



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Digit4Vitis V2
Documentación Técnica**



Presentado por Diego Urbaneja Portal
en Universidad de Burgos — 6 de julio de 2025
Tutores: Carlos Cambra Baseca y Ramón Sánchez
Alonso

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	5
Apéndice B Anexo: Especificación de Requisitos	11
B.1. Introducción	11
B.2. Objetivos generales	11
B.3. Catálogo de requisitos	12
B.4. Especificación de Casos de Uso	15
Apéndice C Anexo: Especificación de Diseño	21
C.1. Introducción	21
C.2. Diseño de datos	21
C.3. Diseño arquitectónico	22
C.4. Guía de estilo	26
Apéndice D Anexo: Documentación Técnica de Programación	31
D.1. Introducción	31
D.2. Estructura de directorios	31
D.3. Manual del programador	33

D.4. Instalación de Herramientas Necesarias	34
D.5. Compilación, instalación y ejecución del proyecto	36
D.6. Pruebas del sistema	37
Apéndice E Anexo: Documentación de Usuario	39
E.1. Introducción	39
E.2. Requisitos de usuario	39
E.3. Instalación y Ejecución	40
E.4. Manual del usuario	43
Apéndice F Anexo de sostenibilización curricular	51
F.1. Introducción	51
F.2. Sostenibilidad Ambiental: Hacia una Agricultura más Eficiente	51
F.3. Sostenibilidad Social: Mejora de la salud pública y costes de cultivo	52
F.4. Sostenibilidad Económica: Optimización y Rentabilidad Agrícola	53
F.5. Contribución personal a la Sostenibilidad	53
Bibliografía	55

Índice de figuras

A.1. Diagrama de Evolución del Proyecto por Fases.	5
B.1. Diagrama de Actividad para el Caso de Uso 1 (CU-1).	17
B.2. Diagrama de Actividad para el Caso de Uso 2 (CU-2).	19
C.1. imagen del color usado para el fondo así como sus valores RGB entre otros	26
C.2. imagen del color primario usado para los elementos de la interfaz así como sus valores RGB entre otros	27
C.3. imagen del color secundario usado para los elementos de la interfaz así como sus valores RGB entre otros	27
C.4. imagen del color usado para el texto así como sus valores RGB entre otros	28
C.5. Logotipo diseñado para la aplicación	29
C.6. Ejemplo de utilización de los emoticonos en los botones	29
D.1. mensaje en caso de que ambas imágenes no sean de la misma hoja. . .	37
D.2. mensaje en caso de que el formato de los archivos subidos no sean los adecuados.	38
E.1. Interfaz principal de la aplicación una vez ejecutada	43
E.2. imagen de la interfaz una vez los archivos se han subido correctamente	44
E.3. Sección de resultados del análisis	45
E.4. Imagen donde se muestra el resultado de la superposición automática .	46
E.5. Comparación de ambas imágenes subidas inicialmente en formato RGB	47
E.6. Comparación de ambas imágenes subidas inicialmente trinarizadas . .	48
E.7. Botones para descargar las trinarizadas de los archivos subidos	48
E.8. Pestaña Acerca del TFG	49

Índice de tablas

A.1. Coste anual estimado del proyecto.	7
A.2. Bibliotecas utilizadas en el proyecto y sus licencias.	9
B.1. CU-1 - Realizar análisis dual con sustracción de ruido	16
B.2. CU-2 - Inspeccionar y descargar resultados	18

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apéndice de la memoria se va a estudiar y discutir la viabilidad del proyecto, tanto en términos económicos como legales así como de la planificación temporal seguida durante el desarrollo del mismo.

A.2. Planificación temporal

Para la gestión del proyecto se adoptó una metodología de desarrollo ágil, estructurada en *sprints*. Este enfoque moderno y flexible permitió un progreso iterativo y transparente, con objetivos claros definidos para cada semana y un seguimiento riguroso del trabajo a través de los *commits* registrados en el repositorio de GitHub del proyecto.

La elección de esta metodología, en lugar de un modelo tradicional en cascada, ha sido estratégica para poder adaptarse con agilidad a los desafíos técnicos y refinar los requisitos de forma continua. Esta flexibilidad resultó crucial para cumplir con el hito principal del proyecto: la entrega de un Producto Mínimo Viable (MVP) robusto y funcional a principios de mayo, que sirviera como base para las fases de validación y refinamiento posteriores.

Fase de Desarrollo General y MVP (Sprints 1-8)

Sprint 1: Kick-off y estructura del proyecto (7 – 13 de marzo)

El proyecto se inauguró con el *kick-off* técnico. El objetivo fundamental fue establecer una arquitectura de software sólida y escalable. Se crearon los direc-

torios clave (`src`, `recursos`, `src/funciones`) y los ficheros iniciales (`main.py`, `requirements.txt`, `.gitignore`) partiendo del proyecto ya existente como base. Durante este *sprint* se definió un esqueleto de aplicación sobre el que construir las futuras funcionalidades, asegurando desde el principio un entorno de trabajo limpio y organizado.

Sprint 2: Refactorización de la arquitectura (14 – 20 de marzo)

Tras una revisión inicial, se detectó que la lógica de la aplicación tenía tendencia a concentrarse en un único fichero. En este sprint se realizó una refactorización profunda para aplicar el principio de Separación de Intereses (*Separation of Concerns*). Se crearon módulos específicos para la interfaz de usuario (`interfaz.py`), el procesamiento de imágenes (`procesamiento.py`) y la gestión de archivos (`archivos.py`). Esta reorganización, fue un paso crítico para garantizar la mantenibilidad y escalabilidad del código a largo plazo.

Sprint 3: Consolidación de funcionalidades y primer prototipo (21 – 27 de marzo)

El objetivo fue implementar el flujo de datos de extremo a extremo y obtener el primer prototipo funcional. Se desarrolló la lógica para la carga de ficheros hiperspectrales (`.bil` y `.hdr`), su interpretación mediante la biblioteca *Spectral* y su conversión a un hipercubo de *NumPy*.

Sprint 4: Nuevo método de detección (28 de marzo – 3 de abril)

El algoritmo de umbral inicial demostró ser insuficiente para la detección precisa del cobre. Este sprint se enfocó en la investigación de las firmas espectrales del compuesto. Se analizaron los histogramas de reflectancia de múltiples bandas. Se implementó un nuevo método de detección basado en umbrales compuestos sobre la banda 164 (728.24 nm), lo que permitió discriminar con mayor fiabilidad los píxeles con tratamiento de los que no lo tenían.

Sprint 5: Mejora de fidelidad y simplificación de la interfaz (4 – 10 de abril)

Se observó que los filtros de suavizado genéricos, aunque reducían el ruido, también eliminaban detalles finos de las gotas. Se tomó la decisión técnica de eliminar toda supresión de ruido previa para trabajar con los datos en crudo. En paralelo, se simplificó la interfaz de usuario, eliminando controles superfluos y centrando el flujo de trabajo en la secuencia Cargar -> Procesar -> Analizar.

Sprint 6: Mejoras visuales y Prototipo en vídeo (11 – 17 de abril)

Este sprint se dividió en dos objetivos: mejorar la experiencia de usuario (UX) y explorar el análisis en tiempo real. Se implementó una hoja de estilos (CSS) para dotar a la aplicación de una estética profesional y una paleta de colores corporativa. El mayor desafío técnico fue la creación de la rama *video*, donde se adaptó el pipeline de procesamiento para que pudiera ejecutarse sobre fotogramas de un *stream* de vídeo, optimizando el rendimiento para lograr una salida estable y demostrando la viabilidad de la tecnología para aplicaciones de campo.

Sprint 7: Branding y DevOps (18 – 24 de abril)

Con el prototipo madurando, el enfoque se desplazó hacia las buenas prácticas de *DevOps* y la profesionalización del producto. Se configuró un *Dockerfile* detallado para crear un entorno de ejecución contenido, aislado y 100 % reproducible, eliminando problemas de dependencias. Se añadieron los logos de las entidades colaboradoras y se pulieron elementos de la interfaz, asegurando una presentación coherente y profesional.

Sprint 8: Preparación para versión demostrable (MVP) (25 de abril – 1 de mayo)

El objetivo de este sprint fue empaquetar una versión estable y fácilmente ejecutable del software, constituyendo el Producto Mínimo Viable (MVP). Se creó el script `lanzar_app.bat` para simplificar la ejecución en Windows. Se crearon las ramas `demo` y `video` como instantáneas congeladas y estables del código, obteniendo así versiones fiables listas para demostración.

Implementación de Sustracción de Ruido (Sprints 9-12)**Sprint 9: Ajustes Post-MVP e inicio de funcionalidades avanzadas (2 – 8 de mayo)**

Tras validar el MVP, se inició una nueva fase de desarrollo enfocada en el núcleo innovador del proyecto: la eliminación de ruido por sustracción. Esto marcó un cambio de paradigma: en lugar de filtrar el ruido, el nuevo enfoque consistiría en detectar todos los posibles artefactos en una hoja de control para luego sustraerlos de la hoja tratada. Implementando así un método diferencial mucho más robusto y científicamente riguroso.

Sprint 10: Mejora del Ajuste Automático (9 – 15 de mayo)

Después de realizar una investigación sobre posibles métodos para aplicar el algoritmo de sustracción de ruido, se implementó el pipeline completo de alineación automática de imágenes: detección de bordes con Canny, extracción de puntos clave con ORB, emparejamiento por fuerza bruta (*BFMatcher*) y estimación robusta de la transformación afín con RANSAC para descartar correspondencias erróneas.

Sprint 11: Sprint de estabilización (16 – 22 de mayo)

Tras la implementación de una funcionalidad tan compleja como la alineación automática, este *sprint* se dedicó por completo a la estabilización. No se añadieron nuevas características; en su lugar, se realizaron pruebas exhaustivas, se corrigieron errores (*bugs*) y se optimizó el rendimiento del nuevo algoritmo para asegurar su robustez y precisión en diferentes escenarios.

Sprint 12: Pruebas y validación (23 de mayo – 29 de mayo)

Durante este *sprint* se realizaron diversas pruebas y validaciones para verificar el correcto desempeño de la aplicación así como la validación de todas las funcionalidades. Asegurando así un correcto desempeño por parte de todos los componentes que conforman la aplicación. Una vez superadas estas pruebas se dió por finalizado el desarrollo de las funcionalidades de la aplicación, Dando comienzo así a la fase de documentación.

Refinamiento y Documentación final (Sprints 13+)**Sprints 13, 14 y 15: Inicio de documentación (30 de mayo – 19 de junio)**

Durante estos *sprints* se realizaron los primeros borradores de la documentación así como su revisión por parte de los tutores. Durante este periodo de tiempo no se realizó ninguna modificación al código de la aplicación.

Sprint 16 : Modificación de la interfaz y diseño del logotipo (20 de junio - 26 de junio)

Durante este *sprint* se implementó una interfaz de usuario (UI) totalmente nueva dando así un nuevo *look and feel* a la aplicación, modificando esquema de colores y disposición de los elementos de la interfaz. Además se diseñó un logotipo para la aplicación con el objetivo de darle un aspecto más profesional y terminada a la aplicación.

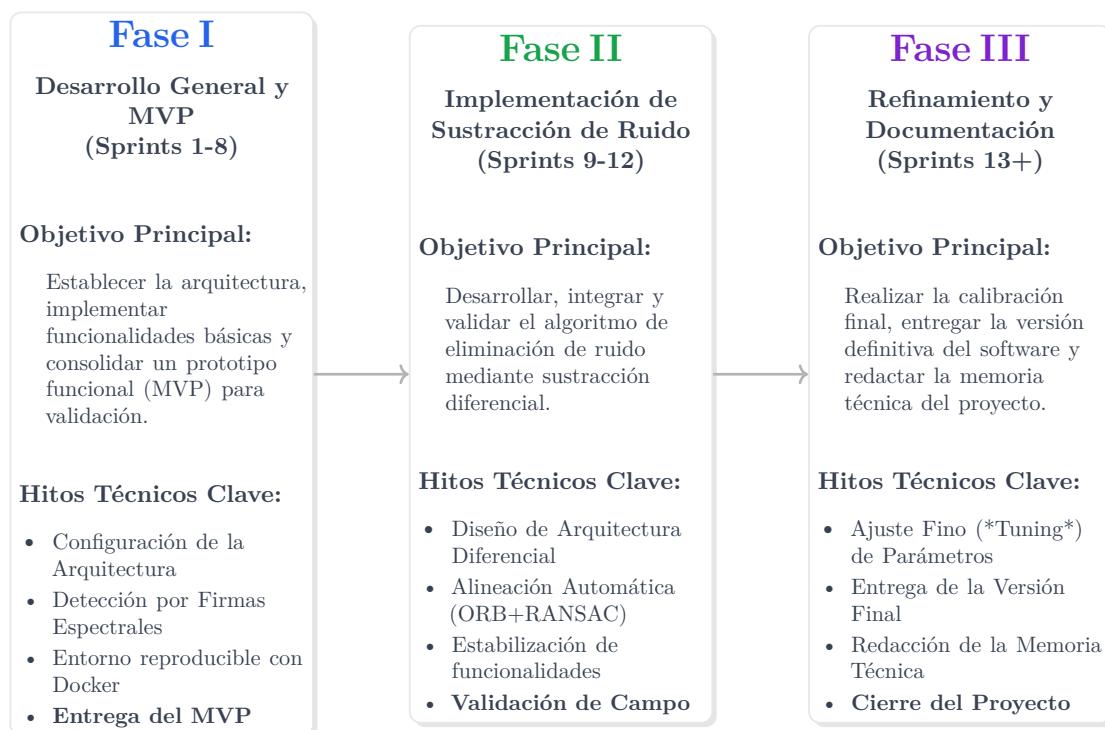


Figura A.1: Diagrama de Evolución del Proyecto por Fases.

A.3. Estudio de viabilidad

En este apartado se va a realizar un análisis de la viabilidad del proyecto. Tanto desde la perspectiva económica como desde la legal para determinar la factibilidad y sostenibilidad del mismo .

Viabilidad económica

Para determinar la viabilidad económica del proyecto, se ha de realizar una estimación de los costes asociados a su desarrollo. Es importante remarcar el hecho de que este estudio de viabilidad es una simulación de un entorno profesional donde es necesario reconocer y cuantificar los costes asociados proyecto para poder estudiar la viabilidad del desarrollo del mismo. Al tratarse de un Trabajo de Fin de Grado no se persigue ningún beneficio económico directo. Sin embargo uno de sus objetivos es la adquisición de conocimiento y la implementación de una herramienta funcional.

Coste de Hardware

A continuación, se detalla el equipamiento y se calcula su coste anual mediante una amortización estándar de 5 años.

- **Ordenador portátil (LG Gram):** 1.300 €
- **PC de sobremesa (Torre, 2 monitores y periféricos):** 1.700 €

El coste total del hardware asciende a 3.000 €. Aplicando una amortización a 5 años, el coste anual del hardware para el proyecto sería de:

$$\frac{3,000 \text{ €}}{5 \text{ años}} = 600 \text{ €/año}$$

Coste de Software

El software utilizado en el proyecto ha sido, en su mayoría, de código abierto o con licencias gratuitas para uso académico y personal.

- **Sistemas Operativos:** Windows 10 (portátil) y Windows 11 (PC), preinstalados en los equipos (licencia OEM), por lo que su coste es marginal para el proyecto teniendo un impacto de 0 €.
- **Entorno de Desarrollo (IDE):** Para la realización del proyecto se empleó Visual Studio Code (en su versión *Insiders*), de uso gratuito.
- **Editor de documentación:** Para la redacción de la documentación se utilizó Overleaf, utilizando su plan gratuito.
- **Bibliotecas de Python:** Todas las bibliotecas empleadas (Numpy, OpenCV, Streamlit, etc.) son de código abierto y gratuitas.

Teniendo esto en cuenta se puede estimar que dado que no se ha requerido la adquisición de licencias de software de pago, el coste total en este apartado es de **0 €**.

Coste de Personal

Para este cálculo, se simula la contratación de un programador con perfil *Junior* a jornada parcial (50 % del tiempo) en una empresa para el desarrollo del proyecto.

- **Salario mensual (bruto):** 1.800 € (12 pagas)

- **Dedicación al proyecto:** 50 % (media jornada)

El coste mensual del personal dedicado al proyecto a media jornada sería:

$$1,800 \text{ €/mes} \times 0,50 = 900 \text{ €/mes}$$

El coste anual del sueldo del programador contratado sería de:

$$900 \text{ €/mes} \times 12 \text{ meses} = 10,800 \text{ €/año}$$

Otros Costes

No se han identificado otros costes relevantes para el desarrollo del proyecto. Los gastos ocasionados por la electricidad o la conexión a internet necesarios para llevar a cabo el proyecto se considera que no tienen un impacto directo del proyecto por estar asociados al entorno de trabajo.

Coste Total y Conclusión

Sumando todos los costes anuales del proyecto, obtenemos el coste total teórico del proyecto en un año.

Recurso	Coste Anual
Coste de Hardware	600,00 €
Coste de Software	0,00 €
Coste de Personal	10.800,00 €
Otros Costes	0,00 €
Total	11.400,00 €

Tabla A.1: Coste anual estimado del proyecto.

Observando este coste desde una perspectiva meramente académica este coste de desarrollo del proyecto es demasiado elevado si lo que se busca es la rentabilidad ya que estos costes no se recuperarían en ningún momento. Esto debido a que no se generan ingresos directos por la realización del proyecto. Desde un punto de vista empresarial, solo sería viable el proyecto en el caso de que el producto desarrollado sea lo suficientemente completo como para generar interés por parte de potenciales clientes que quisieran adquirir el producto bien como producto consumible final o como patente para aplicar en sus propios proyectos.

Sin embargo la viabilidad del proyecto no radica únicamente en los posibles beneficios económicos asociados al mismo. También hay que considerar los intangibles que produce el desarrollo de un producto de estas características. Se

trata de la adquisición de competencias técnicas y personales, la creación de una herramienta funcional para el proyecto Dig4Vitis y la contribución a la investigación en agricultura de precisión.

Viabilidad legal

En este apartado se van a analizar las licencias de las herramientas y bibliotecas de software utilizadas para asegurar la conformidad legal del proyecto. Además, se propondrá una licencia para el código fuente generado.

Licencias del Software y Bibliotecas Utilizadas

El proyecto se fundamenta en el uso de Python y un conjunto de bibliotecas de código abierto. A continuación, se describen las licencias más relevantes empleadas:

- **Licencia MIT:** Es una de las licencias de software libre más permisivas. Permite la reutilización, modificación y distribución del software para cualquier fin (incluso comercial), exigiendo únicamente que se mantenga el aviso de derechos de autor y la licencia original en las copias del software.
- **Apache License 2.0:** Es otra licencia permisiva que permite el uso, modificación y distribución libremente. Requiere que se conserven los avisos de derechos de autor y patentes. Una característica importante es que otorga una licencia de patente explícita de los contribuidores a los usuarios.
- **BSD 3-Clause License:** Similar a la licencia MIT, es muy permisiva pero incluye una cláusula que prohíbe el uso del nombre de los autores o contribuidores para promocionar productos derivados sin permiso explícito.

La siguiente tabla resume las principales bibliotecas utilizadas y sus respectivas licencias.

El uso de estas bibliotecas es totalmente compatible con el desarrollo de un proyecto académico y no impone restricciones que limiten su uso o distribución.

Licencia del Proyecto

Dada la naturaleza académica del proyecto así como el análisis de las distintas licencias y restricciones de las bibliotecas utilizadas durante el desarrollo del mismo, se ha llegado a la conclusión de que la licencia que más se adecúa al entorno de desarrollo y condiciones presentadas para este proyecto es la **Licencia MIT**.

Biblioteca/Herramienta	Licencia
Python	Python Software Foundation License (Compatible con GPL)
Numpy	BSD 3-Clause License
OpenCV-Python	Apache License 2.0
Pandas	BSD 3-Clause License
Streamlit	Apache License 2.0
Spectral Python (Spy)	Licencia MIT
Scikit-Image (skimage)	BSD 3-Clause License
Pillow	Historical Permission Notice and Disclaimer (HPND)

Tabla A.2: Bibliotecas utilizadas en el proyecto y sus licencias.

La decisión de esta licencia sobre las demás se fundamenta en su simplicidad y en su carácter permisivo, que facilita la colaboración, la reutilización del código en futuras investigaciones y la posible integración con otros proyectos sin imponer complejas restricciones legales.

La adopción de esta licencia MIT permite que cualquier persona pueda realizar cualquier modificación o mejora sobre el proyecto ya desarrollado. Teniendo en cuenta que el requisito impuesto por esta licencia es dar crédito al autor original del proyecto. De esta manera se consigue fomentar el espíritu de colaboración de la comunidad de software de código abierto.

Apéndice B

Anexo: Especificación de Requisitos

B.1. Introducción

En este apartado se detallan los requisitos funcionales y no funcionales de la aplicación de software **EcoVid**. El objetivo es proporcionar una descripción clara y sin ambigüedades de las funcionalidades del sistema en su versión final. Este documento sirve como base para el desarrollo, las pruebas y la validación de la aplicación.

Se ha dividido en tres partes principales:

- **Objetivos generales:** Descripción de las metas del proyecto.
- **Catálogo de requisitos:** Listado detallado de los requisitos funcionales (RF) y no funcionales (RNF) que definen el comportamiento y las características del sistema.
- **Especificación de Casos de Uso:** Descripción detallada de las interacciones clave del usuario con la aplicación para cumplir los requisitos funcionales.

B.2. Objetivos generales

Los objetivos principales de la aplicación desarrollada son los siguientes:

- **Automatizar el análisis comparativo:** Proporcionar una herramienta que automatice el proceso de detección de producto fungicida sobre hojas de vid, utilizando un enfoque diferencial que compara una muestra tratada con una muestra de control.

- **Mejorar la precisión mediante sustracción de ruido:** Implementar un algoritmo robusto para la eliminación de falsos positivos. Este sistema se basa en la alineación geométrica precisa de la imagen tratada con su control para sustraer las variaciones de reflectancia naturales de la propia hoja (nervios, brillos), aislando únicamente el producto real.
- **Facilitar la usabilidad:** Ofrecer una interfaz de usuario web, intuitiva y sencilla, que guíe al usuario a través del proceso de carga y análisis dual sin requerir conocimientos técnicos avanzados en visión por computador.
- **Visualizar y exportar resultados para validación:** Ofrecer una visualización clara del resultado final (porcentaje y mapa de recubrimiento), junto con vistas intermedias que permitan al investigador validar la calidad del proceso (ej. la precisión del alineamiento), y permitir la descarga de dichas imágenes.

B.3. Catálogo de requisitos

Requisitos Funcionales (RF)

- RF-01:** El sistema debe permitir al usuario cargar dos pares de imágenes hiperespectrales en formato ENVI (.bil + .hdr): uno para la imagen de referencia **SIN tratamiento** y otro para la imagen **CON tratamiento**.
- RF-02:** El sistema debe validar que para cada muestra (SIN y CON) se hayan subido ambos ficheros (.bil y .hdr) antes de poder iniciar el procesamiento.
- RF-03:** Al pulsar el botón "Iniciar Procesamiento", el sistema debe ejecutar de forma automática un pipeline de análisis dual completo.
- RF-04:** El sistema debe generar una imagen de resultado final que visualice la hoja (área común) y las detecciones de producto ya depuradas (sin ruido).
- RF-05:** El sistema debe calcular y mostrar de forma prominente el **porcentaje de recubrimiento** final, definido como el área de producto real detectado respecto al área total de la hoja común.
- RF-06:** El sistema debe alinear geométricamente la imagen SIN tratamiento con la imagen CON tratamiento para corregir desplazamientos, rotaciones y ligeros cambios de escala.
- RF-07:** El proceso de alineación debe basarse en la detección de los contornos del **limbo foliar** (previa eliminación morfológica del pecíolo), el emparejamiento

de puntos característicos mediante el algoritmo **ORB**, y una estimación robusta de la transformación afín con **RANSAC**.

- RF-08:** El sistema debe ser capaz de identificar el **área de la hoja común** (la intersección de ambas máscaras de hoja una vez alineadas) para asegurar que el análisis comparativo se realiza sobre una región de interés idéntica.
- RF-09:** El sistema debe implementar una lógica de **sustracción de ruido**: un píxel se considerará "producto final" si, y solo si, es detectado como producto en la imagen CON y NO es detectado como tal en la imagen SIN alineada.
- RF-10:** El sistema debe ofrecer una opción principal para descargar la imagen del **resultado final** en formato PNG.
- RF-11:** El sistema debe disponer de una sección expandible ("Ver detalles y descargas adicionales") para mostrar visualizaciones de diagnóstico y validación.
- RF-12:** Dentro de la sección de detalles, el sistema debe mostrar una imagen de **superposición de alineamiento** (p.ej., máscara CON en verde y SIN alineada en rojo) para permitir al usuario inspeccionar visualmente la calidad de la alineación.
- RF-13:** Dentro de la sección de detalles, el sistema debe mostrar las imágenes de **segmentación base** (trinarizadas SIN y CON, antes del alineamiento y sustracción) y sus correspondientes vistas RGB.
- RF-14:** El sistema debe permitir la descarga individual de las **imágenes trinarizadas base** (CON y SIN) en formato PNG desde la sección de detalles.
- RF-15:** El sistema debe incluir una pestaña informativa (Acerca del TFG) que describa el contexto, los objetivos y la metodología del proyecto, proporcionando transparencia sobre su funcionamiento.
- RF-16:** El pipeline de procesamiento debe ser capaz de manejar imágenes de entrada con ligeras diferencias de dimensiones, creando un lienzo común para evitar recortes antes del alineamiento.
- RF-17:** El sistema debe aplicar un filtro de post-procesado sobre la máscara de producto final para eliminar artefactos o detecciones de tamaño insignificante (inferior a un umbral de píxeles predefinido).

Requisitos No Funcionales (RNF)

- RNF-01:** La aplicación debe ser accesible a través de un navegador web estándar (Chrome, Firefox, Edge).
- RNF-02:** La interfaz de usuario debe ser intuitiva, con un flujo de trabajo claro y guiado (Carga -> Proceso -> Resultados).
- RNF-03:** La aplicación debe presentar un diseño visual coherente y profesional, definido en un fichero CSS externo, utilizando una paleta de colores oscura para mejorar la legibilidad y reducir la fatiga visual.
- RNF-04:** El sistema debe proporcionar retroalimentación visual al usuario (spinner de *Analizando imágenes...*) durante las operaciones que consuman un tiempo considerable.
- RNF-05:** El sistema debe gestionar los ficheros subidos en una carpeta temporal ('archivos_subidos') y garantizar su eliminación automática al finalizar la sesión de la aplicación para no consumir espacio en disco de forma innecesaria.
- RNF-06:** La aplicación debe estar desarrollada íntegramente en lenguaje Python, utilizando bibliotecas de código abierto como Streamlit, OpenCV, Spectral, NumPy y Scikit-image.
- RNF-07:** La estructura del código del proyecto debe ser modular, separando la lógica de la interfaz ('interfaz.py'), el procesamiento ('procesamiento.py'), el alineamiento ('alignment.py') y la gestión de archivos ('archivos.py').
- RNF-08:** El sistema debe ser robusto frente a errores de carga de ficheros, mostrando mensajes de error claros al usuario si no se proporcionan los pares de ficheros (.bil y .hdr) requeridos.
- RNF-09:** El estado de la aplicación debe ser gestionado de forma centralizada a través del mecanismo de estado de sesión ('st.session_state') de Streamlit, para preservar los datos procesados entre interacciones del usuario y evitar recálculos innecesarios.
- RNF-10:** La aplicación debe ser distribuible como una imagen de Docker, garantizando la reproducibilidad del entorno de ejecución y la encapsulación de todas las dependencias.
- RNF-11:** El código fuente del proyecto debe estar gestionado bajo un sistema de control de versiones (Git) y disponible en un repositorio público para fomentar la transparencia y la colaboración.

RNF-12: La tipografía principal de la aplicación será 'Segoe UI' o una fuente sans-serif estándar para garantizar la legibilidad, tal y como se define en la hoja de estilos.

B.4. Especificación de Casos de Uso

A continuación se detallan los casos de uso que describen la funcionalidad de la aplicación EcoVid.

ID del Caso de Uso CU-1	
Nombre	Realizar análisis dual de imágenes hiperespectrales con sustracción de ruido
Actor Principal	Usuario (Investigador/Técnico)
Requisitos Asociados	RF: 01-09, 16, 17 RNF: 01, 02, 04, 08, 09
Descripción	El usuario carga los pares de imágenes de una hoja SIN y CON tratamiento. Al pulsar un botón, el sistema ejecuta un proceso automático que alinea ambas imágenes, sustrae el ruido, calcula el porcentaje de recubrimiento real del producto y muestra los resultados.
Precondición	El usuario ha accedido a la aplicación y tiene en su equipo local los cuatro ficheros hiperespectrales necesarios (par .bil y .bil.hdr para la muestra SIN y par .bil y .bil.hdr para la CON).
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario accede a la pestaña <i>Cargar y Procesar Imágenes</i>. 2. Carga los ficheros (.bil y .hdr) de la hoja SIN tratamiento. 3. Carga los ficheros (.bil y .hdr) de la hoja CON tratamiento. 4. Pulsa el botón <i>Iniciar Procesamiento</i>. 5. El sistema valida los archivos y muestra un indicador de progreso (cubre RF-02, RNF-04). 6. Se ejecuta el pipeline de análisis completo, que incluye el alineamiento con ORB/RANSAC, la identificación del área común y la sustracción del ruido (cubre RF-03, RF-06, RF-07, RF-08, RF-09). 7. Se calculan y muestran los resultados finales: imagen y porcentaje de recubrimiento (cubre RF-04, RF-05).
Postcondición	Los resultados del análisis son visibles en la interfaz, con los datos almacenados en la sesión para su posterior inspección y descarga.
Excepciones	E-1: Si faltan archivos, el sistema muestra un error y detiene el proceso (cubre RNF-08).
Importancia	Muy Alta
Tabla B.1: CU-1 - Realizar análisis dual con sustracción de ruido	

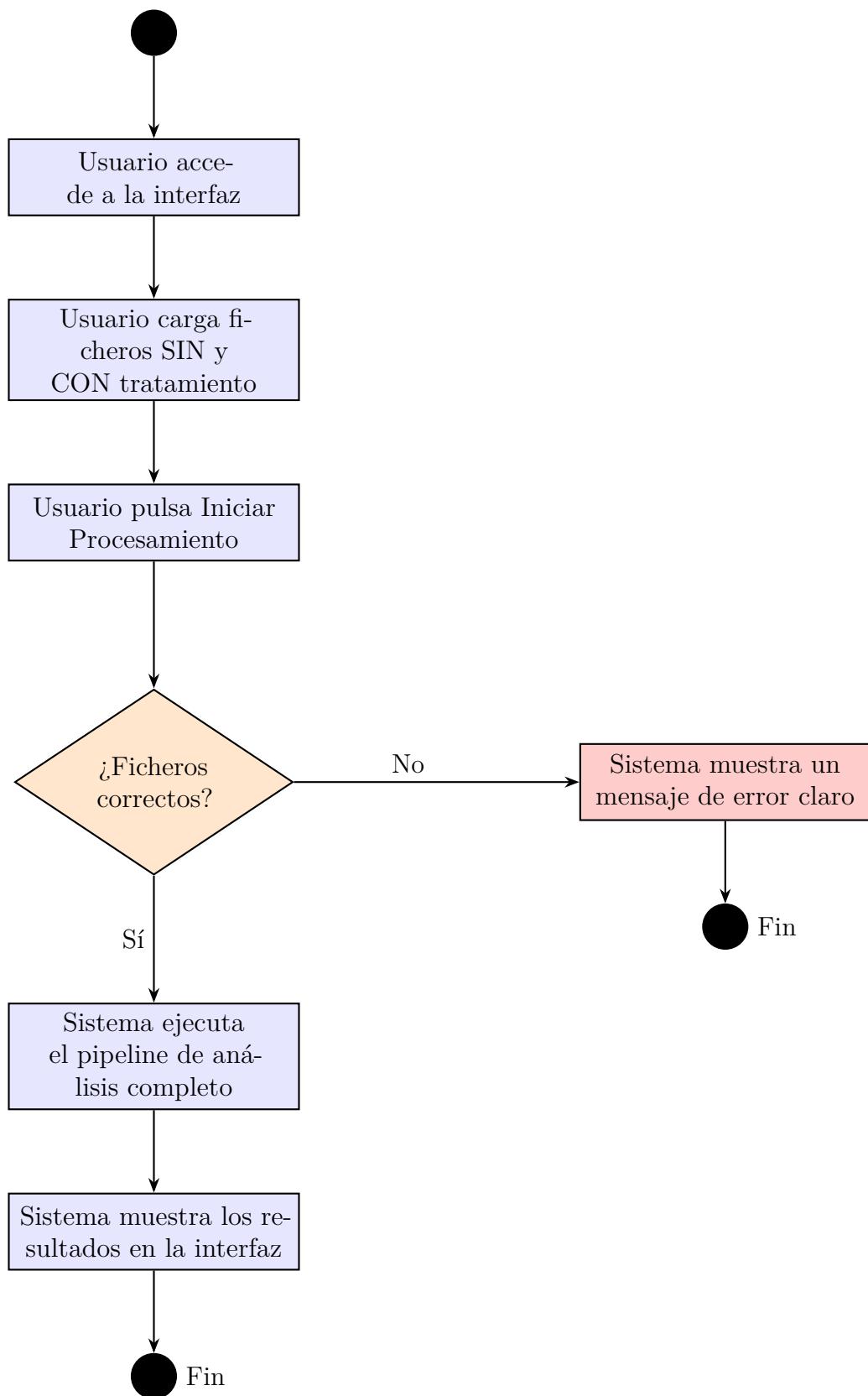


Figura B.1: Diagrama de Actividad para el Caso de Uso 1 (CU-1).

ID del Caso de Uso CU-2	
Uso	
Nombre	Inspeccionar y descargar resultados del análisis
Actor Principal	Usuario (Investigador/Técnico)
Requisitos Asociados	RF: 10, 11, 12, 13, 14
Descripción	Una vez completado el análisis, el usuario puede descargar el resultado principal. Además, puede acceder a una sección de detalles para validar la calidad del proceso y descargar imágenes intermedias.
Precondición	El Caso de Uso 1 (CU-1) se ha completado con éxito.
Flujo Principal	<ol style="list-style-type: none"> 1. El usuario observa el resultado principal (imagen y porcentaje). 2. Pulsa "Descargar Imagen de Resultado" para guardar la imagen final (cubre RF-10). 3. (Opcional) Expande la sección "Ver detalles y descargas adicionales" (cubre RF-11). 4. Dentro de la sección, puede inspeccionar la superposición del alineamiento (cubre RF-12) y visualizar/descargar las imágenes RGB y trinarizadas base (cubre RF-13, RF-14).
Postcondición	El usuario ha guardado en su equipo local las imágenes que necesita para su informe o análisis posterior.
Excepciones	Ninguna.
Importancia	Alta

Tabla B.2: CU-2 - Inspeccionar y descargar resultados

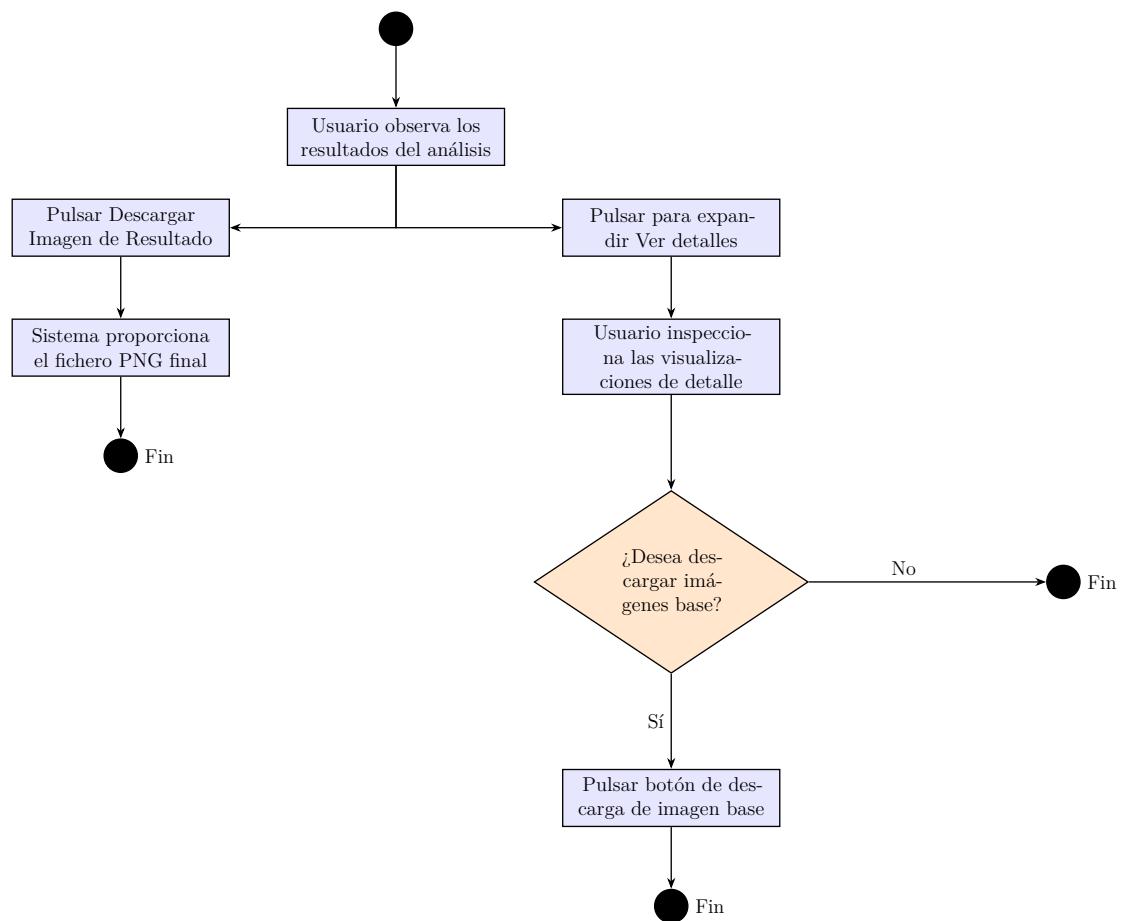


Figura B.2: Diagrama de Actividad para el Caso de Uso 2 (CU-2).

Apéndice C

Anexo: Especificación de Diseño

C.1. Introducción

En este apartado se explicará el diseño de los datos y la arquitectura de la aplicación, detallando cómo se estructuran y comunican los diferentes componentes del software desarrollado para el proyecto.

C.2. Diseño de datos

En la aplicación no se han usado ninguna base de datos. La aplicación trabaja con imágenes hiperespectrales y los datos incluidos en ellas. El manejo de estos datos se realiza de manera temporal mediante los estados de sesión de Streamlit (`st.session_state`) asociados a la sesión de usuario de tal manera que una vez finaliza la sesión y el usuario cierra la aplicación, los datos empleados se eliminan. De esta manera se consigue una experiencia de usuario fluida con tiempos de carga intermedios (una vez subidas las imágenes por primera vez) reducidos y un uso fluido de la aplicación.

Los estados de sesión clave que utiliza la aplicación son:

- **processed**: Variable de tipo Booleano (verdadero o falso) que se activa ('True' o Verdadero) una vez que el usuario ha subido las imágenes y pulsa el botón para iniciar el procesado de las imágenes. De esta manera se controla las previsualizaciones, haciendo que los pasos siguientes solo se muestren al usuario en el caso de que haya datos cargados.
- **cube_sin** y **cube_con**: Su objetivo es almacenar los objetos de datos hiperespectrales, denominados hipercubos, obtenidos a partir de las imágenes

en formato ENVI proporcionadas por el usuario para las muestras CON gotas y SIN gotas respectivamente. Estos datos son la base principal para todo el procesamiento posterior y detección del producto sobre las hojas.

- **rgb_sin** y **rgb_con**: Se encargan de almacenar las imágenes en formato RGB obtenidas a partir de los hipercubos proporcionados por el usuario. Permitiendo la visualización de las mismas en la aplicación.
- **trin_sin** y **trin_con**: Almacenan respectivamente los resultados de segmentación básica o la trinarización para ambas imágenes. Separando el fondo de la hoja y la hoja del producto anterior a la aplicación de reducción de ruido y alineamiento automático de ambas imágenes.

C.3. Diseño arquitectónico

En cuanto a la estructura de la aplicación web se puede describir mediante un patrón **Frontend-Backend**. De esta manera quedan separadas claramente y diferenciadas la interfaz de usuario (**frontend**) de la lógica de procesamiento de los datos (**backend**). Si bien es cierto que este patrón comparte en gran medida características propias de un patrón **Modelo-Vista-Controlador**, la división en dos capas proporcionada por el modelo empleado es más directa en el caso de una aplicación web desarrollada mediante Streamlit.

Frontend (Capa de Presentación y Control)

La capa de interfaz de usuario ha sido construida al completo sobre la biblioteca Streamlit facilitando de esta manera su implementación y su integración en la aplicación web desarrollada. Se define en los ficheros `src/main.py` y `src/funciones/interfaz.py`. Las principales responsabilidades asociadas a esta capa son:

- **Vista**: Encargada de la renderización de todos los componentes visuales de la aplicación como son los títulos, las distintas columnas, las imágenes y logotipos de la aplicación, los cargadores de archivos (`st.file_uploader`), los botones (`st.button`) y las imágenes de resultados (`st.image`).
- **Controlador**: su objetivo es la gestión los eventos producidos por el usuario al interactuar con la aplicación. En cuanto a su lógica se basa en la captura de los eventos producidos al pulsar los botones (p. ej., `if st.button(...):`). Dicha lógica es condicional y actúa como controlador capturando esos eventos, y llamando a las funciones correspondientes de la capa del Backend para realizar las acciones o análisis pertinentes.

Backend (Capa de Lógica y Datos)

Esta capa agrupa toda la funcionalidad de negocio y las acciones referentes al análisis y procesamiento de las imágenes. Dentro de esta capa cada una de las funcionalidades implementadas está en su módulo correspondiente facilitando la legibilidad del código así como la reutilización de dichos módulos de una manera más sencilla, haciendo además que las modificaciones a realizar sean más sencillas de implementar previniendo posibles errores o bugs o ayudando a su detección. Dichos módulos se encuentran dentro de la carpeta `src/funciones/`

- **Gestión de Archivos** (`archivos.py`): Este módulo contiene aquellas funciones referentes a la carga y almacenamiento temporal de los archivos subidos por el usuario. También se encarga de limpiar el disco una vez la sesión termina para evitar problemas con el almacenamiento debido al gran tamaño de las imágenes con las que se trabaja.
- **Procesamiento de Imágenes** (`procesamiento.py`): Este módulo incluye todas las funciones e implementaciones referentes al análisis de las imágenes. Es el núcleo de la aplicación donde se agrupan la mayoría de las funciones más relevantes para el desarrollo del proyecto. Incluye funciones para:
 - Obtener las máscaras de hoja y producto a partir de bandas espectrales específicas definidas en el código (`_obtener_mascaras`).
 - Generar las imágenes trinarizadas separando fondo de la hoja y producto de la hoja (`_trinarizar`, `trinarizar_final`).
 - Se encarga también de gestionar el flujo completo de análisis dual, que incluye el alineamiento y la sustracción de ruido (`aplicar_procesamiento_dual`).
- **Alineamiento de Imágenes** (`alignment.py`): Este módulo contiene todos los aspectos referentes a la lógica de implementación y aplicación del algoritmo ORB para alinear las dos imágenes (CON y SIN gotas). Dentro de sus funciones destacan principalmente la eliminación del peciolo (aspecto irrelevante de la hoja para el análisis que causaba problemas a la hora de aplicar la alineación y superposición) aislando el limbo de la hoja (`_remove_petiole`). También contiene la lógica referente al cálculo de la matriz de transformación afín usando ORB y RANSAC para obtener una alineación y superposición automática precisa.

Esta separación clara facilita el mantenimiento, la reutilización del código y la posibilidad de cambiar el *frontend* en el futuro sin alterar la lógica de procesamiento del *backend*.

Patrones de Diseño y Principios Arquitectónicos

Además de la separación *Frontend-Backend*, la arquitectura del proyecto se beneficia de la aplicación de varios patrones y principios de diseño que mejoran su calidad, facilitando su mantenimiento, reutilización y futura escalabilidad de la aplicación.

Patrón Módulo (Module Pattern)

La clara organización del proyecto en módulos diferenciados donde cada fichero .py dentro de `src/funciones/` agrupa un conjunto de responsabilidades cohesivas.

- `interfaz.py`: Define la interfaz de usuario.
- `procesamiento.py`: Contiene la lógica principal del análisis hiperespectral.
- `alignment.py`: Encapsula los algoritmos específicos para el alineamiento de imágenes.
- `archivos.py`: Gestiona la lectura y escritura de ficheros.

Este patrón es fundamental para la separación de intereses (*Separation of Concerns*). Facilitando también la legibilidad y comprensión del código de cara a futuras mejoras o modificaciones por parte de otros programadores.

- **Impacto en el Mantenimiento:** Esta división en módulos tiene como consecuencia una fácil depuración del código en caso de errores o bugs. Por ejemplo en el caso de que la alineación de imágenes no funcione como se espera, el programador sabe que en una gran probabilidad el problema se encuentre en el módulo referente a la alineación `alignment.py`. Ahormando de esta manera tiempo y esfuerzo al no tener que revisar el código completo para encontrar el problema, sobre todo en casos donde el programador no es el escritor original del código.
- **Impacto en la Escalabilidad:** De igual manera este enfoque modular permite que a la hora de añadir nuevas funcionalidades estas se puedan desarrollar e implementar de manera aislada sin impactar o modificar código ya existente referente a otras funciones evitando crear conflictos indeseados. En el supuesto caso en el que en el futuro se quiera implementar una nueva funcionalidad, ésta se puede implementar de manera individual en un nuevo módulo sin modificar o afectar a las funcionalidades ya desarrolladas, solo habría modificar la llamada desde el controlador para que incluya a la nueva funcionalidad.

Patrón Fachada (Facade Pattern)

Este patrón se puede observar en la función `aplicar_procesamiento_dual()` dentro del módulo `procesamiento.py`. Esta función actúa como una fachada, es decir proporciona una interfaz simple sobre un sistema mucho más complejo. De esta manera el cliente (en el caso actual sería el código del controlador en `interfaz.py`) solo necesita llamar a una única función (pasando las dos imágenes) y es esta función la que se encarga de distribuir las tareas a realizar entre el resto de tareas como por ejemplo obtener las máscaras, realizar los recortes, eliminar el peciolo, entre otras. Así se le proporciona al cliente una abstracción del sistema ocultando estos detalles concretos y simplificando aparentemente el proceso.

- **Impacto en el Mantenimiento:** Este desacoplamiento entre el cliente (*Frontend*) y el subsistema de procesamiento (*Backend*) permite que se puedan realizar modificaciones a alguna de las funcionalidades como mejorar algoritmos o implementar nuevos sin tener un impacto sobre el aspecto visual de la interfaz de usuario, y viceversa. Facilitando las tareas de mantenimiento o simplificando la implementación de nuevas funcionalidades.
- **Impacto en la Escalabilidad:** Del mismo modo en el caso de que se detectasen cuellos de botella en alguna de las funciones implementadas y se desease mejorarlas para hacerlas más eficientes se podría sustituir el subsistema de procesamiento detrás de la fachada por uno más eficiente o mejorado sin afectar al resto de la aplicación mejorando así la escalabilidad del proyecto.

Gestión de Estado Centralizado (Centralized State Management)

Aunque no se trate de un patrón de diseño clásico de GoF, al usar `st.session_state` en Streamlit, actúa como un patrón de *Registro* o *Singelton de Sesión* proporcionando un único lugar centralizado donde se almacena el estado compartido de la sesión.

De esta manera en vez de tener que pasar los datos necesarios para al aplicación como los hipercubos o las imágenes procesadas mediante múltiples llamadas a las funciones de la interfaz, estos datos se guardan en `st.session_state` y se recuperan cuando es necesario.

- **Impacto en el Mantenimiento:** La aplicación de este patrón simplifica en gran medida el flujo de datos de la aplicación. Además, hace que el estado de la aplicación sea predecible y fácil de rastrear reduciendo así la complejidad

de la aplicación y la probabilidad de errores debido a que no se tiene que pasar el estado explícitamente entre componentes.

- **Impacto en la Escalabilidad:** También el hecho de tener un gestor de estado centralizado hace que si la aplicación crece añadiendo nuevos pasos o funcionalidades tiene como consecuencia que evita que la lógica de la aplicación no se vuelva inmanejable.

C.4. Guía de estilo

Se ha definido la apariencia visual de la aplicación **EcoVid** para que la experiencia de usuario al usarla sea lo más coherente y profesional posible, manteniendo la consistencia de la gama de colores y formas durante todo el uso de la aplicación. La guía de estilo empleada está enfocada en un ambiente profesional de análisis de datos.

Colores y Tipografía

El estilo visual se ha definido en el fichero `src/estilos.css` mediante el lenguaje CSS para la definición de los aspectos visuales y de disposición de la aplicación web. El tema empleado en la aplicación ha sido un tema oscuro, ayudando de esta manera a reducir la fatiga visual durante largas sesiones o ambientes con poca luminosidad.

- **Fondo:** Se usa un color gris muy oscuro ('#161616') para el fondo general de la aplicación.

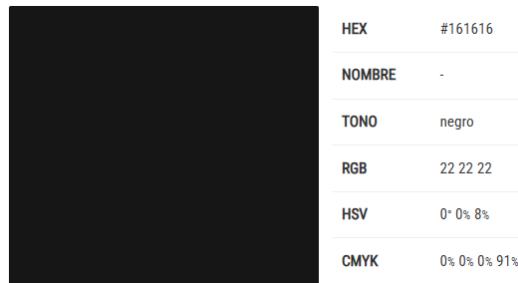


Figura C.1: imagen del color usado para el fondo así como sus valores RGB entre otros

- **Color Primario:** Un naranja fuerte (#D9895B) se utiliza para los elementos destacados como títulos y el borde de la zona de carga de archivos. Los botones usan un gradiente de este mismo naranja. La selección de este color viene motivada por la representación del cobre que tiene un color similar haciendo que en todo momento esté presente este elemento fundamental dentro del trabajo.



Figura C.2: imagen del color primario usado para los elementos de la interfaz así como sus valores RGB entre otros

- **Color Secundario:** Un naranja suave (#E0A17E) a modo de variación del color principal que mantiene el esquema de colores cobrizos de la aplicación, pero en un tono más difuminado usado como contraste para algunos de los detalles de la aplicación.

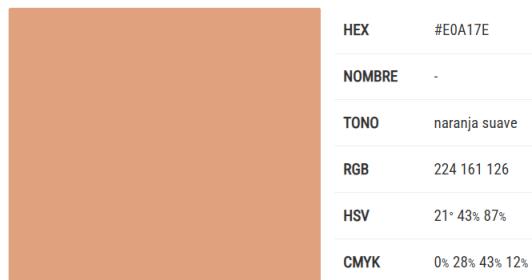
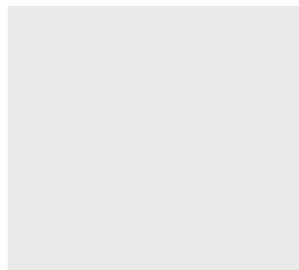


Figura C.3: imagen del color secundario usado para los elementos de la interfaz así como sus valores RGB entre otros

- **Texto:** El color del texto principal es un blanco (#EAEAEA). De esta manera se asegura un contraste adecuado del texto sobre el fondo mejorando así su legibilidad y mejorando la experiencia de usuario.



HEX	#EAEAEA
NOMBRE	-
TONO	blanco
RGB	234 234 234
HSV	0° 0% 91%
CMYK	0% 0% 0% 8%

Figura C.4: imagen del color usado para el texto así como sus valores RGB entre otros

- **Tipografía:** La fuente principal es 'Segoe UI', con alternativas estándar Sans Serif como Tahoma y Verdana. Estas fuentes han sido elegidas buscando una buena legibilidad de los carteles y datos dentro de la aplicación. [ejemplo tipografía]

Nombre e Iconografía

- **Nombre:** El nombre de la aplicación es **EcoVid**. Este nombre refleja una solución que ayuda con la sostenibilidad de los cultivos de vid.
- **Iconografía:** Se ha diseñado un logotipo donde el elemento principal es un racimo de uvas, y a modo de detalle en las hojas de las mismas se han incluido tonos rojizos que hacen referencia al cobre, principal elemento a detectar con la aplicación. Este logotipo refleja el ámbito al que pertenece la aplicación. En la interfaz se prioriza el uso de etiquetas de texto claras e iconos para describir las acciones, manteniendo un diseño limpio y funcional.



Figura C.5: Logotipo diseñado para la aplicación

Además, se han incluido en los botones una serie de emoticonos los cuales ayudan con la identificación de la acción que realiza cada uno de ellos facilitando así su comprensión, además de mejorar la usabilidad general de la aplicación.

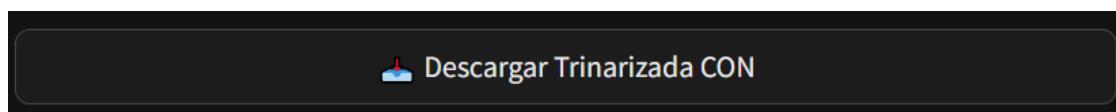


Figura C.6: Ejemplo de utilización de los emoticonos en los botones

- **Logotipos:** En el pie de página se muestran los logotipos de las entidades colaboradoras o financieras del proyecto, como la Unión Europea, la Universidad de Burgos (UBU) y el grupo de investigación GICAP.

Apéndice D

Anexo: Documentación Técnica de Programación

D.1. Introducción

En este apartado se describen de manera detallada aspectos técnicos del proyecto como la organización de los ficheros, también se describe el proceso a seguir para la instalación de los componentes necesarios, así como para la ejecución de la aplicación. Además, se incluye un apartado donde se detallan las pruebas realizadas para verificar el correcto funcionamiento de la aplicación. Este apartado está dirigido principalmente a programadores que necesiten mantener, modificar o extender la funcionalidad de la aplicación.

D.2. Estructura de directorios

La organización de los distintos directorios está diseñada para presentar una separación clara entre el código fuente de las distintas funcionalidades, los recursos visuales como logotipos y los archivos de configuración de la aplicación.

```
.
|-- .gitignore
|-- README.md
'-- src/
    |-- .streamlit/
    |-- Dockerfile
    |-- lanzar_app.bat
    |-- main.py
    |-- estilos.css
    |-- requirements.txt
    |-- bandas.xml
    |-- funciones/
        |   |-- __init__.py
        |   |-- interfaz.py
        |   |-- procesamiento.py
        |   |-- alignment.py
        |   |-- archivos.py
        |   '-- csv.py
    '-- recursos/
        |-- Escudo_ubu.jpg
        |-- gicap_logo.jpeg
        '-- imagen_logo_UE.png
```

- **src/**: Carpeta principal que contiene todo el código fuente, la configuración y los recursos de la aplicación.
- **main.py**: Punto de entrada de la aplicación. Se encarga de la configuración inicial de la página y de lanzar la interfaz.
- **Dockerfile**: Fichero de configuración para crear una imagen de Docker, permitiendo ejecutar la aplicación en un entorno dentro de un contenedor y reproducible.
- **lanzar_app.bat**: *Script* de Windows para ejecutar la aplicación de forma sencilla.
- **estilos.css**: Fichero CSS que define la apariencia visual de la aplicación.
- **requirements.txt**: Lista las dependencias de Python necesarias para ejecutar el proyecto.

- **funciones/**: Paquete de Python donde se encapsula toda la lógica de la aplicación (*backend*).
- **recursos/**: Contiene los archivos de imagen, como los logotipos mostrados en el pie de página.
- **.gitignore**: Especifica los ficheros y directorios que el sistema de control de versiones Git debe ignorar.

D.3. Manual del programador

Esta sección describe los módulos más importantes del sistema y sus responsabilidades.

Punto de Entrada (`main.py`)

Se trata del script principal que se ejecuta con Streamlit. Sus tareas son:

1. Configurar la página de la aplicación (`st.set_page_config`), definiendo el título, el ícono y el *layout* de la página principal de la aplicación.
2. Cargar y aplicar los estilos definidos en `estilos.css`.
3. Registrar la función de limpieza `limpiar_carpeta` para que se ejecute automáticamente al cerrar la aplicación para limpiar el directorio de los archivos subido para el análisis evitando problemas de almacenamiento a la larga.
4. Invocar a la función `cargar_hyper_bin()` del módulo de interfaz, cuyo objetivo es renderizar todos los componentes de la UI.
5. Mostrar un pie de página con los logotipos de las entidades colaboradoras, cargados desde la carpeta `recursos/`.

Módulo de Interfaz (`interfaz.py`)

Este módulo es responsable de toda la interfaz de usuario.

- **cargar_hyper_bin()**: Se trata de la función principal que construye la UI. Para ello define los cargadores de archivos para las imágenes SIN y CON gotas, los botones de *Procesar* y *Ejecutar alineación*, y gestiona el estado de la sesión (`st.session_state`) para mostrar los resultados de forma condicional. Actúa como el controlador principal que responde a las acciones del usuario y llama a las funciones del *backend* para realizar las acciones correspondientes.

Módulo de Procesamiento (`procesamiento.py`)

Es el núcleo de la lógica de análisis de las imágenes.

- **`aplicar_procesamiento_dual()`**: Actúa como una fachada (*Facade Pattern*). Recibe los dos hipercubos (CON y SIN) y dirige todo el flujo de análisis: primero obtiene las máscaras iniciales, posteriormente invoca al módulo de alineamiento, deforma la imagen SIN para que coincida con la CON, y finalmente sustrae las detecciones para eliminar el ruido.
- **`_obtener_mascaras()`**: Su función es segmentar la imagen hiperespectral en hoja y posibles gotas basándose en umbrales de reflectancia predefinidos para bandas espectrales específicas. Realiza la denominada "trinarización" separando fondo de hoja y producto de hoja.
- **`trinarizar_final()`**: Construye la imagen de resultado final una vez aplicados los algoritmos de superposición para reducir el ruido.

Módulo de Alineamiento (`alignment.py`)

Encapsula los algoritmos necesarios para la alineación automática.

- **`_remove_petiole()`**: Función de pre-procesado que utiliza operaciones morfológicas para eliminar el pecíolo de la máscara de la hoja (elemento de la hoja irrelevante para el análisis), permitiendo que el algoritmo se centre en el limbo.
- **`compute_affine_transform()`**: Calcula la matriz de transformación afín (serie de movimientos o transformaciones a realizar sobre una imagen para que coincidan y poder superponerlas) que alinea una imagen sobre la otra. Para ello utiliza el detector ORB sobre los bordes de las hojas y el algoritmo RANSAC para asegurar la robustez del modelo.

D.4. Instalación de Herramientas Necesarias

Para poder ejecutar y desarrollar la aplicación, es necesario tener instaladas las siguientes herramientas en su sistema.

Python

1. **Descarga**: Se recomienda utilizar Python 3.8 o superior. La versión probada y recomendada para este proyecto es Python 3.11. Se puede descargar el

instalador desde la página oficial de Python: <https://www.python.org/downloads/>.

2. Instalación (Windows):

- Ejecutar el instalador descargado.
- **MUY IMPORTANTE:** Marcar la casilla *Add Python to PATH* durante la instalación. Esto facilitará la ejecución de Python desde la línea de comandos.
- Seguir las instrucciones del instalador.

3. Instalación (Linux/macOS):

- Python suele venir preinstalado. Para instalar versiones específicas, se recomienda usar gestores de paquetes como ‘apt’ (Debian/Ubuntu), ‘brew’ (macOS) o ‘pyenv’ para gestionar múltiples versiones de Python.
- Ejemplo con ‘apt’ (Ubuntu): `sudo apt update && sudo apt install python3.11 python3.11-venv`

4. Verificación:

Abrir una terminal o línea de comandos y ejecutar: ‘`python -version`’ o ‘`python3 -version`’. Esto debería mostrar la versión de Python instalada.

Docker Desktop

Docker es una plataforma esencial para la reproducibilidad del entorno de la aplicación.

1. Descarga:

Descargar Docker Desktop desde la página oficial: <https://www.docker.com/products/docker-desktop/>.

2. Instalación (Windows/macOS):

- Ejecutar el instalador y seguir las instrucciones. Requiere virtualización (Hyper-V en Windows o Virtualization.framework en macOS) que Docker Desktop suele configurar automáticamente.
- Asegurarse de que Docker Desktop esté en ejecución (el ícono de Docker debería aparecer en la bandeja del sistema o barra de menú).

3. Instalación (Linux):

- Seguir las instrucciones específicas para su distribución de Linux en la documentación oficial de Docker: <https://docs.docker.com/engine/install/>.

4. **Verificación:** Abrir una terminal y ejecutar: ‘docker –version’. Esto debería mostrar la versión de Docker instalada.

D.5. Compilación, instalación y ejecución del proyecto

El proyecto no requiere compilación. Los pasos para su instalación y ejecución son:

Método 1: Ejecución local con Python

1. **Prerrequisitos:** Asegurarse de tener instalado Python 3.8 o superior (recomendado 3.11) y todas las herramientas mencionadas en la sección anterior.
2. **Instalación de dependencias:** Navegar hasta la raíz del proyecto y ejecutar el siguiente comando para instalar las bibliotecas desde `src/requirements.txt`:

```
pip install -r src/requirements.txt
```

3. **Ejecución:** Ejecutar la aplicación con uno de los siguientes métodos:

- Usando el script por lotes (solo en Windows):

```
cd src
.\lanzar_app.bat
```

- Usando el comando de Streamlit directamente desde la raíz del proyecto:

```
streamlit run src/main.py
```

Método 2: Ejecución con Docker (Recomendado para reproducibilidad)

1. **Prerrequisitos:** Tener Docker Desktop instalado y en ejecución.
2. **Construcción de la imagen:** Desde la raíz del proyecto, ejecutar:

```
docker build -t dig4vitis-app src/
```

3. **Ejecución del contenedor:** Una vez construida la imagen, ejecutar:

```
docker run -p 8501:8501 dig4vitis-app
```

La aplicación estará disponible en el navegador en la dirección **<http://localhost:8501>**.

D.6. Pruebas del sistema

Debido a la naturaleza visual e interactiva del proyecto, las pruebas realizadas se centraron en validaciones independientes de las funciones tanto por parte del desarrollador como por parte de expertos versados en el tema para poder verificar que los resultados obtenidos son correctos o factibles. No se desarrolló una suite de pruebas unitarias automatizadas.

El proceso de pruebas manuales consistió en:

- **Pruebas de Carga:** El primer aspecto a verificar es la correcta subida de los archivos (par de ficheros .bil y .bil.hdr). Además de comprobar la robustez del programa frente a archivos erróneos o corruptos verificando que se mostraban los mensajes de error pertinentes para avisar al usuario de posibles errores en la carga de dichos archivos.

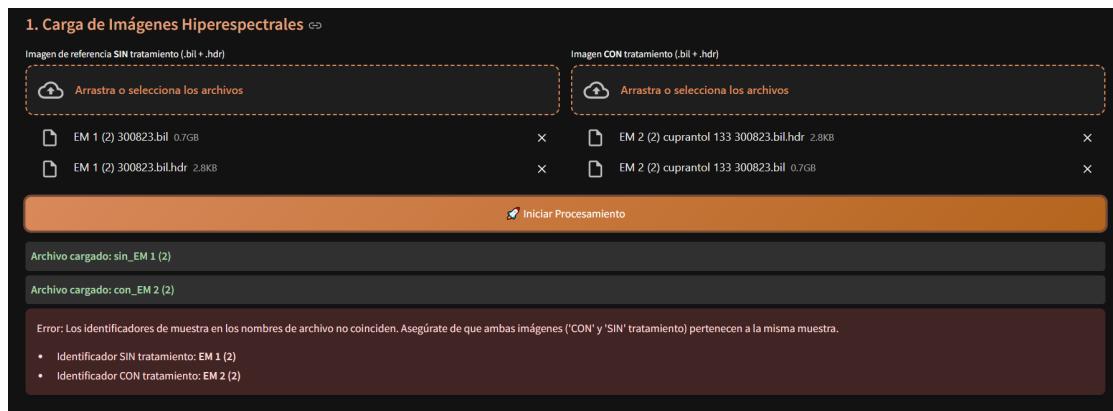


Figura D.1: mensaje en caso de que ambas imágenes no sean de la misma hoja.



Figura D.2: mensaje en caso de que el formato de los archivos subidos no sean los adecuados.

- **Pruebas de Procesamiento y Análisis:** Se utilizó un conjunto de imágenes de muestra para ejecutar el flujo completo. Se inspeccionaron visualmente los resultados en cada paso:

- Correcta generación de las previsualizaciones RGB y trinarizadas base.
- Calidad de la superposición de alineamiento para confirmar que el algoritmo ORB funcionaba correctamente.
- Coherencia del resultado final, verificando que el ruido era eliminado eficazmente en la imagen trinarizada automática.

Estas pruebas fueron supervisadas por expertos para poder verificar la coherencia y factibilidad de los resultados obtenidos.

- **Pruebas de funcionamiento de algoritmos:** para verificar el correcto funcionamiento del sistema de eliminación de ruido por diferencia se realizó un análisis en el que se subieron dos imágenes iguales. Si el algoritmo funciona correctamente, el resultado del análisis debería ser de un porcentaje de recubrimiento de 0 %. Se verificó que el resultado era correcto verificando la hipótesis.

- **Pruebas de Interfaz y Usabilidad:** Por último, se llevó a cabo una revisión de todos los componentes de la interfaz donde se verificó su correcto funcionamiento asegurando que estos se comportaban de la manera esperada, además de verificar que la navegación por la UI es intuitiva.

Apéndice E

Anexo: Documentación de Usuario

E.1. Introducción

En este apartado se detallarán y explicarán los requisitos necesarios para la ejecución de la aplicación así como los detalles de como instalar todas las herramientas necesarias y una guía que sirva como base para la utilización de la aplicación.

E.2. Requisitos de usuario

Los requisitos necesarios para poder ejecutar la aplicación son:

- **Equipo:** Un ordenador personal (Windows, macOS o Linux). Se recomienda un equipo moderno, ya que el procesamiento de imágenes hiperespectrales puede consumir una cantidad considerable de recursos como memoria RAM.
- **Software:** Un navegador web actualizado (como Google Chrome, Mozilla Firefox o Microsoft Edge). Para la instalación se necesitará software adicional (ver sección siguiente).<
- **Datos:** Debes tener preparadas las imágenes hiperespectrales que deseas analizar. Para ello se requiere, para cada muestra de hoja, dos pares de ficheros en formato ENVI:
 - Un fichero de datos con extensión .bil.
 - Un fichero de cabecera con extensión .hdr.

Necesitarás un par de estos ficheros para la imagen de la hoja **SIN tratamiento** y otro par para la hoja **CON tratamiento**.

E.3. Instalación y Ejecución

Para poder ejecutar la aplicación, sigue estos pasos en orden.

Paso 1: Instalar el software previo (Git)

Antes de nada, es imprescindible tener instalado **Git**, una herramienta que nos permitirá descargar el código del proyecto.

- **En Windows:**

1. Dirígete a la página oficial de Git: <https://git-scm.com/download/win>.
2. Descarga y ejecuta el instalador. Puedes aceptar las opciones por defecto durante la instalación.

- **En macOS:**

1. Abre la aplicación "Terminal".
2. Escribe el comando `git -version` y pulsa Enter.
3. Si Git no está instalado, el sistema te ofrecerá automáticamente instalar las *Command Line Developer Tools*. Acepta para completar la instalación.

- **En Linux (Debian/Ubuntu):**

1. Abre una terminal.
2. Ejecuta los siguientes comandos:

```
sudo apt update  
sudo apt install git
```

Paso 2: Descargar el código de la aplicación

Con Git ya instalado, puedes obtener el código fuente. Éste se encuentra en el repositorio: https://github.com/Urbi22/TFG_Dig4Vitis_V2.

Opción A (Recomendada): Clonar con Git

1. Abre una terminal (Símbolo del sistema o PowerShell en Windows).
2. Navega con el comando `cd` hasta la carpeta donde quieras guardar el proyecto (ej. `cd Desktop`).
3. Ejecuta el siguiente comando:

```
git clone https://github.com/Urbii22/TFG_Dig4Vitis_V2.git
```

Esto creará una carpeta llamada `TFG_Dig4Vitis_V2` con todos los archivos.

Opción B: Descargar como .ZIP

1. Desde la página web del repositorio, haz clic en el botón verde `<> Code` y selecciona **Download ZIP**.
2. Una vez descargado, **descomprime el archivo .zip** en la ubicación que prefieras.

Paso 3: Ejecutar la aplicación

Una vez tienes el código, elige uno de los dos métodos siguientes para lanzar la aplicación.

Método 1: Ejecución con Docker (Recomendado)

Este método funciona en cualquier sistema operativo (Windows, macOS, Linux).

1. **Instalar Docker:** Descarga e instala Docker Desktop desde su página web oficial: <https://www.docker.com/products/docker-desktop/>.
2. **Abrir una terminal:** Abre una ventana de terminal (Símbolo del sistema o PowerShell en Windows, Terminal en macOS/Linux).
3. **Navegar a la carpeta del proyecto:** Usa el comando `cd` para situarte en la carpeta donde has descomprimido el proyecto.
4. **Construir la imagen de la aplicación:** Ejecuta el siguiente comando. Este proceso solo es necesario la primera vez y puede tardar varios minutos.

```
docker build -t ecovid-app src/
```

5. **Ejecutar la aplicación:** Una vez construida la imagen, ejecuta este comando para iniciar la aplicación:

```
docker run -p 8502:8502 ecovid-app
```

6. **Abrir la aplicación:** Abre tu navegador web y ve a la dirección **`http://localhost:8502`**.

Método 2: Ejecución local (Para usuarios de Windows)

Este método utiliza un script para facilitar el lanzamiento.

1. **Instalar Python:** Si no lo tienes, instala Python (versión 3.8 o superior, recomendada 3.11) desde <https://www.python.org/>. Asegúrate de marcar la casilla *Add Python to PATH* durante la instalación.
2. **Instalar dependencias:** Abre una terminal en la carpeta del proyecto (sitúate en la carpeta del proyecto en el explorador de archivos, click derecho con el ratón y seleccionar la opción *abrir terminal aquí* o similar, o bien abre una terminal y navega hasta la ubicación del proyecto como se ha descrito con anterioridad) y ejecuta el siguiente comando para instalar las librerías necesarias:

```
pip install -r src/requirements.txt
```

3. **Lanzar la aplicación:** Navega hasta la carpeta `src` y haz doble clic en el archivo `lanzar_app.bat`, o bien navega con la terminal hasta la carpeta `src/` del proyecto y ejecuta el comando:

```
streamlit run main.py
```

4. Se abrirá una ventana de terminal y, tras unos segundos, la aplicación aparecerá en tu navegador web.

En el caso de que el sistema operativo no sea WINDOWS para lanzar la aplicación a través de la línea de comandos será necesario realizarlo mediante el comando

```
streamlit run main.py
```

dentro de la carpeta `src/` del proyecto.

E.4. Manual del usuario

La interfaz de la aplicación está diseñada para guiarte a través de un proceso de 4 pasos.

Cuando ejecutes la aplicación verás una interfaz como la siguiente:



Figura E.1: Interfaz principal de la aplicación una vez ejecutada

En ella verás dos pestañas diferentes, la primera que es la que está por defecto la aplicación en si donde se puede realizar el análisis mientras que la segunda pestaña es un *Acerca del TFG* donde se detallan todos los aspectos relevantes a la aplicación como cual es su objetivo y un breve resumen de su utilización

Paso 1: Carga de las imágenes

Al abrir la aplicación, verás la pantalla principal.

- En la sección 1. *Carga de imágenes hiperespectrales*, encontrarás dos recuadros de carga.
- En el recuadro de la izquierda, sube los dos ficheros (.bil y .hdr) de la hoja sin tratamiento. Para ello podrás tanto pulsar y seleccionarlos desde el explorador de archivos o arrastar dentro del recuadro si así lo deseas.
- En el recuadro de la derecha sube los dos ficheros de la hoja con tratamiento. Con el mismo procedimiento que para el caso anterior

- Si los ficheros se han cargado correctamente verás una imagen como la siguiente:



Figura E.2: imagen de la interfaz una vez los archivos se han subido correctamente

Paso 2: Procesamiento y previsualización

- Una vez cargados los cuatro ficheros, haz clic en el botón naranja **Iniciar procesamiento**.
- La aplicación procesará los datos y, en la sección "2. Resultados del Análisis" mostrará el resultado final en formato imagen de la detección del producto sobre la hoja e indicará el porcentaje de recubrimiento calculado.

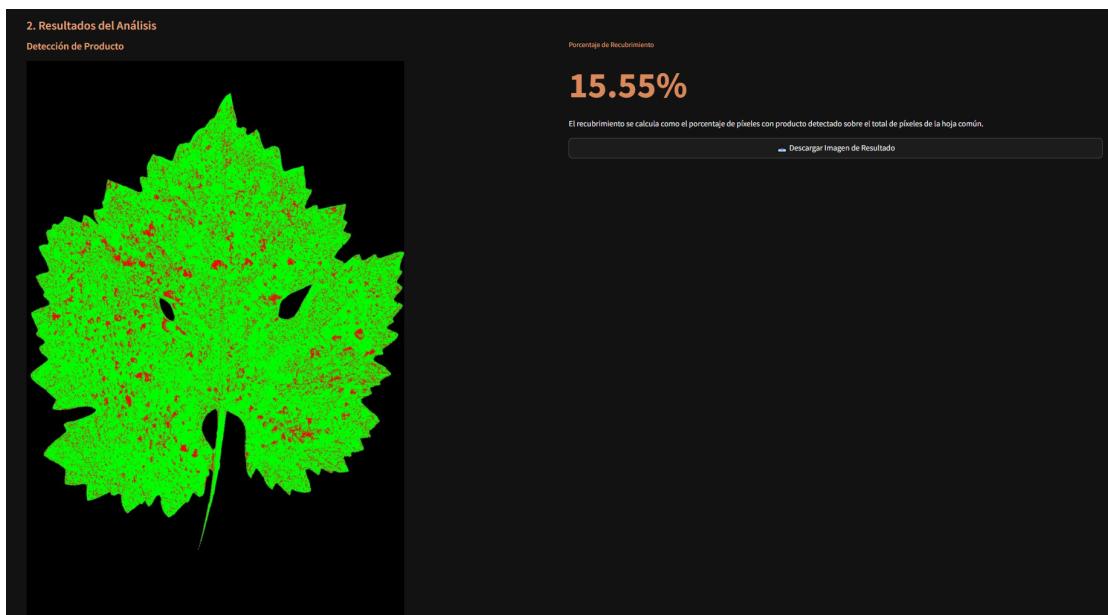


Figura E.3: Sección de resultados del análisis

Se permitirá en caso de desearlo descargar la imagen trinarizada en formato PNG.

Paso 3: visualización de pasos intermedios:

Una vez finalizado el análisis si el usuario lo requiere en la parte inferior, debajo de la imagen trinarizada, puede pulsar sobre el desplegable *Ver detalles y descargas adicionales*. En este desplegable podrá ver tanto la alineación automática realizada como las imágenes subidas en un primer momento (con y sin gotas) tanto en formato RGB como trinarizadas.

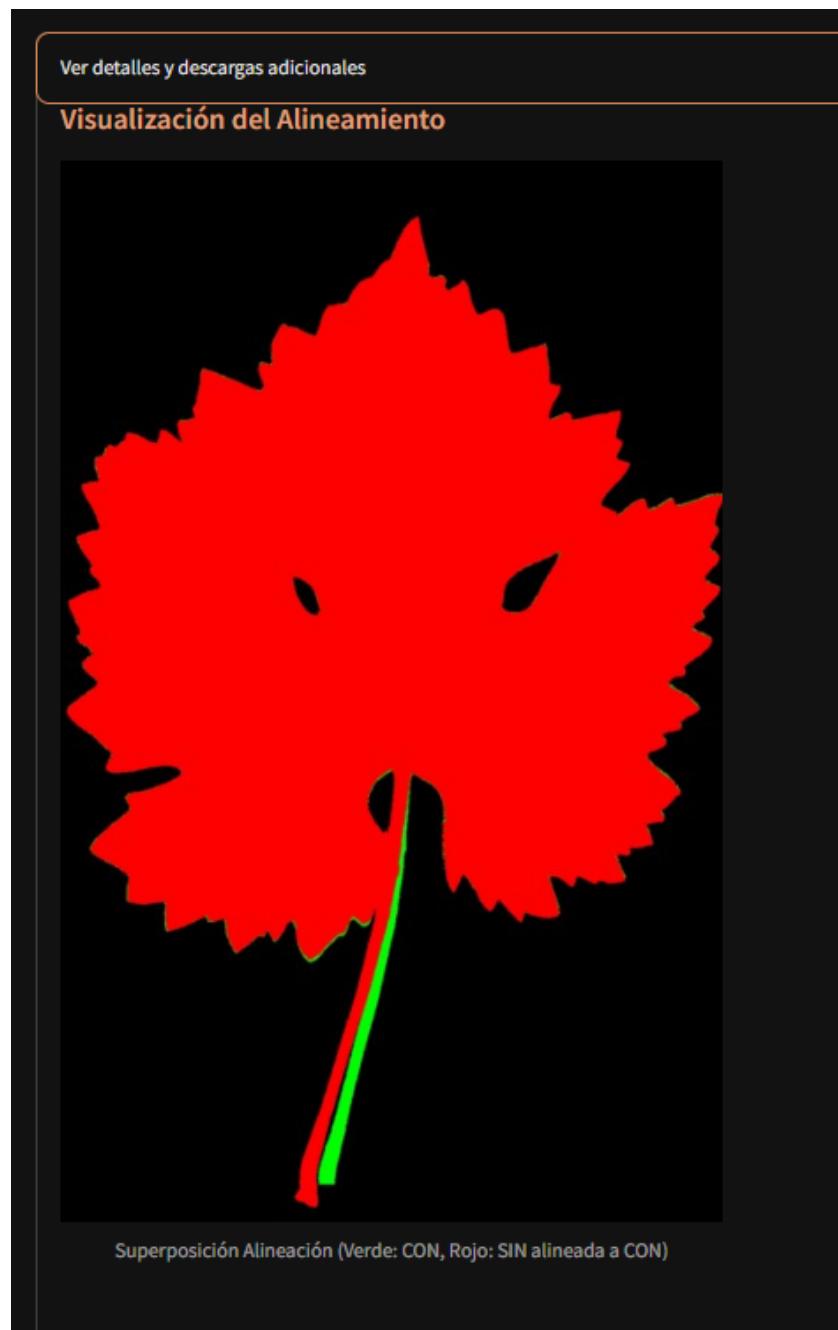


Figura E.4: Imagen donde se muestra el resultado de la superposición automática

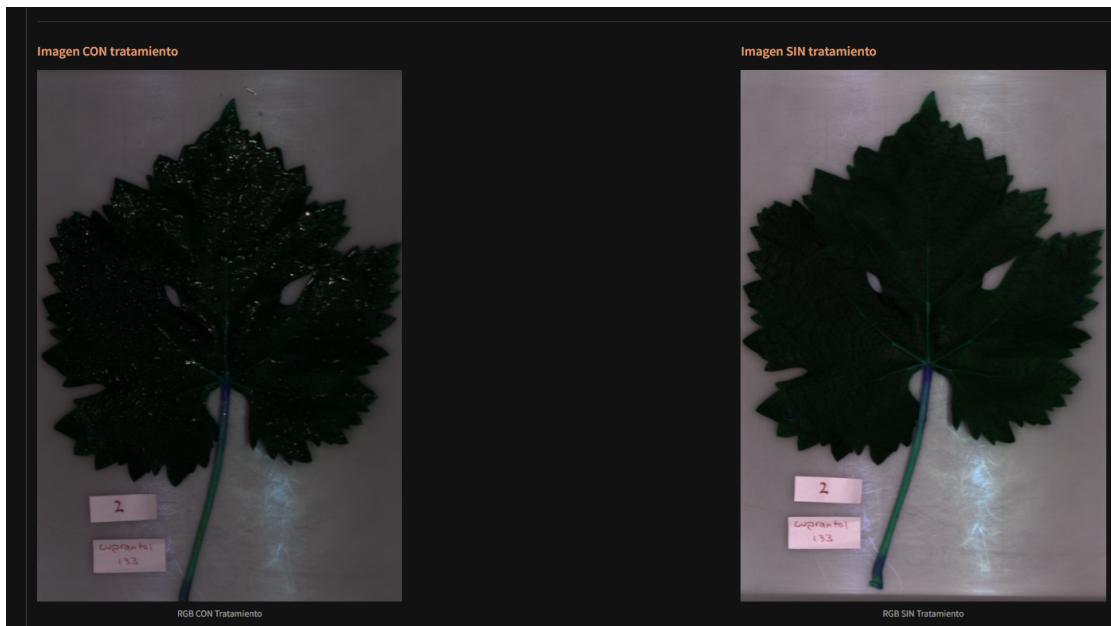


Figura E.5: Comparación de ambas imágenes subidas inicialmente en formato RGB

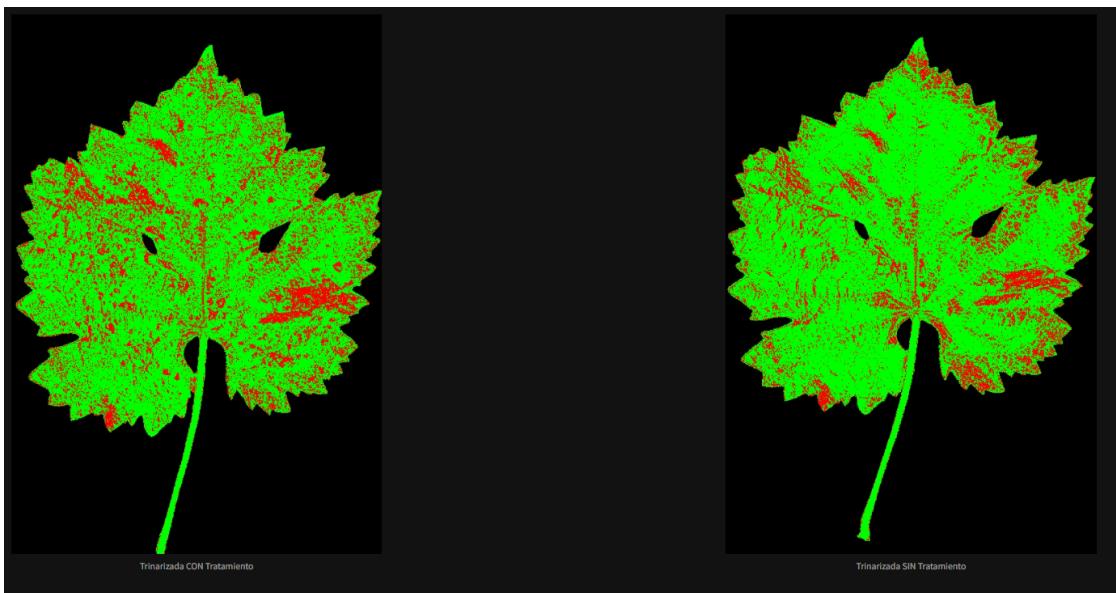


Figura E.6: Comparación de ambas imágenes subidas inicialmente trinariizadas

Paso 4: Descargar los resultados

Una vez finalizado el análisis se pueden descargar las imágenes trinariizadas resultantes, tanto la de resultado final, mediante el botón debajo del porcentaje de recubrimiento calculado, así como las trinariizadas de los archivos subidos (con y sin gotas) mediante los botones en la parte inferior de la aplicación.



Figura E.7: Botones para descargar las trinariizadas de los archivos subidos

Paso 5: visitar la pestaña Acerca del TFG

Si así lo desea antes o después del análisis podrá pulsar en la pestaña de *Acerca del TFG* en la parte superior de la aplicación para obtener algo mas de información sobre el trabajo y la aplicación.



Figura E.8: Pestaña Acerca del TFG

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este apartado se incluye una reflexión personal sobre los aspectos de la sostenibilidad abordados en el presente Trabajo de Fin de Grado (TFG) *Análisis de imágenes hiperespectrales para la detección de cobre en viñedo (proyecto Dig4Vitis v2)*. Se detallan las competencias de sostenibilidad adquiridas y aplicadas durante el desarrollo del proyecto. Para ello se ha usado como guía y base sobre la que fundamentar dichas conclusiones las directrices de sostenibilidad de la CRUE [1]. El objetivo principal de este apartado es indicar y desarrollar aquellos puntos en los que este trabajo contribuye a los principios de desarrollo sostenible, desde un prisma tanto tecnológico como social.

F.2. Sostenibilidad Ambiental: Hacia una Agricultura más Eficiente

El objetivo principal del proyecto realizado se condensa en la detección y cuantificación de tratamientos antifúngicos con base de cobre en hojas de viñedo. Dicho objetivo tiene un impacto directo en la sostenibilidad ambiental. En la actualidad la aplicación de estos tratamientos se realiza de manera indiscriminada resultando en un uso excesivo de los recursos, posible contaminación del suelo y el agua teniendo un impacto negativo sobre la biodiversidad.

Este proyecto se enmarca dentro del concepto de agricultura de precisión. Dicho concepto tiene como objetivo la optimización de los recursos agrícolas. En este caso al realizar una cuantificación precisa de la cantidad de producto en la superficie de la hoja, la herramienta desarrollada facilita una aplicación correcta y óptima del

producto evitando desperdiciar recursos. Reduciendo y optimizando los recursos empleados se obtiene un impacto positivo directo sobre la huella ecológica del uso de estos productos antifúngicos en el entorno de la viticultura. Un menor uso de cobre implica:

- **Reducción de la contaminación:** Menos químicos en el ambiente se traduce en una menor filtración y posterior contaminación hacia acuíferos y suelos adyacentes.
- **Conservación de recursos:** La producción y distribución de estos fungicidas requieren energía y materiales; su uso eficiente conserva estos recursos.
- **Impacto en la biodiversidad:** La reducción de químicos puede proteger a organismos no objetivo, como insectos polinizadores o microorganismos del suelo ayudando a no reducir la biodiversidad en la zona donde se aplica el producto.

La tecnología hiperespectral empleada para la recopilación de los datos y evaluación de los cultivos es en sí misma no invasiva y no destructiva. Por este motivo este tipo de tecnología es sostenible en comparación a con otros tipos de muestreo que implican destrucción de los cultivos o parte de ellos para la obtención de las muestras y datos. La capacidad de analizar grandes áreas de forma eficiente sin alterar el ecosistema es un pilar fundamental de la sostenibilidad en la agricultura moderna.

F.3. Sostenibilidad Social: Mejora de la salud pública y costes de cultivo

Aunque menos directa, la contribución de este proyecto a la sostenibilidad social es igualmente relevante. Una agricultura más eficiente y sostenible beneficia a la sociedad en varios aspectos:

- **Salud Pública:** La reducción en el uso de fungicidas disminuye la exposición de los trabajadores agrícolas a químicos potencialmente dañinos y reduce los residuos de productos en los alimentos finales, mejorando la calidad de vida de los trabajadores y evitando posibles riesgos para la salud de los consumidores finales del producto.
- **Sostenibilidad Económica para el Agricultor:** La optimización de los tratamientos conduce a una reducción de costes para los agricultores, al gastar

menos en fungicidas. Esto mejora la viabilidad económica de las explotaciones agrarias, especialmente para pequeños y medianos viticultores.

La posibilidad de descargar resultados y analizarlos detalladamente fomenta la transparencia y el seguimiento de las prácticas agrícolas. Esto es crucial para la confianza del consumidor y la adopción de estándares de calidad.

F.4. Sostenibilidad Económica: Optimización y Rentabilidad Agrícola

Como se ha mencionado con anterioridad el objetivo del proyecto es permitir a los agricultores hacer un uso más eficiente de sus recursos. Gracias a la detección precisa de las zonas tratadas con producto. Causando así un impacto directo en aspectos como:

- **Reducción de costes:** Menos fungicida significa menos gasto en el producto en sí.
- **Optimización de mano de obra y maquinaria:** Las aplicaciones más precisas pueden reducir el tiempo y el esfuerzo necesarios, así como el consumo de combustible de la maquinaria agrícola.
- **Mejora de la eficiencia:** Uno de los resultados que tiene el asegurar una cobertura adecuada solo donde es necesaria, es maximizar la efectividad del tratamiento, protegiendo mejor la cosecha y reduciendo las pérdidas por enfermedades.

F.5. Contribución personal a la Sostenibilidad

Desde una perspectiva personal, el desarrollo de este Trabajo de Fin de Grado me ha permitido no solo reflexionar sobre como un grado como el de Ingeniería Informática puede tener un impacto en la sostenibilidad, sino ver de primera mano cómo mediante herramientas como la que se ha desarrollado en este proyecto pueden ser de gran utilidad a la hora de abordar desafíos de sostenibilidad. Gracias a este proyecto he sido capaz de obtener una mejor comprensión práctica de como la tecnología se puede aplicar para lograr objetivos ambientales y económicos. Aspectos como la eficiencia del código y la reusabilidad del mismo también se alinean con los principios de sostenibilidad, buscando soluciones que optimicen los recursos computacionales y faciliten la futura implementación de posibles mejoras,

además de prolongar la vida útil del software. Este proyecto me ha proporcionado una oportunidad muy valiosa para poder integrar la concienciación ambiental y social en el desarrollo de software, estando en cierto modo tan alejados conceptos como la concienciación ambiental y social del desarrollo meramente técnico de una herramienta software, proporcionándome una competencia que considero ahora esencial para todos aquellos futuros ingenieros.

Este anexo busca subrayar que el proyecto Dig4Vitis v2 no es solo un avance tecnológico, sino una contribución tangible hacia una viticultura más sostenible, eficiente y respetuosa con el medio ambiente y la sociedad.

Bibliografía

- [1] Crue Universidades Españolas. Directrices para la introducción de la sostenibilidad en el currículum. Technical report, Crue Universidades Españolas, apr 2012.