UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

INGENIERÍA DE COMPUTADORES

Caracterización de granos de trigo mediante Visión por Computador
Characterization of wheat grains using Computer Vision

Realizado por
ALEJANDRO VILLALBA FERNÁNDEZ

Tutorizado por
DR. D. SERGIO GÁLVEZ ROJAS

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

MÁLAGA, Septiembre, 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
INGENIERÍA DE COMPUTADORES

Caracterización de granos de trigo mediante Visión por Computador

Characterization of wheat grains using Computer Vision

Realizado por
Alejandro Villalba Fernández

Tutorizado por
Sergio Gálvez Rojas

Departamento
Lenguajes y Ciencias de la computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

Fecha defensa: X de septiembre de 2023

Agradecimientos

Agradecer con sinceridad a mi tutor el Dr. D. Sergio Gálvez Rojas por su dedicación, paciencia y guía frente a este proyecto. Gracias por su profesionalidad y buena actitud incluso en las malas épocas. Me ha demostrado no solamente la pasión que tiene por enseñar sino lo importante que es en cualquier ámbito de la vida el ser una buena persona.

Gracias a mis compañeros de carrera porque sin algunos de ellos el camino hasta llegar aquí hubiese sido mucho más tedioso y complicado. De ellos me llevo una enseñanza muy importante: el trabajo en equipo siempre va a ser mejor que trabajar solo.

Para terminar, agradecer al mayor soporte que cualquier estudiante debería de tener, mis padres, por el apoyo incondicional ante cualquier situación o barrera y por brindarme su ayuda para cualquier necesidad que surgiese. Por animarme en las peores épocas y por celebrar conmigo las mejores. A ellos les debo el camino hasta aquí recorrido y este trabajo.

Resumen

El trabajo consiste en la creación de un programa en lenguaje Python que, dada una imagen con los granos de trigo a analizar, identifica cada una de las semillas por separado y obtiene ciertas características de las mismas con el objetivo de crear un dataset de forma pseudo-automática. Se realizan pruebas con dos tipos de imágenes: procedentes de cámara digital y procedentes de escáner.

El trabajo se divide en dos partes: en la primera fase, que aborda la identificación individual de cada grano, se logra que, a partir de una imagen con fondo negro, con casillas dibujadas de 1x1cm y donde en el centro de cada casilla se sitúa un grano, se apliquen una serie de filtros para detectar los bordes de cada grano y se obtenga por separado una imagen de cada uno de ellos.

La segunda fase, centrada en la obtención de características, implica la creación de un dataset con alrededor de 15-30 características basadas en 4 aspectos principales: forma, color, tamaño y textura. El dataset así obtenido, que incluye valores estadísticos básicos, constituye el punto de partida para análisis y estudios subsiguientes fuera del ámbito de este trabajo. Además, el mecanismo pseudo-automático se emplea en el análisis de nuevas muestras.

Palabras Claves: Reconocimiento, Análisis, Detección, Python, Granos de Trigo, Visión por Ordenador.

Abstract

The work involves the creation of a program in Python language that, given an image with wheat grains to analyze, identifies each seed individually and obtains certain characteristics of them with the aim of creating a dataset in a pseudo-automatic manner. Tests are conducted with two types of images: those from a digital camera and those from a scanner.

The work is divided into two parts: in the first phase, which addresses the individual identification of each grain, it is achieved that, from an image with a black background, with 1x1cm squares drawn and where a grain is placed in the center of each square, a series of filters are applied to detect the edges of each grain and an image of each of them is obtained separately.

The second phase, focused on feature extraction, involves the creation of a dataset with around 15-30 characteristics based on 4 main aspects: shape, color, size, and texture. The resulting dataset, which includes basic statistical values, serves as the starting point for subsequent analyses and studies outside the scope of this work. Additionally, the pseudo-automatic mechanism is used in the analysis of new samples.

Keywords: Recognition, Analysis, Detection, Python, Wheat, Computer Vision.

Índice

Resumen **1**

Abstract **1**

Índice **1**

Introducción **3**
1.1 Motivación **3**
1.2 Objetivos **4**
1.3 Estudio de tecnologías **4**
1.4 Metodología de trabajo **6**

Estudio del estado del arte **8**
2.1 Detección de contornos **8**
2.2 Preprocesamiento de Imágenes **9**
2.2.1 Adquisición y corrección de la imagen 10
2.2.2 Conversión a Escala de Grises 10
2.2.3 Filtrado y eliminación de ruido 10
2.2.4 Mejora de contraste 10
2.2.5 Binarización 10
2.2.6 Morfología matemática 11
2.3 Relevancia en la caracterización de granos de trigo **11**
2.4 FiJi y GrainScan **12**
FiJi (ImageJ) 12
GrainScan 14

Análisis **16**
3.1 Requisitos **16**
3.2 Herramientas **17**

Diseño **22**

Implementación **27**
5.1 Segmentación **27**
5.1.1 Estructura Principal y Menús 27
5.1.2 Panel de control 28
5.1.3 Funcionalidad de Dibujo en el Canvas 29
5.1.4 Análisis y Conteo 29
5.1.5 Actualización de Umbral 29
5.1.6 Información del Estado de la Aplicación 30
5.1.7 Funcionalidades de Dibujo y Medición 30
5.1.8 Funciones de Procesamiento de Imagen 30
5.1.9 Análisis de Datos 32
5.1.10 Funciones auxiliares 32
5.1.11 grain_detector 33
5.1.12 Clase ToolTip: 34
5.2 Obtención de características cuantificables **36**

5.2.1 Detalles de la clase Grano: 36
Verificación 43
6.1 Objetivo Principal 43
6.2 Pruebas 43
6.4 Importancia de la Verificación 44
Creación del dataset 45
Conclusiones y trabajo futuro 47
8.1 Conclusiones 47
8.2 Trabajo Futuro 48
Referencias 50
Manual de Usuario 51
I. Interfaz de Usuario 52
I.2 Análisis de Resultados 57

1

Introducción

1.1 Motivación

El trigo ha sido durante milenios el sustento básico para muchas civilizaciones alrededor del mundo. Su relevancia en la economía agraria global es incuestionable, habiéndose convertido en un pilar fundamental en la dieta humana. Pero más allá de su valor nutricional, el trigo ha sido objeto de una evolución constante, pasando por procesos de domesticación que han engendrado una amplia gama de variedades. Cada una de estas variedades, adaptadas a diferentes condiciones climáticas y geográficas, posee propiedades singulares que la hacen apta para distintos usos culinarios, desde el pan hasta la pasta pasando incluso por el cus-cus.

En el contexto globalizado actual, donde la trazabilidad y autenticidad de los productos es esencial, poder identificar y diferenciar estas variedades de trigo se convierte en una herramienta crucial. No solo para garantizar la calidad del grano y evitar posibles fraudes comerciales, sino también para optimizar su utilización en función de sus características intrínsecas. Sin embargo, el proceso manual de clasificación y análisis puede resultar tedioso y propenso a errores. En este escenario, la tecnología y la informática ofrecen soluciones innovadoras para enfrentar este desafío.

Figura 1.1 Ejemplo de tipo de imagen con la que trabaja la aplicación.

1.2 Objetivos

Este Trabajo de Fin de Grado se embarca en la misión de desarrollar una herramienta tecnológica que facilite el proceso de identificación y análisis de granos de trigo. Utilizando Python, un lenguaje de programación ampliamente reconocido por su versatilidad y capacidad para manejar datos, se propone la creación de un programa que, a partir de imágenes de los granos, sea capaz de individualizar cada semilla y extraer características esenciales. Estas características son relevantes no solo a nivel directo, sino que también ayudan de manera indirecta a abordar propiedades más profundas como el contenido proteico, el nivel de gluten y la presencia de carotenos. Como por ejemplo, obtener la gama de colores de un grano puede ser de gran utilidad si se quiere saber si el grano está más o menos hidratado, si es una cosecha con sequías, la procedencia del grano… etcétera.

Figura 1.2: Ejemplo de detección de personas usando openCV

El enfoque metodológico del proyecto se estructura en dos fases esenciales. La primera está dedicada a la identificación y segmentación de cada grano a partir de imágenes específicas. La segunda fase se centra en la extracción de características fundamentales de los granos. El resultado será un conjunto de datos o dataset, que, enriquecido con valores estadísticos, proporcionará una base sólida para futuras investigaciones y aplicaciones en el campo de la agroindustria.

La finalidad de este Trabajo Fin de Grado es ofrecer una solución tecnológica que contribuya a la eficiencia y precisión en el análisis del trigo, potenciando la calidad y trazabilidad en la cadena de suministro y abriendo la puerta a investigaciones futuras en el ámbito agronómico y alimentario.

Se acabará esta introducción hablando del estudio de las tecnologías usadas y la metodología de trabajo empleada.

1.3 Estudio de tecnologías

La visión por computador ha experimentado un crecimiento exponencial en las últimas décadas, propiciado en gran parte por los avances en la computación y la disponibilidad de herramientas de programación especializadas. En el contexto del análisis y clasificación de granos de trigo, diversas tecnologías y conceptos emergen como elementos clave para abordar con precisión y eficiencia este desafío.

Python. Este lenguaje de programación de alto nivel, conocido por su sintaxis clara y legible, se ha consolidado como una de las principales herramientas en el ámbito de la ciencia de datos y la inteligencia artificial. Su versatilidad y la amplia variedad de bibliotecas disponibles lo convierten en una opción ideal para desarrollar aplicaciones complejas relacionadas con la visión por computador. Además, Python facilita la integración con otras tecnologías y herramientas, permitiendo la construcción de sistemas robustos y escalables. Para el estudio de este lenguaje de programación se ha hecho una lectura y seguimiento de las actividades que plantea el libro (Ramírez Jiménez, 2021) y para el desarrollo de interfaces gráficas de usuario se ha utilizado el libro (Meier, 2019) con el que se ha empezado a trabajar en la parte gráfica de la aplicación. También se ha utilizado el repositorio de código de dicho libro que puede hallarse en (Meier B. , 2019)

OpenCV. Open Source Computer Vision Library (OpenCV, 2023) es una de las bibliotecas más populares y completas para la visión por computador. Escrita en C++ y con bindings para Python, ofrece una amplia gama de herramientas y técnicas para procesar imágenes y vídeos, desde operaciones básicas de filtrado y transformación hasta algoritmos avanzados de reconocimiento y aprendizaje automático. En el contexto del análisis de granos de trigo, OpenCV puede ser utilizado para segmentar, identificar y clasificar cada grano, extrayendo características esenciales y proporcionando datos precisos para su posterior análisis.

Tkinter. Es el paquete estándar de Python para la creación de interfaces gráficas de usuario (GUI). Su sencillez y eficacia lo convierten en una herramienta indispensable para desarrollar aplicaciones interactivas, permitiendo a los usuarios interactuar de manera intuitiva con sistemas complejos. En el caso del programa propuesto para analizar granos de trigo, Tkinter podría ser empleado para visualizar imágenes, ajustar parámetros y presentar resultados de manera estructurada y accesible. La documentación puede encontrarse en (Tkinter, 2023)

PIL (Python Imaging Library). (Pillow, 2023) Aunque ahora se conoce más comúnmente como "Pillow" tras su fork, esta biblioteca es esencial para abrir, manipular y guardar distintos formatos de imagen en Python. Junto con OpenCV, Pillow proporciona herramientas adicionales para el procesamiento y análisis de imágenes, tales como operaciones con imágenes en modo RGB y técnicas específicas de filtrado y transformación.

Conceptos de Visión por Computador. La visión por computador engloba una serie de técnicas y conceptos que permiten a las máquinas interpretar y tomar decisiones basadas en imágenes y vídeos. Algunos de estos conceptos, como la segmentación de imágenes, detección de bordes, filtrado y extracción de características, son fundamentales para identificar y analizar cada grano de trigo de manera individual. La capacidad de diferenciar texturas, formas y colores a través de algoritmos permite obtener información detallada sobre cada grano, desde su variedad hasta su calidad y propiedades intrínsecas. El alumno ha cursado también la asignatura de procesamiento de Imágenes y Vídeos donde se dan conceptos de visión por computador que se ha aplicado en este trabajo como por ejemplo el paso de las imágenes a escala de grises

para facilitar los procesos que vienen a continuación como por ejemplo la detección de contornos de los granos.

En resumen, la combinación de estas tecnologías y conceptos abre un abanico de posibilidades en el ámbito de la agroindustria. La fusión de Python, OpenCV, Tkinter y PIL, junto con técnicas avanzadas de visión por computador, ofrece un enfoque innovador para el análisis y clasificación de granos de trigo, potenciando la calidad, trazabilidad y eficiencia en la cadena de suministro agrícola.

1.4 Metodología de trabajo

La metodología de trabajo adoptada para este proyecto se basa en SCRUM, una de las metodologías ágiles más populares y ampliamente adoptadas en la industria del software.
SCRUM es especialmente conocido por su flexibilidad y capacidad para adaptarse a cambios, lo cual es crucial en proyectos donde los requisitos pueden evolucionar o no estar completamente definidos desde el principio.

El trabajo se organiza en sprints o iteraciones de dos semanas. Estos sprints son períodos de tiempo durante los cuales se establecen objetivos claros y se trabaja para alcanzarlos. Al finalizar cada sprint, se lleva a cabo una reunión de revisión con el tutor.

Estas reuniones son esenciales para el proceso, ya que ofrecen una oportunidad para evaluar y demostrar el trabajo realizado durante el sprint, recibir retroalimentación directa del tutor, que actúa como el cliente, revisar los requisitos ya implementados y garantizar que se alinean con las expectativas y por último determinar y acordar los próximos pasos y requisitos a implementar en el siguiente sprint.

Dichas reuniones han sido cruciales para que el proyecto se mantenga en el camino correcto y que cualquier desviación o malentendido a la hora de asimilar los conceptos se corrijan lo antes posible.

Aunque las reuniones fuesen cada dos semanas, se ha establecido una comunicación constante en caso de una duda menor.

A continuación, se hablará sobre las diferentes fases en las que se ha dividido el proyecto:

Estudio del estado del arte: Antes de sumergirse en la solución del problema, es fundamental conocer las soluciones existentes. En esta fase, se realiza una investigación exhaustiva de aplicaciones y software que tengan objetivos similares. Este estudio permite identificar las mejores prácticas, evitar reinventar la rueda y, posiblemente, descubrir características adicionales o enfoques innovadores que se podrían incorporar.

Análisis: Una vez que se tiene una comprensión clara del panorama general, la siguiente etapa implica documentar los requisitos del proyecto y cómo se planea abordar el problema. Esta fase es esencial para definir claramente el alcance del proyecto y las expectativas. Es en esta etapa donde se estudiaron las opciones que se tenían a mano

respecto a qué librería usar para crear la interfaz gráfica de la aplicación, qué otras librerías serían útiles para facilitar el trabajo más adelante y qué IDE se escogería para trabajar.

Diseño: Con los requisitos en mano, se procede a planificar la estructura general del software. Se evalúan diferentes opciones de implementación, se consideran patrones de diseño apropiados y se establece la arquitectura del sistema.

Implementación: Es aquí donde el diseño cobra vida. Esta fase se subdivide en dos partes:

Segmentación: Sección donde se aborda la identificación y demarcación de áreas específicas en las imágenes.
Primero se empezó a hacer una interfaz gráfica muy básica pero funcional donde sólo las opciones de cerrar la aplicación y escoger una imagen funcionaban hasta el momento. Una vez se tuvo el esqueleto de la interfaz gráfica se empezó a trabajar en el algoritmo de detección de granos. Una vez que dicho algoritmo funcionaba se retocó la interfaz gráfica para que fuese más atractiva y pulir algunos errores o funcionalidades con resultados no deseados.

Obtención de características cuantificables: Una vez segmentadas las imágenes, se extraen características y métricas específicas de las áreas identificadas. Con una interfaz ya decente y un algoritmo que funciona bien, comenzó la etapa de creación de funciones para crear un dataset a partir de los granos de trigo detectados.

Durante la implementación, es fundamental considerar aspectos como el manejo de errores, la validación de parámetros de entrada y la optimización del rendimiento. También es importante comentar el código para tener un recuerdo de qué o cómo se había planteado el código no solamente para el tutor sino para el propio alumno ya que, al ser un trabajo de meses, es fácil que se olviden ciertos detalles.

Verificación: Una vez desarrollado el software, es crucial asegurarse de que funcione como se espera. Para ello, se realizan pruebas unitarias que comprueban la funcionalidad de cada componente individualmente y su comportamiento con diferentes sets de imágenes.

Creación del dataset: Con el software verificado, se procede a aplicar los procedimientos definidos a un conjunto de imágenes de entrada. El objetivo es producir un dataset compuesto por las métricas y características extraídas de estas imágenes.

Figura 1.3: Ejemplo de detección de objetos y trazado de los contornos usando openCV

2

Estudio del estado del arte

La caracterización de granos de trigo a través de la visión por computador es un campo que ha ganado relevancia en la última década, permitiendo una mejor clasificación y análisis de estos granos en base a sus propiedades físicas. Para adentrarse en este campo, es esencial comprender los fundamentos de la detección de contornos y el preprocesamiento de imágenes. El siguiente capítulo ofrece una revisión detallada de estos temas y su relevancia en la caracterización de granos de trigo.

2.1 Detección de contornos

Los contornos son discontinuidades significativas en una imagen y son esenciales para identificar y separar objetos en una imagen. En el contexto de granos de trigo, los contornos permiten identificar y separar individualmente cada grano en una muestra.

Existen múltiples técnicas y algoritmos que se centran en la detección de contornos, entre los más populares y ampliamente adoptados se encuentran:

- **Operador de Sobel:** Utiliza núcleos convolucionales para detectar rápidamente los cambios horizontales y verticales en la intensidad de la imagen.

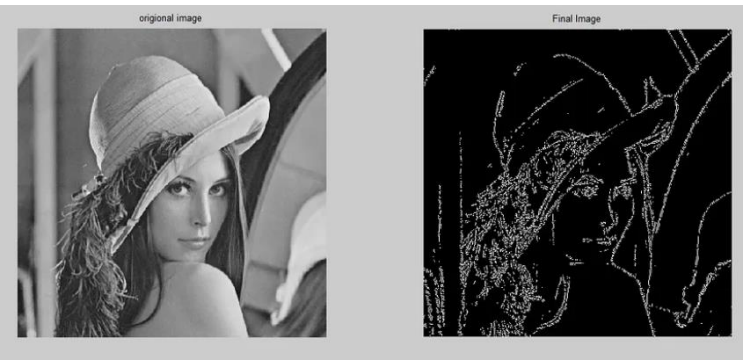


Figura 2.1: Imagen tratada con el operador Sobel

- **Operador de Canny:** Reconocido por su capacidad para detectar una amplia variedad de bordes en imágenes. Funciona mediante la eliminación de ruido, el cálculo del gradiente de la imagen y la supresión de no máximos.

Figura 2.2: En la figura de la izquierda tenemos una imagen borrosa y en la segunda y tercera tenemos la imagen con el operador de Canny aplicado con diferentes parámetros

- **Detección de bordes de Prewitt:** Similar al operador de Sobel, pero utiliza un kernel diferente.

Figura 2.3: Imagen original y la misma imagen pero con el detector de Prewitt aplicado

- **Algoritmo Suzuki:** Se centra en la identificación de contornos jerárquicos en imágenes binarias. Esto significa que no solo detecta los contornos externos de los objetos, sino también los contornos internos, lo cual es útil para identificar objetos que están dentro de otros objetos.

En el caso de este proyecto, la librería con la que se ha trabajado, es decir, OpenCV, tiene una función llamada "findContours()" que usa el algoritmo de Suzuki para la detección de contornos pero lo combina con el operador Canny para aplicar la detección de bordes.

2.2 Preprocesamiento de Imágenes

El preprocesamiento de imágenes es una etapa esencial antes de cualquier operación de análisis o interpretación de la imagen. Esta etapa prepara y mejora la calidad de la imagen, garantizando que las características relevantes estén bien definidas y que las no deseadas se minimicen o eliminen. Aquí se describen las fases y técnicas más comunes en el preprocesamiento de imágenes:

2.2.1 Adquisición y corrección de la imagen

Antes de cualquier operación, se debe obtener una imagen de calidad. Las imágenes pueden presentar defectos debido a la calidad de la cámara, las condiciones de iluminación o incluso el movimiento. Algunas correcciones comunes incluyen:

- **Corrección de Iluminación:** Ajustar la iluminación de la imagen para garantizar una visibilidad uniforme.

- **Corrección de Distorsión:** Las lentes de la cámara a veces pueden distorsionar la imagen. La corrección de distorsión trata de rectificar estas deformaciones.

- **Estabilización de Imágenes:** En casos donde el movimiento pueda ser un problema, se utiliza estabilización para reducir el desenfoque.

2.2.2 Conversión a Escala de Grises

Como se mencionó anteriormente, convertir una imagen a escala de grises simplifica la información y la prepara para técnicas de detección de bordes y características. Este paso es imprescindible a la hora de detectar contornos y resaltarlos.

2.2.3 Filtrado y eliminación de ruido

- **Filtro Gaussiano:** Reduce el ruido de alta frecuencia.
- **Filtro Mediano:** Es eficaz contra el ruido "sal y pimienta".
- **Filtro Bilateral:** Suaviza sin comprometer los bordes.
- **Desconvolución:** Técnica avanzada que busca revertir el efecto del ruido introducido por un proceso conocido.

2.2.4 Mejora de contraste

Mejorar el contraste de una imagen puede facilitar la identificación de características relevantes.

- **Ecualización de histograma:** Ajusta la distribución de intensidad en la imagen para mejorar el contraste.
- **Estiramiento de contraste:** Aumenta la diferencia entre los píxeles más oscuros y más claros para resaltar características.

2.2.5 Binarización

En algunas aplicaciones, especialmente en la detección de bordes o en la segmentación, puede ser útil convertir la imagen en una imagen binaria, donde los píxeles se clasifican como fondo u objeto.

- **Umbralización:** Selecciona un umbral y clasifica los píxeles como 'objeto' o 'fondo' según si su intensidad está por encima o por debajo del umbral.
- **Métodos adaptativos:** Estos métodos ajustan el umbral según las características locales de la imagen, permitiendo una binarización más precisa en imágenes con iluminación variable.

2.2.6 Morfología matemática

Consiste en procesar imágenes basadas en la forma. Se utiliza para limpiar y mejorar estructuras en imágenes binarias.

- **Erosión y dilatación:** Operaciones básicas que permiten reducir o aumentar características en una imagen binaria.
- **Apertura y cierre:** Combinaciones de erosión y dilatación que permiten eliminar ruido, llenar huecos y mejorar contornos.

Para concluir, muchas de estas características están recogidas dentro de la librería OpenCV, es por ello que se ha escogido dicha librería para facilitar el trabajo.

A continuación se mostrarán algunos ejemplos de funciones ya implementadas en la librería OpenCV:

- **getOptimalNewCameraMatrix y undistort:** Correción de distorsiones de lente
- **cvtColor:** Convertir imágenes a escala de grises usando la bandera **COLOR_BGR2GRAY**
- **GaussianBlur:** Filtro Gaussiano
- **medianBlur:** Filtro Mediano
- **bilateralFilter:** Filtro Bilateral
- **equalizeHist:** Ecualización del Histograma de la imagen
- **threshold :** Umbralización de las imágenes
- **adaptiveThreshold:** Métodos adaptativos
- **erode:** Erosión
- **dilate:** Dilatación

2.3 Relevancia en la caracterización de granos de trigo

El preprocesamiento y la detección de contornos son esenciales para la caracterización de granos de trigo. Una vez que los granos están adecuadamente identificados y separados, se pueden extraer métricas y características relevantes, como el área, la forma, la textura y otros parámetros relevantes. Estas características son esenciales para clasificar los granos, identificar posibles defectos y comprender mejor su calidad y propiedades.

Para saber diferenciar qué características son importantes y cuales no, se ha seguido el artículo científico (Rohit Sharma, 2021) donde se puede llegar a la conclusión que sólo con el alto, ancho y el contorno se pueden sacar una serie de características más completas como por ejemplo la redondez, el área del contorno de cada grano o la solidez.

Para las características donde entran en juego los colores es necesario la imagen a color pero es cierto que con ciertas funciones de la librería OpenCV se pueden sacar fácilmente.

2.4 FiJi y GrainScan

FiJi (ImageJ)

Es una distribución popular del programa de procesamiento de imágenes de código abierto llamado ImageJ. ImageJ es ampliamente reconocido en la comunidad científica, especialmente entre aquellos que trabajan en biología y microscopía, por su versatilidad y capacidad para ser personalizado según las necesidades del usuario. FiJi amplía esta funcionalidad al incluir una serie de plugins que facilitan muchos procesos de análisis de imágenes.

Distribución de ImageJ: En su esencia, FiJi es una versión "mejorada" de ImageJ, ya que incorpora una amplia colección de plugins que extienden las capacidades básicas de ImageJ.

Diseñado para la biología: Aunque es útil en una variedad de aplicaciones de procesamiento de imágenes, FiJi ha sido especialmente diseñado para satisfacer las necesidades de los biólogos. Las herramientas incluidas son ideales para el análisis de imágenes biológicas, especialmente en el ámbito de la microscopía.

Plugins incluidos: FiJi viene con muchos plugins que no están presentes en la versión estándar de ImageJ. Estos plugins facilitan tareas como la corrección de iluminación, la segmentación de imágenes, el seguimiento de partículas, entre otros.

Actualizaciones constantes: FiJi cuenta con un sistema de actualización que permite a los usuarios mantener sus plugins al día. Esto es esencial en el ámbito científico, donde las técnicas y metodologías evolucionan rápidamente.

Interfaz amigable: A pesar de la amplia gama de herramientas y funcionalidades, FiJi presenta una interfaz de usuario que es relativamente fácil de navegar, especialmente para aquellos familiarizados con ImageJ.

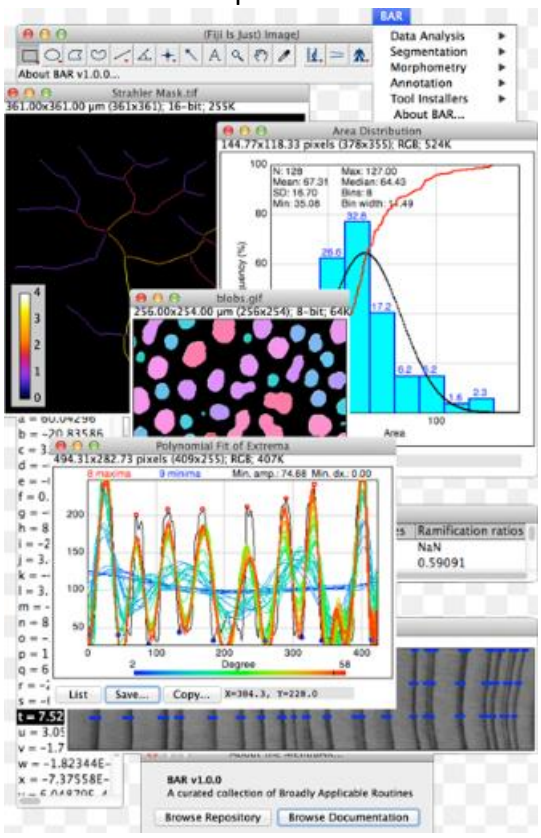


Figura 2.4: Demostración de lo potente que puede llegar a ser FiJi

Soporte y comunidad: Dado que FiJi es una herramienta ampliamente utilizada en la comunidad científica, cuenta con un amplio soporte en línea. Hay numerosos tutoriales, foros y recursos disponibles para ayudar a los usuarios a resolver problemas y aprender a usar la aplicación de manera más efectiva.

Código abierto: Al igual que ImageJ, FiJi es de código abierto, lo que significa que los usuarios avanzados pueden modificar y personalizar el software según sus necesidades. Además, esto permite a la comunidad contribuir al desarrollo del software.

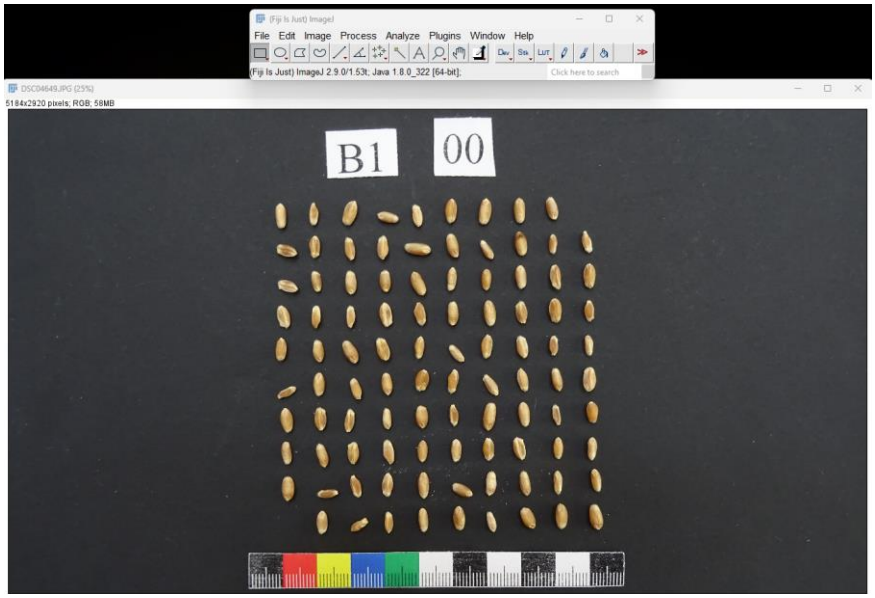


Figura 2.5: Aplicación FiJi con una imagen abierta

Con su robusto conjunto de herramientas y plugins especializados en el análisis de imágenes biológicas, se presenta como una guía invaluable para un proyecto centrado en la detección y procesamiento de granos de trigo. Su capacidad para manejar y procesar imágenes microscópicas y macroscópicas con alta precisión puede ser crucial para identificar y analizar características individuales de cada grano.

Además, dado que FiJi es ampliamente utilizado en la comunidad científica para tareas similares en otros campos de estudio, su enfoque y metodologías pueden ofrecer una base sólida sobre la cual desarrollar técnicas específicas para el análisis de granos de trigo.

Las funcionalidades de segmentación, corrección de iluminación y seguimiento de partículas, entre otros, pueden adaptarse y ajustarse para abordar los desafíos únicos presentados por el estudio de estos granos, haciendo de FiJi una referencia esencial para garantizar un enfoque metodológico eficaz y preciso. También se utilizó el esqueleto de la GUI de FiJi como guía para comenzar con una base.

A continuación se muestra un ejemplo de uso de la herramienta FiJi.

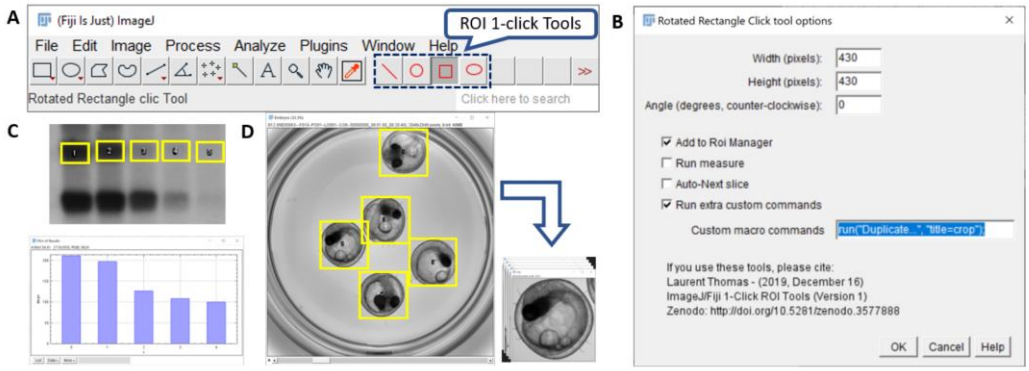


Figura 2.6: (A) El conjunto de herramientas ROI en la barra de herramientas de Fiji. Al hacer doble clic en el ícono se muestra (B) la ventana de opciones asociada para establecer las formas de ROI y las acciones personalizadas a ejecutar al hacer clic. (C) Uso de la herramienta de rectángulo para la cuantificación de la intensidad de bandas en geles de electroforesis. (D) Uso de la herramienta de rectángulo para anotar la posición de la muestra. Usando un comando personalizado (destacado en B), las regiones anotadas se recortan automáticamente.

GrainScan

Específicamente diseñado para la cuantificación y análisis de características de semillas y granos, emerge como una herramienta de referencia muy útil para cualquier proyecto centrado en la detección y procesamiento de granos de trigo. Su especialización en el análisis granular asegura una precisión y eficiencia adaptadas a las necesidades específicas de tales investigaciones.

El hecho de que GrainScan haya sido desarrollado teniendo en cuenta los desafíos intrínsecos a la morfología y variabilidad de los granos, lo convierte en una fuente rica de metodologías y técnicas validadas.

Además, su capacidad para proporcionar medidas detalladas y precisas sobre características como el tamaño, la forma y la textura de los granos puede ofrecer una hoja de ruta para desarrollar o perfeccionar algoritmos en proyectos similares.

Por lo tanto, al considerar GrainScan como una guía, un proyecto puede beneficiarse no solo de sus técnicas establecidas, sino también de su enfoque centrado en los granos, garantizando un análisis más eficaz y detallado de los granos de trigo.

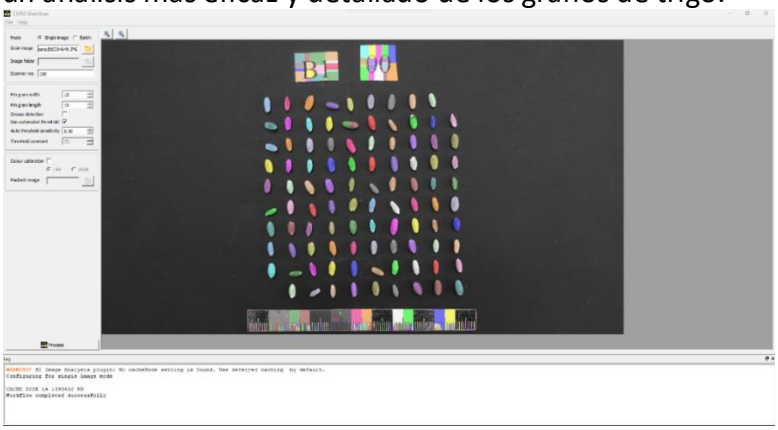


Figura 2.7: Aplicación GrainScan con una imagen procesada

GrainScan ha desempeñado un papel fundamental en la conceptualización y diseño de nuestra interfaz de usuario. Esta herramienta no solo ha sido una fuente de inspiración para estructurar y organizar la interfaz gráfica, sino que también nos ha proporcionado una dirección clara sobre las variables esenciales que deberían incorporarse en el panel de control. Algunos ejemplos notables de estas variables son la longitud mínima, la anchura mínima y el umbral, entre otros.

Uno de los aspectos más innovadores que adoptamos de GrainScan es la forma en que maneja la visualización de imágenes. A diferencia de otros programas, como FiJi, que abren imágenes en una ventana emergente separada, GrainScan presenta la imagen directamente dentro de la interfaz principal de la aplicación. Reconocimos el valor de esta funcionalidad y decidimos integrarla en nuestro diseño.

Esta decisión no solo mejora la estética y coherencia de la aplicación, sino que también la hace considerablemente más intuitiva y amigable para el usuario. Al eliminar la necesidad de gestionar múltiples ventanas o pop-ups, facilitamos la experiencia del usuario y proporcionamos un flujo de trabajo más fluido y eficiente.

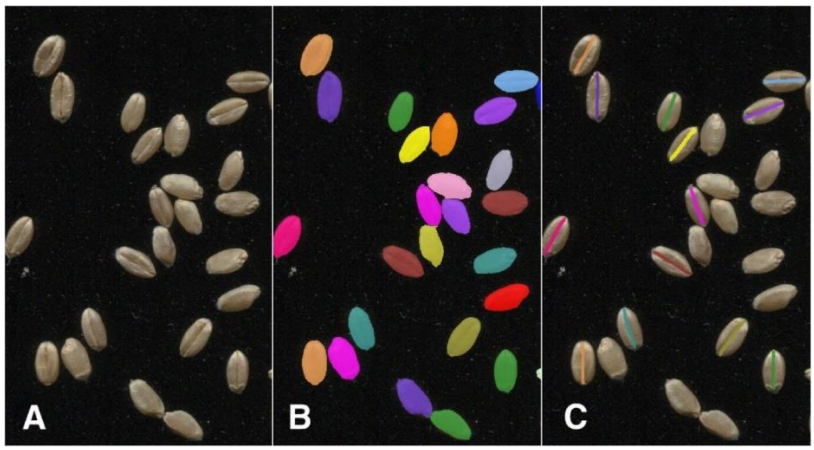


Figura 2.8: Ejemplos de entrada y salida de GrainScan. Panel A: Granos de trigo escaneados para la entrada de GrainScan. Panel B: Salida de GrainScan destacando los granos segmentados según lo determinado por el software. Diferentes colores indican diferentes granos. Panel C: Salida opcional de detección de pliegues destacando las regiones identificadas como pliegue del grano.

3

Análisis

El análisis consta de varios aspectos que se deben de cumplir antes de pasar a la siguiente fase. El objetivo principal del análisis es tener una documentación concisa de cuál es el objetivo del proyecto y cómo se va a abordar dicho proyecto.

Este análisis constará de dos partes:

3.1 Requisitos

En los requisitos del proyecto se explicará detalladamente qué es lo que se ha pedido hacer y con qué recursos.

En este caso se partía de una idea explicada de forma oral, es decir, no había ningún tipo de estructura ni esqueleto base del que partir. Es por ello que el apartado del Estudio del arte fue muy importante para investigar qué tecnologías habría que usar para llevar a cabo el objetivo principal.

La idea principal consistía en la detección de granos de trigo para su posterior procesamiento, para ello se dio una serie de fotos para probar el algoritmo.

Dichas fotos tienen un formato muy específico para que la aplicación funcione correctamente. Se muestran granos de trigo separados uno de otros en cuadrículas de 1x1 centímetros con un fondo negro u oscuro para su mejor detección. Además aparecerá en algún lugar de la imagen una regla para tener una medida real, dicha regla tendrá que ser usada por la aplicación posteriormente.

Una vez estén los granos detectados se procederá a desarrollar los algoritmos pertinentes, explicados uno a uno aquí. En la implementación y segmentación del código se explicará todo más al detalle y cómo se ha hecho.

El resultado de todos estos algoritmos tendrán que ser exportados a un archivo CSV para así poder crear un dataset de todos los granos detectados.

También se especificó que era obligatorio el uso de la biblioteca OpenCV para el reconocimiento de las semillas y si era necesario en el tratamiento de las mismas una vez detectadas.

OpenCV es una de las bibliotecas más populares y completas para la visión por computador. Diseñada para proporcionar una infraestructura común para aplicaciones de visión por computador, se utiliza ampliamente en la detección en tiempo real, en la extracción de características, en la creación de descripciones, en el reconocimiento de patrones, entre muchas otras aplicaciones.

3.2 Herramientas

Para el desarrollo e implementación del proyecto se han usado las siguientes herramientas:

Módulo "csv" (CSV, 2023)**:** Es una librería de Python que se utiliza para trabajar con archivos CSV (valores separados por comas). Permite leer y escribir en archivos CSV. Este módulo se ha usado únicamente para la función que, una vez estén todos los granos detectados, usa todos los algoritmos desarrollados y exporta los resultados en un fichero CSV de manera muy sencilla, un "cabecero" o "header" donde aparece el nombre de cada característica y justo debajo tantas filas como granos se hallan detectado con todas las características rellenas.

Módulo "math": Es una librería usada en una gran cantidad de lenguajes de programación, entre las que se encuentra Python. se utiliza para realizar operaciones matemáticas más avanzadas. Incluye funciones matemáticas como seno, coseno, tangente, logaritmos y más.

En este caso se ha usado para la función math.sqrt() que calcula la raíz cuadrada de un número.

Tkinter: Es otra librería Python que se ha utilizado para la interfaz gráfica de usuario (GUI) de la aplicación. Cuenta con una serie de widget (botones, tooltips, textfields, etc) que son útiles para crear una aplicación GUI.
Tkinter es un conjunto de envoltorios que implementan los widgets Tk como clases de Python. Las principales virtudes de tkinter son que es rápido y que generalmente viene incluido con Python. Aunque su documentación estándar es débil, se dispone de buen material, que incluye: referencias, tutoriales, libros, etcétera.

Módulo "filedialog": Es un submódulo de Tkinter que se encarga de proporcionar una interfaz para mostrar mensajes al usuario, en este proyecto se ha usado para la apertura del explorador de archivos y así elegir un fichero con un formato de imagen (png, jpeg, jpg).

También se ha usado en el guardado del fichero csv resultante después de toda la detección y procesamiento de los granos de trigo.

Módulo "messagebox": Es otro submódulo de Tkinter que proporciona una interfaz para mostrar mensajes al usuario,
este módulo se ha usado para mostrar al usuario errores o información de que el fichero se ha guardado correctamente, se ha usado a modo de pop-up para informar.

CV2: Es el módulo principal para la detección de bordes en este proyecto también es parte de la biblioteca OpenCV. El módulo **cv2** en sí es una interfaz Python para esta biblioteca, permitiendo que los desarrolladores accedan a una amplia gama de herramientas y técnicas de procesamiento de imágenes y visión por computador directamente desde el lenguaje Python.

Dentro de las capacidades del módulo **cv2**, se encuentran funciones para la lectura y escritura de imágenes, operaciones básicas y avanzadas de procesamiento de imágenes como la transformación de color, filtrado, detección de bordes y segmentación.

También proporciona herramientas para la detección y reconocimiento de características, la correspondencia de características y la estimación de movimiento, así como algoritmos para la detección de objetos, seguimiento y reconocimiento de patrones.

Adicionalmente, **cv2** ofrece capacidades para la calibración de cámaras, la reconstrucción 3D, la estimación de la pose y la creación de interfaces gráficas. Dada su versatilidad y robustez, OpenCV, y por extensión el módulo cv2, se ha convertido en una herramienta esencial para académicos, investigadores y profesionales que trabajan en áreas relacionadas con la visión por computador, la robótica, la inteligencia artificial y el procesamiento de imágenes.

Con una comunidad activa y un desarrollo constante, la biblioteca sigue evolucionando, añadiendo regularmente nuevas características y mejorando las existentes para abordar los desafíos emergentes en el campo de la visión por computador.

Módulo "PIL" (Pillow, 2023)**:** Corresponde a las siglas "Python Imaging Library", es una biblioteca de procesamiento de imágenes para el lenguaje de programación Python. Aunque el desarrollo activo de PIL se detuvo hace tiempo, ha sido sucedido y ampliado por "Pillow", un fork que extiende las capacidades de PIL y garantiza su compatibilidad con las versiones modernas de Python. Sin embargo, a menudo se utiliza el término "PIL" para referirse a "Pillow", dado el legado y la historia de la biblioteca original.

El módulo PIL proporciona capacidades para realizar una amplia variedad de operaciones de procesamiento de imágenes. Entre sus características se encuentran las herramientas para abrir, manipular y guardar muchos formatos de imagen diferentes. Esto incluye tareas básicas como la lectura y escritura de imágenes, conversiones de formato, operaciones de pixel y manejo de colores.

Además, permite realizar operaciones más avanzadas como el filtrado, la transformación (por ejemplo, rotación, escalado y recorte), la mejora de la imagen y la detección de características.
Uno de los aspectos más valiosos de PIL es su simplicidad y facilidad de uso. Aunque no es tan extenso o especializado como otras bibliotecas como OpenCV, su naturaleza ligera y su interfaz intuitiva lo hacen ideal para tareas de procesamiento de imágenes que no requieren herramientas de visión por computador avanzadas o para desarrolladores que buscan una solución rápida y eficiente para la manipulación básica de imágenes en Python.

En este proyecto se han aprovechado varias de las funcionalidades que ofrece PIL, de las que destaca la apertura de los iconos de los botones y en una función que redimensiona la imagen que se quiere tratar.

Módulo "numpy" (NumPy, 2023)**:** Representa una biblioteca fundamental para la computación científica en Python. Esta biblioteca proporciona soporte para trabajar con arrays multidimensionales, así como una colección extensiva de funciones matemáticas de alto nivel para operar sobre estos arrays. La fortaleza de numpy radica en su capacidad para realizar operaciones numéricas de manera eficiente, gracias a su implementación interna en C y Fortran, lo que garantiza una rápida ejecución.

El componente central de numpy es el objeto ndarray, que es un array multidimensional de tamaño fijo. A diferencia de las listas en Python, que pueden ser heterogéneas, un ndarray es homogéneo; todos los elementos deben ser del mismo tipo. Esto, junto con la capacidad de numpy de operar en bloques de datos contiguos en memoria, permite operaciones matriciales y vectoriales altamente optimizadas.

Además de las capacidades básicas de manipulación de arrays, numpy ofrece una amplia variedad de submódulos que abordan diferentes dominios de la computación científica. Estos incluyen herramientas para la álgebra lineal, la transformada de Fourier, la generación de números aleatorios y más. Es esta combinación de funcionalidad general y específica lo que hace de numpy una herramienta indispensable en campos que van desde la física y la biología hasta la economía y la ingeniería.

Otro aspecto importante de numpy es su interoperabilidad. Sirve como pilar fundamental para otras bibliotecas en el ecosistema de ciencia de datos de Python, como pandas para manipulación de datos y matplotlib para visualización, por mencionar solo algunas. Así, cualquier persona que trabaje en análisis de datos, aprendizaje automático, o ciencias en general, encontrará en numpy una herramienta esencial para el procesamiento y análisis numérico en Python.

Respecto a este proyecto, numpy se ha usado sobre todo en el desarrollo de los algoritmos que nos daban las características de cada grano usando en repetidas ocasiones su función zeros() para crear máscaras.

Módulo "sys" (Sys, 2023)**:** Es una de las bibliotecas integradas en el núcleo de Python, proporcionando acceso a algunas variables y funciones específicas del intérprete. Sirve como una ventana al entorno en el que Python se ejecuta, permitiendo a los desarrolladores interactuar con el sistema subyacente y el propio intérprete.

Uno de los usos más comunes de **sys** es manipular la lista de argumentos pasados a un script de Python. Mediante **sys.argv**, es posible obtener una lista de argumentos de línea de comandos, lo que facilita la creación de interfaces de usuario basadas en la terminal.

Además, **sys** proporciona herramientas para controlar las entradas y salidas estándar del sistema, como **sys.stdin**, **sys.stdout** y **sys.stderr**. Estos objetos representan, respectivamente, la entrada estándar, la salida estándar y la salida de errores estándar, permitiendo una interacción más detallada y controlada con el usuario o con otros programas.

Otra característica útil es **sys.path**, que contiene la lista de directorios en los que el intérprete busca módulos antes de importarlos. Esto es esencial para la gestión de paquetes y módulos en Python, especialmente cuando se quiere agregar o modificar rutas de búsqueda para módulos personalizados.

El módulo también ofrece información sobre el entorno de ejecución, como **sys.platform,** que indica el sistema operativo en el que se está ejecutando el script, o **sys.version**, que devuelve la versión del intérprete de Python en uso.

Finalmente, **sys** proporciona funciones para manipular el flujo de ejecución del intérprete, como **sys.exit()**, que permite terminar la ejecución de un script y devolver un código de salida al sistema operativo, o **sys.exc_info()**, que recupera información sobre excepciones que están siendo manejadas en ese momento.

Este caso es un tanto especial porque la aplicación final no utiliza este módulo pero si se ha estado usando a lo largo de toda la implementación para controlar entradas, salidas y errores estándar del sistema. También se ha hecho uso de **sys.exit()** en una versión temprana del proyecto incluso **sys.version().**

Módulo "skimage": Forma parte de la biblioteca **scikit-image,** (scikit-image, 2023) una colección de algoritmos para el procesamiento de imágenes en Python. Este módulo, en particular, se centra en la detección de características y patrones dentro de las imágenes, proporcionando herramientas y técnicas para identificar y describir propiedades específicas en las imágenes, como esquinas, bordes y texturas.

Dentro de **skimage.feature**, las funciones **graycomatrix** y **graycoprops** están dedicadas a la extracción de características texturales de las imágenes utilizando el método de Matriz de Co-ocurrencia de Niveles de Gris (GLCM, por sus siglas en inglés). La textura es una propiedad importante de las imágenes, especialmente en aplicaciones como el análisis de imágenes médicas, la clasificación de materiales y la detección de defectos.

La función **graycomatrix** calcula la GLCM para una imagen en escala de grises. La GLCM es esencialmente una tabla que indica con qué frecuencia dos píxeles,
con ciertos valores y en una relación espacial específica, ocurren en una imagen, dicha tabla es representada en el código como una matriz. Por ejemplo, podría indicar cuántas veces un píxel con intensidad 5 está al lado de un píxel con intensidad 7. Estos datos se utilizan para describir la textura de la imagen, ya que diferentes texturas producirán diferentes distribuciones de co-ocurrencia.

Una vez que se tiene la GLCM, la función **graycoprops** se utiliza para extraer propiedades estadísticas de la matriz. Estas propiedades incluyen medidas como el contraste (cuánta diferencia hay entre los píxeles de una imagen), la homogeneidad (cómo se distribuyen los valores de píxeles cercanos entre sí), la energía (suma de los valores al cuadrado de la GLCM) y la correlación (mide la dependencia lineal de niveles de gris en pares de píxeles). Estas propiedades cuantifican la textura de la imagen de manera que pueda ser comparada, clasificada o analizada de otras maneras.

Estas dos funciones se han usado para la elaboración de características relacionadas con las texturas de los granos de trigo y donde intervenga la GLCM.

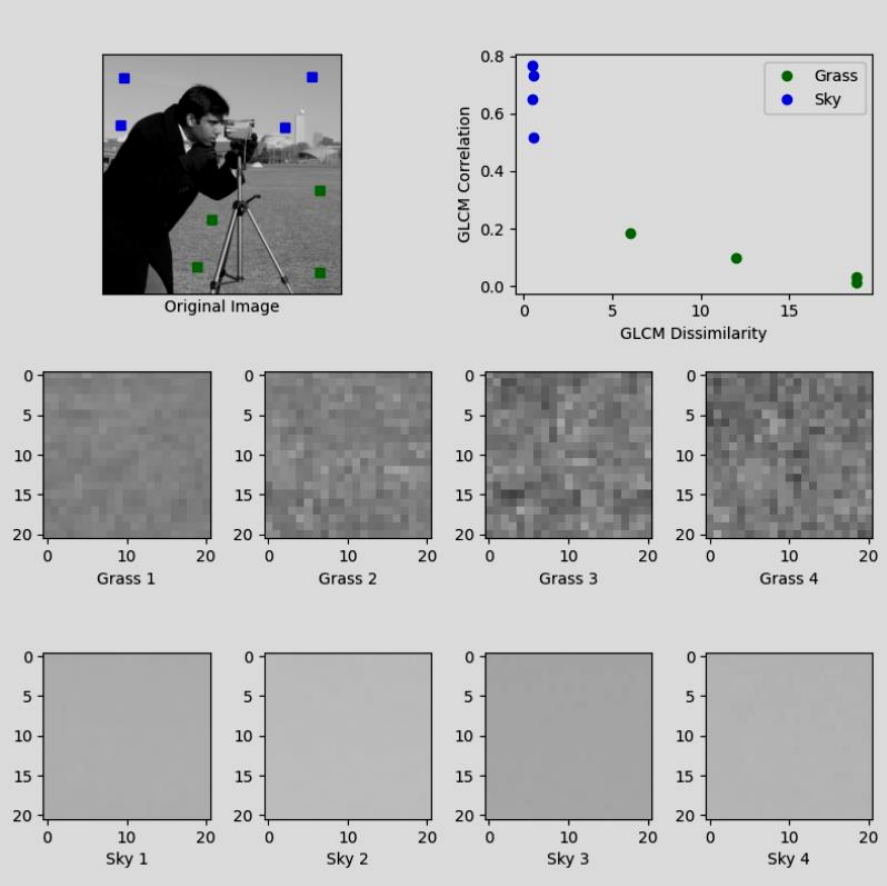


Figura 3.1: Este ejemplo ilustra la clasificación de texturas utilizando matrices de co-ocurrencia de niveles de gris (GLCMs, por sus siglas en inglés). Una GLCM es un histograma de valores en escala de grises coexistentes a un desplazamiento dado sobre una imagen.

4

Diseño

El diseño (ProjectManager, 2019), dentro del proceso de desarrollo de software, juega un papel crucial como puente entre la concepción y la implementación del software. Es en esta fase donde las ideas abstractas y los requisitos definidos se traducen en un plan estructurado y detallado que guiará la construcción del software.

Propósito del Diseño:
Con los requisitos ya establecidos y comprendidos, el diseño se encarga de determinar "cómo" se llevarán a cabo esos requisitos. Mientras que la fase de análisis responde a la pregunta "¿qué?", el diseño responde a "¿cómo?". Esta distinción es esencial, pues es en el diseño donde se definen las soluciones técnicas y se estructura el enfoque de desarrollo.

Figura 4.1: Ciclo del desarrollo software donde nos dejan claro que antes de empezar a programar hay que preestablecer claramente un buen diseño

Evaluación de Opciones:
El diseño no se trata simplemente de elegir una solución y seguir adelante. Implica evaluar diferentes opciones de implementación, considerar las ventajas y desventajas de cada una, y seleccionar la que mejor se adapte a los requisitos, al contexto y a las restricciones del proyecto.

Esta evaluación es esencial para garantizar que el software será eficiente, escalable y mantenible.

Patrones de Diseño:
Estos son soluciones probadas y comprobadas a problemas comunes de diseño en el desarrollo de software. Utilizar patrones de diseño no sólo facilita la implementación, sino que también puede mejorar la eficiencia, la modularidad y la reusabilidad del código. Al considerar patrones de diseño apropiados durante la fase de diseño, se asegura una base sólida para la construcción del software.

Arquitectura del Sistema:
Es en la fase de diseño donde se establece la arquitectura del sistema. Esto implica definir cómo se organizarán y comunicarán los diferentes componentes o módulos del software. Una arquitectura bien definida es esencial para garantizar que el software sea robusto, escalable y fácil de mantener. Además, ayuda a identificar posibles problemas o desafíos antes de que comience la implementación.

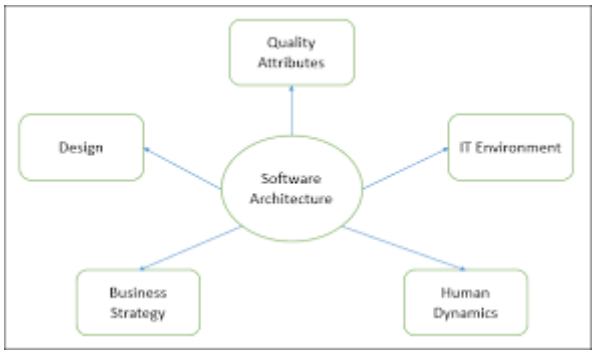


Figura 4.2: Ejemplo de arquitectura de sistema para un proyecto software

Documentación:
Una parte esencial del diseño es la documentación. Los diagramas, las especificaciones y otros documentos producidos durante esta fase sirven como una hoja de ruta para los desarrolladores. Facilitan la comunicación entre los miembros del equipo y garantizan que todos tengan una comprensión clara y coherente de cómo debe construirse el software.

Iteración y Retroalimentación:
El diseño no es un proceso lineal. A menudo implica iteraciones, donde las soluciones propuestas se revisan, se modifican y se refinan a medida que se adquiere una comprensión más profunda de los desafíos y las soluciones posibles. La retroalimentación, tanto interna como de los stakeholders, puede influir y mejorar el diseño, asegurando que el producto final cumpla con las expectativas.

En este proyecto se empezó a diseñar la estructura de la aplicación en un solo fichero python con la convicción de que no haría falta dividir el código en varios ficheros. En el momento que surgió la necesidad de hacer una clase llamada **Grano** para asociar a cada grano de la imagen un objeto con una longitud, anchura, contorno y centro.

Una vez se creó esta nueva clase, se llegó a la conclusión de que era mejor implementar las funciones que extraerían las características de cada grano en esta nueva clase ya que en la gran mayoría de funciones se necesitaban únicamente alguna de las características del grano ya mencionadas (longitud, anchura…).

Respecto a las funciones que nos calculaban características relacionadas con el color o la textura del grano se necesitaba también la imagen original para su correcto funcionamiento.

Más adelante apareció la mejora de los tooltips que no es más que una ayuda al usuario para comprender de manera rápida para qué sirve cada botón y sin tener que acudir al manual de usuario.

Surgió de nuevo la necesidad de crear una nueva clase y para ello se creó otro fichero llamado **tooltips.py** que contendría la clase Tooltip y su completo desarrollo para que así solamente hiciese falta crear una línea de código para cada botón y así tener un tooltip funcional de manera muy sencilla.

Es así como se pasó de un solo fichero principal a tres, se estuvo pensando en dividir el proyecto en cuatro ficheros, los dos que ya hemos mencionado y el principal dividirlo en dos, uno donde estuviese única y exclusivamente la parte meramente gráfica de la aplicación y en el otro todas las funciones, es decir, en uno los botones, campos de texto… Y en el otro qué hacía cada función de cada botón u opción que ofrece la aplicación en sí.

Respecto a la arquitectura del sistema se puede decir que como en un principio se propuso, funciona como un solo fichero, es decir, existe un fichero "main" que importa las otras dos clases que están en los otros dos ficheros, las clases **Tooltip** y **Grano** importan a su vez módulos que les facilitan al desarrollo de sus funciones pero no importan clases creadas por el alumno. El fichero principal importa tanto las clases secundarias mencionadas anteriormente como otros módulos para la creación de la interfaz gráfica u otras facilidades.

También cabe destacar la creación y procesamiento de los iconos e imágenes que contiene la aplicación. Todas de imágenes han sido preprocesadas para su posterior uso, ya sea porque habían sido sacadas de un lugar con un formato inadecuado, porque el tamaño era demasiado grande o por el contrario demasiado pequeño, sea como sea, en todas las imágenes ha habido un proceso de preparación previa a su uso. Por ejemplo, para la redimensión las imágenes se ha usado alguna aplicación web como Image Resizer.

La mayoría de iconos han sido sacados de una página web especializada en iconos totalmente gratuita con una gran variedad de estos, se consideró otras muchas opciones pero finalmente se optó por iconos8 porque te daba no una amplio abanico de posibilidades sino que también te daba la posibilidad de elegir entre diferentes tamaños para cada una de las imágenes.

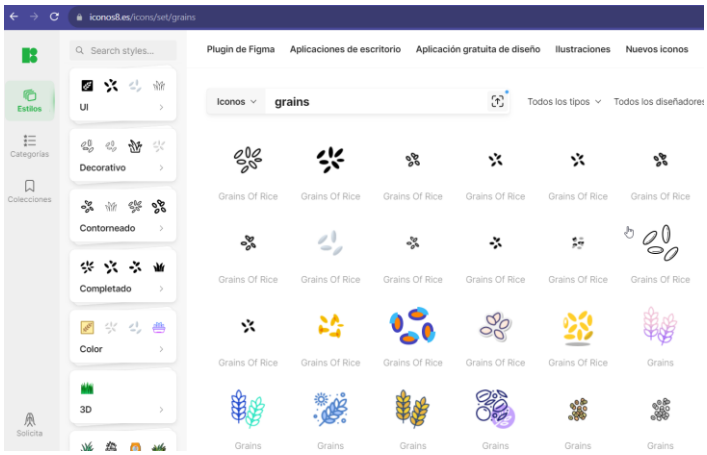


Figura 4.3: Página web iconos8 donde se muestra la variedad de imágenes

La retroalimentación ha sido extremadamente importante en este proyecto porque como dijimos al principio de esta lectura, no solamente han existido las reuniones cada dos semanas, sino que ha habido un contacto casi diario cuando ha sido necesario donde se esclarecían requisitos que no acababan de ser interiorizados o simplemente a media que se avanzaba en el proyecto surgían nuevas necesidades o requisitos.
Un ejemplo de una nueva necesidad fue la de crear la clase Tooltip y separar el fichero main en dos: A medida que se iba haciendo la interfaz gráfica surgió la mejora de añadir iconos a los botones para obtener una aplicación más vistosa para el usuario, el problema de esta mejora es que si no se le agrega un tooltip puede llegar a ser confuso el funcionamiento del botón para un usuario promedio que no ha usado la aplicación anteriormente ya que con una sola imagen es complicado entender qué hace dicho botón.

Algo importante a destacar respecto a la documentación del proyecto es que no solamente existe un manual de usuario sino que mientras se estaba implementando el proyecto se han estado guardando y produciendo diferentes versiones del software, no solamente como checkpoints que funcionan como requisitos tachados dentro de una to-do list sino que también sirve como snapshot o copia de seguridad en caso de cometer un error que fuerce al usuario a volver a atrás pudiendo hacer un rollback de manera inmediata y segura.

Además, estas versiones servían para enviarlas al tutor del proyecto con la certeza de que le iba a funcionar, es decir, aunque la última versión del alumno se haya guardado hace dos días, el alumno le mandaría dicha versión en vez de la versión en la que está trabajando actualmente para así asegurarse que al tutor le funcionara aunque no fuese la última novedad.

Para terminar con el capítulo, se mostrará el diagrama UML de la aplicación:

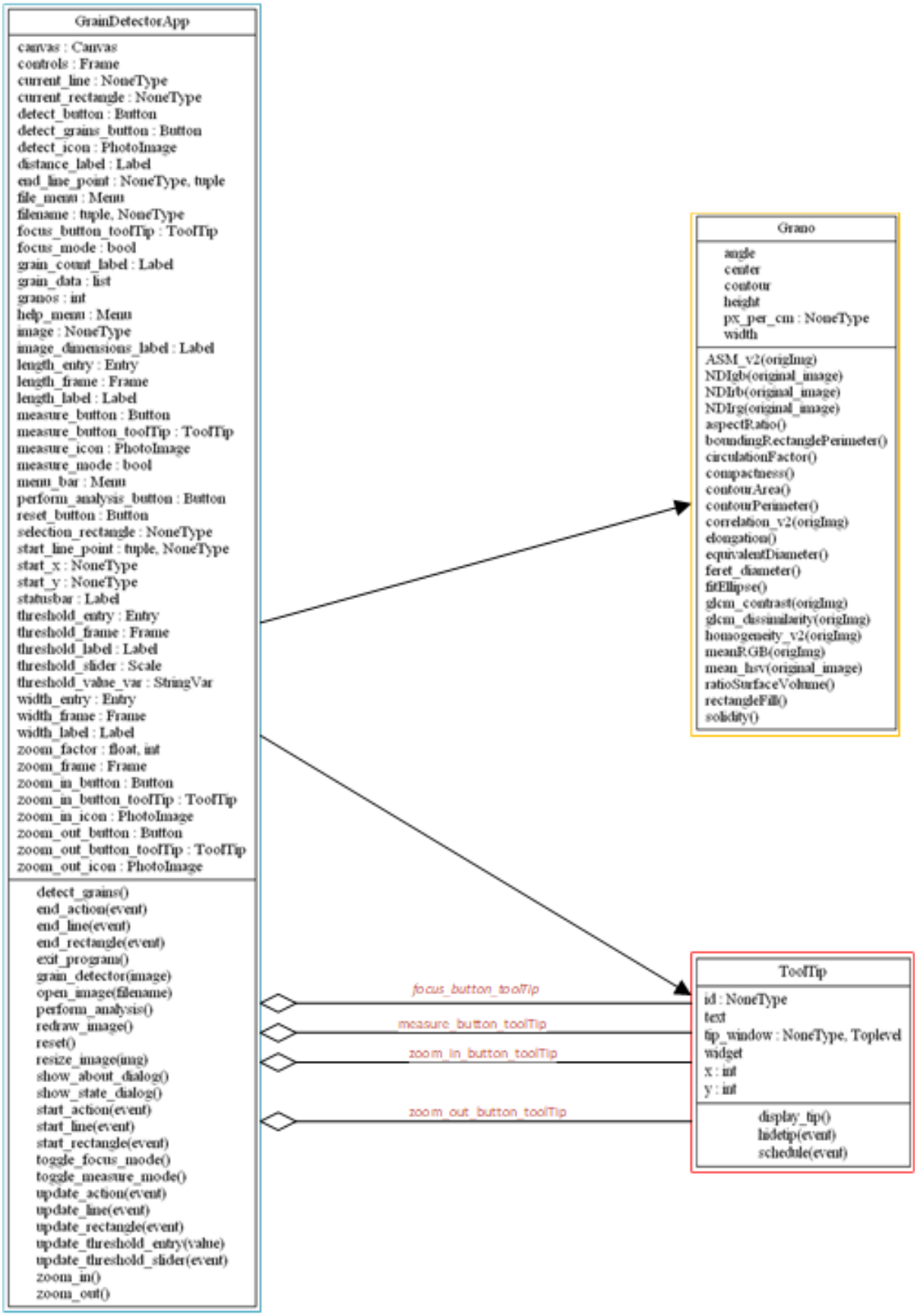


Figura 4.4: Diagrama UML del proyecto.

5

Implementación

Como ya se mencionó anteriormente, esta fase consta de dos subfases dentro de este proyecto en específico, la segmentación, que trata la detección de cada grano y saca algunas características principales de los propios granos como puede ser la altura, anchura, el contorno y su punto céntrico y por otro lado tenemos la obtención de características cuantificables, dichas obtenciones se basan en las características básicas de cada grano y un ejemplo puede ser la solidez, la homogeneidad o el área del contorno de cada grano.

5.1 Segmentación

Dentro de la segmentación se ha incluido también la parte de la interfaz gráfica de usuario (GUI) que es con la que se comenzará hablando.

Es un componente esencial para que los usuarios interactúen con la aplicación de manera intuitiva y eficaz. El código presentado revela una estructura clara y detallada de la GUI para el proyecto **Grain Detector**.

La clase **GrainDetectorApp** se basa en el módulo **tkinter** y proporciona la estructura principal de la GUI.

5.1.1 Estructura Principal y Menús

La aplicación inicia con una lista vacía, **grain_data**, que se destina a almacenar datos relacionados con los granos detectados, dichos datos son los que se mencionaron anteriormente, es decir, el centro, al longitud, anchura… etcétera.

A través de **iconbitmap**, se establece un ícono para la ventana de la aplicación. El título se configura como **Grain Detector** y se maximiza al inicio con el método **state**.

La GUI presenta una barra de menú principal que incluye opciones para abrir una imagen, cerrar la aplicación y un menú llamado **Help** que consta de dos submenús, el primero llamado **About** trata de información sobre alumno que ha hecho este proyecto y de quién es el tutor que le ha ido guiando. El segundo submenú se llama **Show State** y muestra el estado actual del programa, con el estado actual se refiere a características de la aplicación como por ejemplo si está el zoom activado, en definitiva características que veremos más adelante.

5.1.2 Panel de control

En el panel de control, los usuarios pueden configurar parámetros para la detección de granos, como la longitud y el grosor mínimos de los granos, así como ajustar el umbral para la segmentación de la imagen.

Si el usuario está constantemente procesando imágenes puede llegar a ser ciertamente tedioso el hecho de estar rellenando dichos campos dentro del panel de control, es por ello que se han puesto unas medidas por defecto, el umbral tiene un rango entre 0 y 255, es por ello que se le ha puesto un valor por defecto de 128 mientras que para la longitud y el grosor se le ha puesto un valor por defecto de 10 píxeles (recordar que estos valores por ahora están en píxeles hasta que no se haga la conversión).

Se proporcionan botones para realizar funciones clave, como **detectar granos**, **enfocar áreas específicas** y **medir distancias** en la imagen. Cada botón está acompañado de un ícono visual para mejorar la experiencia del usuario.

Detectar granos: Este botón utiliza las variables insertadas en el panel de control y el botón para enfocar áreas específicas para así detectar los granos con las características deseadas y dentro de la región deseada.

Enfocar áreas específicas: Esta función llegó debido a un problema que surgió en el planteamiento del proyecto. En la especificación del proyecto se dejó claro que todas las imágenes debían de tener un formato en el que apareciese una regla para hacer más tarde una conversión de píxeles a centímetros pero no solo eso, sino que además la mayoría de imágenes están etiquetadas por lo que no se puede aplicar el algoritmo de detección de granos a la imagen en su totalidad. Es por ello que surgió la necesidad de que el usuario eligiese la zona que más le interesase para trabajar y es de lo que se encarga este botón.

Los controles de zoom permiten a los usuarios acercar y alejar la vista de la imagen.

Una barra de estado en la parte inferior proporciona retroalimentación sobre las acciones del usuario o muestra mensajes relevantes. Como se verá más adelante cuando se enseñe cómo están hechas las funciones podremos ver claramente que esta barra de estado se actualiza única y exclusivamente cuando se le da de nuevo al botón **Detectar granos** esto está pensado adrede para que la barra de estado no esté constantemente cambiando por cualquier campo que se toque, sino que cambie solamente cuando hay un nuevo caso de detección.

5.1.3 Funcionalidad de Dibujo en el Canvas

El canvas, esencial para visualizar y manipular imágenes, está configurado para responder a eventos de mouse, lo que permite a los usuarios dibujar regiones de interés (ROI) o medir distancias.

Las variables **start_x** y **start_y** rastrean las coordenadas iniciales de un ROI, mientras que **current_rectangle** y **selection_rectangle** almacenan información sobre el **ROI** dibujado.

Se introduce un modo de enfoque, **focus_mode**, para permitir a los usuarios especificar un área de la imagen en la que desean centrarse durante la detección de granos.

5.1.4 Análisis y Conteo

Los usuarios pueden realizar análisis en las imágenes a través del botón **Analyze**.
Este botón se ocupa de aplicar todas las fórmulas calculadas en la clase Grano y exportarlas a un fichero CSV es por ello que hace uso del módulo **csv.**

Un etiqueta **grain_count_label** muestra el número total de granos detectados en la imagen, esta etiqueta está inicializada a cero y en el momento que se activa el botón **detectar granos**, esta etiqueta lee el número de elementos que hay dentro de la lista **grain_data** y los muestra en la interfaz gráfica.

Además, se introdujo un modo de medición, **measure_mode**, para permitir a los usuarios medir distancias en la imagen y poder hacer una conversión de píxeles a centímetros, como la longitud de un grano. Las variables **start_line_point**, **end_line_point** y **current_line** ayudan en esta funcionalidad, permitiendo a los usuarios dibujar y visualizar una línea de medición.

Hasta aquí la interfaz gráfica del programa, ahora se explicará detalladamente las funciones que tienen que ver con la segmentación y detección de granos de trigo, también se añadirán algunas funciones extras como el medidor píxeles por centímetro.

5.1.5 Actualización de Umbral

update_threshold_entry y **update_threshold_slide**r: Estas funciones se encargan de sincronizar el valor del umbral entre una barra deslizante y un cuadro de entrada. Si el usuario cambia el valor en uno de estos widgets, el otro se actualiza automáticamente.

Estas funciones se implementaron para una mayor precisión a la hora de introducir un valor del umbral, por cómo está hecho el slider del umbral, los valores se actualizaban de 3 en tres, es por ello que para el umbral hay un slider para una mayor comodidad pero también hay un textfield si lo que se desea es una mayor precisión.

5.1.6 Información del Estado de la Aplicación

show_state_dialog: Esta función muestra una ventana emergente con detalles del estado actual de la aplicación, como el título, el archivo cargado, el factor de zoom y los modos activos. Es útil para que los usuarios puedan comprobar rápidamente la configuración actual.

__str__: Esta función especial convierte la información de estado del programa en una cadena de texto. Esto permite una representación textual fácil de la configuración y estado actual de la aplicación.

Con esta representación se puede después usar la información en forma de debug o para futuras funciones que la puedan usar.

5.1.7 Funcionalidades de Dibujo y Medición

Las funciones **start_action**, **update_action** y **end_action** controlan la interacción del usuario con el canvas. Dependiendo de si el usuario está en modo de enfoque o medición, estas funciones llaman a los métodos adecuados para dibujar un rectángulo o una línea.

En un primer momento existían en vez de tres funciones, seis, debido a las funciones **focus** y **measure**. Para cada una de estas dos funciones debía de existir estas tres pero al final se optó por hacer tres funciones para las dos y dependiendo de en qué estado estaba el programa se usaba para una función u otra.

Un ejemplo básico: Si el usuario presionaba el botón que activa el modo **Focus** o de **Enfoque** la variable correspondiente a **Measure** o **Medición** se ponía a **False** y viceversa.

Las funciones **start_line**, **update_line** y **end_line** gestionan el proceso de dibujar y calcular una línea en la imagen, que representa una medida. La distancia en píxeles se convierte y muestra en centímetros.

Al contrario que **start_action**, **update_action** y **end_action**, estas tres funciones sí son específicas para la función de **Medición** ya que el modo **Enfoque** no hace uso de una línea recta sino de un rectángulo.

5.1.8 Funciones de Procesamiento de Imagen

5.1.8.1 detect_grains

Esta es una función clave que inicia el proceso de detección de granos en la imagen. Si hay un área seleccionada (ROI), se extrae esa región y se procesa para detectar granos. Luego se actualiza la imagen en el canvas.
Al ser una función crucial la describiremos muy detalladamente:

Verificación de Imagen Cargada:

if self.image is not None: Verifica que haya una imagen cargada en el programa.

Reapertura y Copia de la Imagen:

self.open_image(self.filename): Abre nuevamente la imagen usando el nombre de archivo guardado.
processed_image = self.image.copy() : Crea una copia de la imagen para procesarla sin modificar la imagen original.

Verificación y Procesamiento del Rectángulo Seleccionado (ROI):

if self.selection_rectangle is not None : Verifica si hay un rectángulo seleccionado (Region Of Interest, ROI).

Las siguientes líneas escalan las coordenadas del ROI según el factor de zoom actual. Esto es necesario porque la imagen mostrada puede estar ampliada o reducida respecto a la original.

roi = self.image[y1:y2, x1:x2]: Extrae el ROI de la imagen.

processed_roi, _ = self.grain_detector(roi): Procesa el ROI para detectar granos. La función grain_detector devuelve una imagen procesada con los granos resaltados y una lista de datos de los granos detectados.

Si las dimensiones del ROI original y el procesado no coinciden, se ajusta el tamaño del ROI procesado para que coincida con el original.

processed_image[y1:y2, x1:x2] = processed_roi: Inserta el ROI procesado de nuevo en la imagen.

Redimensionamiento y Centrado de la Imagen:

image = self.resize_image(processed_image): Redimensiona la imagen procesada para adaptarla al tamaño del canvas manteniendo su relación de aspecto.

Actualización de la Etiqueta de Dimensiones:

Muestra las dimensiones de la imagen en una etiqueta.

Conversión y Visualización de la Imagen:

image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB): Convierte la imagen de BGR (formato de OpenCV) a RGB (formato estándar).

tk_img = ImageTk.PhotoImage(Image.fromarray(image)): Convierte la imagen procesada a un formato compatible con Tkinter.

self.canvas.create_image(0, 0, anchor=tk.NW, image=tk_img): Muestra la imagen en el canvas.

self.canvas.image = tk_img: Guarda una referencia a la imagen para evitar que sea eliminada por el recolector de basura de Python.

Actualización de la Barra de Estado:

self.statusbar.config(text=str(self)): Actualiza la barra de estado con la información actual del programa.

5.1.8.2 resize_image

Ajusta el tamaño de una imagen para que se ajuste al tamaño del canvas, manteniendo la relación de aspecto.

5.1.8.2 open_image

Permite al usuario abrir una imagen desde su computadora y la muestra en el canvas.

5.1.9 Análisis de Datos

perform_analysis: Una vez detectados los granos, esta función permite al usuario guardar los datos analizados en un archivo CSV. Los datos incluyen dimensiones, proporciones, colores medios y características texturales de cada grano.

Esta función llama a la clase Grano, donde se encuentran todas las fórmulas que calculan las características necesarias para su posterior exportación.

5.1.10 Funciones auxiliares

exit_program: Cierra la aplicación.

toggle_focus_mode y toggle_measure_mode: Estas funciones activan o desactivan los modos de enfoque y medición, respectivamente.

redraw_image, zoom_in, y zoom_out: Estas funciones controlan el zoom de la imagen y la redibujan en el canvas.

show_about_dialog: Muestra una ventana con información sobre el programa y sus creadores.

5.1.11 grain_detector

Para finalizar se explicará la clase **grain_detector**, esta función se encarga de la detección de granos en una imagen dada y es por ello que al ser la más importante se ha dejado para el final, se desglosará parte por parte como se hizo con **detect_grain:**

5.1.11.1 Extracción de Parámetros:

min_length = int(self.length_entry.get()): Obtiene la longitud mínima que debe tener un grano para ser considerado válido.

min_width = int(self.width_entry.get()): Obtiene el ancho mínimo que debe tener un grano para ser considerado válido.

threshold_value = self.threshold_slider.get(): Obtiene el valor del umbral definido por el usuario a través de un control deslizante.

5.1.11.2 Conversión a escala de grises:

into_gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY): Convierte la imagen a escala de grises.

5.1.11.3 Binarización:

_, threshold = cv2.threshold(into_gray, threshold_value, 255, cv2.THRESH_BINARY): Binariza la imagen utilizando el valor de umbral especificado.

5.1.11.4 Operadores morfológicos:

Se crea un kernel rectangular de tamaño 5x5.

img_processed = cv2.morphologyEx(threshold, cv2.MORPH_OPEN, kernel): Aplica una operación morfológica de apertura para eliminar ruido y cerrar pequeñas brechas en los contornos.

5.1.11.5 Detección de contornos:

outlines, _ = cv2.findContours(img_processed, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE): Encuentra los contornos en la imagen procesada.

5.1.11.6 Filtrado y recopilación de datos de granos:

Se inicializan dos listas vacías: **grains** (para contornos) y **grain_data** (para datos de granos).

Para cada contorno detectado, se verifica si cumple con las dimensiones mínimas especificadas. Si es así, se recopila información sobre el grano y se crea una instancia del objeto Grano, que se añade a la lista **grain_data**.

5.1.11.7 Mostrar información:

Se imprime la información de cada grano en la consola, esta impresión está hecha con el objetivo de depuración y control de errores en caso de que alguna función no esté funcionando bien, es con este print se han detectado errores al no haber una similitud entre los datos volcados en el fichero csv y la consola.

Se actualiza una etiqueta en la GUI para mostrar el número total de granos detectados.

5.1.11.8 Mostrar información:

cv2.drawContours(image, grains, -1, (0, 255, 0), 2): Dibuja los contornos detectados en la imagen original en color verde, se ha elegido este color para que se diferencie bien lo que es el grano de su contorno.

5.1.11.9 Actualización de la lista de datos de granos:

self.grain_data = grain_data: Actualiza la lista de datos de granos de la clase con la lista recién generada.

5.1.11.10 Retorno:

return image, grain_data: Devuelve la imagen con los granos resaltados y la lista de datos de los granos.

5.1.12 Clase ToolTip:

La clase **ToolTip** se encarga de proporcionar un cuadro emergente informativo (también conocido como "tooltip") que se muestra cuando el cursor del ratón se encuentra sobre un widget específico en una interfaz gráfica. El propósito principal de esta herramienta es ofrecer información adicional al usuario sobre la funcionalidad o el propósito del widget en cuestión.

Atributos:

widget: Es el widget al cual se le asociará el tooltip.

text: Es el texto que se mostrará dentro del tooltip.

tip_window: Es una ventana de nivel superior que contiene el tooltip. Inicialmente es None y se crea cuando es necesario mostrar el tooltip.

id: Es el identificador de un evento programado. Se utiliza para programar la aparición del tooltip después de un cierto retraso.

x e y: Son las coordenadas del cursor del ratón cuando se desencadena el evento de entrada en el widget.

Métodos:

__init__(self, widget, text):

Constructor de la clase. Inicializa los atributos y vincula varios eventos del widget (<Enter>, <Leave> y <ButtonPress>) a sus respectivos manejadores.

display_tip(self):

Muestra el tooltip en una posición ligeramente desplazada respecto a la posición actual del cursor en el widget.

Crea una ventana de nivel superior (Toplevel) sin bordes para contener el texto del tooltip.

Ajusta la geometría de la ventana según la posición del cursor.

Crea y empaca una etiqueta (Label) con el texto del tooltip en esta ventana.

schedule(self, event):

Programa la aparición del tooltip después de un retraso (500 milisegundos en este caso).

Guarda las coordenadas actuales del cursor del ratón.

hidetip(self, event=None):

Oculta y destruye el tooltip si está siendo mostrado.

Cancela cualquier aparición programada del tooltip si existe.

Vinculación de eventos:

Al mover el cursor sobre el widget (<Enter>), el tooltip se programa para mostrarse después de un retraso utilizando el método schedule.

Al mover el cursor fuera del widget (<Leave>) o al hacer clic en el widget (<ButtonPress>), el tooltip se oculta inmediatamente mediante el método hidetip.

5.2 Obtención de características cuantificables

Para llevar a cabo esta segunda fase de la implementación se decidió crear una clase llamada **Grano** que representa un grano individual en una imagen.

La clase ofrece una variedad de métodos para calcular características y propiedades del grano, que pueden ser útiles para la clasificación, análisis o cualquier otro tipo de operación de procesamiento de imágenes.

Estas características incluyen medidas geométricas (como el área y el perímetro), medidas de forma (como la solidez y la elongación), y medidas basadas en el color y la textura (utilizando GLCM - Matriz de Co-ocurrencia de Niveles de Gris).

5.2.1 Detalles de la clase Grano:

5.2.1.1 Atributos:

px_per_cm: Es un atributo de clase que sacamos de la función **Measure** en la clase principal. Es una constante que representa la relación entre píxeles y centímetros.

center: El centro del grano.

width: Ancho del grano.

height: Altura del grano.

angle: Ángulo del grano.

contour: Contorno del grano en la imagen.

5.2.1.2 Métodos:

__init__(self, center, width, height, angle, contour): Constructor que inicializa los atributos del grano.

__str__(self): Representación en cadena del objeto Grano.

contourArea(self): Devuelve el área del contorno del grano usando la función **contourArea** de OpenCV.

contourPerimeter(self): Devuelve el perímetro del contorno del grano usando la función **arcLength** de OpenCV.

solidity(self): Calcula y devuelve la solidez del grano, que es la proporción del área del contorno del grano al área del casco convexo del grano.

fitEllipse(self): Ajusta y devuelve una elipse al contorno del grano usando la función fitEllipse de OpenCV.

rectangleFill(self): Calcula y devuelve la proporción del área del grano al área del rectángulo que lo rodea. Para ello crea una máscara de ceros del tamaño del rectángulo delimitador, después traslada el contorno a la esquina superior izquierda de la máscara, dibuja el contorno relleno en la máscara, cuenta cuántos píxeles ocupa el grano y por último calcula el relleno necesario. $\frac{Pixel\ Count}{height*width}$

boundingRectanglePerimeter(self): Calcula y devuelve la proporción del perímetro del contorno del grano al perímetro del rectángulo que lo rodea. $\frac{Perimeter}{2*(height+width)}$

equivalentDiameter(self): Calcula y devuelve el diámetro equivalente del grano, que es el diámetro de un círculo con el mismo área que el grano. $2 * \sqrt{\frac{A}{\pi}}$

circulationFactor(self): Calcula y devuelve el factor de circulación del grano, que es la relación entre el perímetro del contorno y $\pi . \frac{P}{\pi}$

compactness(self): Calcula y devuelve la compacidad del grano. $\frac{4\pi A}{P^2}$

elongation(self): Calcula y devuelve la elongación del grano. $\frac{M-m}{M+m}$

aspectRatio(self): Calcula y devuelve la relación de aspecto del grano. $\frac{L}{B}$

feret_diameter(self): Calcula y devuelve el diámetro de Feret del grano. Es una función auxiliar para más adelante calcular la relación entre la superficie y el volumen de un grano.

ratioSurfaceVolume(self): Calcula y devuelve la relación entre la superficie y el volumen del grano. $\frac{A}{M^3}$

meanRGB(self, origImg): Calcula y devuelve el valor medio de los canales RGB del grano. Para ello crea una máscara binaria del mismo tamaño que la imagen original inicializada con ceros, el contorno del grano se dibuja en esta máscara con un valor de 255 (blanco).

Usando la función cv2.mean() se calcula el valor medio de cada canal en la imagen original donde la máscara es blanca y los valores medios se devuelven en un diccionario teniendo como claves los colores RGB.

NDIrg(self, original_image), NDIrb(self, original_image), NDIgb(self, original_image): Calculan y devuelven los índices de diferencia normalizada entre los canales de color R, G, y B.

Primero se obtiene los valores medios de los canales RGB utilizando la función **meanRGB** después se calcula el NDI utilizando la fórmula $\frac{|R-G|}{R+G+\epsilon}$ donde épsilon es una pequeña constante que evita la división por cero. Las dos fórmulas se calculan de manera similar pero con sus respectivas fórmulas $\frac{|R-B|}{R+B+\epsilon}$ $\frac{|G-B|}{G+B+\epsilon}$

mean_hsv(self, original_image): Calcula y devuelve el valor medio de los canales HSV del grano.

La imagen original en formato BGR se convierte al espacio de colores de HSV, luego se crea una máscara binaria del contorno del grano al igual que en **meanRGB**.

Se utiliza la función **cv2.mean** para calcular el valor medio de cada canal HSV dentro del contorno el grano y por último se devuelven los valores medios en un diccionario teniendo como claves **hue**, **saturation** y **value**.

glcm_contrast(self, origImg): Calcular el contraste de la textura del grano utilizando la GLCM. Para ello convierte la imagen original a escala de grises, crea una máscara binaria que representa el contorno del grano, usa la máscara para aislar el grano en la imagen en escala de grises.

Calcula la matriz GLCM y utiliza dicha matriz para calcular el contraste, que mide la diferencia local entre la imagen.

Su fórmula es la siguiente: $\sum_{i,j=0}^{levels-1} \boldsymbol{P_{i,j}(i-j)^2}$

glcm_dissimilarity(self, origImg): Calcular la disimilitud de la textura del grano utilizando la GLCM. Para ello sigue los mismos pasos que **glcm_contrast** para obtener la matriz GLCM y posteriormente calcula la disimilitud, que es una medida de variación de la propia matriz.

Su fórmula es la siguiente: $\sum_{i,j=0}^{levels-1} \boldsymbol{P_{i,j}\,|i-j|}$

homogeneity(self, origImg): Calcular la homogeneidad de la textura del grano utilizando la GLCM. Sigue los mismos pasos que las anteriores fórmulas para sacar la GLCM y la homogeneidad se encarga de medir la proximidad de la distribución de elementos en la matriz GLCM al diagonal de la matriz, es así como podemos calcular dicha homogeneidad.

Su fórmula es la siguiente: $\sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2}$

ASM(self, origImg): Calcular el Medida de Energía Angular Segunda (ASM) de la textura del grano utilizando la GLCM. Sigue los mismos pasos que las anteriores fórmulas para sacar la GLCM, el ASM es una medida de homogeneidad de la textura en la imagen. Es la suma de los cuadrados de cada elemento en la GLCM. También se le llama **energía**.

Su fórmula es la siguiente: $\sum_{i,j=0}^{levels-1} P_{i,j}^2 \sqrt{ASM}$

correlation(self, origImg): Calcular la correlación de la textura del grano utilizando la GLCM. Sigue los mismos pasos que las anteriores fórmulas para sacar la GLCM. La correlación mide la dependencia lineal de los niveles de gris en la GLCM. Es un valor que oscila entre -1 y 1. Un valor cercano a 1 indica que los niveles de gris en la imagen son linealmente dependientes entre sí.

Su fórmula es la siguiente: $\sum_{i,j=0}^{levels-1} P_{i,j}[\frac{(i-\mu_i)(j-\mu_j)}{\sigma_i^2\sigma_j^2}]$

A continuación se muestra una tabla con todas las magnitudes de los atributos calculadas por la aplicación y se explica brevemente cada uno de ellos. Esta información es la que se almacena en el fichero CSV de características:

Atributo	Significado	Magnitud	Ejemplo
Center	Coordenadas del centro del grano de trigo	Pixeles	(268.5, 247.5)
Witdh	Anchura del grano de trigo	Cm	0.716332051
Height	Altura del grano de trigo	Cm	0.315186103
Contour Area	Superficie del área del grano de trigo	Cm^2	0.17911714
Contour Perimeter	Perímetro del contorno del grano de trigo	Cm	1.80798832
Solidity	Cociente entre el área real del grano de trigo y el área convexa del polígono que lo contiene	No tiene. El valor fluctúa entre 0 y 1. 1 el grano es totalmente sólido. 0 no se ha detectado grano.	0.969390274
Fit Ellipse	Devuelve un triplete con el centro de la elipse, las longitudes de los ejes mayor y menor y el ángulo de rotación de la elipse	Centro y longitud de ejes en pixeles. Rotación en grados.	((268.51,247.22), (32.75,77.42), 176.66)
Rectangle Fill	Cociente entre el número de pixeles del grano y el área del mínimo rectángulo que lo contiene	No tiene. El valor fluctúa entre 0 y 1. 0 no se ha detectado grano y 1 el grano coincide totalmente con su rectángulo	0.734116196
Bounding Rectangle Perimeter	Cociente entre el perímetro del grano y el perímetro del mínimo rectángulo que lo contiene	No tiene. El valor fluctúa entre 0 y 1. 0 no se ha detectado grano y 1 el grano coincide	0.880727819

		totalmente con su rectángulo	
Equivalent Diameter	Diámetro del grano de trigo	Cm	0.443179827
Circulation Factor	Cociente entre el perímetro y PI. Es una variante de una métrica común de circularidad	No tiene. Valores cercanos a 1 indican que el grano tiene forma circular.	0.560632
Compactness	Cociente entre 4*PI*Area del grano y su perímetro al cuadrado	No tiene. El valor fluctúa entre 0 y 1. 0 no se ha detectado grano y 1 los pixeles del grano detectado coinciden totalmente con el área selecciónada.	0.628454517
Elongation	Cociente que indica cómo de alargado es el grano	No tiene. El valor fluctúa entre 0 (M=m, el grano es un círculo o un cuadrado) y 1 (m = 0, es una línea recta o extremadamente elongada)	0.435816155
Aspect Ratio	Cociente entre la altura y la anchura del grano	No tiene. Tampoco fluctúa entre dos valores, dependerá de las características de cada grano.	2.544943757
Ratio Surface of Volume	Cociente entre el área de la superficie y el volumen al cubo del grano. Este ratio proporciona una medida de cuánta superficie externa tiene un objeto en relación con su volumen	No tiene. Un valor alto indica que el objeto tiene una gran superficie externa en compraración con su volumen. Un valor bajo indica superficie externa pequeña en comparación con su volumen.	1.77473E-11
Mean RGB	Diccionario que contiene la media de cada canal de color del grano (Blue, Green, Red)	No tiene. Cada canal va de 0 a 255. 0 indica ausencia de señal y 255 máxima intensidad.	{'blue': 105.03, 'green': 99.53, 'red': 93.46}
NDIrg	Es una métrica utilizada en análisis de imágenes, para determinar la "verdez" o salud de la vegetación. Calcula la diferencia	No tiene. Las áreas con valores más altos de NDIrg suelen indicar vegetación más saludable o densa,	0.02422632

	normalizada entre los canales rojo y verde.	mientras que valores más bajos pueden indicar áreas estresadas o menos densas.	
NDIrb	Calcula la diferencia normalizada entre los canales rojo y azul.	No tiene.	0.052792742
NDIgb	Calcula la diferencia normalizada entre los canales verde y azul.	No tiene.	0.028603005
Mean HSV	Diccionario que contiene la media de cada canal de cada grano en términos de sus componentes de Tono, Saturación y Valor.	No tiene. Cada canal va de 0 a 255. 0 indica ausencia de señal y 255 máxima intensidad.	{'hue': 106.52, 'saturation': 25.61, 'value': 103.88}
GLCM contrast	Métrica que se utiliza para examinar la textura de una imagen. Analiza cómo se distribuyen las combinaciones de píxeles adyacentes (o píxeles separados por una distancia específica) en una imagen.	No tiene. Un valor alto de contraste indica que hay una gran diferencia en los niveles de gris de los píxeles adyacentes. Un valor bajo indica que hay pequeñas diferencias en los niveles de gris de los píxeles adyacentes, lo que sugiere que la textura es más suave o uniforme.	0.460662441
GLCM Dissimilarity	Métrica que se utiliza para describir la variación en la uniformidad de una imagen.	No tiene. El valor mínimo de disimilitud es 0, lo que ocurre cuando todos los píxeles adyacentes tienen la misma intensidad. El valor máximo de disimilitud es L−1, que sería el caso en el que las diferencias de intensidad entre píxeles adyacentes son siempre el máximo posible.	0.005928495
Homogeneity	Métrica que se utiliza para medir cuán uniformes son	No tiene. El valor de la homogeneidad varía	0.999594225

	los pares de píxeles en una imagen, es decir, cuán similares son en intensidad los píxeles adyacentes.	entre 0 y 1. Un valor cercano a 1 indica una imagen más homogénea, mientras que un valor cercano a 0 sugiere una mayor variabilidad o diferencia en las intensidades de los píxeles adyacentes.	
ASM	Métrica que mide la uniformidad o regularidad de una imagen.	No tiene. El valor de ASM varía entre 0 y 1. Un valor de ASM cercano a 1 indica que la imagen tiene una textura muy constante o uniforme, porque hay pocos valores en la GLCM que tienen altas probabilidades. Por otro lado, un valor de ASM cercano a 0 sugiere una imagen con textura más variada.	0.998795631
Correlation	Métrica que mide cómo está correlacionada una píxel con su vecino a lo largo de toda la imagen.	No tiene. La correlación varía entre -1 y 1: Un valor cercano a 1 indica que la textura tiene una dependencia lineal positiva entre los niveles de gris de los píxeles vecinos. Un valor cercano a -1 indica una dependencia lineal negativa. Un valor cercano a 0 indica la ausencia de cualquier dependencia lineal.	0.95802476

Tabla 5.1. Explicación detallada de los atributos que se recogen por cada grano de trigo, magnitudes y valores de ejemplo.

6

Verificación

La **verificación** es una fase fundamental en el ciclo de vida del desarrollo de software. Su objetivo principal es asegurarse de que el software que se ha desarrollado cumple con las especificaciones y requerimientos iniciales. A continuación, se detalla más profundamente este concepto:

6.1 Objetivo Principal

La verificación se centra en responder a la pregunta: "¿Se está construyendo el proyecto correctamente?". En otras palabras, se asegura de que el software se esté desarrollando de la manera correcta y que cumpla con los estándares de calidad definidos.

6.2 Pruebas

Para llevar a cabo una buena verificación en este proyecto, se ha considerado que no hacía falta una automatización de pruebas o generar un proceso iterativo para cumplir con los estándares preestablecidos sino que bastaba con centrar los esfuerzos en hacer una batería de pruebas manualmente.

Durante estas pruebas, cada componente se prueba de manera aislada del resto del sistema para asegurarse de que funcione correctamente en una variedad de escenarios o conjuntos de datos. Cabe destacar que aunque es cierto que probar los componentes de manera aislada es una buena práctica, ha habido ciertas situaciones donde era más conveniente probar dos o incluso tres componentes aislados pero funcionando entre sí para comprobar cómo interactuaban entre ellos.

Por ejemplo, una prueba unitaria podría centrarse en una función que ajusta el contraste de una imagen. Se probaría esa función con varias imágenes y niveles de contraste para asegurarse de que produce los resultados esperados.

Hubo un ejemplo real que ocurrió probando los métodos de la clase Grano que explica claramente el párrafo anterior. Probando dichos métodos, surgió un problema, y es que al ser una cantidad de datos tan grande, la consola python mostraba tantos datos que era prácticamente imposible interpretarlos y si tenían sentido o no.

Para solucionar este problema se optó por dejar las instrucciones print solo en algunos de los métodos, dejando así una consola más limpia y legible. Por otra parte se volcaron los resultados de todos los métodos en un archivo csv haciendo uso de la función **perform_analysis** dentro de la clase principal.

Con esto, los datos de los granos se veían mucho mejor y además se podían comparar con los datos obtenidos en consola y ver si efectivamente coincidían o si por el contrario, los métodos en conjunto no funcionaban bien.

6.4 Importancia de la Verificación

Detecta problemas temprano: La detección temprana de errores puede reducir significativamente los costos de corrección. Es mucho más costoso corregir un error después de que el software ha sido desplegado que durante las etapas iniciales de desarrollo.

Garantiza la calidad: La verificación asegura que el software cumple con los estándares de calidad y que cada componente funciona como se espera.

Proporciona confianza: Al verificar cada componente individualmente, se puede tener confianza en que cada pieza del software funciona correctamente. Esto es esencial antes de pasar a la fase de validación, donde se prueba el sistema como un todo.

Es cierto que la verificación de cada componente de manera individual es un trabajo tedioso pero es así como se aseguró su correcto funcionamiento y además organiza mucho el código, es decir, si está verificado que todos los métodos hechos hasta cierto punto funcionan correctamente, sabes que lo que hace que la aplicación no funcione correctamente es el último método con el que estás trabajando. Es una forma de "pisar sobre seguro" que hace que el trabajo sea menos caótico en el caso de que empiecen a surgir errores.

7

Creación del dataset

Aunque ya hemos hablado en la implementación del código de cómo se crea el dataset, se explicará de manera detallada el método encargado de esta tarea y más adelante cómo interpretar los datos.

Dentro de los propios requisitos del proyecto estaba especificado que se necesitaba que los el dataset estuviese guardado en un archivo csv, es por ello que se hizo uso del **módulo csv** del que ya se habló anteriormente.

En un primer lugar, el objetivo era crear un dataset de manera automática, es decir, dada un conjunto de imágenes, exportar los datos de todas las imágenes a un fichero csv de manera automática pero este requisito fue descartado por el ROI.

Las regiones de interés se implementaron debido a que las imágenes no solamente tenían granos en ellas sino que también contaban con una regla para calibrar e incluso una etiqueta. Al tener otros elementos que no eran granos, el algoritmo podía detectar dichos elementos o partes de estos como granos y salían resultados erróneos.

Al implementar el ROI era imposible automatizar el procesamiento de muchas imágenes a la vez porque hacía falta que manualmente el usuario creara el ROI para cada imagen.

A continuación se explicará qué hace paso por paso la función **perform_analysis**:

Primero verifica si hay datos de granos disponibles para el análisis. Si no se han detectado granos aún, muestra una advertencia al usuario solicitando que detecte granos primero. (función **Focus** o **Enfoque**)

También verifica si se ha establecido el número de píxeles por centímetro que tiene la imagen y si no se ha establecido aún, muestra una advertencia al usuario solicitando que primero utilice la función de medición primero. (función **Measure** o **Medición**)

Se le pide al usuario que proporcione un nombre para el archivo CSV donde se guardarán los datos. Si el usuario no proporciona ningún nombre la función termina.

Se define una lista llamada **Header** que contiene los nombres de las columnas para el archivo CSV, en dicha lista están los nombres de todas las fórmulas o ecuaciones que se aplicarán a cada grano.

Se abre el archivo CSV en modo escritura.

Se escribe la cabecera en el archivo CSV.

Para cada grano en los datos de grano (**self.grain_data**), se calculan diferentes métricas y propiedades usando varios métodos del objeto grain.

Cada conjunto de métricas para un grano se escribe como una fila en el archivo CSV.

Si ocurre un error al procesar un grano (por ejemplo, si falta un atributo o si hay un error en el cálculo), se imprime un mensaje de error, pero la función continúa procesando los siguientes granos. Esta impresión se da en la consola python para depurar y testear el programa pero no aparece como un pop-up.

Una vez que se han procesado todos los granos y se han escrito los datos en el archivo CSV, se muestra un mensaje informativo al usuario indicando que el análisis se ha completado y se proporciona la ruta del archivo CSV donde se guardaron los datos.

En resumen, **perform_analysis** es una función que analiza métricas y propiedades de granos, las escribe en un archivo CSV y proporciona retroalimentación al usuario a través de cuadros de diálogo. Es una función integral que combina varias etapas del proceso de análisis y exportación en una sola operación.

Existen una serie de medidas importantes a la hora de cómo interpretar el dataset que se explicarán en el manual de usuario.

Center	Width (Cm)	Height (Cm)	Contour Area (Cm^2)	Contour Perimeter (Cm)	Solidity	Fit Ellipse
(46.0, 263.0)	0.70610687	0.305343511	0.1639801	1.781590424	0.954677975	((46.98044967651367, 261.8159484863281), (31.311605453491 21, 75.045166015625), 176.0702056884
(154.5, 256.0)	0.667938931	0.391221374	0.203950819	1.850021464	0.981165134	((153.4955139160156 2, 255.8802490234375), (38.59513473510742, 75.92758178710938), 164.0247039
(259.5, 250.5)	0.715648855	0.333969466	0.188791082	1.814168801	0.979452055	((259.9624328613281, 250.3329620361328), (35.03662109375, 76.06427764892578), 176.5581054687 5
(39.746150970458984, 153.43075561523438)	0.662146852	0.273115744	0.150595828	1.657825522	0.972369195	((40.066856384277344, 154.43061828613 28), (30.331193923950195, 74.38349914550781), 176.102600
(147.5, 152.0)	0.667938931	0.295801527	0.150914501	1.675950467	0.974140464	((147.6634979248047, 151.526733398437 5), (29.767507553100586, 71.62716674804688), 172.0089111
(254.5, 145.5)	0.677480916	0.276717557	0.149139036	1.692719167	0.970379147	((254.2387237548828, 145.89199829101562), (28.414756774902344, 75.5962905883789), 4.904316902
(34.519229888916016, 53.403846740722656)	0.29567127	0.581985867	0.137621278	1.534457843	0.966741286	((34.02816390991211, 54.29928207397461), (30.945302963256836, 62.94706726074219), 10.04337787
(137.6321563720703, 50.54936218261719)	0.611180822	0.231548684	0.106801031	1.486069793	0.960687961	((137.8792877197265 6, 50.239280700683594), (23.5683650970459, 65.81059265136719), 163.0940704
(249.0, 40.5)	0.696564885	0.286259542	0.161339666	1.764424498	0.963829209	((248.8631591796875, 41.439510345458984), (30.229564666748047, 77.2925033569336), 178.0243682

Figura 7.1: Ejemplo de dataset o resultados

8

Conclusiones y trabajo futuro

8.1 Conclusiones

Tras un análisis detenido y exhaustivo del proyecto desarrollado para la detección y posterior tratamiento de granos de trigo, es evidente que estamos ante una herramienta que puede llegar a tener un gran valor tanto para el sector agrícola como para la comunidad científica y tecnológica.

A continuación, se presentarán mis conclusiones y reflexiones sobre este proyecto:

La estructura de la aplicación, la herramienta de desarrollo, así como el haber usado la librería openCV hacen que el proyecto sirve de base para futuras adaptaciones a distintos usos, hecho que es importante para favorecer la evolución tecnológica en aspectos que se desconocen hoy en día. Creo que puede servir de andamio para apoyar otros proyectos relacionados con otros sectores de producción. Es motivador sentir que tu esfuerzo pueda servir para otros desarrolladores.

Me ha gustado trabajar en un proyecto de este tipo porque a medida que me iba informando sobre el tópico, me iba dando cuenta de que no es una herramienta trivial, sino que puede llegar a tener una utilidad real y que se pueda llegar a usar para fines empresariales o personales y eso me motivó mucho a la hora de buscar estructuras fáciles de usar y fiables para que una programado fuese capaz de interpretar el código sin demasiadas complicaciones.

El hecho de que el trigo sea una de las tres plantas más importantes del mundo para el ser humano junto con el arroz y el maíz hace, que si la aplicación se sigue mejorando y actualizándose según los nuevos requisitos que vayan apareciendo en el sector puede llegar a ser una herramienta importante para el sector o como mínimo útil.

Centrándome un poco más en mi persona, debo de decir antes que nada que ha sido una experiencia gratificante por poder dar un paso atrás y ver cómo una idea que estaba en el aire la he podido transformar en una realidad.

No ha sido algo sencillo porque la libertad que se ha dado a la hora de qué decisiones tomar a veces era abrumador (por otro lado, me ha permitido trabajar como lo exigen en las empresas de informática que trabajan en equipos de trabajo empoderados con objetivos y en las que te exigen tener habilidades blandas como la proactividad y ser resolutivo) al no tener experiencia en este tipo de proyectos pero viéndolo ahora en perspectiva es una sensación agradable.

En un principio esperaba que la complejidad del proyecto fuese menor de la que ha sido pero como he hablado en el párrafo anterior, el tener carta blanca para hacer el proyecto como estimase oportuno cumpliendo siempre con los requisitos principales ha hecho de este proyecto todo un reto donde ha habido muchos pasos hacia adelante y algunos hacia atrás.

Al haber estudiado el grado de computadores, en ninguna asignatura se me ha enseñado a hacer interfaces gráficas y he tocado muy poco sobre el tema de creación de aplicaciones, tampoco había programado python anteriormente, por ello, antes de empezar a organizar todo, estuve un mes estudiando por mi cuenta los conceptos básicos de python (sobre todo estuve leyendo este libro *Ramírez, O(2021). Python a fondo, Primera Edición, Spain, Marcombo)* y leyendo algún libro sobre GUI en python como el que he dejado en las referencias *Meier, B (2019). Python GUI Programming, Tercera Edición, Birminghan, Packt.*

El hecho de aprender por mi cuenta ciertos conocimientos que no me habían impartido anteriormente ha fortalecido mi capacidad autodidacta, he aprendido mucho de Python, sobre cómo crear GUIs, sobre cómo funciona una aplicación y también sobre cuales son los procesos necesarios para la detección y el posterior tratamiento de uno o varios objetos.

En resumen, este proyecto me ha permitido ampliar los conocimientos adquiridos en el grado y estar más preparado para la inserción laboral ya que, no sólo he aprendido a programar en lenguajes de que usan la mayoría de las empresas, sino que también he mejorado mis habilidades organizativas, de comunicación escrita, de análisis y crítica, de auto aprendizaje y de aprendizaje permanente, habilidades muy apreciadas en cualquier empresa.

8.2 Trabajo Futuro

Algo que me ha ocurrido cuando estaba desarrollando los métodos era que a medida que avanzaba, más ideas se me ocurrían para que en un futuro alguien pueda recoger el relevo y seguir con una segunda parte de este proyecto. Por ejemplo, haciendo uso de IA se podría hacer una interpretación de los datos que la aplicación extrae para identificar qué tipo de grano de trigo es, de qué tipo de cultivo procede, si pertenece a

una cosecha de sequía… Además por cómo está estructurado el código y las clases, es muy fácil añadir nuevos métodos al dataset.

Otra idea interesante sería la de mejorar imágenes con un alto contraste, es decir, tomas al sol y con mucho brillo o luz o por el contrario, a la sombra y con poca luz.

Aunque creo que esta idea en cierta parte ya podría funcionar según el caso, mejorar el algoritmo para que los granos de las imágenes no tengan que estar tan separados y ordenados para el correcto funcionamiento de la aplicación.

Es muy probable que las personas que usen esta aplicación estén en el campo o al menos usen la aplicación allí, por ello, es improbable que se tenga siempre una regla a mano, la mejora sería que al darle al botón de Medición, el usuario pudiese elegir entre diferentes elementos con los que poder medir, tendríamos la clásica regla pero por ejemplo también poder medirlo con una moneda estándar de un euro, siempre tiene el mismo tamaño y se podría hacer fácilmente una conversión.

La primera mejora que se me vino a la cabeza es la de desplegar esta aplicación en la web ya que creo que gana mucho desde el punto de vista del usuario sobre todo por la comodidad que esto supone.

Como se mencionó anteriormente en esta memoria, un problema que surgió a partir del formato de las imágenes era la obligación de crear un ROI (Región de Interés) porque las imágenes contenían otros elementos que no eran granos y esto producía un mal funcionamiento del algoritmo. Es por ello que una posible mejora es la de encontrar alguna forma de poder crear un ROI de manera automática para un listado de imágenes y que el propio algoritmo fuese capaz de reconocer cuáles serían los valores adecuados para la detección de granos, en resumen, automatizar el trabajo que tiene que hacer el usuario para un listado de imágenes completo sin tener que ir una por una.

Referencias

CSV. (2023). *CSV File Reading and Writing*. Obtenido de https://docs.python.org/3/library/csv.html

Meier, B. (2019). *Python GUI Programming* (3ª ed.). Birminghan: Packt.

Meier, B. (2019). *Python-GUI-Programming-Cookbook-Third-Edition*. (P. Publishing, Editor) Obtenido de Python-GUI-Programming-Cookbook-Third-Edition: https://github.com/PacktPublishing/Python-GUI-Programming-Cookbook-Third-Edition

NumPy. (2023). *NumPy documentation*. Obtenido de https://numpy.org/doc/stable/index.html

OpenCV. (2023). *OpenCV-Python Tutorials*. Obtenido de https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

Pillow. (2023). *Pillow documentation*. Obtenido de https://pillow.readthedocs.io/en/stable/

ProjectManager. (2019). *How to structure your project*. Obtenido de https://www.projectmanager.com/blog/project-organization-101

Ramírez Jiménez, O. (2021). *Python a Fondo.* Marcombo.

Rohit Sharma, M. K. (2021). Image processing techniques to estimate weight and nmorphological parameters for selected wheat refractions. *Scientific Reports*. Obtenido de https://rdcu.be/dk4pC

scikit-image. (2023). *scikit-image API reference*. Obtenido de https://scikit-image.org/docs/stable/api/skimage.feature.html

Sys. (2023). *System-specific parameters and functions*. Obtenido de https://docs.python.org/3/library/sys.html

Tkinter. (2023). *Tkinter-Python interface to Tcl/Tk*. Obtenido de https://docs.python.org/3/library/tkinter.html

Apéndice I

Manual de Usuario

Bienvenido al manual de usuario de la Aplicación de Detección de Granos de Trigo. Esta herramienta ha sido diseñada con el propósito fundamental de facilitar y optimizar el proceso de identificación, análisis y tratamiento posterior de granos de trigo mediante técnicas avanzadas de procesamiento de imágenes y visión por computadora.

La creciente demanda en la industria agrícola de herramientas precisas y eficientes para la clasificación y análisis de granos ha llevado al desarrollo de esta aplicación, que busca no solo mejorar la calidad y eficiencia del proceso, sino también proporcionar una base sólida para investigaciones y desarrollos futuros en el campo.

Este manual tiene como objetivo proporcionar a los usuarios una guía detallada y comprensible sobre cómo utilizar la aplicación, maximizando sus beneficios y comprendiendo a fondo todas sus características y funcionalidades. Se ha estructurado de manera que tanto usuarios novatos como experimentados puedan encontrar respuestas claras y directas a sus preguntas y llevar a cabo sus tareas de forma eficiente.

Requisitos del sistema: Especificaciones necesarias para garantizar el funcionamiento óptimo de la aplicación. Una de las características positivas de esta aplicación es que no necesita ningún tipo requisito especial para el correcto funcionamiento de la aplicación, está hecha para ser ejecutada en PC y no en dispositivos móviles.

Instalación y Configuración: Pasos para instalar y configurar la aplicación en su sistema.

Interfaz de Usuario: Una descripción detallada de la interfaz de la aplicación, sus componentes y cómo interactuar con ellos.

I. Interfaz de Usuario

Para empezar a explicar cómo manejar la aplicación se enseñará una imagen de la interfaz general nada más abrirla.

Figura I.1: Imagen general de la aplicación al ser ejecutada

Existe un pequeño menú arriba con las siguientes funciones básicas:

Por un lado está el menú **File** donde se encuentran los submenús **Open** que se encarga de abrir una imagen y **Exit** que se encarga de cerrar la aplicación.

El otro menú se llama **Help** y tiene dos submenús, el primero llamado **About** donde al pinchar en él nos aparece un pop-up con información sobre quién ha hecho esta aplicación y quién ha sido la persona encargada de guiarle. El segundo submenú se llama **Show State**, es otro pop-up que nos muestra información del estado actual de la aplicación.

Figura I.2: Pop-up resultante de pinchar en la opción **About**

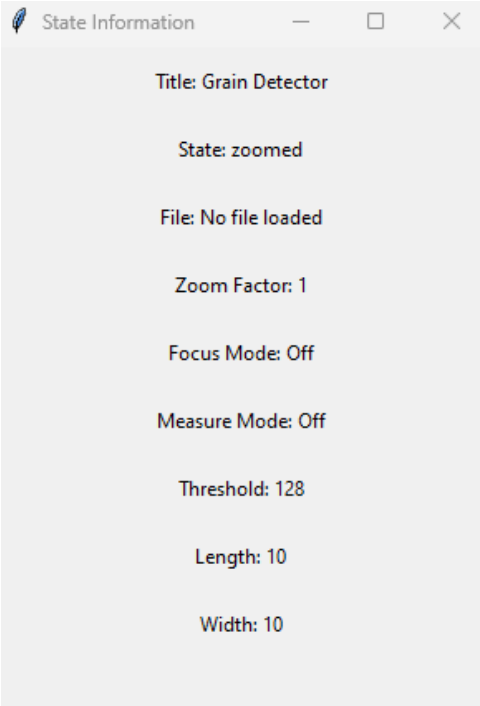


Figura I.3: Pop-up resultante de pinchar en la opción **Show State**

Los valores de las características recogidas en esta imagen son los valores estándar que tiene la aplicación nada más abrirse.

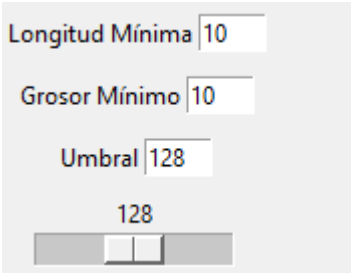


Figura I.4: Panel de control de la aplicación

Estos son los valores que el usuario puede controlar a la hora de detectar los granos dentro de una imagen, cuanto mayor sea el tamaño de los granos, mayor valor se le deberá de dar en los campos de texto **Longitud Mínima** y **Grosor Mínimo**. Estos valores son muy importantes ya que por ejemplo si la imagen dada tiene pequeñas partículas de color claro, el algoritmo podría confundirlas con granos. Es por ello que necesitamos un grosor y longitud mínima, para crear una especie de filtro manual según la imagen que el usuario necesite.

El umbral tiene un rango de 0 a 255, cuanto menor sea este valor, objetos más oscuros detectará y cuanto mayor sea el rango, objetos más claros serán detectados. Pongamos un ejemplo de dos imágenes:

Figura I.5: Imagen de ejemplo

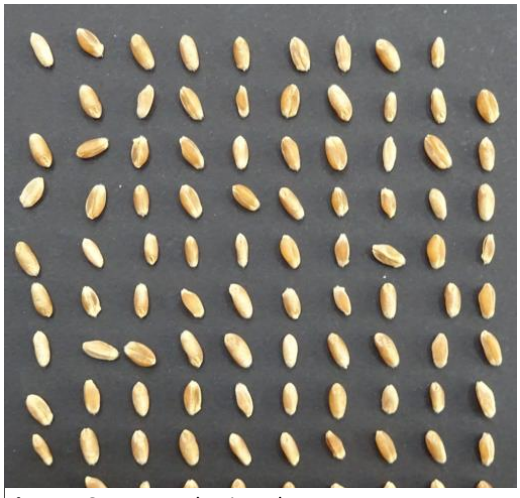


Figura I.6: Imagen de ejemplo

En la figura I.5 el tono de colores de la imagen se podría decir que es normal o incluso un poco oscuro, es por ello que para que el algoritmo funcione correctamente hemos tenido que poner un umbral menor que la media, en específico el valor del umbral era de 82.

Por otro lado, la figura I.6 tiene un tono bastante más claro y por ello, se ha incrementado el valor del umbral a 120.

Este paso no es trivial porque cada grano de manera individual puede tener varios tonos de colores, algunos claro y otros oscuros y por ello hay que ir probando poco a poco hasta encontrar el umbral que recoja el mejor contorno posible.

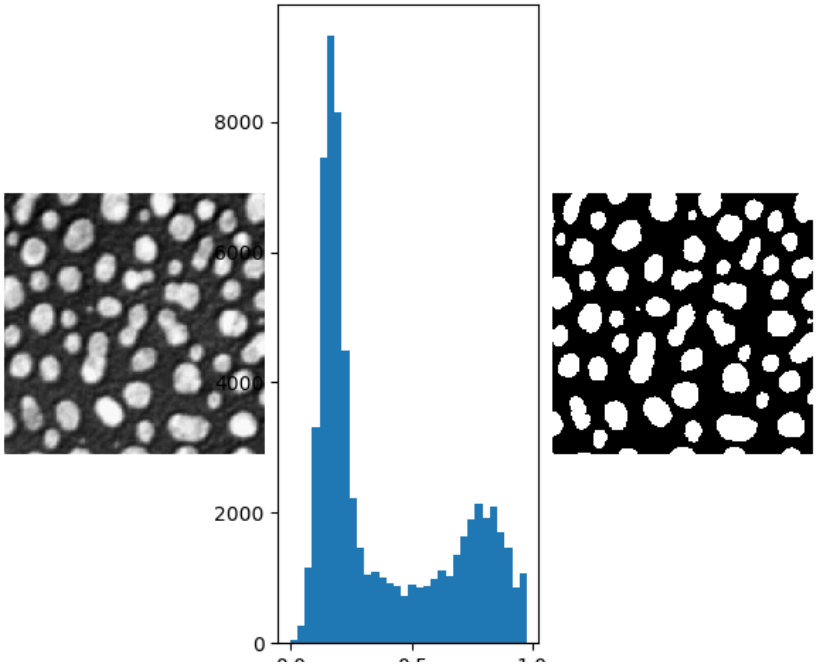


Figura I.7: Ejemplo de imagen original // Su histograma // Imagen resultante con una umbralización aplicada

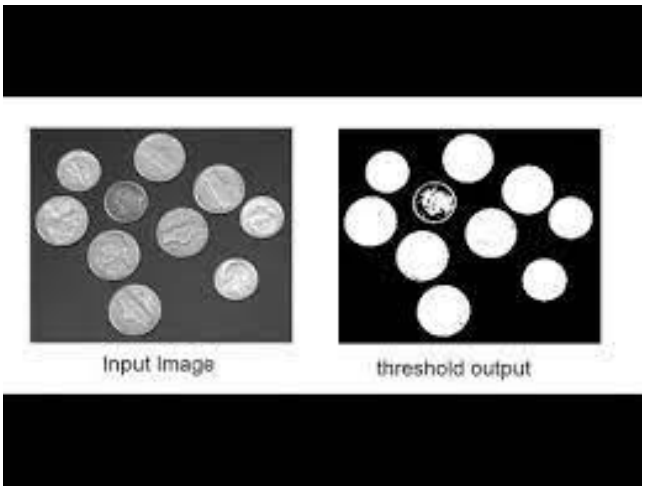


Figura I.8: Otro ejemplo sobre el resultado de una imagen al aplicarle un umbral

Ahora se explicará los diferentes botones que hay en la aplicación:

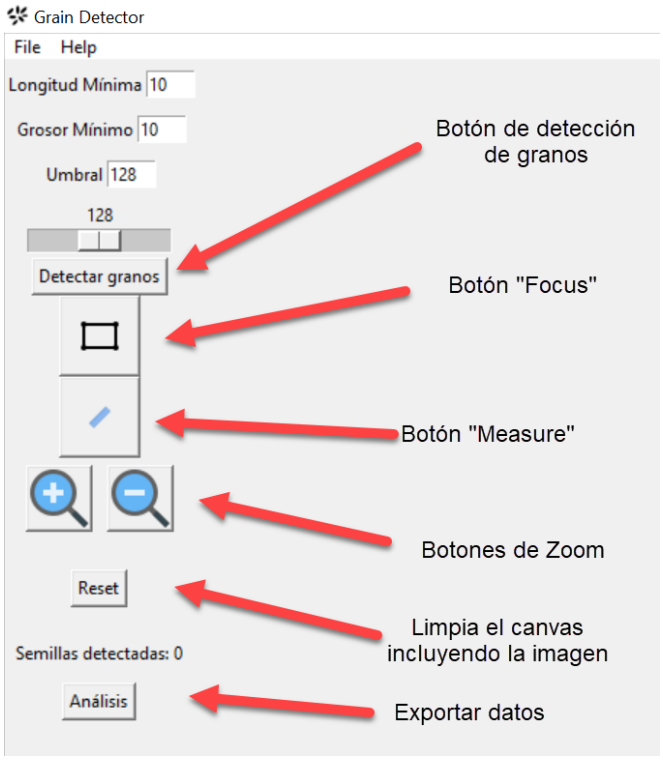


Figura I.9: Explicación de los botones de la aplicación

Para explicar los botones primero se explicará el orden en el que se aconseja hacer las cosas para el correcto funcionamiento de la aplicación.

Primero hay que rellenar los campos de texto de arriba con los valores que el usuario considere oportuno, como ya se ha mencionado, estos campos de textos traen un valor estándar pero es aconsejable cambiarlos según qué imagen.

Después se debe de elegir una región de interés usando el botón **Focus**. Una vez se pinche en el botón, hay que pinchar en una zona de la imagen y arrastrar, esto generará un rectángulo rojo.

El algoritmo funcionará **SOLAMENTE** en la porción de imagen del rectángulo, si el usuario se equivoca o quiere probar otra ROI se puede volver a pinchar el botón sin problemas.

Después de haber usado el botón **Focus**, es recomendable usar la función de medición o **Measure**, una vez se haya pinchado en el botón solamente hay que pulsar en el principio de la regla que tiene cada imagen y soltar cuando se haya llegado a la distancia de 10 centímetros.

La función **Measure** está programada para reglas de 10 centímetros, hay algunas de las reglas que tienen hasta 11 centímetros pero con quedarse con los primeros diez centímetros **Measure** funcionaría bien.

Los botones del **Zoom** es una funcionalidad para el usuario pero no afecta en nada al funcionamiento de la detección de granos.

El botón **Reset** sirve para eliminar todos los ROIs, la imagen abierta y todas las mediciones que se hicieron para en vez de tener que cerrar la aplicación y volverla a abrir, darle al botón de reset en caso de querer empezar de cero.

Por último el botón **Analyze**, este botón es el encargado de exportar todos los datos y características de los granos a un fichero CSV, para ello es OBLIGATORIO el haber hecho uso anteriormente de los botones **Focus** y **Measure** y evidentemente de haberle dado al botón **Detectar granos**. Es decir, debemos de tener un ROI, una medición de la imagen en centímetros y haber lanzado el algoritmo sobre dicho ROI.

Con todos estos pasos dados, analyze podrá llamar a todos los métodos y crear el dataset correspondiente.

Para terminar con la parte de la aplicación en sí, se mostrará una imagen con un ROI y una medición de la imagen:

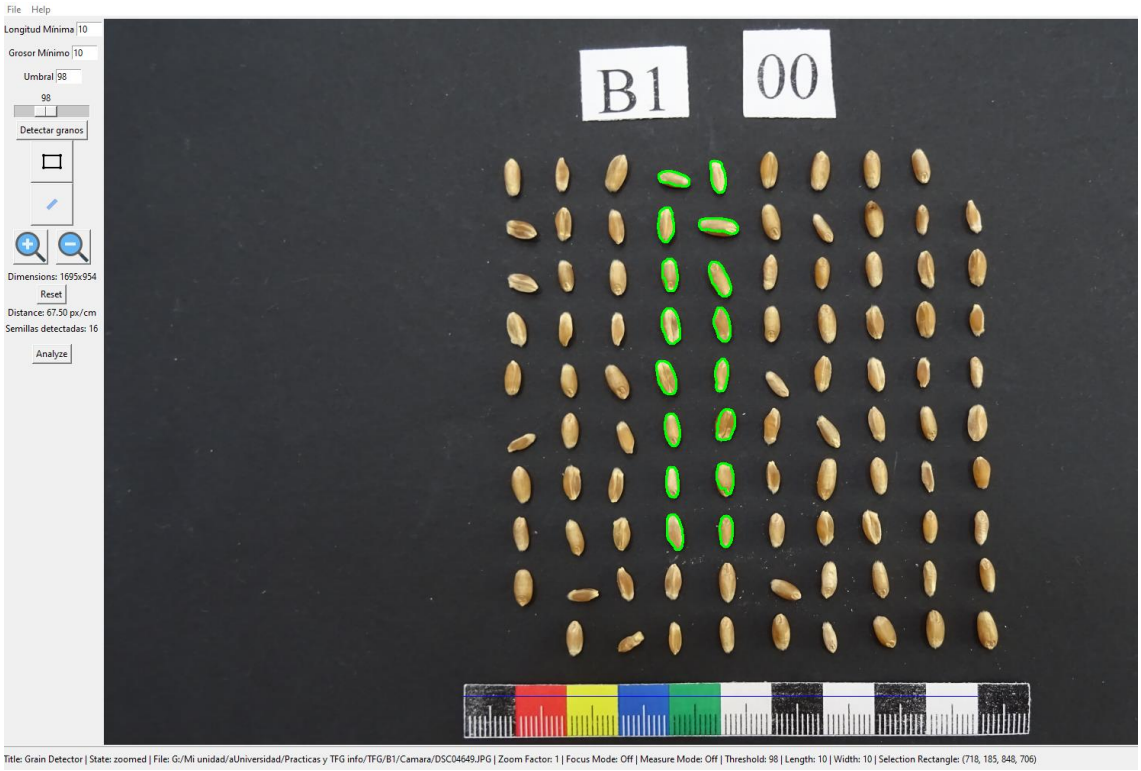


Figura I.10: Aspecto de la aplicación con una imagen cargada y con granos detectados. Nótese que el botón "measure

El ROI que se ha elegido son dos columnas casi completas para demostrar que la región no tiene por qué ser la imagen completa, también puede ser un área de dicha imagen.

Podemos observar la línea recta azul que está atravesando la regla, esa línea recta que cuenta 10 centímetros es el medidor de píxeles por centímetro.

Debajo del todo podemos ver la barra de estados de la aplicación donde aparece información del estado actual de la aplicación como por ejemplo la dirección de la imagen que se está tratando, el zoom, el valor de las variables de control…
Se puede observar también que al abrir una imagen nos aparece información de las dimensiones de la imagen por si es de utilidad, al usar el botón de medición también aparece cuántos píxeles por centímetro tiene la imagen y al pinchar en el botón **Detectar granos** también aparece cuántas semillas se han detectado.

I.2 Análisis de Resultados

En este apartado se explicará cómo interpretar los resultados obtenidos del fichero CSV que hemos generado anteriormente.

Estos resultados son mostrados de forma internacional (anglosajona) lo que conlleva a una forma de mostrarlos diferente a la española, se explicará cómo hay que configurar el excel para que las unidades no se malinterpreten.

Para que Excel interprete y lea números en el formato inglés (por ejemplo, usar un punto como separador decimal y una coma para separar miles), es necesario ajustar la configuración regional del sistema o modificar la configuración de Excel directamente. A continuación se describe cómo realizarlo en ambos casos:

Cambiar la configuración regional del sistema (Windows):

1. Abrir el Panel de Control (se puede escribir "Panel de Control" en la barra de búsqueda de Windows).
2. Seleccionar "Reloj y región".
3. Posteriormente, seleccionar "Región".
4. En la ventana que se despliega, acceder a la pestaña "Formatos".
5. En la sección "Formato:", elegir "Inglés (Estados Unidos)" u otro formato en inglés de preferencia.
6. Finalizar haciendo clic en "Aceptar" para aplicar los cambios.

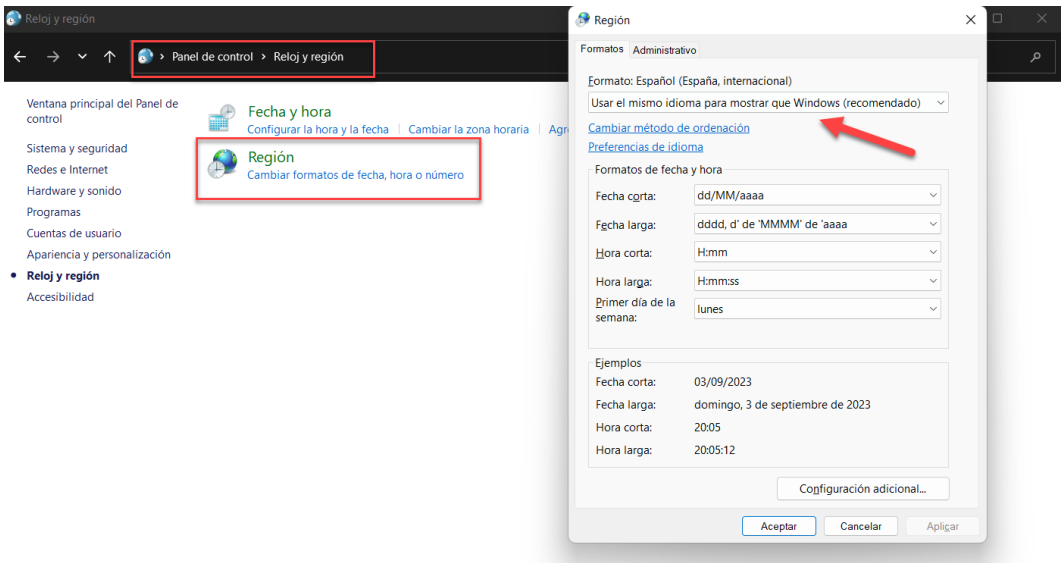


Figura I.11: Guía sobre cómo cambiar el formato de números al inglés

Una vez que está excel con la nomenclatura anglosajona abrimos los datos que estarán guardados con el nombre y en el directorio que se le haya indicado al programa.

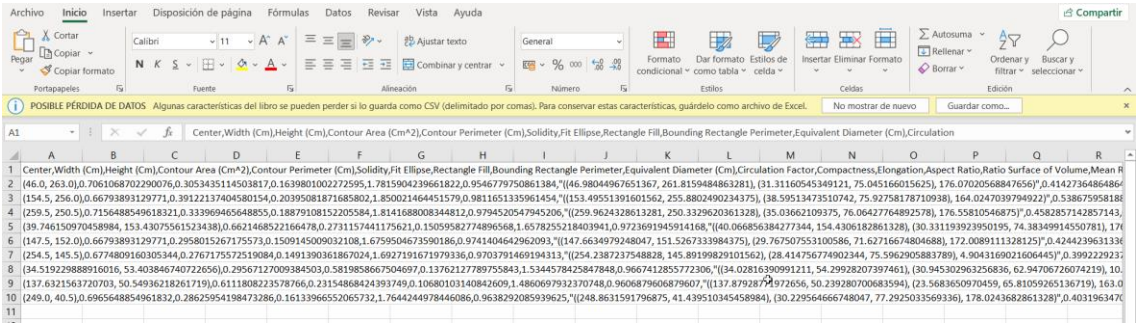


Figura I.12: Datos o resultados mostrados de manera "cruda"

Aparecerá algo parecido a la imagen de arriba, seguramente con la misma advertencia informándonos de que el archivo que hemos abierto es del tipo CSV.

Para que los datos aparezcan con un mejor formato se hará lo siguiente:

1. Abrir la sección **Datos**.
2. Pinchar en la opción **Obtener datos**.
3. Elegir la opción **De un archivo**.
4. Elegir **Desde el texto/CSV**.
5. Seleccionar de nuevo el archivo que deseamos.
6. Le damos a **Cargar**.

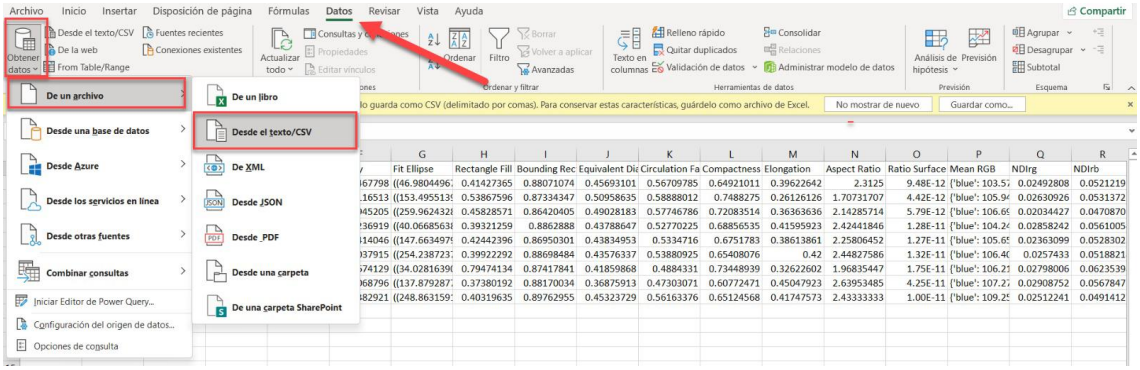
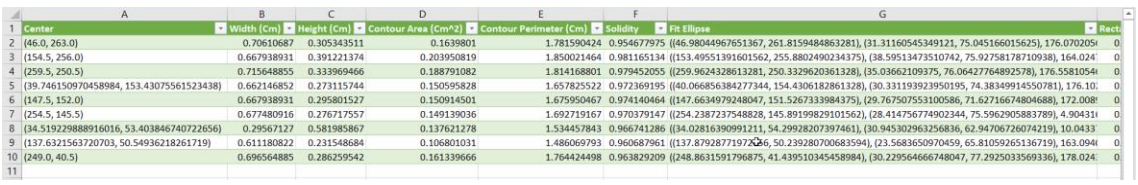


Figura I.13: Guía sobre cómo mostrar los datos de una mejor forma

Una vez hecho todos los pasos, el fichero debería de aparecer así:



Center	Width (Cm)	Height (Cm)	Contour Area (Cm^2)	Contour Perimeter (Cm)	Solidity	Fit Ellipse
(46.0, 263.0)	0.70610687	0.305343511	0.1639801	1.781590424	0.954677975	((46.98044967651367, 261.8159484863281), (31.31160545349121, 75.04516601562...
(154.5, 256.0)	0.667938931	0.391221374	0.203950819	1.850021464	0.981165134	((153.4955139160156, 255.8802490234375), (38.59513473510742, 75.92758178710...
(259.5, 250.5)	0.715648855	0.333969466	0.188791082	1.814168801	0.979452055	((259.9624328613281, 250.3296203613281), (35.03662109375, 76.06427764892578...
(39.746150970458984, 153.43075561523438)	0.662146852	0.273115744	0.150595828	1.657825522	0.972369195	((40.06685638427734, 154.4306182861328), (30.33119392395019...
(147.5, 152.0)	0.667938931	0.295801527	0.150914501	1.675950467	0.974140464	((147.6634979248047, 151.5267333984375), (29.76750755310058...
(254.5, 145.5)	0.677480916	0.276717557	0.149139036	1.692719167	0.970379147	((254.2387237548828, 145.8919982910156), (28.41475677490234...
(34.519229888916016, 53.403846740722656)	0.29567127	0.581985867	0.137621278	1.534457843	0.966741286	((34.02816390991211, 54.29928207397461), (30.94530296325683...
(137.63215637207031, 50.54936218261719)	0.611180822	0.231548684	0.106801031	1.486069793	0.960687961	((137.87928771972656, 50.23928070068359...
(249.0, 40.5)	0.696564885	0.286259542	0.161339666	1.764424498	0.963829209	((248.8631591796875, 41.43951034545898...

Figura I.14: Resultados mostrados con un mejor formato

Algo importante a tener en cuenta es que si a la hora de guardar el fichero se elige un directorio como la nube, ya sea One Drive o Google Drive, es posible que la aplicación de error al guardar por permisos, intentar guardar en directorios locales.

Para finalizar con este manual se volverá a explicar algunos campos que pueden ser confusos:

Fit Ellipse
((46.98044967651367, 261.8159484863281), (31.31160545349121, 75.045166015625), 176.07020568847656)
((153.49551391601562, 255.8802490234375), (38.59513473510742, 75.92758178710938), 164.0247039794922)
((259.9624328613281, 250.3329620361328), (35.03662109375, 76.06427764892578), 176.55810546875)
((40.066856384277344, 154.4306182861328), (30.331193923950195, 74.38349914550781), 176.10260009765625)
((147.6634979248047, 151.5267333984375), (29.767507553100586, 71.62716674804688), 172.0089111328125)
((254.2387237548828, 145.89199829101562), (28.414756774902344, 75.5962905883789), 4.9043169021606445)
((34.02816390991211, 54.29928207397461), (30.945302963256836, 62.94706726074219), 10.043377876281738)
((137.87928771972656, 50.239280700683594), (23.5683650970459, 65.81059265136719), 163.0940704345703)
((248.8631591796875, 41.439510345458984), (30.229564666748047, 77.2925033569336), 178.0243682861328)

Figura I.15: Datos del atributo Fit Ellipse

El campo **Fit Ellipse** a diferencia de la mayoría de campos, está formado una tripleta con la siguiente forma:
((centro de la elipse),(longitudes de los ejes mayor y menor), ángulo de rotación de la elipse)

Mean RGB
{'blue': 103.57931034482758, 'green': 98.08806366047745, 'red': 93.31671087533155}
{'blue': 105.94072164948453, 'green': 100.39733676975945, 'red': 95.25}
{'blue': 106.69569245020843, 'green': 101.1324687355257, 'red': 97.09958314034274}
{'blue': 104.24394463667821, 'green': 98.65167243367935, 'red': 93.16897347174164}
{'blue': 105.6545768566494, 'green': 99.65227403569374, 'red': 95.05123776626368}
{'blue': 106.40837696335078, 'green': 100.98022105875509, 'red': 95.91157649796394}
{'blue': 106.21527777777779, 'green': 99.1439393939394, 'red': 93.74684343434345}
{'blue': 107.27893890675242, 'green': 101.4871382636656, 'red': 95.75}
{'blue': 109.2524219590958, 'green': 104.12109795479009, 'red': 99.01776103336921}

Figura I.16: Datos del atributo Mean RGB

El campo **Mean RGB** devuelve un diccionario con tres valores:

{'blue': (media del color azul), 'green': (media del color verde), 'red': (media del color rojo)}

Mean HSV
{'hue': 105.92519893899204, 'saturation': 25.315119363395223, 'value': 103.57931034482758}
{'hue': 105.35524054982818, 'saturation': 25.67955326460481, 'value': 105.94072164948453}
{'hue': 107.2487262621584, 'saturation': 22.94071329319129, 'value': 106.69569245020843}
{'hue': 105.15801614763552, 'saturation': 27.08823529411765, 'value': 104.24394463667821}
{'hue': 106.88773747841107, 'saturation': 25.568796776050664, 'value': 105.6545768566494}
{'hue': 105.11401977894124, 'saturation': 25.179755671902267, 'value': 106.40837696335078}
{'hue': 106.94381313131314, 'saturation': 29.923611111111114, 'value': 106.21527777777779}
{'hue': 105.16318327974277, 'saturation': 27.356109324758844, 'value': 107.27893890675242}
{'hue': 104.8891280947255, 'saturation': 23.861141011840687, 'value': 109.2524219590958}

Figura I.17: Datos del atributo Mean HSV

El campo **Mean HSV** devuelve un diccionario con otros tres valores:
{'hue': (matiz o tono), 'saturation': (saturación), 'value': (valor)}

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA