sites/default/files/2021-01/breve%20historia_de%20la%20metrologia_doc.pdf.

[4] Elias Resendiz, *Se inventa teleautógrafo creado por el Ing. Elisha Gray*, Evento registrado el 1 de enero de 1888, 2018. dirección: https://time.graphics/es/event/723179.

[5] IBM Corporation, *Punched Card*. dirección: https://www.ibm.com/history/punched-card.

[6] Sebastián Vargas Romo, *Cinta perforada 1970*, Evento registrado el 27 de febrero de 1970, 2018. dirección: https://time.graphics/event/1754028.

[7] Franco Castro, *Pantelégrafo*. dirección: https://proyectoidis.org/pantelegrafo/.

[8] National Institute of Standards and Technology, *First Digital Image*, 2022. dirección: https://www.nist.gov/mathematics-statistics/first-digital-image.

[9] IDIS, *Fotografía digital*. dirección: https://proyectoidis.org/fotografia-digital/.

[10] Tera Digital, *Cúal fue el primer producto comercial vendido con un escáner de código de barras?* Dirección: https://tera-digital.com/es/blogs/barcodes/first-barcode-scanner-commercial-product.

[11] Microtek International Inc., *Microtek Scanner and Scanner Software Test*. dirección: https://microtek.com/en/history.

[12] mlauhp24, *El scanner*, 2015. dirección: https://mlauhp24.blogspot.com/2015/06/el-scanner.html.

[13] Timetoast, *Evolución de las Impresoras Multifuncionales*. dirección: https://www.timetoast.com/timelines/impresoras-multifuncionales.

[14] U. N. de La Plata, *Análisis Multidimensional de Imágenes Digitales*, 2017. dirección: https://core.ac.uk/download/pdf/296407369.pdf.

[15] 3D Systems, *Guía de escáneres 3D*. dirección: https://es.3dsystems.com/3d-scanner/scanner-guide.

[16] Kreon3D, *How does laser triangulation 3D scanning technology improve manufacturing processes*. dirección: https://www.kreon3d.com/es/article/how-does-laser-triangulation-3d-scanning-technology-improve-manufacturing-processes.

[17] L. C., *Escáner láser 3D o escáner de luz estructurada*, Idioma: español, 2019. dirección: https://www.3dnatives.com/es/escaner-laser-3d-escaner-luz-estructurada-mejor-080820192/.

[18] Artec 3D, *Aprende sobre el escaneo 3D con láser*. dirección: https://www.artec3d.com/es/learning-center/laser-3d-scanning.

[19] M. González Ballester y J. Cabestany, *Fundamentos de procesamiento digital de imágenes*. Editorial Reverté, 2001.

[20] Canon España, *Explicación de los sensores de imagen*, Accedido: 17 de mayo de 2025, 2023. dirección: https://www.canon.es/pro/infobank/image-sensors-explained/.

[21] Epson, *Especificaciones del escáner Epson Perfection V19*, Datos técnicos como resolución, profundidad de color, velocidad, tipo de sensor, botones, conexión USB, etc., 2025. dirección: https://epson.co.cr/Para-el-hogar/Esc%C3%A1neres/Fotos/Esc%C3%A1ner-Epson-Perfection-V19/p/B11B231201.

[22] Canon España, *Canon CanoScan LiDE 400 - Escáner plano en negro*, Ficha de producto en tienda oficial, Canon España, 2025. dirección: https://www.canon.es/business/products/scanners/flatbed-scanners/canoscan-lide-400/specifications/.

[23] Canon España, *Especificaciones del escáner CanoScan LiDE300*, Ficha técnica en sitio de Canon Business, 2025. dirección: https://www.canon.es/business/products/scanners/flatbed-scanners/canoscan-lide-300/specifications/.

[24] R. González Duque, *Python para todos*. España: MundoGeek, Licencia Creative Commons Reconocimiento 2.5 España. dirección: http://mundogeek.net/tutorial-python/.

[25] S. Delgado Quintero, *Aprende Python*, Licencia GNU GPL v3, Curso gratuito de programación Python, 2022. dirección: https://example.com/aprende-python.

[26] Kansert, *Calibre pie de rey miniatura 110-07 (DIN 862)*, Calibre de precisión pequeño, acero inoxidable templado, cursor monobloque, 4 modos de medición, 2025. dirección: https://kansert.es/mme/producto/calibre-pie-de-rey-miniautura-110-07/.

[27] Mitutoyo, *Micrómetro de exteriores 103-137 (0–25 mm)*, Graduación de 0,01 mm, precisión ± 0,002 mm, acabado esmaltado, tope de carraca, 2025. dirección: https://forteindustria.com/producto/mitutoyo-micrometro-de-exteriores-103-137-0-25mm/.

[28] J. Rodrigo, I. Puertas y C. J. Luis, *Estudio acerca de la tipología de las máquinas medidoras por coordenadas (MMC)*, Artículo en Interempresas.net sobre caracterización de máquinas de medición por coordenadas, 2012. dirección: https://www.interempresas.net/Medicion/Articulos/102185-Estudio-acerca-de-la-tipologia-de-las-maquinas-medidoras-por-coordenadas-(MMC).html.

[29] C. T. Ferrero, *Computer Vision I*, Material docente, Control Engineering Group, Departamento TEISA, Universidad de Cantabria, 2025.

[30] EdutoolsTec, *Spyder-Anaconda*. dirección: https://edutools.tec.mx/es/colecciones/tecnologias/spyder-anaconda.

[31] Spyder IDE, *Spyder: The Scientific Python Development Environment*. dirección: `https://www.spyder-ide.org`.

[32] Python Software Foundation, *Python Programming Language – Official Website*. dirección: `https://www.python.org/`.

[33] B. Jähne, *Digital Image Processing*. Springer, 2005.

[34] Universidad de Murcia, *Procesamiento digital de imágenes: Tema 6 – Mejora de la calidad de la imagen*, 2006. dirección: `https://www.um.es/geograf/sigmur/teledet/tema06.pdf`.

[35] S. Torres Ruiz, *Práctica 3: Filtrado de imágenes*, Departamento de Informática, Universidad de Jaén, 2021. dirección: `https://www4.ujaen.es/~satorres/practicas/practica3_vc.pdf`.

[36] C. T. Ferrero, *Computer Vision III*, Material docente, Control Engineering Group, Departamento TEISA, Universidad de Cantabria, 2025.

[37] C. T. Ferrero, *Computer Vision II*, Material docente, Grupo de Ingeniería de Control, Departamento TEISA, Universidad de Cantabria, 2025.

[38] C. T. Ferrero, *Computer Vision IV*, Material docente, Grupo de Ingeniería de Control, Departamento TEISA, Universidad de Cantabria, 2025.

[39] R. Marín, *¿Qué es OpenCV? Instalación en Python y ejemplos básicos*, Revista Digital INESEM, Informática y TICS, 2020.

[40] A. S. Alberca, *La librería NumPy*, Manual de Python, Aprende con Alf (docencia/python/manual/numpy), 2020.

[41] A. Lane, *Inclusión de texto en Matplotlib*, Blog Analytics Lane, 2022.

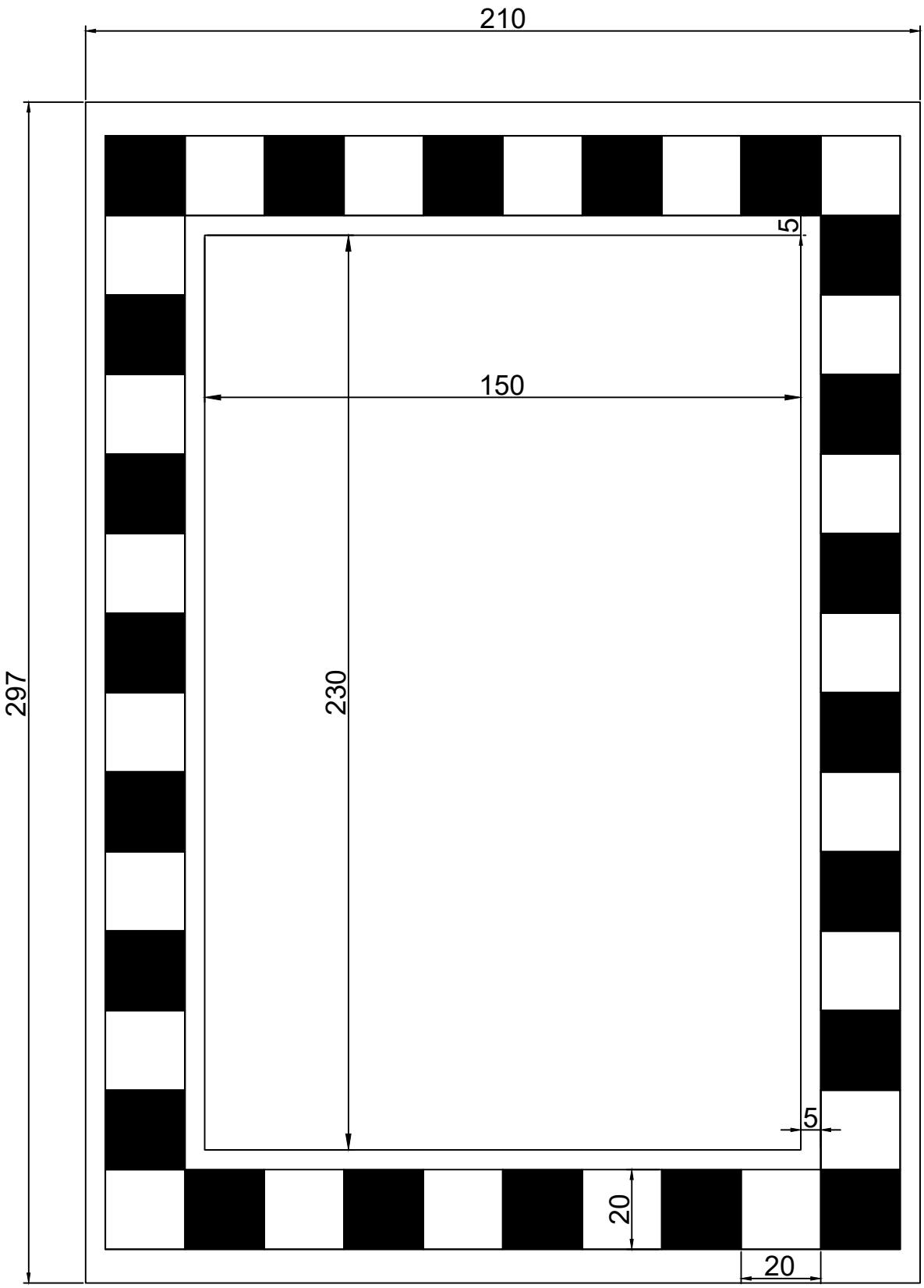
[42] W. Curo, *Biblioteca Tkinter*, Blog de Walther Curo, 2025.

[43] A. Lane, *Pandas: cambiar los tipos de datos en los DataFrames*, Blog Analytics Lane, 2021.

[44] Singole, *What is os-module*, Medium, 2025.

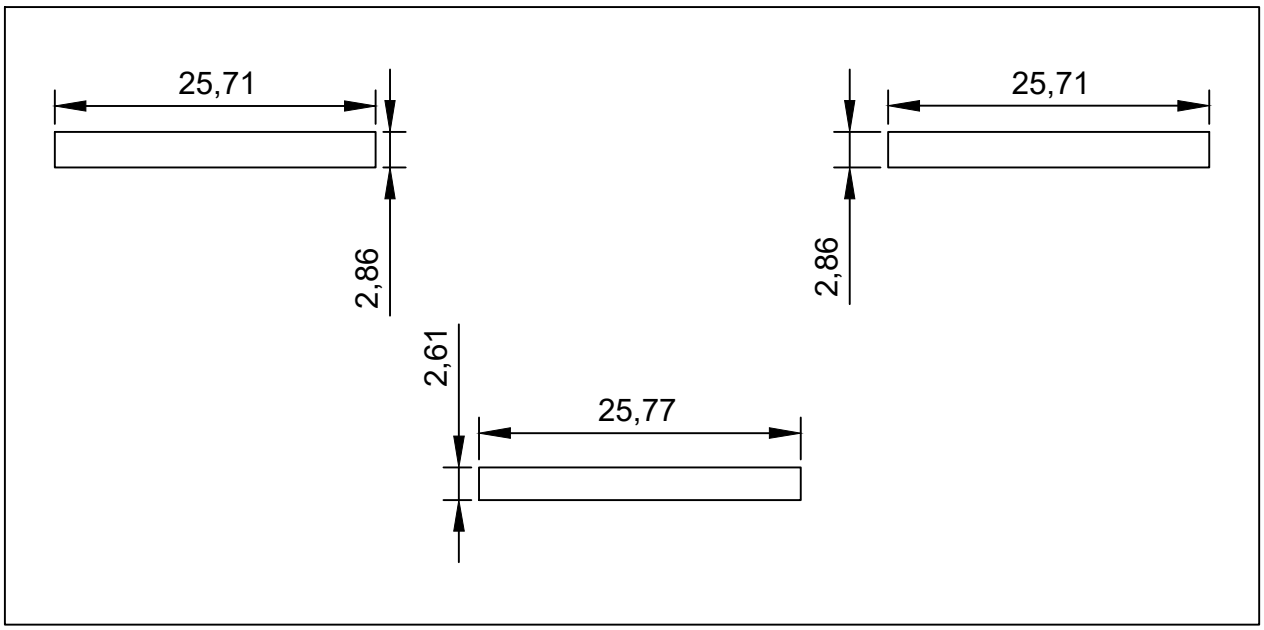
ANEXO A.

PLANOS

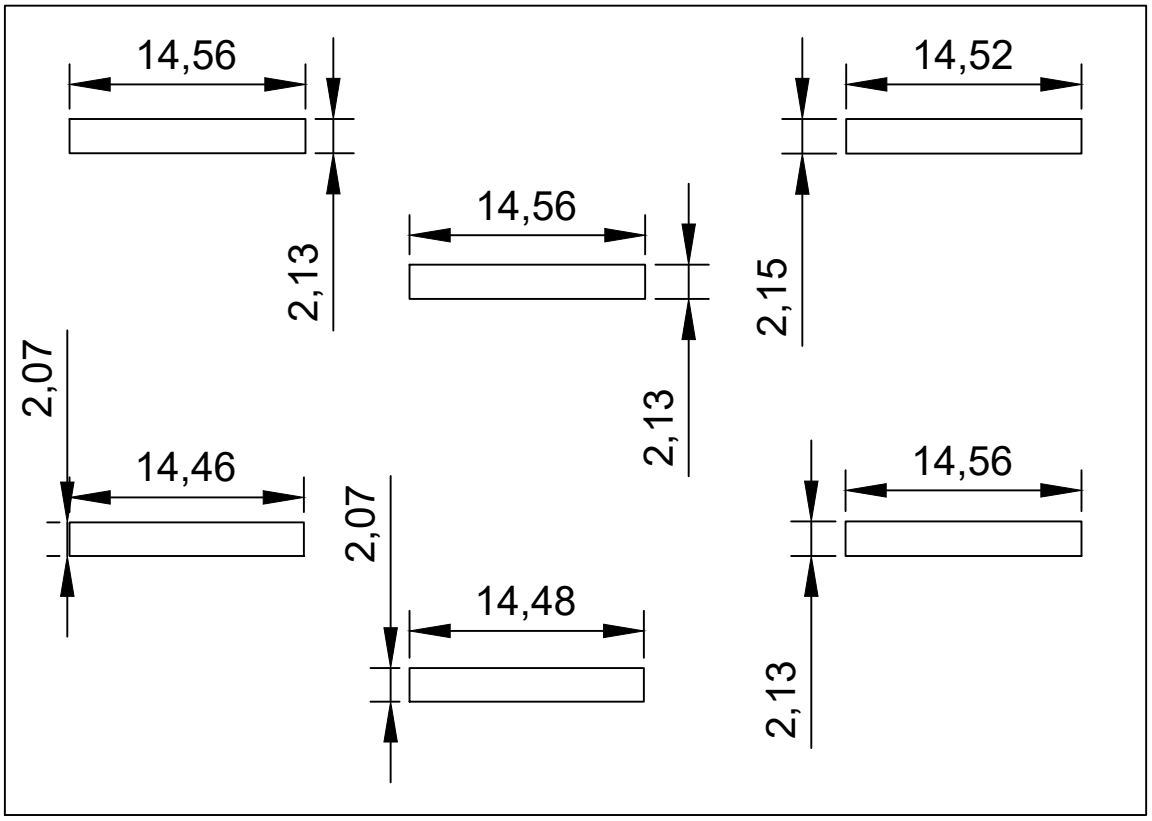


	Fecha	Nombre	
Dibujado	25/08/25	Helena Ruiz Fernández	Universidad de Cantabria (UC)
Comprobado	25/08/25	Helena Ruiz Fernández	

Escala: S/E	Nombre de plano: Marco de calibración	Referencia de Proyecto: METROLOGÍA DE PIEZAS
Formato: DIN A4		No. plano: 1/3



	Fecha	Nombre	
Dibujado	25/08/25	Helena Ruiz Fernández	Universidad de Cantabria (UC)
Comprobado	25/08/25	Helena Ruiz Fernández	
Escala: S/E	Nombre de plano: Pieza 1		Referencia de Proyecto: METROLOGÍA DE PIEZAS
Formato: DIN A4			No. plano: 2/3



	Fecha	Nombre		
Dibujado	25/08/25	Helena Ruiz Fernández	Universidad de Cantabria (UC)	
Comprobado	25/08/25	Helena Ruiz Fernández		
Escala: S/E	Nombre de plano: Pieza 2		Referencia de Proyecto: METROLOGÍA DE PIEZAS	
Formato: DIN A4			No. plano: 3/3	

ANEXO B.

CÓDIGO PYTHON

```python
# ----------------------------- LIBRERIAS -----------------------------
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tkinter as tk
import pandas as pd
import os
from tkinter import messagebox
from skimage.measure import label, regionprops
from openpyxl.utils.dataframe import dataframe_to_rows
from openpyxl.utils import get_column_letter
from openpyxl.styles import Font, Alignment, Border, Side, PatternFill
from openpyxl import Workbook, load_workbook

def recortar_piezas_por_posicion(I_color):

    # Convertir a escala de grises
    I_gray = (
              cv2.cvtColor(I_color, cv2.COLOR_BGR2GRAY)
              if len(I_color.shape) == 3
              else I_color
              )

    # Binarización con Otsu
    _, BW = cv2.threshold(I_gray, 0, 255,
                          cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)

    # Contornos externos
    contornos, _ = cv2.findContours(BW, cv2.RETR_EXTERNAL,
                                    cv2.CHAIN_APPROX_SIMPLE)
    if not contornos:
        raise ValueError("No se detectaron contornos en la imagen.")

    # Contorno más grande
    contorno_max = max(contornos, key=cv2.contourArea)

    # Limites del folio
    x, y, v, h = cv2.boundingRect(contorno_max)

    # Separra el marco de la zona de estudio
    resto = I_color[y:y+h, x:x+v]

    alto_total, ancho_total = I_color.shape[:2]
    mask = np.ones((alto_total, ancho_total), dtype=np.uint8) * 255
    cv2.rectangle(mask, (x, y), (x+v, y+h), 0, -1)

    marco = cv2.bitwise_and(I_color, I_color, mask=mask)

    return resto, marco

# FUNCIÓN TRATAMIENTO DE LA IMAGEN DE CALIBRACIÓN
def Tratamiento_Imagen_calibracion(I, px_por_mm, alto_mm, ancho_mm):

    # Estableccer distancia máxima y mínima de los cuadrados
    d_min_mm = 15
```

```
d_max_mm = 25

# Convertir a escala de grises
I_gray = cv2.cvtColor(I, cv2.COLOR_BGR2GRAY)

# Binarización con umbral fijo
_, BW = cv2.threshold(I_gray, 127, 255, cv2.THRESH_BINARY_INV)

# Limpieza morfológica
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
BW_clean = cv2.morphologyEx(BW, cv2.MORPH_OPEN, kernel)

# Etiquetar regiones
L = label(BW_clean)
propiedades = regionprops(L)

# Enumeración de las regiones
propiedades = sorted(propiedades,
                     key=lambda p: (p.centroid[0], p.centroid[1]))

# Inicializar variables
resultados = []
vis = I.copy()
idx = 0

# Descartar las regiones que no cumplan con las reglas de descarte
for prop in propiedades:

    fila_min, col_min, fila_max, col_max = prop.bbox
    ancho_px = col_max - col_min
    alto_px  = fila_max - fila_min

    # Conversión a mm
    ancho_mm_med = ancho_px / px_por_mm
    alto_mm_med  = alto_px  / px_por_mm

    # Reglas de descarte
    desc_por_pequena = ((ancho_mm_med < d_min_mm) and
                        (alto_mm_med < d_min_mm))
    desc_por_grande  = ((ancho_mm_med > d_max_mm) or
                        (alto_mm_med  > d_max_mm))

    # Hacer el descarte
    if desc_por_pequena or desc_por_grande:
        continue

    idx += 1
    resultados.append({
        "id": idx,
        "ancho_mm": ancho_mm_med,
        "alto_mm": alto_mm_med,
        "err_w": ancho_mm_med - ancho_mm,
        "err_h": alto_mm_med  - alto_mm,
        "centro": prop.centroid,
        "bbox": prop.bbox
    })
```

```
    return resultados, BW_clean, I_gray, vis

# FUNCIÓN CÁLCULO DE LOS FACTORES DE CORRECCIÓN DEL ESCÁNER
def calcular_factores_correccion(resultados, ancho_real, alto_real, px_por_mm):

    if not resultados:
        raise ValueError("No hay regiones.")

    # Media de las medidas de los cuadrados
    media_medidas_x = np.mean([r["ancho_mm"] for r in resultados])
    media_medidas_y = np.mean([r["alto_mm"]  for r in resultados])

    # Factor de corrección
    k_v = ancho_real / media_medidas_x
    k_h = alto_real / media_medidas_y

    return k_v, k_h

# FUNCIÓN CREACIÓN DE LOS PERFILES DE INTENSIDAD
def perfil_intensidad_por_region(I_gray, BW_image):

    # Etiquetar las regiones de la figura
    etiquetas = label(BW_image)
    regiones = regionprops(etiquetas)
    regiones = regiones[1:]

    # Altura y anchura de la imagen
    alto, ancho = I_gray.shape

    # Posiciones en las que se realiza el perfil dentro de cada región
    posiciones = [0.5]

    # Establecer posiciones para el perfil vertical y el horizontal
    perfiles_h = [[] for _ in posiciones]
    perfiles_v = [[] for _ in posiciones]

    for region in regiones:

        # Obtener las posiciones por las que se delimitan las regiones
        min_fila, min_col, max_fila, max_col = region.bbox

        # Expandir un poco los bordes de la región
        min_fila = max(0, min_fila - 10)
        min_col = max(0, min_col - 10)
        max_fila = min(alto, max_fila + 10)
        max_col = min(ancho, max_col + 10)

        # Recortar la imagen en función de la posición obtenida
        I_recortada = I_gray[min_fila:max_fila, min_col:max_col]

        # Tamaño de la región: número de filas y columnas
        n_filas, n_cols = I_recortada.shape

        # Perfiles horizontales
        # Calculo de las filas en las que se realiza los perfiles
```

```
            filas = [min(max(int(pos * n_filas), 0),
                         n_filas - 1) for pos in posiciones]
            for i, fila in enumerate(filas):

                # Guarda los datos de la fila en la que se realiza el perfil
                perfil = I_recortada[fila, :]

                # Obtener el perfil y guardarle
                perfiles_h[i].append(perfil)

            # Perfiles verticales
            # Calculo de las columnas en las que se realiza los perfiles
            cols = [min(max(int(pos * n_cols), 0),
                        n_cols - 1) for pos in posiciones]
            for i, col in enumerate(cols):

                # Guarda los datos de la fila en la que se realiza el perfil
                perfil = I_recortada[:, col]

                # Obtener el perfil y guardarle
                perfiles_v[i].append(perfil)

        # Mostrar los perfiles

        # === Dibujar perfiles horizontales ===
        for i, perfiles in enumerate(perfiles_h):
            plt.figure(figsize=(8, 4))
            for j, perfil in enumerate(perfiles):
                plt.plot(perfil, label=f'Región {j + 1}')
            plt.title(f'Perfil horizontal - {posiciones[i]*100:.0f}% altura')
            plt.xlabel("Columna")
            plt.ylabel("Intensidad (0–255)")
            plt.grid(True)
            plt.legend(fontsize='small')
            plt.tight_layout()
            plt.show()

        # === Dibujar perfiles verticales ===
        for i, perfiles in enumerate(perfiles_v):
            plt.figure(figsize=(8, 4))
            for j, perfil in enumerate(perfiles):
                plt.plot(perfil, label=f'Región {j + 1}')
            plt.title(f'Perfil vertical - {posiciones[i]*100:.0f}% ancho')
            plt.xlabel("Fila")
            plt.ylabel("Intensidad (0–255)")
            plt.grid(True)
            plt.legend(fontsize='small')
            plt.tight_layout()
            plt.show()

        return perfiles_h, perfiles_v

    # FUNCIÓN NIVEL PARA BINARIZACION
    def Nivel_binarizacion(perfil, ancho_mm_deseado, dpi):

        # Conversión de los datos para poder ser tratados
```

```
    perfil = np.array(perfil)

    # Paso del ancho de la pieza de calibración a pixeles
    ancho_px_deseado = (ancho_mm_deseado * dpi) / 25.4

    # Inicializar las variables
    mejor_nivel = None
    mejor_diferencia = float('inf')


    for nivel in range(0, 256):

        # Indice del perfil en el que se cumple que el valor de iluminación es
        # menor al guardado en nivel
        indices = np.where(perfil < nivel)[0]

        # Si no hay dos indices guardados no hay valle y no se puede medir
        if len(indices) < 2:
            continue

        # Cuando ya tenemos los dos puntos, ya se puede calcular la distancie
        # entre ambos
        ancho_valle = indices[-1] - indices[0]
        diferencia = abs(ancho_valle - ancho_px_deseado)

        # Registro de las distancias. Solo se actualiza si la distancia medida
        # se asemeja más a la deseada que la anterior
        if diferencia < mejor_diferencia:
            mejor_diferencia = diferencia
            mejor_nivel = nivel

    return mejor_nivel

# FUNCIÓN DE BINARIZACIÓN DE LA PIEZA A MEDIR
def Tratamiento_Imagen(I, umbral):

    # Convertir la imagen a escala de grises
    I_gray = cv2.cvtColor(I, cv2.COLOR_BGR2GRAY)

    umbral = umbral

    # Binarización de la imagen conociendo ya el umbral con el que se debe
    # realizar
    _, I_final = cv2.threshold(I_gray, umbral, 255, cv2.THRESH_BINARY_INV)

    return I_final, I_gray


# FUNCIÓN EXTRACCIÓN DE PROPIEDADES
def Etiquetar(BW_image, anchura_min_mm, pixel_a_mm):

    # Etiquetar los objetos detectados en la imagen original
    L = label(BW_image)

    # Extraer propiedades de cada región
    propiedades = regionprops(L)
```

```
    propiedades = propiedades[1:]

    # Calcular altura mínima
    tolerancia_mm = 0.2

    # Convertir a píxeles
    anchura_min_px = (anchura_min_mm - tolerancia_mm) / pixel_a_mm
    anchura_max_px = 40 / pixel_a_mm

    # Crear imagen de salida vacía
    BW_filtrada = np.zeros_like(BW_image, dtype=np.uint8)

    # Lista para almacenar propiedades filtradas
    propiedades_filtradas = []

    # Filtrar regiones por altura
    for prop in propiedades:
        fila_min, colum_min, fila_max, colum_max = prop.bbox
        anchura = colum_max - colum_min

        if anchura >= anchura_min_px and anchura <= anchura_max_px:
            BW_filtrada[L == prop.label] = 255
            propiedades_filtradas.append(prop)

    # Reetiquetar la imagen filtrada
    L_filtrado = label(BW_filtrada)
    propiedades_actualizadas = regionprops(L_filtrado)


    # Devolver propiedades actualizadas (solo regiones válidas)
    return propiedades_actualizadas


# FUNCIÓN PANTALLA INTERACTIVA
def click(evento, x, y, flags, parametros):

    # Guardar los valores de parametros en sus variables correspondientes
    propiedades, escala, pixel_a_mm, pixel2_a_mm2, k_v, k_h = parametros

    # Solo se ejecuta esta parte del código cuando se hace clic con el botón
    # izquierdo sobre la pantalla que se abre
    if evento == cv2.EVENT_LBUTTONDOWN:

        # Bucle para recorrer todas las regiones detectadas
        for i, prop in enumerate(propiedades):

            # Coordenadas del centroide de las regiones y teniendo en cuenta
            # la escala
            cy, cx = prop.centroid
            cx_scaled = int(cx * escala)
            cy_scaled = int(cy * escala)

            # Distancia desde donde se hace el click hasta el centroide de la
            # región
            distancia = np.sqrt((x - cx_scaled)**2 + (y - cy_scaled)**2)
```

```
# Se tolera un desfase de 15 píxeles de error
if distancia < 15:

    # Se obtienen las propiedades de la región y se pasan a mm
    area_mm2 = prop.area * pixel2_a_mm2 * k_v * k_h
    orientacion = prop.orientation # En radianes
    excentricidad = prop.eccentricity
    centroide = prop.centroid
    fila_min, colum_min, fila_max, colum_max = prop.bbox

    # Coordenadas del centroide convertidas a mm
    cx_mm = centroide[1] * pixel_a_mm * k_v
    cy_mm = centroide[0] * pixel_a_mm * k_h

    # CIRCUNFERENCIA
    if excentricidad < 0.6:

        # Cálculo del radio
        radio_mm = np.sqrt(area_mm2 / np.pi)

        # Crear el mensaje con los datos que se quieren mostrar
        mensaje = (
            f"Objeto {i+1} - CÍRCULO\n"
            f"Área: {area_mm2:.4f} mm²\n"
            f"Radio: {radio_mm:.4f} mm\n"
            f"Excentricidad: {excentricidad:.4f}\n"
            f"Posición: ({cx_mm:.4f}, {cy_mm:.4f}) mm\n"
            f"Orientación: {np.degrees(orientacion):.4f}°"
        )

    # RECTÁNGULO
    else:

        # Cálculo de altura y anchura en píxeles
        altura_px = fila_max - fila_min
        ancho_px = colum_max - colum_min

        # Corrección teniendo en cuenta la orientación
        sin_orient = np.abs(np.sin(orientacion))
        altura_mm = altura_px * pixel_a_mm * sin_orient * k_v
        ancho_mm = ancho_px * pixel_a_mm * sin_orient * k_h

        # Crear el mensaje con los datos que se quieren mostrar
        mensaje = (
            f"Objeto {i+1} - RECTÁNGULO\n"
            f"Área: {area_mm2:.4f} mm²\n"
            f"Lados: {ancho_mm:.4f} x {altura_mm:.4f} mm\n"
            f"Excentricidad: {excentricidad:.4f}\n"
            f"Posición: ({cx_mm:.4f}, {cy_mm:.4f}) mm\n"
            f"Orientación: {np.degrees(orientacion):.4f}°"
        )

    # Creación de la ventana emergente con la información
    root = tk.Tk()
    root.withdraw()
    messagebox.showinfo("Información del objeto", mensaje)
```

```
                break


# FUNCIÓN MOSTRAR LA INFORMACIÓN INTERACTIVA
def mostrar_info_interactiva(I_color, propiedades, pixel_a_mm, pixel2_a_mm2,
                             k_v, k_h):

    # Ancho máximo de la imagen
    max_ancho = 700

    # Altura y anchura de la imagen
    alto, ancho = I_color.shape[:2]

    # Cálculo de la escala
    escala = min(1.0, max_ancho / ancho)

    # Redimensionar la imagen y se pasa a formato RGB
    I_redimen = cv2.resize(I_color.copy(), (int(ancho * escala),
                                            int(alto * escala)))
    I_RGB = cv2.cvtColor(I_redimen, cv2.COLOR_BGR2RGB)

    # Dibujar los números sobre cada centroide escalado
    for i, prop in enumerate(propiedades):
        y, x = prop.centroid
        x = int(x * escala)
        y = int(y * escala)
        numero = str(i + 1)
        font = cv2.FONT_HERSHEY_SIMPLEX
        font_scale = 0.6
        thickness = 2

        # Medir tamaño del texto
        (text_width, text_height), _ = cv2.getTextSize(numero, font,
                                                       font_scale, thickness)

        # Ajustar para centrar el texto en el centroide
        x_text = int(x - text_width / 2)
        y_text = int(y + text_height / 2)

        # Se muestra en rojo el número sobre las regiones
        cv2.putText(I_RGB, numero, (x_text, y_text), font, font_scale,
                    (255, 0, 0), thickness)

    # Creación de la ventana donde se va a mostrar la imagen
    cv2.namedWindow("Haz clic en un numero", cv2.WINDOW_NORMAL)

    # Asociación con la función de hacer clic
    cv2.setMouseCallback("Haz clic en un numero", click,
                         param=(propiedades, escala, pixel_a_mm, pixel2_a_mm2,
                                k_v, k_h))

    # Hasta que no se pulsa ESC se mantiene en la ventana mostrando la imagen
    while True:
        cv2.imshow("Haz clic en un numero", cv2.cvtColor(I_RGB,
                                                        cv2.COLOR_RGB2BGR))
        if cv2.waitKey(20) & 0xFF == 27:  # Código ASCII de ESC
```

```
            break

    # Cierre de la ventana generada
    cv2.destroyAllWindows()


# FUNCIÓN QUE GENERA LA TABLA CON LOS RESULTADOS
def generar_tabla_resultados(propiedades, pixel_a_mm, pixel2_a_mm2, k_v, k_h):

    # Inicializar lista de datos
    datos = []

    for i, prop in enumerate(propiedades):

        # Obtención de las propiedades y pasarlas a mm, mm^2 y º
        area_mm2 = prop.area * pixel2_a_mm2 * k_v * k_h
        excentricidad = prop.eccentricity
        orientacion = np.degrees(prop.orientation)
        centroide = prop.centroid
        cx_mm = centroide[1] * pixel_a_mm * k_v
        cy_mm = centroide[0] * pixel_a_mm * k_h

        # CIRCUNFERENCIA
        if excentricidad < 0.6:

            tipo = "Círculo"

            # Calculo del radio y no se tiene en cuenta el ancho y el alto
            radio_mm = np.sqrt(area_mm2 / np.pi)
            ancho_mm = alto_mm = np.nan
            desviacion_area = np.nan

        # RECTANGULO
        else:

            tipo = "Rectángulo"

            # Calculo de altura y anchura y tener en cuenta la orientación
            # de la pieza
            min_fila, min_col, max_fila, max_col = prop.bbox
            altura_px = max_fila - min_fila
            ancho_px = max_col - min_col
            sin_orient = np.abs(np.sin(prop.orientation))
            alto_mm = altura_px * pixel_a_mm * sin_orient * k_v
            ancho_mm = ancho_px * pixel_a_mm * sin_orient * k_h
            radio_mm = np.nan

            # Área ideal de un rectángulo perfecto
            area_ideal = alto_mm * ancho_mm

            # Desviación relativa en porcentaje
            desviacion_area = ((area_mm2 - area_ideal) / area_ideal) * 100

        # Guardar en la lista de datos un diccionario con las propiedades
        datos.append({
            "Objeto": i + 1,
```

```
            "Tipo": tipo,
            "Área [mm²]": area_mm2,
            "Radio [mm]": radio_mm if tipo == "Círculo" else np.nan,
            "Ancho [mm]": ancho_mm if tipo == "Rectángulo" else np.nan,
            "Alto [mm]": alto_mm if tipo == "Rectángulo" else np.nan,
            "Desviación Área [%]": desviacion_area,
            "Excentricidad": excentricidad,
            "Orientación [°]": orientacion,
            "Centro X [mm]": cx_mm,
            "Centro Y [mm]": cy_mm
        })

    #♣ Guardar los resultados teniendo en cuenta las tabulaciones y estructura
    resultados = pd.DataFrame(datos)

    return resultados

# FUNCIÓN QUE GUARDA LOS DATOS DE LA TABLA EN UN EXCEL
def guardar_tabla_en_excel(tabla, nombre_archivo="Resultado_Pieza.xlsx"):

    # Si el libro ya está creado
    if os.path.exists(nombre_archivo):

        # Abrir archivo existente
        wb = load_workbook(nombre_archivo)

    else:

        # Crear un nuevo libro si no existe
        wb = Workbook()

    # Establecer el nombre de la hoja
    titulo = "Resultados"
    idx = 1
    while f"{titulo} {idx}" in wb.sheetnames:
        idx += 1
    nombre_hoja = f"{titulo} {idx}"

    # Crear la hoja nueva
    hoja = wb.create_sheet(title=nombre_hoja)

    # Combinar celdas para el título de la tabla
    hoja.merge_cells(start_row=1, start_column=1, end_row=1,
                     end_column=len(tabla.columns))
    celda_titulo = hoja.cell(row=1, column=1)

    # Escribir el titulo de la tabla
    celda_titulo.value = "Resultados de la pieza escaneada"

    # Fuente y alineación de la celda del titulo de la tabla
    celda_titulo.font = Font(bold=True)
    celda_titulo.alignment = Alignment(horizontal="center", vertical="center")
    celda_titulo.fill = PatternFill(start_color="DDDDDD", end_color="DDDDDD",
                                    fill_type="solid")

    # Escribir encabezados y datos desde la fila 2
```

```
    for r_idx, row in enumerate(dataframe_to_rows(tabla, index=False, header=True), 2):
        for c_idx, value in enumerate(row, 1):
            cell = hoja.cell(row = r_idx, column = c_idx, value = value)
            # Alineación de las celdas
            cell.alignment = Alignment(horizontal='center', vertical='center')
            if r_idx == 2:
                # Fuente del encabezado de la tabla (negrita)
                cell.font = Font(bold=True)

    # Ajustar ancho de columnas
    for col_idx, col in enumerate(hoja.columns, 1):
        max_length = max(len(str(cell.value)) for cell in col if cell.value)
        col_letter = get_column_letter(col_idx)
        hoja.column_dimensions[col_letter].width = max_length + 2

    # Añadir bordes
    thin = Side(border_style="thin", color="000000")
    for row in hoja.iter_rows(min_row=1, max_row = hoja.max_row, min_col=1,
                              max_col = hoja.max_column):
        for cell in row:
            cell.border = Border(top=thin, left=thin, right=thin, bottom=thin)

    # Guardar cambios
    wb.save(nombre_archivo)

# FUNCIÓN MAIN
def main():

    # Cerrar todas las ventanas
    plt.close('all')

    # -------------- REQUERIR INFORMACIÓN POR LA TERMINAL --------------

    # Preguntar el nombre del archivo
    nombre_archivo = input("Introducir el nombre del archivo "
                           "de imagen (incluyendo extensión): ")

    # Preguntar la distancia mínima
    anchura_min_mm = float(input("Introduce la anchura mínima "
                                 "del objeto (en milímetros): "))

    # Preguntar por el tipo de calibración
    print("\n********** MENÚ DE CALIBRACIÓN **********\n")
    print("1. Marco de calibración en papel")
    print("2. Marco de calibración en pieza")
    print("\n*********************************\n")

    opcion = input("\nElige la opción que desee: ")

    # Leer la imagen escaneada
    I = cv2.imread(nombre_archivo)

    # ----------------- DEFINIR VARIABLES NECESARIAS -------------------

    # DPI
    dpi = 4800
```

```python
# Relaciónes pixel-milímetro
px_por_mm = dpi / 25.4
pixel_a_mm = 25.4 / dpi
pixel2_a_mm2 = pixel_a_mm ** 2

# Medidas conocidas
ancho_mm_deseado = 20
alto_mm_deseado = 20

# Factores de corrección
k_h = 1
k_v = 1

# --------------- PROCESO DE TRATAMIENTO DE LA IMAGEN ---------------

# PASO 1: SEPARA IMAGEN DE CALIBRACIÓN Y ESTUDIO

pieza,calibracion = recortar_piezas_por_posicion(I)


# PASO 2: CALIBRACIÓN
resultados,BW_cali,I_gray_cali,vis_cali = (Tratamiento_Imagen_calibracion(
    calibracion, px_por_mm, alto_mm_deseado, ancho_mm_deseado))

# En función del tipo de calibración calcular los factores de corrección
# o el umbral
match opcion:
    case "1":

        k_v, k_h = calcular_factores_correccion(resultados,
                                                ancho_mm_deseado,
                                                alto_mm_deseado,
                                                px_por_mm)

        nivel_medio =  145

    case "2":

        # Obtener los perfiles de intensidad y guardar el del centro
        # de la pieza para realizar el estudio
        perfiles_h, perfiles_v = perfil_intensidad_por_region(I_gray_cali,
                                                              BW_cali)
        perfil_h_r4 = perfiles_h[0][0]
        perfil_v_r4 = perfiles_v[0][0]

        # Cálculo del nivel de binarización horizontal y vertical y
        # obtener la media para binarizar la pieza que se quiere estudiar
        nivel_ancho = Nivel_binarizacion(perfil_h_r4,
                                         ancho_mm_deseado, dpi)
        nivel_alto = Nivel_binarizacion(perfil_v_r4,
                                        alto_mm_deseado, dpi)
        nivel_medio = int(round((nivel_ancho + nivel_alto)/2))

    case _:
        print("Opción inválida")
```

```
# PASO 3: TRATAMIENTO DE LA PIEZA

BW_pieza,I_gray_pieza = Tratamiento_Imagen(pieza,nivel_medio)

# PASO 4: DETECCIÓN DE REGIONES Y EXTRACCIÓN DE PROPIEDADES

propiedades = Etiquetar(BW_pieza, anchura_min_mm, pixel_a_mm)

# PASO 5: SALIDAS

while True:
    print("\n********** MENÚ DE SALIDA **********\n")
    print("1. Imagen original enumerada y ampliación de regiones")
    print("2. Pantalla interactiva")
    print("3. Excel")
    print("4. Salir")
    print("\n************************************\n")

    opcion = input("\nElige la opción que desee: ")

    # Generación imagen enumerada y ampliación de las regiones detectadas
    if opcion == "1":

        # Obtener centroides de las regiones filtradas
        centros = np.array([prop.centroid for prop in propiedades])

        # Convertir la imagen de BGR a RGB
        I_rgb = cv2.cvtColor(pieza, cv2.COLOR_BGR2RGB)

        # Mostrar la imagen en RGB
        plt.figure(figsize=(8, 6))
        plt.imshow(I_rgb)
        plt.title("Regiones etiquetadas")
        plt.axis('off')

        # Añadir número en el centroide de cada región filtrada
        for i, (y, x) in enumerate(centros):
            plt.text(x, y, str(i + 1), color='red', ha='center',
                     va='center', weight='bold')
        plt.show()

        # Número de regiones
        num_regiones = len(propiedades)

        # Calcular el tamaño del grid
        cols = 5
        rows = (num_regiones + cols - 1) // cols  # redondeo hacia arriba
        margen = 10

        # Dibujar las regiones detectadas
        plt.figure(figsize=(3*cols, 3*rows))

        for i, prop in enumerate(propiedades):
            min_fila, min_col, max_fila, max_col = prop.bbox
```

```
                # Ampliar la bbox con margen y asegurarse de no salirse
                min_fila = max(min_fila - margen, 0)
                min_col = max(min_col - margen, 0)
                max_fila = min(max_fila + margen, BW_pieza.shape[0])
                max_col = min(max_col + margen, BW_pieza.shape[1])

                # Recortar la región de la imagen binarizada
                region = BW_pieza[min_fila:max_fila, min_col:max_col]

                # Mostrar cada región en un subplot
                plt.subplot(rows, cols, i + 1)
                plt.imshow(region, cmap='gray')
                plt.title(f'Región {i+1}')
                plt.axis('off')

        # Generar la ventana interactiva
        elif opcion == "2":

            # Llamar a la función que genera la ventana que muestra la imagen
            # con los datos de las regiones
            mostrar_info_interactiva(pieza, propiedades, pixel_a_mm,
                                     pixel2_a_mm2, k_v, k_h)

        # Generar excel
        elif opcion == "3":

            # Llamar a la función de generar una tabla con los resultados
            # de cada región
            tabla = generar_tabla_resultados(propiedades, pixel_a_mm,
                                             pixel2_a_mm2, k_v, k_h)

            # Llamar a la función que genera el Excel con los datos guardados
            guardar_tabla_en_excel(tabla, "Resultado_Pieza.xlsx")

        # Salir del programa
        elif opcion == "4":

            print("\nSalir del programa")
            break
        else:
            print("\nOpción no valida")


if __name__ == "__main__":
    main()
```