



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática



TFG del Grado en Ingeniería Informática

**Dieta por Dientes
Documentación Técnica**



Presentado por Ismael Tobar García
en Universidad de Burgos — 29 de diciembre de 2016
Tutor: D. Álvar Arnaiz González y Dr. José
Francisco Diez Pastor

Índice general

Índice general	I
Índice de figuras	III
Apéndice A Planificación	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	19
Apéndice B Especificación de Requisitos	21
B.1. Introducción	21
B.2. Objetivos generales	21
B.3. Catalogo de requisitos	22
B.4. Especificación de requisitos	23
Apéndice C Especificación de diseño	33
C.1. Introducción	33
C.2. Diseño de datos	33
C.3. Diseño procedural	40
C.4. Diseño arquitectónico	42
Apéndice D Documentación técnica de programación	47
D.1. Introducción	47
D.2. Estructura de directorios	47
D.3. Manual del programador	49
D.4. Compilación, instalación y ejecución del proyecto	51
D.5. Pruebas del sistema	52
Apéndice E Documentación de usuario	57
E.1. Introducción	57

E.2. Requisitos de usuarios	57
E.3. Instalación	58
E.4. Manual del usuario	58
Apéndice F Procesado automático de la imagen	69
F.1. Introducción	69
F.2. Extracción de bordes mediante procesado de imagen	69
F.3. Extracción de bordes mediante Deep Learning	75

Índice de figuras

A.1. Burndown del sprint 1	4
A.2. Burndown del sprint 2	5
A.3. Burndown del sprint 3	6
A.4. Burndown del sprint 4	7
A.5. Burndown del sprint 5	8
A.6. Burndown del sprint 6	10
A.7. Burndown del sprint 7	11
A.8. Burndown del sprint 8	13
A.9. Burndown del sprint 9	14
A.10.Burndown del sprint 10	15
A.11.Burndown del sprint 11	16
A.12.Burndown del sprint 12	17
A.13.Burndown del sprint 13	18
A.14.Burndown del sprint 14	19
B.1. Diagrama general de casos de uso.	25
B.2. Diagrama de casos de uso extendido.	26
C.1. Diagrama de clases inicial	34
C.2. Diagrama de clases final parte uno.	35
C.3. Diagrama de clases final parte dos.	36
C.4. Diseño de la interfaz de usuario	37
C.5. Diseño de la interfaz de usuario	38
C.6. Barra de herramientas de la aplicación	38
C.7. Barra de herramientas de la imagen	39
C.8. FigureCanvas de la imagen	39
C.9. Pestañas	40
C.10.Diagrama de paquetes inicial	43
C.11.Diagrama de paquetes inicial	44

D.1. Imágenes prueba 1	55
D.2. Imágenes prueba 2	55
D.3. Imágenes prueba 3	56
D.4. Imágenes prueba 4	56
E.1. Ventana inicial de la aplicación	59
E.2. Opción para cargar una imagen	59
E.3. Seleccionador de ficheros	60
E.4. Opciones de apertura de imagen.	61
E.5. Opción para abrir un proyecto.	61
E.6. Opción para abrir un proyecto.	62
E.7. Como se muestra un proyecto cargado o abierto.	63
E.8. Como cambiar el idioma.	63
E.9. Imagen válida para detección de estrías modo semiautomático.	64
E.10. Selección de la orientación de la imagen.	64
E.11. Selección de las opciones disponibles para este modo.	65
E.12. Pasos para seleccionar el color correctamente.	65
E.13. Pasos para añadir un segmentos manualmente.	66
E.14. Selección de la longitud mínima de corte.	66
E.15. Segmentos calculados por el modo automático.	67
E.16. Segmentos englobados en el cuadrado.	68
E.17. Opciones para guardar tabla.	68
F.1. Resumen visual filtros usados.	76
F.2. Pasos del procesado parte uno.	77
F.3. Pasos del procesado parte dos.	78
F.4. Pasos del procesado parte tres.	79

Apéndice A

Planificación

A.1. Introducción

Para la realización de este proyecto hemos seguido la metodología Scrum, la cual consiste en, tener una estrategia de desarrollo incremental del producto, semanalmente, juntar varias fases en paralelo del proyecto en vez del modo contrario que seria en cascada o secuencial. Por lo que, hemos usado la metodología Scrum [?], así en cada iteración vamos dando valor añadido al proyecto, siempre tenemos versiones nuevas y nuestro proyecto va teniendo más valor.

Desde GitHub seguimos estos pasos cada semana:

- Creamos una Milestone con duración de una semana.
- Creamos la issue o tarea correspondiente a la reunión semanal de una hora.
- Vamos creando issues correspondientes a las demás tareas aquellas que son parecidas se incluyen en la misma tarea que se subdivide en puntos.
- Respecto a la gestión temporal se realiza desde ZenHub, que es una herramienta incluida en el navegador e integrada en GitHub, las tareas se pasan de nuevas a abiertas
- A medida que vamos finalizando las tareas las vamos incluyendo en cerradas, para poder ver el gráfico o burdownchart que nos indica el progreso perfecto frente al progreso real.

A.2. Planificación temporal

En esta sección vamos a entrar en detalle de lo que se ha hablado tanto en los sprint meetings «reuniones semanales de sprint» como de lo que se ha conseguido al terminar cada iteración. También vamos a observar en las gráficas como ha ido desarrollándose el trabajo en la linea temporal.

Sprint 0 (9/9/2016 - 16/9/2016)

Se ha especificado a grandes rasgos en que consiste el problema a resolver y unas recomendaciones, por parte de los tutores, para enfrentarnos a dicho problema.

Se va a hacer un mini prototipo para evaluar las herramientas, librerías y algoritmos necesarios. Posteriormente a la reunión con el cliente (Rebeca García [?])¹ se decidirá el lenguaje y librerías a utilizar.

En esta primera iteración se ha hablado de evaluar las distintas herramientas de gestión, documentación y de programación y las tareas son:

- Probar L^AT_EX
- Probar gestores de tarea:
 - Trello
 - Zenhub
 - Version One
- Gestores de versiones
 - GitHub
 - Bitbucket
- Examinar el problema, evaluar el notebook y las posibilidades de las librerías
- Echar un vistazo al artículo

Cumplido:

Esta semana como aun no sabía muy bien cómo usar el repositorio la gráfica no nos dice nada, porque tuvimos que cambiar el uso de los milestones.

¹ Dra. Rebeca García González [?], que estudia paleobiología y paleoecología de homínidos en la Universidad de Burgos

El milestone inicial, ya que no era posible con GitHub usarlo como deseábamos, hasta la semana 1 no pondré burndown porque no refleja nada del trabajo hecho.

Todos los puntos han sido realizados y destacar, la implementación para el algoritmo ha sido amena, y ha funcionado aunque aun tiene cosas que otras semanas mejoraremos.

Sprint 1 (16/9/2016 - 22/9/2016)

En esta semana vamos a tener tareas de interfaz gráfica , de documentación y de codificación y las tareas son:

- Mejora de la detección de las líneas que quedas solapadas.
- Analizar herramientas de interfaces gráficas y comparativa.
 - PyQt4.
 - WxWidget.
- Prototipado inicial de la herramienta y documentar el prototipo.

Cumplido:

Esta semana hemos hecho algunos de los puntos mas relevantes del proyecto ya que la interfaz ha sido realizada correctamente con uso de layouts para facilitar el re-escalado de las pantallas sin que se oculten o descoloquen botones.

Hemos ampliado el rango de frameworks de interfaces con Tkinter y WxPython porque al investigar vimos que también eran muy relevantes en este campo.

En cuanto a la mejora de la detección de líneas también mejoramos el algoritmo ajustando los parámetros y cambiando algunas propiedades.

También al aveces fallar y como aun no sabemos si es posible dejar pasar el fallo hemos implementado por recomendación de los tutores un modo manual para encontrar las líneas que no encontraba el algoritmo, a su vez también valdrá para pintar una imagen vacía manualmente.

En la figura A.1 se muestra el gráfico del Sprint 1.

Sprint 2 (22/9/2016 - 30/9/2016)

En la semana dos vamos a abarcar puntos de la interfaz y puntos de la documentación del proyecto y las tareas son:



Figura A.1: Burndown del sprint 1

- Documentación.
 - Aspectos relevantes.
 - Técnicas y herramientas.
 - Planificación temporal.
- Listas en la interfaz gráfica (Usar tablas para mostrar las rectas que hemos añadido manualmente).
- Cargar imágenes con el file chooser.

Cumplido:

Hemos finalizado los objetivos aunque mas adelante y después de una revisión seguramente tengamos que modificar algunas cosas, añadir mas documentación ya que es la primera semana de documentación del proyecto.

Respecto al punto de la tabla donde aparezcan las listas de líneas que vamos añadiendo queda preguntar, si vendría bien añadir tres botones mas al modo manual.

Cosa que en la reunión con el Cliente (Rebeca García [?])² voy a exponer y posteriormente si parece bien desarrollar.

En cuanto al punto de cargar las imágenes con un *file chooser* de paso, como me sobró algo de tiempo añadí una pantalla de inicio con un mensaje,

² Dra. Rebeca García González [?], que estudia paleobiología y paleoecología de homínidos en la Universidad de Burgos

así la primera vez que abramos la herramienta no se muestren tablas vacías ni una imagen predefinida.

Opte por hacer usar una pagina de inicio a modo de fachada y cuando se cargue la imagen ya iniciar todas las funcionalidades de la aplicación. En la figura A.2 se muestra el gráfico del Sprint 2.



Figura A.2: Burndown del sprint 2

Sprint 3 (30/9/2016 - 7/10/2016)

En la semana tres vamos a abarcar puntos de la interfaz GUI , generar el informe, y pasar actualizar el código de PyQt4 a PyQt5 y las tareas son:

- Acabar la GUI.
 - Mostrar todas las líneas en la tabla.
 - Poder borrar la línea seleccionada.
- Informe.
 - Calcular estadísticas.
 - Mirar documentación Python.
- Pasar código de PyQt4 a PyQt5.

Cumplido:

Hemos finalizado los objetivos de esta semana y hemos generado funcionalidad a la tabla para agregar las líneas, también añadido que se ilumine

la línea seleccionada dentro de la imagen en color amarillo. Hemos añadido funcionalidad para borrar la línea que tenemos seleccionada y también para poder limpiar la tabla completa.

Respecto a la tarea del informe, hemos calculado los estadísticos de todas las líneas, también su clasificación y escritura dentro de un fichero CSV, también la generación de una tabla L^AT_EX que se actualiza a cada ejecución con los datos que han salido para poder pegarla fácilmente a un informe.

Como nos dimos cuenta que la versión de PyQt se había actualizado de la cuatro a la cinco pues hemos pasado el código a la nueva versión y no ha sido muy difícil.

En la figura A.3 se muestra el gráfico del Sprint 3.



Figura A.3: Burndown del sprint 3

Sprint 4 (7/10/2016 - 14/10/2016)

En esta semana vamos a abarcar el diseño software de la aplicación así como sacarlo de los NoteBooks y pasarlo a un IDE en condiciones con su subdivisión en clases y paquetes y las tareas son:

- Diseño de la aplicación.
 - Diagrama de clases.
 - Diagrama de paquetes.
- Implementación del código.
- Corrección de las memorias.

- Herramienta SonarQube.
 - Ejecutar la aplicación.
 - Corregir las horas de débito.

Cumplido:

Hemos finalizado los objetivos del sprint. En paralelo hemos implementado el diseño y el código a medida que teníamos una parte del diseño, de forma incremental, por lo que no se queda del todo reflejado cuando cerramos las tareas.

Hemos elegido Eclipse como IDE junto con su plugin PyDev para Python.

La división en clases hemos conseguido tener una primera estimación de como estaba la aplicación. Respecto a la herramienta SonarQube hemos corregido todos los errores y defectos que salían, a mencionar que no había código repetido ni errores graves.

También detectamos un Bug y fue corregido: Al mostrar las imágenes, tenía un fallo, consistía en que la imagen estaba como si viéramos su reflejo en un espejo, porque el eje de las « y » debe ir inverso a como lo conocemos, en vez de « $0 - tam - Image$ » debía ir de « $tam - Imagen - 0$ ».

En la figura A.4 se muestra el gráfico del Sprint 4.

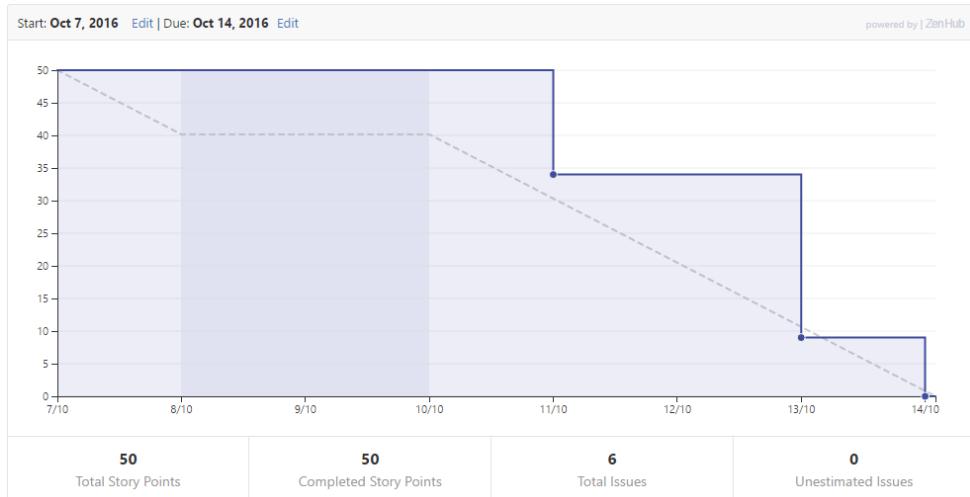


Figura A.4: Burndown del sprint 4

Sprint 5 (13/10/2016 - 22/10/2016)

Esta semana vamos a continuar con el diseño de la aplicación aplicar los patrones correspondientes y paquetes de las clases.

- Aplicar patrones de diseño
 - Fachada.
 - Mediador.
 - Comando.
- Documentar sobre los patrones usados.
- XML mirar documentación sobre ello.

Cumplido:

Esta semana, hemos finalizado los objetivos, hemos documentado los patrones y decidido que con el mediador ya nos servía y el fachada como entrada a la aplicación.

El patrón comando en Python no le hemos visto mucho sentido ya que en Python el connect con la función se hace en una sola linea por lo que no hace falta utilizar un patrón comando.

Respecto al XML hemos mirado documentación y lo hemos implementado que guarde los nombres de todos los archivos que generan un proyecto.

En la figura A.5 se muestra el gráfico del Sprint 5.



Figura A.5: Burndown del sprint 5

Sprint 6 (22/10/2016 - 28/10/2016)

Esta semana vamos a intentar ir terminando las tareas para poder la semana que viene tener un prototipo entregable de la aplicación.

- Completar menús de la GUI.
 - Guardar.
 - Cargar.
 - Acerca de.
 - Ayuda.
- Completar documentación de los fuentes.
 - Paquete código.
 - Paquete gui.
- Pruebas unitarias.
 - Test paquete código.
 - Test paquete gui.

Cumplido:

Esta semana, hemos finalizado los objetivos, aunque en el gráfico no queda reflejado por el ataque del día 21 de octubre de 2016 [?] no pude cerrar las tareas del anterior sprint ni crear las de este. Respecto a los menús de la gui, hemos añadido y renombrado los campos.

Las funcionalidades añadidas son: Poder guardar los cambios, podamos abrir y modificar un proyecto, el acerca de y el botón sobre el que va a colgar la ayuda. También hemos documentado los métodos de las clases del código fuente Python para poder generar la documentación automática con PyDoc.

También hemos desarrollado las pruebas unitarias de la aplicación de todas aquellas funciones que se podía comprobar.

En la figura A.6 se muestra el gráfico del Sprint 6.

Sprint 7 (28/10/2016 - 04/11/2016)

Esta semana vamos a añadir funcionalidades para que el usuario tenga mas comodidad al usar la aplicación, evitando que pueda cometer errores o falsos positivos.

- Detectar y calcular la referencia.



Figura A.6: Burndown del sprint 6

- Detectar la referencia.
- Calcular cuanto es la referencia.
- Detectar la región pintable.
- Revisar toda la aplicación.
 - Revisar el código con SonarQube.
 - Ajustar los cálculos en algunos puntos.
 - Añadir botón de cerrar.
- Corregir la documentación.
 - Memorias.
 - Anexos.

Cumplido:

Esta semana, hemos finalizado todos los objetivos que nos habíamos propuesto, también hemos documentado bastante, no solo corregir lo que teníamos planteado.

El gráfico de esta semana se ha quedado muy desplazado hacia la derecha, porque hemos echo unas pruebas en ZenHub y al sacar varias tareas que ya estaban en cerradas, se ha despintado de la gráfica, pero todas ellas fueron terminadas en tiempo.

Las funcionalidades añadidas son:

- Detectar la región pintable: Era uno de los puntos que nos faltaban, para poder conseguir que solo se pudieran pintar, a mano, las líneas dentro del cuadrado pintado por el usuario, en las imágenes. Porque solo está permitido para este estudio las líneas que se encuentran dentro del recuadro en la imagen.
- Detectar y calcular la referencia: En las imágenes de microscopio, se les hace una marca con los aumentos con los que se han hecho las imágenes, la referencia, que marca la escala y otros datos importantes. Nuestro trabajo ha sido detectarlo y calcular cuantos píxeles son.
- Hemos añadido el botón de cerrar la aplicación que era un punto en el que aun no habíamos pensado y es necesario.
- La mejora de la calidad del software pasando la herramienta SonarQube para corregir las posibles mejoras que nos proponga dicha herramienta.

En la figura A.7 se muestra el gráfico del Sprint 7.

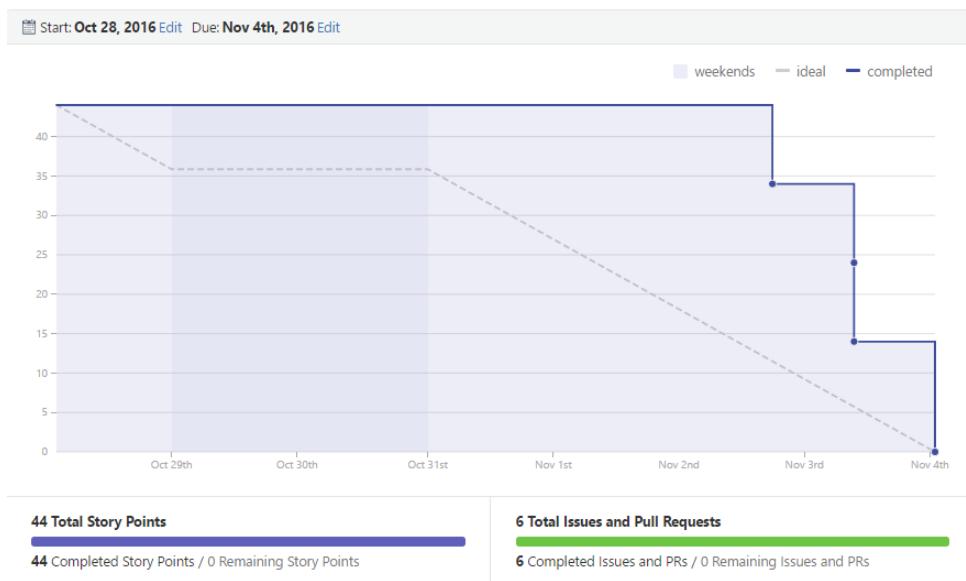


Figura A.7: Burndown del sprint 7

Sprint 8 (04/11/2016 - 11/11/2016)

Esta semana, va a consistir en investigar sobre el modo automático, dar valor al código en cuanto a calidad aplicando algún concepto de patrones de diseño, haciendo una fachada para la entrada salida.

- Diseño:

- Desacoplar del mediador lo que debe ir en la fachada.
- Comprobar que los mediadores solo interfieren con aspectos de la GUI.
- Añadir Configuraciones al cálculo.
 - Añadir las repeticiones.
 - Añadir la longitud mínima para filtrar rectas.
- Investigación detección de bordes por procesado
 - Vamos a usar todos los algoritmos y métodos disponibles en las librerías.
 - Programar otros conocidos pero que no existan en la librería.
 - Buscar mucha mas información sobre los metodos conocidos para detección de bordes.
- Investigación detección de bordes por Deep Learning [?] ³.
 - StrudturedForest.
 - Retina-Unet.

Cumplido:

En esta semana, hemos finalizado los objetivos, además de buscar información sobre los métodos hemos creado un notebook con las pruebas de las ejecuciones de todos ellos.

La semana pasada implementamos la configuración, pero estaba en el código, no podía el usuario cambiarlo en función de sus necesidades. Por lo que hemos extraído esa parte a la interfaz gráfica, así puede modificar los valores de configuración, guardarse en el XML, para cuando se cargue el proyecto, también se carguen esos valores.

En la figura A.8 se muestra el gráfico del Sprint 8.

Sprint 9 (04/11/2016 - 11/11/2016)

Esta semana, vamos a centrarnos en ejecutar y probar todos los métodos estudiados en la semana anterior, algunas modificaciones para facilitar al usuario, su simplicidad y mejorar, los posibles escenarios de ejecución.

- Extracción de características.

³Conjunto de algoritmos de aprendizaje que modela abstracciones en datos, usando transformaciones no lineales

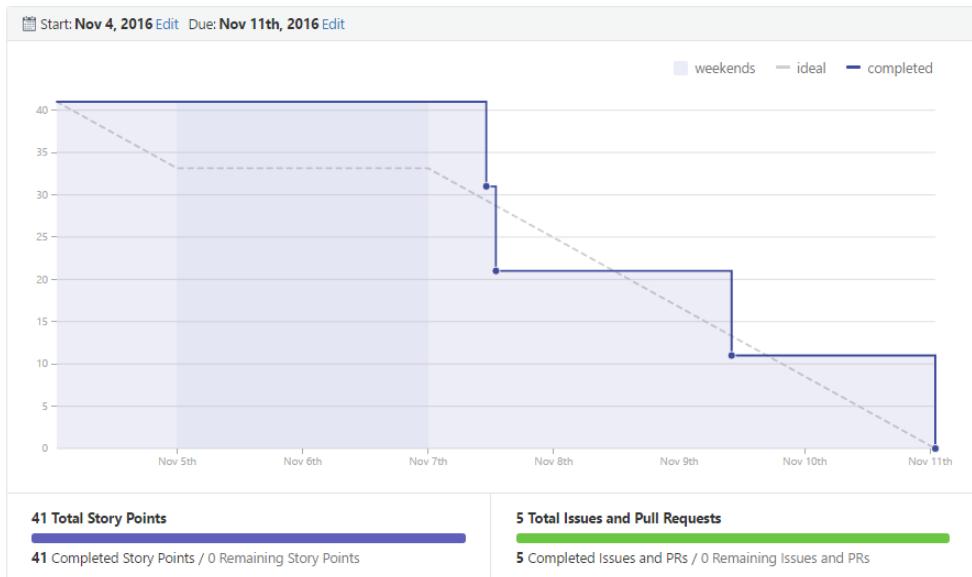


Figura A.8: Burndown del sprint 8

- Aplicar Hough con valores ajustados.
- Extraer y procesar los segmentos.
- Investigar y probar Deep Learning para Python 2.7.
 - Probar en Python 2.7 ya que en 3.5 no funcionan los algoritmos.
- Ejecutable sin dependencias.
 - Crear un .exe pero no ha sido posible ya que no hay versiones actuales de esas herramientas para Python 3.5
 - Crear una alternativa.
- Documentar.
 - Documentar Memorias.
 - Documentar Anexos.
- Independencia del color.
 - Modificar la distancia al color para que funcione con todos los colores.

Cumplido:

Esta semana hemos finalizado los objetivos, pero no todo ha sido posible su implementación o realización. Los algoritmos de Deep Learning, no hemos sido capaces de ejecutarlos, ya que están bastante desactualizados y las dependencias, no están demasiado claras, su ejecución no ha sido posible.

Todos los demás puntos del sprint, han sido realizados con éxito. La parte de la independencia del color, nos ha permitido que ahora nuestra aplicación, funcione para todas las líneas indiferentemente, del color en el que estén pintadas, el usuario elije que color tienen y las busca.

Respecto a la extracción de características, del modo automático por procesado, hemos conseguido que sea decente partiendo de que las líneas no son fácilmente detectables y las que están mas marcadas no son las que nos interesan.

Para el ejecutable sin dependencias las opciones de crear el .exe no han sido posibles dada la desactualización de las herramientas destinadas a estos fines. Como alternativa hemos creado una distribución de Miniconda específica para nuestro propósito que solo contendrá las librerías necesarias para ello.

En la figura A.9 se muestra el gráfico del Sprint 9.

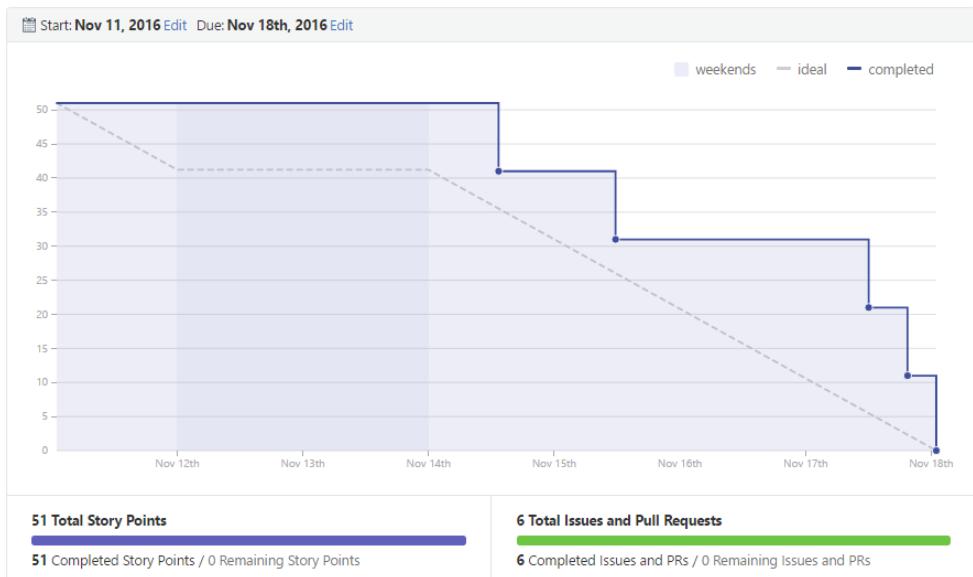


Figura A.9: Burndown del sprint 9

Sprint 10 (18/11/2016 - 25/11/2016)

Esta semana las tareas en las que nos vamos a centrar consistirán en investigar sobre aprendizaje automático, repasar el funcionamiento de la aplicación

y permitir seleccionar el área a evaluar.

- Añadir una forma de pintar el milímetro cuadrado que sera la región pintarle.
- Investigar mas sobre los algoritmos de deep learning.
- Repasar código para ver si esta correcto todo después de las ultimas modificaciones.
- Documentar y corregir las memorias.

Cumplido:

Hemos implementado con éxito, la selección del área a evaluar que se corresponde con 0,56 milímetros cuadrados, hemos repasado el código y retirado líneas comentadas, que ya no se usaban y hemos documentado y corregido las memorias y anexos. La parte de Deep Learning no hemos sido capaces de ejecutar ninguno de los los proyectos que habíamos encontrado.

En la figura A.10 se muestra el gráfico del Sprint 10.



Figura A.10: Burndown del sprint 10

Sprint 11 (25/11/2016 - 02/12/2016)

Esta semana las tareas que vamos a realizar consistirán en, crear una ayuda para el usuario al usar la aplicación, investigación, acoplar la parte automática a la aplicación y documentar.

- Documentar.
- Investigar Deep Learning.
- Crear la ayuda.
- Acoplar la parte de automático a la aplicación.

Cumplido:

Esta semana hemos finalizado con éxito los objetivos, acoplando la parte automática a nuestra aplicación de escritorio y la ayuda de usuario ha sido creada, también hemos documentado los aspectos que hemos utilizado, pero la parte de Deep Learning ha vuelto a quedarse sin poder ejecutar al no funcionar las nuevas herramientas de esta semana.

En la figura A.11 se muestra el gráfico del Sprint 11.

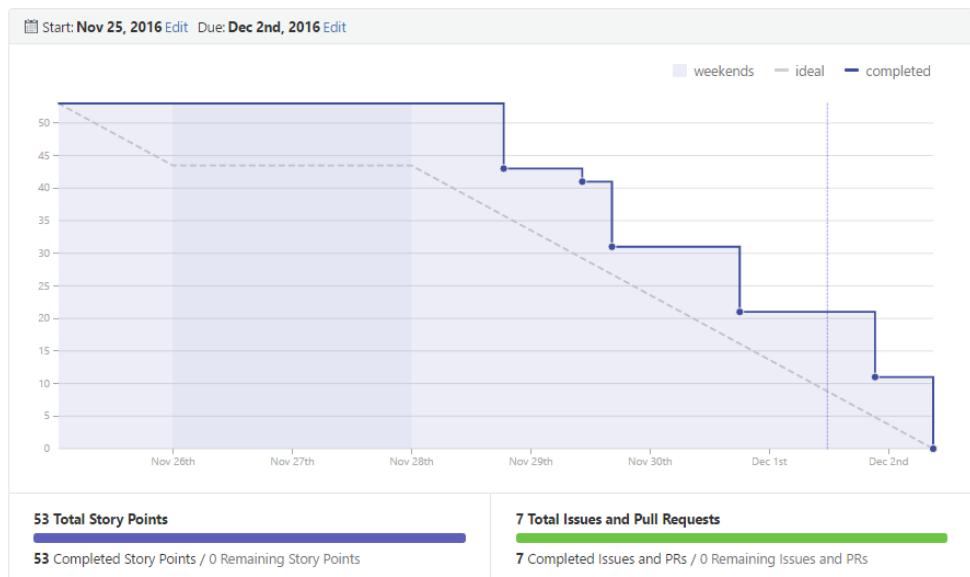


Figura A.11: Burndown del sprint 11

Sprint 12 (02/12/2016 - 09/12/2016)

Esta semana hemos cambiado la dirección de la investigación y las pruebas sobre algoritmos de Deep Learning al no haber tenido éxito probando los existentes en Python, hemos probado algoritmos en Matlab y hemos documentado las memorias y anexos.

- Probar algoritmos de Matlab

- Documentar.
- Documentar parte dos.

Cumplido:

Esta semana hemos finalizado los objetivos aunque los resultados de la parte de Deep Learning no han sido demasiado buenos, ya que en la ejecución de los proyectos propuestos faltaban clases y no funcionaban dichas aplicaciones. En la parte de documentación hemos avanzado bastante y hemos ido completando aspectos de las memorias y anexos. En la figura A.12 se muestra el gráfico del Sprint 12.

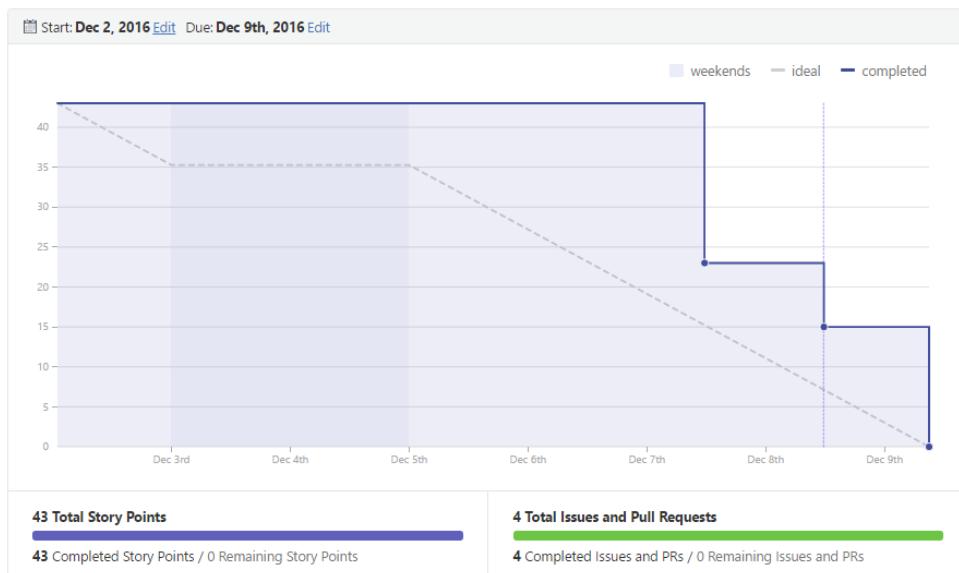


Figura A.12: Burndown del sprint 12

Sprint 13 (09/12/2016 - 16/12/2016)

Esta semana vamos a crear un test de calidad para saber si nuestras predicciones de las estrias son buenas o no, y poder medir posteriormente como de bueno es otra forma de detectarlo. Tambien vamos a documentar anexos y memorias de nuestro proyecto y vamso a probar otro algoritmo de Deep Learning propuesto por el tutor.

- Crear una medida de calidad de nuestra aplicación.
- Documentar anexos y memorias de nuestro proyecto.
- Probar algoritmo propuesto por el tutor.

Cumplido:

Esta semana hemos finalizado el objetivo de crear la medida de calidad para las imágenes detectadas, como prototipo, también hemos realizado la documentación de anexos y memorias de las partes que habíamos planteado, hemos probado el algoritmo pero tampoco hemos sido capaces de hacerlo funcionar.

En la figura A.13 se muestra el gráfico del Sprint 13.

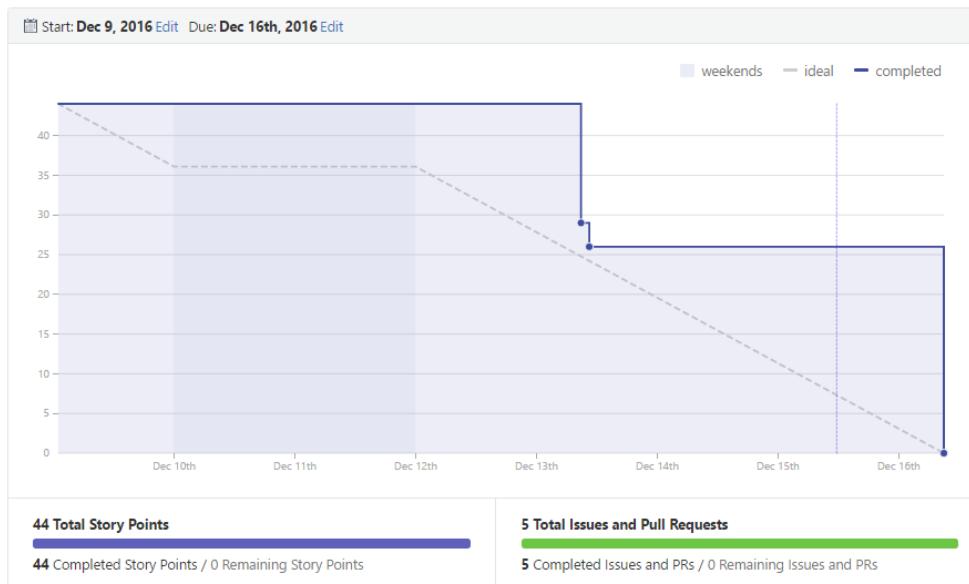


Figura A.13: Burndown del sprint 13

Sprint 14 (16/12/2016 - 23/12/2016)

Esta semana hemos planteado que ya casi tenemos todo terminado y que la medida de calidad debería ser especificada como un test dentro del programa por lo que hemos creado un test específico para dicha medida, y la documentación vamos a ir revisando los puntos mas importantes y completándolos.

- Documentar.
- Implementar test de calidad como test unitario.

Cumplido:

Esta semana hemos finalizado los objetivos marcados sin ningun contratiempo y sin imprevistos tangibles.

En la figura A.14 se muestra el gráfico del Sprint 14.

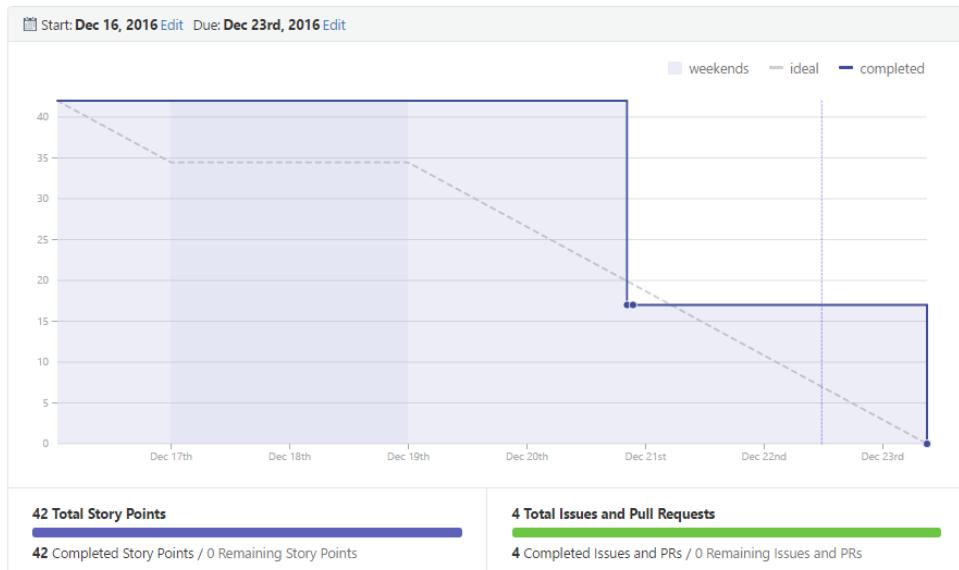


Figura A.14: Burndown del sprint 14

Sprint 15 (23/12/2016 - 30/12/2016)

A.3. Estudio de viabilidad

Viabilidad económica

En este punto, vamos a hacer es un estudio de viabilidad económica del proyecto, vamos a analizar los gastos que hubiera supuesto, si fuese un proyecto real, para una empresa, en cuanto a tiempo de programación e infraestructura necesaria para ello.

Costes de personal

En este proyecto el personal ha estado compuesto por un programador que ha estado durante 4 meses y medio trabajando en sacar adelante el proyecto. Como las horas han sido detalladas en la herramienta ZenHub podemos estimar que a día de hoy el proyecto consta de 480 Horas a 10€ la hora:

$$10e/Hora * 480Horas = 4800e$$

A esta cuantía debemos añadirle el valor que se debería llevar la Seguridad Social que equivaldría a:

- 23,6 %: Por Seguridad Social.
- 5,5 %: Por desempleo.

Cuadro A.1: Tabla de librerías y sus licencias

Librería	Versión	Descripción	Licencia
Jinja2	2.8	Un motor de plantillas escrito en Python.	BSD
Numpy	1.11.1	Procesamiento de arrays para números, strings, records y objetos.	BSD
Scipy	0.18.1	Librería científica para Python.	BSD
Scikit-image	0.12.3	Librería para tratamiento de imágenes para SciPy.	BSD
Networkx	1.11	Librería Python para crear y analizar redes complejas.	BSD
Pillow	3.3.1	Librería Python para tratamiento de imágenes.	PIL license similar MIT
Pyqt	5.6.0	Librería Python para crear GUI.	GPLv2 o GPLv3
Matplotlib	1.5.3	Librería Python para mostrar gráficos 2D.	PSF-based
Tesseract	3.02	Ocr para ejecutar en Windows.	Apachev2

- 0,6 %: Por formación profesional.
- 0,2 %: Por Fogasa.
- Total: 29,9 %

Y esto equivale a:

Viabilidad legal

En este punto, lo que vamos a hacer es analizar las librerías que hemos usado, en nuestro proyecto. Buscar los tipos de licencias que tienen cada uno, luego analizar, dependiendo de las compatibilidades, que tipos de licencia podemos aplicar nuestro proyecto y seleccionar uno de ellos.

BSD de tres cláusulas es compatible con GPL (esto está en construcción)

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este anexo contendrá los objetivos de la aplicación y los requisitos que va a tener esta. Consistirán en breves descripciones de todos los objetivos principales, como se subdividen en objetivos mas cortos para conseguir realizarlos y las funcionalidades implementadas.

Requisitos: Contendrá el que consiste el proyecto y las definiciones de los requisitos y objetivos.

B.2. Objetivos generales

Los objetivos que buscamos con este proyecto son:

- Construcción del prototipo de procesado de las imágenes: Con este prototipo la funcionalidad buscada era, hacer las pruebas sobre el código, que mas adelante introduciríamos en la aplicación real, pero así podríamos ajustar los parámetros para que funcionase bien. También de esta forma estaríamos seguros que el proyecto es realizable.
- Construcción de la aplicación: Una vez que tengamos el prototipo funcionando, tendríamos que construir una aplicación para ejecutar de una forma mas profesional que simplemente desde un notebook, esto es algo muy avanzado para un usuario sin experiencia, así facilitar la ejecución sin necesidad de depender del notebook.
- Construcción del prototipo para el modo automático: Como hemos realizado el trabajo en tiempo, hemos optado por acoplar un modo de detección de bordes automatizado, para ver si eramos capaces de detectar

los segmentos que detectaría un humano. El problema de este modo es que algunos segmentos no son casi perceptibles por un humano por lo tanto los que estén mas marcados son los que podremos detectar.

- Añadir el modo automático a la aplicación: Como hemos echo anteriormente con el modo de detección de las lineas pintadas. Esta vez también lo acoplaremos a nuestra aplicación, facilitando la ejecución y aplicación de este modo sobre imágenes que no estuvieran pintadas.

B.3. Catalogo de requisitos

El proyecto consiste en:

- Construcción del prototipo de procesado de las imágenes.
 - Lectura de imagen.
 - Binarización para detectar las estrías pintadas.
 - Procesado de la imagen binaria.
 - Extracción de características.
 - Procesado de las características.
- Construcción de la aplicación.
 - Lectura de imágenes y cargar en el panel.
 - Construcción del panel de pestañas
 - Construcción del modo de automático para lineas pintadas.
 - Construcción del modo manual para corregir los errores o editar las lineas que hemos pintado.
 - Construcción de la pestaña para el modo completamente automático.
- Construcción del prototipo para el modo automático.
 - Lectura de la imagen.
 - Equalizacion de la imagen para distribuir el histograma.
 - Binarización para extraer bordes.
 - Procesado de la imagen binaria para limpiar de ruido.
 - Calculo de las características de la imagen.
 - Procesado de características.

B.4. Especificación de requisitos

- RF-1: Construcción de la ayuda en la aplicación.
 - RF-1.1: Construcción de los menús donde mostrar las opciones de la ayuda en la aplicación.
 - RF-1.2: Construcción de la ventana que visualizara el HTML.
- RF-2: Construcción de las opciones para cargar un proyecto o crear un proyecto.
 - RF-2.1: Construcción de los menús donde mostrar las opciones de cargar o crear proyectos en la aplicación.
 - RF-2.2: Crear la funcionalidad para escribir un proyecto junto con todos los ficheros necesarios para ello.
 - RF-2.3: Crear la funcionalidad para leer un proyecto y cargarlo para continuar su edición.
- RF-3: Guardar un proyecto, este constara de las imágenes original y pintada, estadísticas, CSV con las líneas, fichero de configuración, y tabla L^AT_EX.
 - RF-3.1: Construcción de los menús donde mostrar las opciones de guardar proyectos en la aplicación.
 - RF-3.2: Crear la funcionalidad para guardar un proyecto junto con todos los ficheros necesarios para ello.
- RF-4: Detectar las estrías de dieta contenidas sobre la imagen.
 - RF-4.1: Construcción de los botones que implementaran las opciones de calcular las estrías dependiendo del modo de la aplicación.
 - RF-4.2: Crear la funcionalidad para los botones de calcular las estrías en un proyecto junto con pintar sobre la imagen los segmentos detectados.
- RF-4.1: Construcción del modo para calcular las estrías pintadas en las imágenes.
 - RF-4.1.1: Seleccionar el color de las estrías pintadas que queremos detectar.
 - RF-4.1.2: Seleccionar la orientación de la imagen, la longitud mínima que debe ignorar y el numero de repeticiones que quiere para calcular los segmentos.
 - RF-4.1.3: Pedir a la aplicación que use el algoritmo implementado para la detección de las estrías pintadas.

- RF-4.1.4: Mostrar los segmentos que ha calculado con el algoritmo en el panel donde se muestra la imagen.
- RF-4.2: Construcción del modo manual para corregir los errores o editar las líneas que hemos pintado.
 - RF-4.2.1: Pintar líneas sobre la imagen, pero solo sobre el cuadrado, para añadir nuevos segmentos.
 - RF-4.2.2: Añadir las líneas que ha calculado el algoritmo de detección de estrías pintadas.
 - RF-4.2.3: Borrar y limpiar la tabla de los segmentos obtenidos, que se corresponden con las estrías.
 - RF-4.2.4: Pintar, permitir mover y fijar el cuadrado donde queremos pintar las estrías manualmente.
 - RF-4.2.5: Guardar los segmentos y obtener las estadísticas y mediciones de todos los segmentos que ha calculado el algoritmo y se muestran en la tabla.
- RF-4.3: Construcción del modo completamente automático para detectar las estrías en imágenes sin pintar.
 - RF-4.3.1: Seleccionar la orientación de la imagen y la longitud mínima que debe ignorar para calcular los segmentos.
 - RF-4.3.2: Mandar calcular al algoritmo las estrías que contiene la imagen.
 - RF-4.3.3: Mostrar los segmentos que ha calculado con el algoritmo en el panel donde se muestra la imagen.
 - RF-4.3.4: Pintar, permitir mover y fijar el cuadrado donde queremos quedarnos con las estrías.

Diagrama de casos de uso

El diagrama que contiene los caso de uso de nuestra aplicación esta contenido en la figura [B.1](#).

El diagrama extendido del caso de uso 4 esta contenido en la figura [B.2](#).

A partir del diagrama principal y el diagrama extendido, se han diseñado las siguientes tablas de los diagramas de casos de uso, se muestran en las siguientes figuras, [B.1](#), [B.2](#), [B.3](#), [B.4](#) y este se subdivide en [B.5](#), [B.6](#) y [B.7](#).

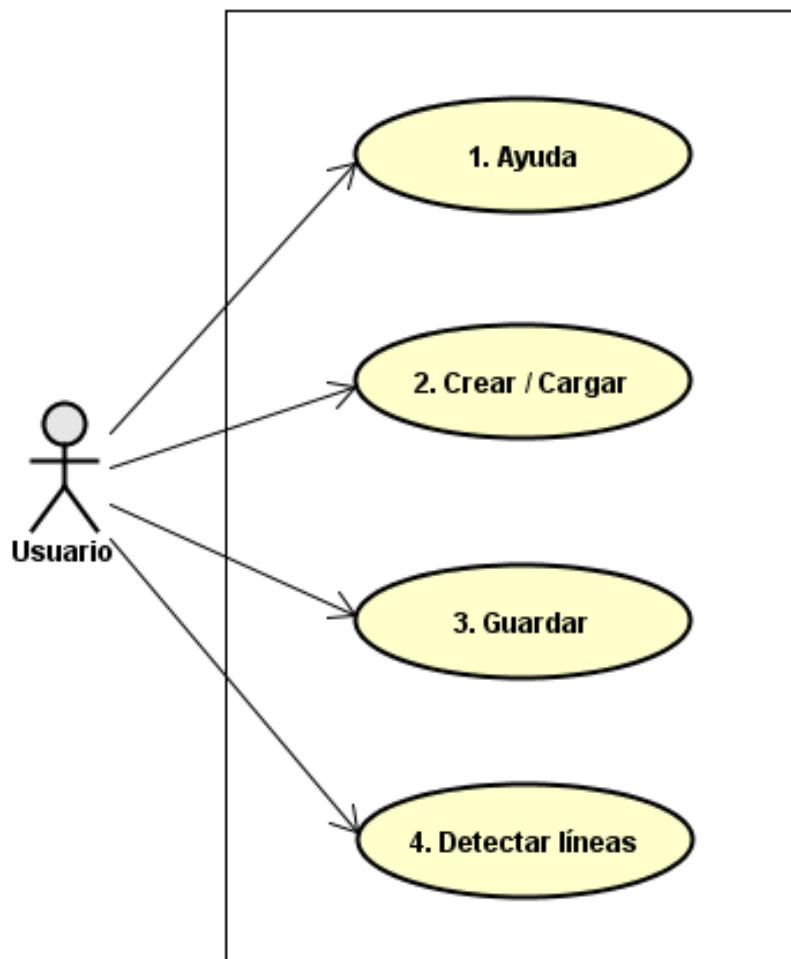


Figura B.1: Diagrama general de casos de uso.

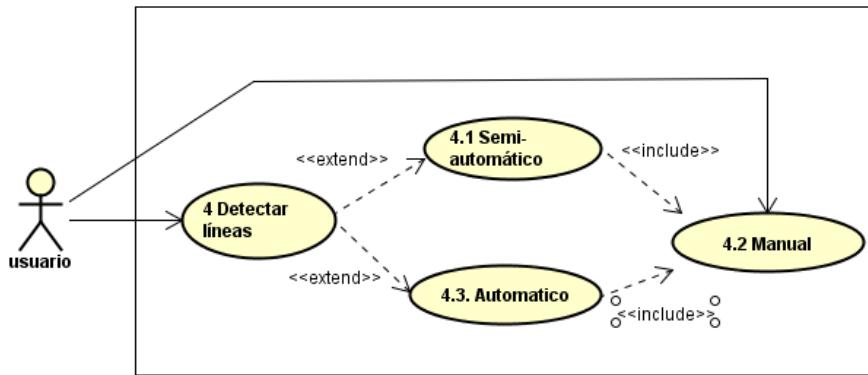


Figura B.2: Diagrama de casos de uso extendido.

Cuadro B.1: Tabla del caso de uso 1

Caso de uso 1	Mostrar ayuda de la aplicación
Versión	1
Autor	Ismael Tobar García
Requisitos	RF-1 RF-1.1 RF-1.2
Descripción	El usuario podrá consultar la ayuda de la aplicación, para aprender o para consultar algún aspecto que considere oportuno.
Precondiciones	No tiene
Acciones	1 Ejecutar la aplicación. 2 Abrir el menú principal de la parte superior. 2.1 Seleccionar de la pestaña desplegable la opción ayuda
Postcondiciones	Se abrirá una ventana emergente con la ayuda.
Excepciones	No pueden saltar
Importancia	Baja

Cuadro B.2: Tabla del caso de uso 2

Caso de uso 2	Mostrar ayuda de la aplicación
Versión	1
Autor	Ismael Tobar García
Requisitos	RF-2 RF-2.1 RF-2.2 RF-2.3
Descripción	El usuario podrá cargar o abrir un proyecto ya existente para su edición.
Precondiciones	Para cargar proyecto: Que exista un proyecto. Para nuevo proyecto: Que exista la imagen a cargar.
Acciones	1 Ejecutar la aplicación. 2 Abrir el menú principal: 2.1 Seleccionar nuevo proyecto. 2.1.1 Buscar la imagen que queremos analizar. 2.1.2 Abrir la imagen seleccionada previamente. 2.2 Seleccionar abrir proyecto: 2.2.1 Buscar el proyecto existente que queremos abrir. 2.2.2 Abrir el directorio previamente seleccionado.
Postcondiciones	Se tiene que crear la ventana con la imagen y las opciones disponibles.
Excepciones	No exista el proyecto seleccionado No sea una imagen válida
Importancia	Alta

Cuadro B.3: Tabla del caso de uso 3

Caso de uso 3	Guardar un proyecto desde la aplicación.
Versión	1
Autor	Ismael Tobar García
Requisitos	RF-3 RF-3.1 RF-3.2
Descripción	El usuario podrá guardar los datos calculados en un proyecto.
Precondiciones	El usuario deberá haber ejecutado alguno de los tres modos disponibles y tener la tabla con estrías que guardar.
Acciones	1 Ejecutar la aplicacion. 2 Cargar o abrir proyecto 3 Ejecutar alguno de los tres modos. 3.1 Modo semiautomático para estrías pintadas. 3.2 Modo automático para imágenes sin pintar. 3.3 Modo manual de detección de estrías. 4 Hacer click sobre el menú o botón de guardar. 5 Seleccionar un directorio sobre el que guardar
Postcondiciones	Se tiene que crear un directorio llamado Proyecto que contenga todos los ficheros que componen a este.
Excepciones	Si tenemos abierto los ficheros CSV del proyecto nos dirá que no puede guardar
Importancia	Alta

Cuadro B.4: Tabla del caso de uso 4

Caso de uso 4	Detectar las estrías de dieta contenidas en la imagen.
Versión	1
Autor	Ismael Tobar García
Requisitos	RF-4 RF-4.1 RF-4.2
Descripción	El usuario podrá detectar las estrías que aparezcan en la imagen para producir las estadísticas y todas las salidas generadas con esto.
Precondiciones	Debe haber una imagen cargada sobre la que calcular las estrías.
Acciones	<p>1 Ejecutar la aplicación.</p> <p>2 Cargar un proyecto o abrir uno nuevo.</p> <p>3 Determinar por que modo vamos calcular las estrías.</p> <p>3.1 Dependiendo del modo debemos seguir unos pasos u otros.</p> <p>3.2 Procederemos a ejecutar el algoritmo correspondiente.</p> <p>3.3 Podremos visualizar que estrías han sido detectadas.</p> <p>3.4 Habilitaremos las funcionalidades de guardar.</p>
Postcondiciones	Debemos haber detectado la mayoría de las estrías que contienen las imágenes.
Excepciones	No puede saltar ninguna ya que lo hemos preparado para esto.
Importancia	Alta

Cuadro B.5: Tabla del caso de uso 4.1

Caso de uso 4.1	Ejecución del modo semi-automático para detectar las estrías ya pintadas.
Versión	1.0
Autor	Ismael Tobar García.
Requisitos	RF-4.1 RF-4.1.1 RF-4.1.2 RF-4.1.3 RF-4.1.4
Descripción	El usuario ejecutar el cálculo de estrías pintadas sobre la imagen de forma automática.
Precondiciones	Tener cargada una imagen
Acciones	<p>1 El usuario deberá seleccionar el color en el que están pintadas las estrías.</p> <p>2 El usuario deberá seleccionar una longitud mínima que ignorara el algoritmo.</p> <p>3 El usuario deberá seleccionar la orientación de la imagen.</p> <p>4 El usuario deberá seleccionar el número de repeticiones que puede ejecutar.</p>
Postcondiciones	Pintar los segmentos detectados en la imagen.
Excepciones	No puede saltar ninguna ya que los botones se activan solo cuando ha seleccionado todos los pasos bien.
Importancia	Alta

Cuadro B.6: Tabla del caso de uso 4.2

Caso de uso 4.2	Ejecución del modo manual para pintar estrías en la imagen.
Versión	1.0
Autor	Ismael Tobar García
Requisitos	RF-4.2 RF-4.2.1 RF-4.2.2 RF-4.2.3 RF-4.2.4 RF-4.2.5
Descripción	El usuario podrá pintar nuevas estrías sobre la imagen o pintar la imagen desde cero.
Precondiciones	Tener una imagen cargada.
Acciones	1 Deberá seleccionar el botón de corregir líneas 1.1 Solo podrá seleccionar puntos dentro del cuadrado. 1.1 Deberá seleccionar el inicio de la estría a pintar. 1.2 Deberá seleccionar el final de la estría a pintar. 2 Deberá hacer click en el botón añadir puntos.
Postcondiciones	Ninguna
Excepciones	No pueden saltar ya que solo podrá seleccionar puntos dentro de la parte de la imagen contenida en el cuadrado
Importancia	Media

Cuadro B.7: Tabla del caso de uso 4.3

Caso de uso 4.3	Ejecución del modo completamente automático para imágenes sin pintar.
Versión	1.0
Autor	Ismael Tobar García
Requisitos	RF-4.3 RF-4.3.1 RF-4.3.2 RF-4.3.3 RF-4.3.4
Descripción	Este modo permitirá al usuario ejecutar el algoritmo para la detección de las estrías en imágenes sin pintar.
Precondiciones	Tener una imagen sin pintar cargada.
Acciones	1 El usuario deberá seleccionar una longitud mínima que ignorara el algoritmo. 2 El usuario deberá ejecutar el algoritmo que detecta las estrías que hay en la imagen. 4 El usuario deberá fijar el cuadrado para quedarse con las estrías detectadas que considere efectivo.
Postcondiciones	Pintar los segmentos detectados en la imagen.
Excepciones	No puede saltar ninguna ya que los botones se activan solo cuando ha seleccionado todos los pasos bien.
Importancia	Alta

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado vamos a explicar que hemos usado para la resolución del problema en cuanto a herramientas de diseño.

Para la elaboración de los diagramas de paquetes y de clases hemos usado UML como metodología y el programa Astah para implementar los diagramas de clases.

Primero, hemos planeado un prototipo de diseño de cada una de las clases, como podemos observar en la figura C.1 el diagrama de clases inicial, a medida que las implementábamos las hemos ido completando, para que se correspondieran en lo que finalmente va a quedar.

C.2. Diseño de datos

En este punto vamos a hacer el diseño inicial de las clases de la aplicación y del diseño de paquetes en el que estarán contenidas.

Diseño inicial

Esta parte se va a corresponder con un diagrama inicial que hemos realizado pero seguramente va a sufrir cambios para ser mas acordes al diseño y cumplir los requisitos que esto conlleva.

Clases

En esta parte vamos a hacer una breve descripción del contenido de cada clase y su función.

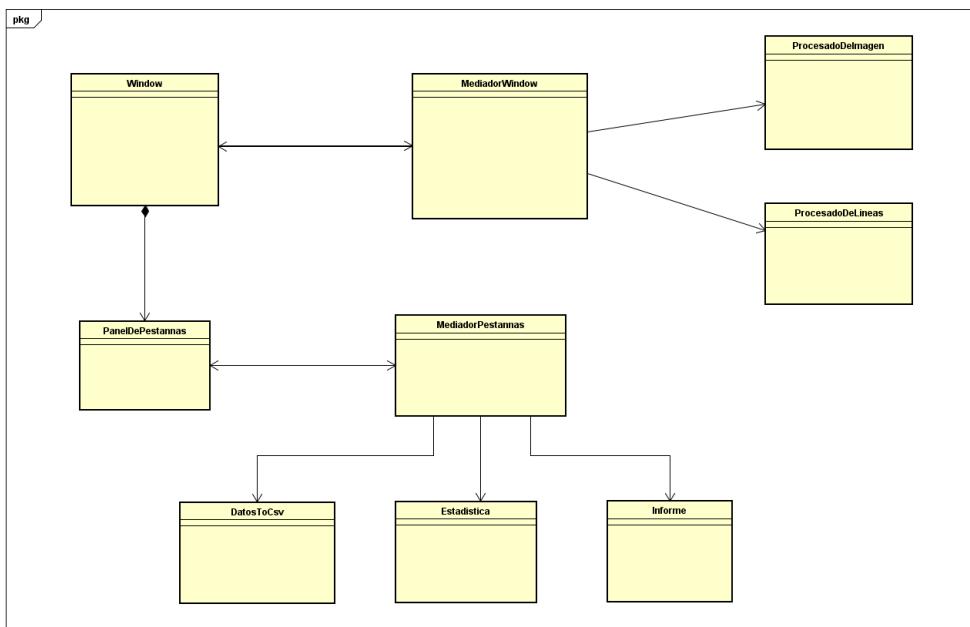


Figura C.1: Diagrama de clases inicial

- **VentanaInicio:** Esta clase simplemente hará de fachada entre la ejecución y el acceso a la aplicación real cuando hayamos elegido una imagen que procesar
- **Window:** Esta clase contendrá la ventana principal, menús, el panel de pestañas y la imagen sobre la que dibujar.
- **PanelDePestannas:** Esta clase contendrá la botonería y la interacción con la imagen a evaluar.
- **MediadorPestannas:** Esta clase va a ser un mediador para las pestañas para deslocalizar código y complejidad.
- **MediadorVentana:** En esta clase va ser un mediador entre la ventana principal y sus métodos para así reducir complejidad en la clase de ventana.
- **ProcesadoDeImagen:** Esta clase contendrá el tratamiento que tiene que llevar la imagen para la extracción de características a evaluar.
- **ProcesadoDeLineas:** Esta clase contendrá el procesado de las líneas obtenidas por el procesado de la imagen y el resultado final que obtendremos.
- **Informe:** Esta clase contendrá la generación del informe (Tabla) de estadísticas en formato L^AT_EX para poder exportar fácilmente a un documento PDF.

- DatosToCsv: Esta clase contendrá el paso de los datos que contiene la tabla a un documento en formato csv del que desde Excel podemos extraer fácilmente los datos.
- Estadistica: Esta clase contendrá el cálculo de los datos estadísticos y la clasificación de las líneas según su orientación.

Diseño Final

Este apartado va a consistir en mostrar los diagramas de clases finales que han salido pero los vamos a dividir en dos partes ya que sino sera un diagrama demasiado grande y así facilitar el entendimiento de ello.

Clases

Los diagramas los podemos visualizar en las figuras C.2 y C.3.

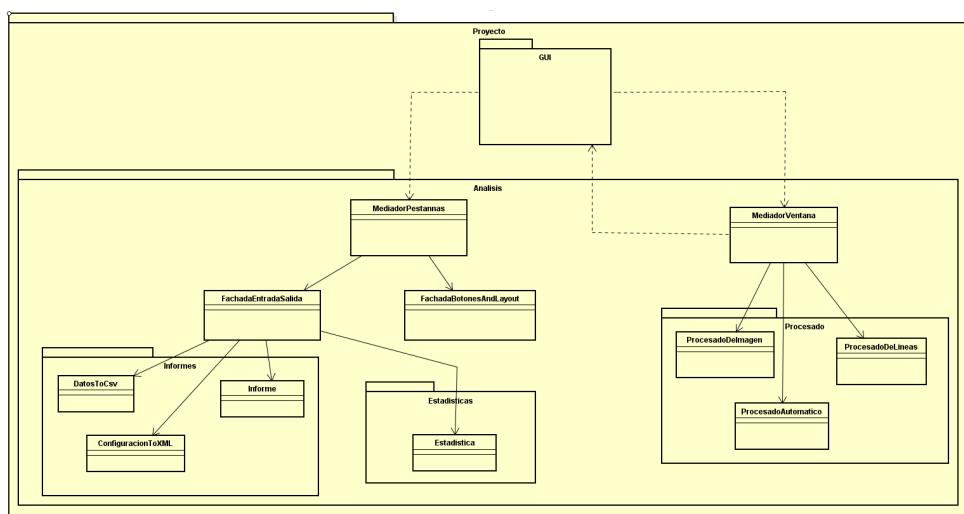


Figura C.2: Diagrama de clases final parte uno.

Las clases que hemos mantenido desde el primer diseño pueden ser encontradas en la sección Diseño inicial. Las clases que hemos añadido nuevas para facilitar el diseño las podemos vamos a explicar a continuación.

- VisorHTML: Esta clase va a servir para mostrar la ayuda en formato HTML ya que la herramienta para implementar la ayuda no estaba disponible para PyQt solo estaba para Qt.
- PintarRectangulo: Esta clase servirá para mostrar el triangulo por la ventana donde queremos evaluar las estrías que aun no están pintadas dentro de la imagen.

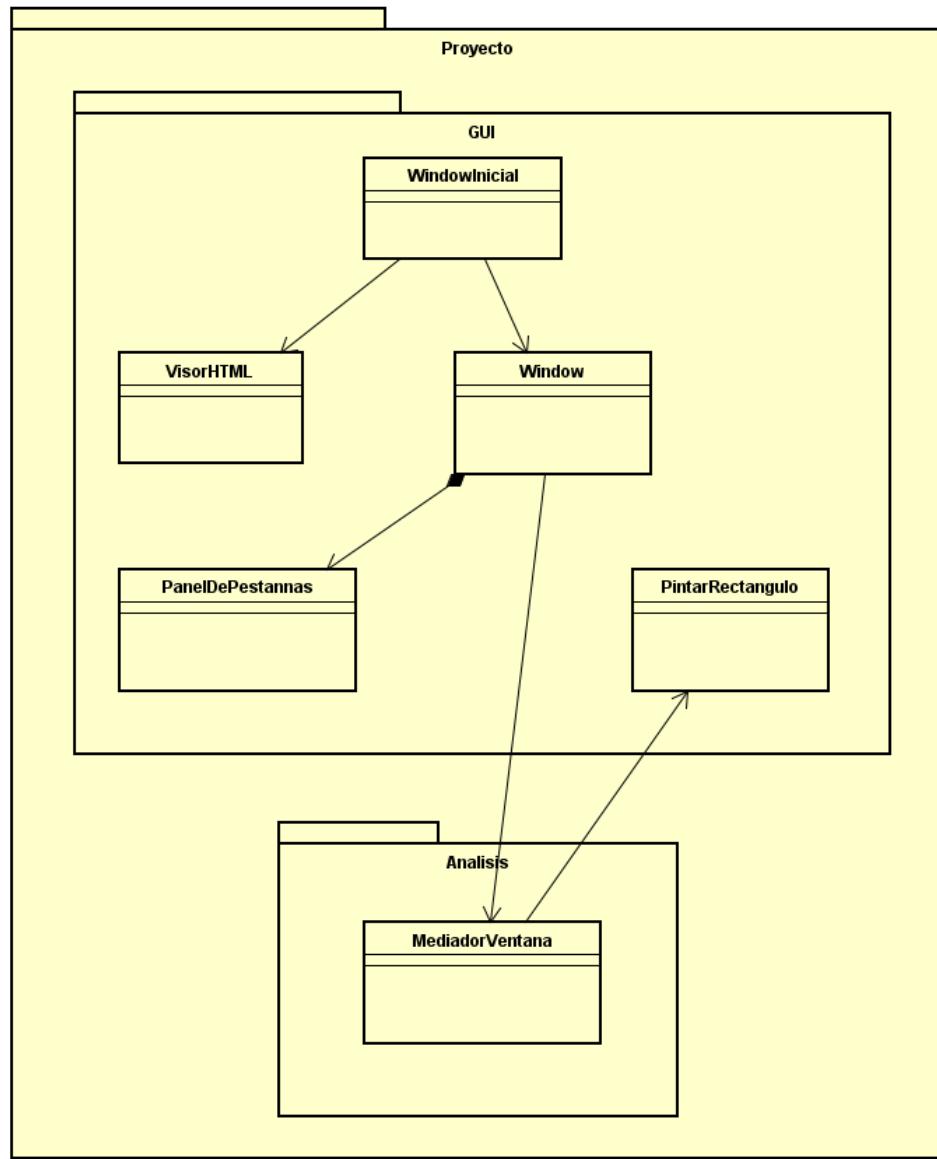


Figura C.3: Diagrama de clases final parte dos.

- **ProcesadoAutomatico:** Esta clase sera la encargada de hacer el procesado para las imágenes que estén sin pintar y se corresponde con el nuevo modo que hemos añadido como extra a la aplicación.
- **FachadaBotonesAndLayout:** Esta clase va a contener las declaraciones e inicializaciones de la botonería y los layout correspondientes al panel de pestañas.
- **FachadaEntradaSalida:** Esta clase se va a encargar de comunicar los

procedimientos de entrada y salida para crear y gestionar los resultados que vamos a guardar en el fichero de proyecto generado por la aplicación.

- **ConfiguracionToXML:** Esta clase se va a encargar de guardar las configuraciones que hemos aplicado a las imágenes para obtener los resultados que hemos conseguido.

Diseño de la Interfaz

Este apartado se va a corresponder con el diseño de la interfaz gráfica que hemos optado desde el primer momento.

Primera versión

Para el desarrollo de la interfaz gráfica pensé en una planificación espacial acorde con los elementos que esta contendría. Un layout que sería muy adecuado podría ser un border layout pero como no existe en PyQt4 lo he tenido que simular gracias a apilar layouts de otros tipos. Consiguiendo tener una botonería arriba del todo y dos columnas debajo claramente diferenciadas en la que en una estuviera la imagen que vamos a mostrar y en la otra las funcionalidades, como se ilustra en la imagen C.4.

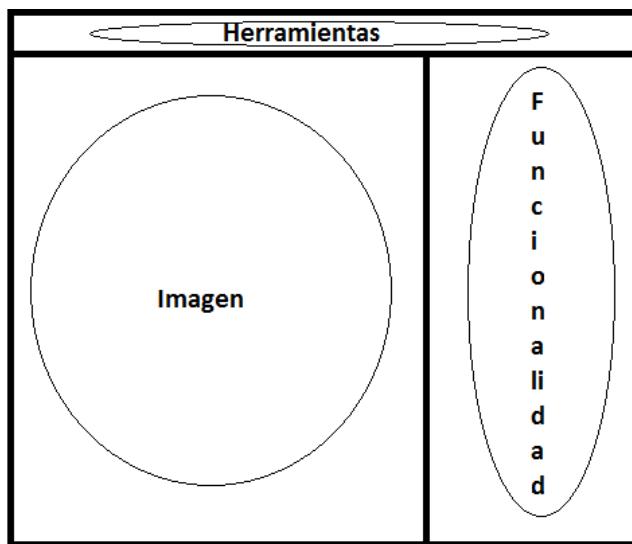


Figura C.4: Diseño de la interfaz de usuario

Versión a evaluar

En otro Sprint del proyecto lo que he usado han sido pestañas para así tener en la zona de funcionalidades los modos de trabajo de la aplicación claramente separados, como podemos observar en la figura C.5.

- Detección líneas rojas.
- Corrección y pintar líneas.
- Automático.

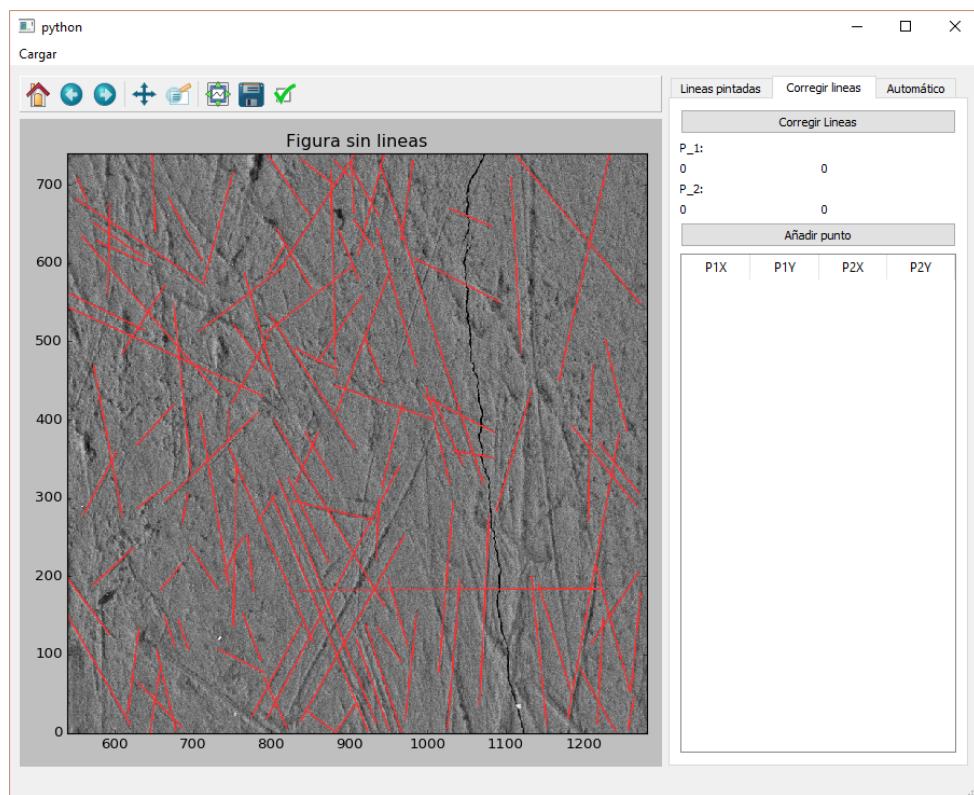


Figura C.5: Diseño de la interfaz de usuario

Barra de herramientas Es una sección de la interfaz que nos va permitir de momento la carga de imágenes para su procesado [C.6](#).

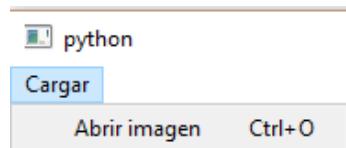


Figura C.6: Barra de herramientas de la aplicación

Barra de herramientas de la imagen Es una sección de la interfaz que nos permitirá manipular la imagen para realizar estas acciones, como podemos observar en la figura [C.7](#).

- Volver al principio.
- Retroceder un paso.
- Avanzar un paso.
- Desplazar la imagen.
- Aumentar la región que seleccionemos.
- Configurar la región, bordes, etc.
- Guardar la imagen con su escala.
- configurar el tamaño y numeración de los ejes.
- Coordenadas actuales del ratón.
- Niveles de color de los canales R G B del espacio RGB.



Figura C.7: Barra de herramientas de la imagen

FigureCanvas de la imagen Es la región donde se muestra la imagen que va a ser ligeramente mas grande que la parte de las pestañas [C.8](#).

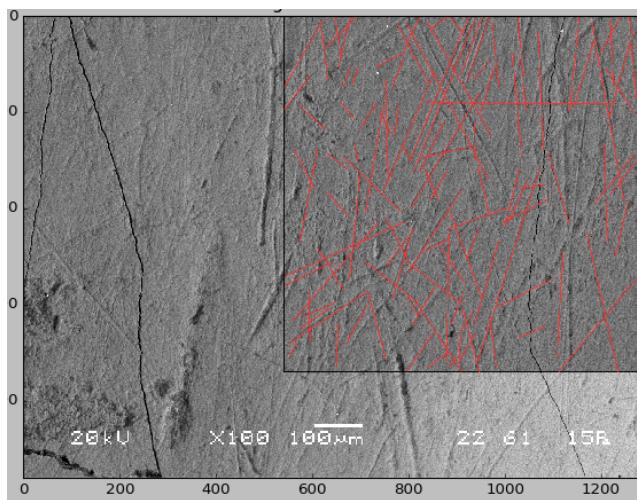


Figura C.8: FigureCanvas de la imagen

Pestañas Sección de la imagen donde estarán implementadas las funcionalidades para visualizar las líneas que son detectadas por el algoritmo. se ilustra en la figura [C.9](#).

- El modo primero es la detección de los surcos en rojo en los dientes
- El modo segundo es la corrección/detección manual de las líneas por una persona y quedando reflejadas en la imagen.
- El modo tercero es la ultima parte del proyecto que consistirá en hacer todo el proceso anterior de forma automática.

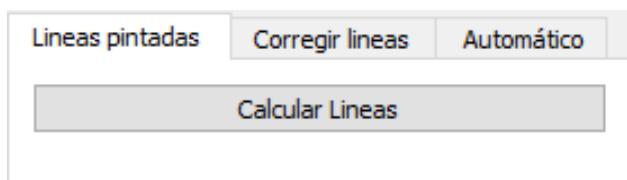


Figura C.9: Pestañas

Diseño salida CSV

Para las salida de datos en formato CSV hemos optado por usar los siguientes formatos.

- Estadísticas:
 - Tipo: Nos indica la orientación de la estría.
 - Numero: Nos indica cuantas estrías de esa orientación hay.
 - MediaLon: Nos indica la media de las estrías de esa orientación.
 - DesviacionTip Nos indica la desviación típica de las estrías con dicha orientación.
- Salida Estrias:
 - Línea: Aparecerán las coordenadas de los dos puntos que forman el segmento.
 - Ángulo: Aparecerá el ángulo que forma la estría con el eje x.
 - Tamaño: Aparecerá la longitud de la estría.
 - Tipo: Aparecerá la orientación de la estría

C.3. Diseño procedimental

En este apartado se van a explicar los pasos que siguen los distintos modos de la aplicación.

- Modo semiautomático: La finalidad de este modo de trabajo en nuestra aplicación, servirá para detectar las estrías que previamente se han pintado en alguna imagen. Para extraer las características de estas.
 - Comienza cuando el usuario cargue la imagen en la aplicación.
 - Posteriormente a la carga, el usuario deberá elegir los parámetros que puede modificar, para la ejecución del algoritmo.
 - Una vez seleccionados los parámetros, podremos continuar mandando ejecutar a la aplicación el algoritmo, para la detección de las estrías pintadas.
 - Este nos va a devolver un conjunto de segmentos, que se corresponderán con las estrías que contiene la imagen pintadas, las añadirá a la tabla donde podremos editarlas o añadir algunas que podría no haber detectado dicho algoritmo.
 - por ultimo podremos guardar las estadísticas, las imágenes original, pintada, longitudes de los segmentos, ángulos, orientaciones, la tabla rellena en formato LATEX, para incluir en futuros informes y el fichero de configuración del proyecto.
- Modo automático: La finalidad de este modo de trabajo de la aplicación, consiste en detectar las estrías de dieta de una imagen en blanco desde el cero. Para extraer características de dichas estrías.
 - Primero el usuario deberá cargar una imagen, sin que tenga las estrías pintadas, en la aplicación.
 - posteriormente, deberá elegir los parámetros disponibles para el modo automático, este consistirá en seleccionar la orientación y la longitud mínima que ignora el algoritmo.
 - Una vez que hayan sido seleccionados se habilitara la función de detección de las estrías. Al ejecutar este algoritmo nos mostrara todas las estrías que detecto en la imagen y nos permitirá mover la región de interés que queremos analizar.
 - Al mover la región de interés que nos conviene analizar y fijarlo, desechará las estrías que no estén contenidas y también los trozos de las detectadas que se salgan fuera de la región.
 - Por último, el paso anterior nos mostrara en la tabla de edición de los segmentos las estrías que ha detectado, pudiendo modificar o eliminar aquellas que nosotros no consideremos útiles. También nos permitirá guardar el proyecto tal y como estaba para obtener las estadísticas y demás datos relevantes de las estrías detectadas.
- Modo manual: La finalidad de este modo no es otra mas que poder incluir el factor humano en la aplicación y corregir los posibles errores o modificar ciertas estrías que nuestro algoritmo ha detectado.

- Despues de cargar cualquier imagen en la aplicación podremos acceder a las funciones de este modo.
- Para pintar desde cero una imagen, sin que tenga estrías, podemos hacer clic en el la pestaña de corregir lineas y mover la región de iteres por la imagen en la zona que queremos pintar.
- Dentro de la pestaña, anteriormente mencionada, en el botón con el mismo nombre al hacer clic, la primera vez fijaremos el cuadrado y procederemos a seleccionar el origen y el final de la estría que queremos detectar, para añadirla simplemente tendremos que hacer click sobre el botón añadir punto y esta quedará incluida en la tabla. Para pintar nuevas estrías en una imagen que ya contenga estrías pintadas procederemos de forma similar, pero no podremos mover la región de interés.
- Una vez que tengamos alguna estría pintada en nuestra imagen podremos proceder a guardar las estadísticas y todos los datos relevantes del proyecto, de forma similar a los otros modos.

C.4. Diseño arquitectónico

En esta sección del anexo de diseño, vamos a explicar y mostrar de forma general como esta diseñado el proyecto, en cuanto a la distribución de paquetes y los patrones de diseño que hemos aplicado.

Diseño de paquetes inicial

En este punto vamos a hacer una breve descripción de cada paquete y de su contenido dentro de la aplicación.

- Gui: Este paquete contendrá todos los elementos gráficos de la aplicación y todos lo que se corresponde con interacción con el usuario.
 - Mediadores: Este paquete va a contener los mediadores de la interfaz para deslocalizar código y complejidad en esta.
- Código: Este paquete contendrá todas las clases de cálculo que necesitará la aplicación:
 - Estadísticas: Este paquete contendrá las clases del cálculo de estadísticas.
 - Informes: Este paquete contendrá la generación del informe LATEX
 - Procesado: Este paquete se va a encargar tanto del procesado de la imagen como del procesado de los segmentos extraídos hasta conseguir los segmentos finales.

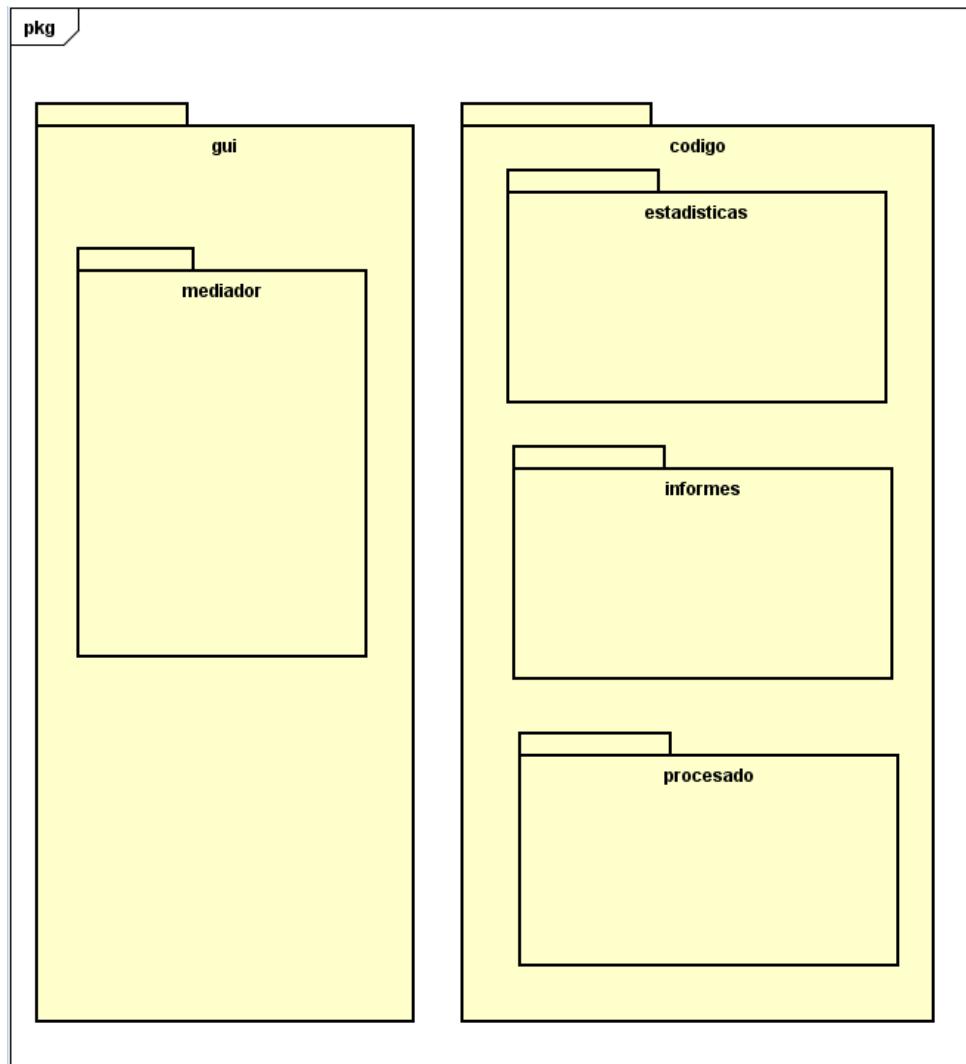


Figura C.10: Diagrama de paquetes inicial

Los paquetes que ya hayan sido explicados en el diagrama inicial no van a aparecer su explicación en este diagrama. **Src**

- Proyecto: Este paquete contendrá todos los fuentes del proyecto menos los main.
 - Análisis:
 - Diccionario: Este paquete tendrá las clases para poder internacionalizar la aplicación.
 - GUI: Contendrá los elementos que componen la interfaz.

- Test: Este paquete contendrá los test que hemos realizado sobre el código.
 - Calidad: Contendrá las fuentes que testan la calidad de la detección en una imagen.

Diseño de paquetes Final

Este apartado se va a corresponder con la descripción de los paquetes que han cambiado desde el diseño inicial de la aplicación, también mostraremos el diagrama actualizado con los paquetes que hemos incluido. Como podemos observar en la figura C.11

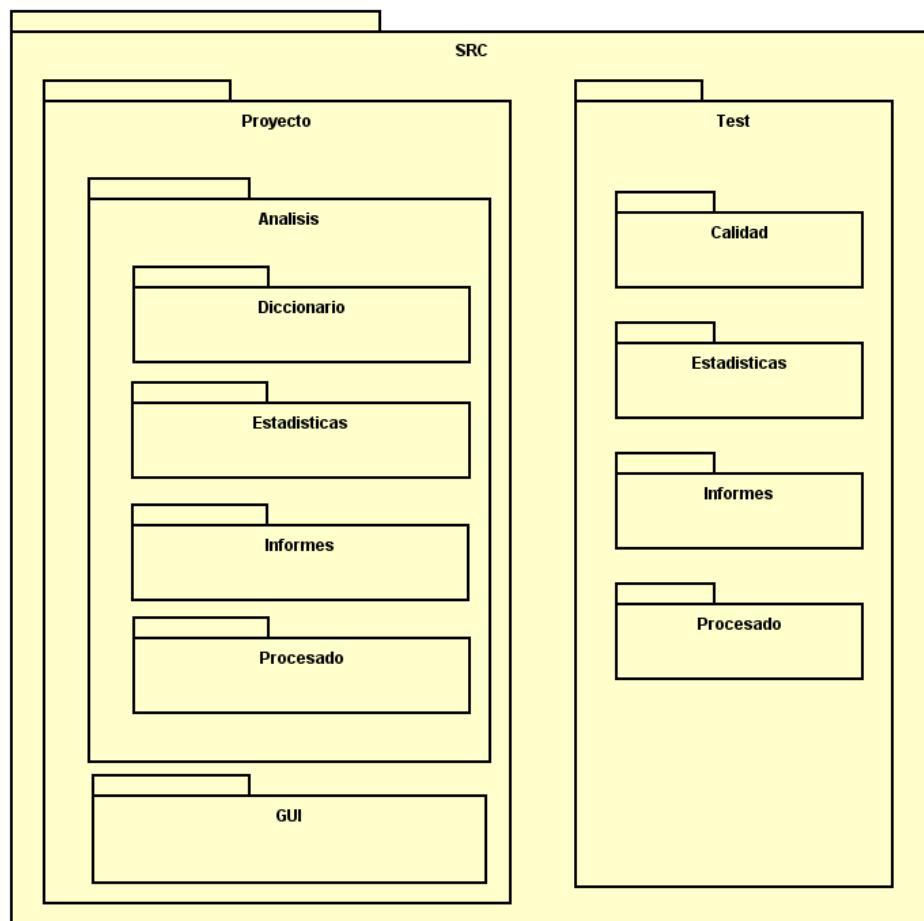


Figura C.11: Diagrama de paquetes inicial

Patrones de diseño

En este apartado vamos a usar de los patrones de diseño que hemos utilizado en nuestra aplicación.

El patrón más importante y con el que vamos a empezar es el medidor que consigue hacer que la interfaz esté limpia simplemente con las declaraciones de las variables que más adelante va a utilizar. La base de los patrones de diseño, está explicado en su libro más significativo, el cual es [?].

- **Mediador [?]:** Se utiliza para encapsular las interacciones entre los objetos de la interfaz por lo que se clasifica como patrón de comportamiento. Como dato curioso, es de los pocos patrones que permiten la bidireccionalidad de comunicación entre las clases, por eso es el patrón idóneo para la interacción de las clases de la interfaz gráfica. Aplicando este patrón reducimos la dependencia del código por lo que de esta forma las interfaces no ven lo que tienen por debajo y reduce la complejidad.

Obtenemos:

- Conseguir desacoplamiento.
 - Simplificar y aclarar la comunicación.
 - Centralizar el control.
 - Interfaz simple para un sistema complejo.
- **Comando [?]:** Se utiliza para encapsular el contenido de una función, así ni el emisor de la operación ni el receptor, sabe que hay por debajo, pero si poder acceder a las funciones que hay.

Obtenemos:

- Permitir la parametrización.
 - Visualizar que órdenes se ejecutan en cada paso.
 - Construir operaciones complicadas a partir de otras más sencillas.
 - Permitir saber qué hacer para deshacer una operación.
- **Fachada [?]:** Es un patrón de diseño estructural, sirve para reducir la complejidad con la división en clases más pequeñas y reduciendo sus dependencias.

Obtenemos:

- Separar en niveles la aplicación.
- Reducir su complejidad.

- Desacoplar el código.
- Interfaz simple para un sistema complejo.
- Potable y reutilizable.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Esta sección está dirigida a otros desarrolladores, de modo que puedan leer continuar el proyecto y entenderlo. En él se describen en detalle el funcionamiento del proyecto y los aspectos que podrían mejorarse o modificarse en el futuro.

D.2. Estructura de directorios

Ejecutables:

- EjecutarGui: Este fichero se corresponde con el ejecutable para lanzar la aplicación en modo GUI.
- EjecutarTest: Este fichero se corresponde con el ejecutable para lanzar los test de la aplicación en modo consola.
- Documentar: Este fichero se corresponde con el ejecutable para documentar la aplicación de forma recursiva y crear los ficheros HTML correspondientes.

Documentación:

En esta carpeta estará contenida la documentación en formato HTML.

- pydoc: Esta carpeta se corresponderá con la documentación de pydoc

Src:

Esta carpeta contendrá todos los ficheros de código fuente del proyecto, el OCR, y los test.

Proyecto:

Esta carpeta contendrá todos los ficheros de código fuente del proyecto, todos los módulos o paquetes.

Código

- Estadísticas.
 - Estadística.
- Informes:
 - ConfiguracionToXML.
 - DatosToCsv.
 - Informe.
 - InGuardarDatos
- Procesado:
 - ProcesadoAutomatico.
 - ProcesadoDeImagen
 - ProcesadoDeLineas
- Diccionario:
 - Diccionario
 - DiccionarioING
- Gui:
 - Fachadas:
 - FachadaBotonesAndLayaout.
 - FachadaEntradaSalida.
 - Mediadores:
 - MediadorPestannas.
 - MediadorVentana.
 - PanelDePestannas
 - PintarRectangulo
 - VentanaInicio
 - VisorHtml
 - Window

Tesseract:

En esta carpeta contendremos el ejecutable del OCR junto con sus ficheros de configuración para poderlo ejecutar.

Test:

En esta carpeta contendremos los test del código para comprobar que funcionan correctamente.

Código:

- Estadísticas:
 - TestEstadistica
- Informes:
 - TestConfiguracionToXML
 - TestDatosToCsv
 - TestInforme
- Procesado:
 - TestProcesadoDeImagen
 - TestProcesadoDeLineas

D.3. Manual del programador

En esta sección vamos a describir como se han programado las partes mas importantes que son las que mas nos interesan.

Procesado

Para programar el procesado hemos encadenado sucesivos pasos, que dividiremos en tres secciones.

- ProcesadoDeImagen: Este se corresponde con el procesado de la imagen para la extracción de las características.
 1. Hemos calculado de la imagen la distancia a un color: Se corresponde con el que estén pintadas los segmentos, lo selecciona el usuario. Evitamos cualquier color que hiciese que falle el algoritmo, filtrando la selección por el canal del modelo de color HSV, que indica la saturación. En nuestra imagen lo único que tiene una saturación

alta son los segmentos pintados ya que la imagen esta en escala de grises.

2. Binarizamos la imagen con la distancia al color para obtener la máscara con los objetos que queremos detectar y reducimos el grosor de estos.
 3. Sobre la máscara reducida calculamos los segmentos por la transformada de Hough.
- ProcesadoDeLineas: Este módulo contiene el procesado de las lineas, detectadas en el ultimo paso del punto anterior.
 1. Primero de todo, lo que tenemos no son segmentos completos, sino subsegmentos que forman los segmentos reales. Tenemos que unirlos aquellos que sean muy similares, caracterizados por distancia y ángulo.
 2. Para realizar la unión lo primero que haremos sera ir añadiendo, un camino entre los dos segmentos, a un grafo, de aquellos que cumplan lo anterior.
 3. Obtendremos las uno componentes del grafo, se explica en las memorias pero podemos encontrarlo aquí también [?], que se corresponden, con los clusters, que son los segmentos buscados.
 - ProcesadoAutomatico: Este apartado aunque lleva proceso similares a los anteriores no puede ir junto ya que necesita otros pasos intermedios.
 1. Usaremos, el detector de bordes que hemos implementado.
 2. Ecualizaremos la imagen para distribuir su histograma.
 3. Calculamos los autovectores de la matriz Hessiana y nos quedamos con los autovectores largos.
 4. Binarizamos la imagen y procesamos dicha imagen. Queda reflejado en anexo F **F**.

GUI

- Imagen y líneas: Para pintar las imágenes, hemos usado el backend de Matplotlib, que nos muestra la imagen sobre unos ejes de coordenadas, que nos indican e informan sobre las coordenadas de cada pixel, gracias a esto y a los métodos de conexión sobre los eventos, hemos podido simplificar al máximo la obtención de los puntos, la forma de pintar los segmentos detectados y la interacción de usuario.
- OCR: Para leer la referencia de la imagen hemos usado un OCR conocido que se llama Tesseract a este le pasamos el recorte de la imagen que contiene la referencia y nos devuelve el numero que contiene.

D.4. Compilación, instalación y ejecución del proyecto

Compilación:

En Python no hace falta compilar el proyecto ya que es un lenguaje interpretado. Necesitaremos únicamente tener Python instalado, a través de Miniconda que es una distribución Python.

Instalación:

Para poder ejecutar deberemos instalar Miniconda en C.Users.TuUsuario, Es el directorio predefinido por Miniconda, una vez que lo tengamos instalado podremos proceder a hacer doble clic sobre el ejecutable EjecutarGui.bat que instalará las dependencias a las librerías que necesitamos para su ejecución.

Ejecución del proyecto:

Para la primera ejecución, si no tenemos las librerías instaladas, al hacer doble clic sobre EjecucionGui.bat, tardará un rato en descargarlas y crear el entorno virtual de Miniconda con únicamente las librerías que se usan. No obstante, en lo sucesivo únicamente ejecutara la aplicación.

Si la descarga de las librerías se interrumpe por una caída de la red u otro problema, nos dará un error. Pero si volvemos a ejecutar se iniciará donde se paró la descarga, es decir no perdemos lo que ya haya conseguido descargar.

Conclusiones:

Hemos seguido otras formas de conseguir crear un entorno de ejecución, hemos intentado crear el ejecutable como se detalla en las memorias pero no hemos sido capaces, por problemas de versiones, aun no están disponibles.

Python es un lenguaje interpretado y necesitamos tener Python 3.5. junto con las librerías usadas, pero instalar una a una dichas librerías es un trabajo complejo, para un usuario. Por eso para facilitar la instalación de Python hemos optado por usar Miniconda, es una distribución que facilita dicha instalación pero sin ninguna librería instalada, por lo que es mas ligero, 50 Megas, frente a Anaconda, cerca de 500 Megas. Quedando a nuestra disposición indicar que librerías instalar.

En la primera ejecución del código nos va a descargar e instalar todas las librerías que necesitamos.

D.5. Pruebas del sistema

En esta sección vamos a informar como ejecutar los test que están ubicados en la carpeta Test dentro de la carpeta src del proyecto.

Test Python:

Dichos test están escritos en Python usando la librería unittest.

Hay muchas comprobaciones sobre las funciones de cálculo y aquellas que escriben y leen ficheros, pero para la parte de la interfaz gráfica no hay hechas pruebas ya que no se puede controlar del todo esta parte, pero ha sido probada por mí y todos los aspectos añadidos y no produce fallos conocidos.

Para ejecutarlos basta con ejecutarlos desde Eclipse, el main de los test mainTest.py o también si preferimos podemos ejecutarlos desde la terminal con el mismo fichero. Otra opción para ejecutar los test es dar doble clic sobre el ejecutable EjecutarTest que está en la carpeta de los ejecutables.

Aparte de los datos proporcionados en la ejecución hemos puesto una salida informativa para cada test y las partes que comprueba.

Test de calidad

En este apartado vamos a mostrar las medidas que vamos a usar para conseguir la calidad de las imágenes y explicar las formulas.

- True positive rate (TPR): Mide la proporción de los elementos que se han considerado como positivos y son positivos.

$$TPR = \frac{TP}{TP + FN}$$

- True negative rate (TNR):Mide la proporción de los elementos que se han considerado como negativos y son negativos.

$$TNR = \frac{TN}{FP + TN}$$

- Positive predictive value (PPV): Mide la proporción de elementos clasificados como verdaderos positivos entre los positivos.

$$PPV = \frac{TP}{TP + FP}$$

- Negative predictive value (NPV): Mide la proporción de elementos clasificados como verdaderos negativos entre los negativos.

$$NPV = \frac{TN}{TN + FN}$$

- False positive rate (FPR): Mide la proporción de los elementos que se han considerado como falsos positivos entre los que son negativos reales.

$$FPR = 1 - TNR$$

- False discovery rate (FDR): Mide la proporción de los elementos que se han considerado como falsos positivos entre los que son positivos de verdad.

$$FDR = 1 - PPV$$

- Miss rate or false negative rate (FNR): Mide la proporción de los elementos que se han considerado como falsos negativos entre los que son positivos.

$$FNR = 1 - TPR$$

- Accuracy (ACC): Mide el error observable.

$$ACC = \frac{TP + TN}{P + N}$$

- Valor F1: Este valor muestra la media armónica entre la precision y la sensibilidad del modelo.

$$F1 = \frac{2TP}{2TP + FP + FN}$$

- Matthews correlation coefficient (MCC): Esta medida se usa en Maching Learning para medir la calidad de la clasificación en dos clases.

$$F1 = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}}$$

- Informedness or Bookmaker Informedness (BM): Esta medida informa de la generalización de un problema multi-clases y estima la probabilidad de decisión.

$$BM = TPR + TNR - 1$$

- Markedness (MK): Esta es la medida de cuando una variable es marcada como predicto o causa posible de otra.

$$MK = PPV + NPV - 1$$

		Predicted condition	
		Positive prediction	Negative prediction
True Condition	Positive class	True positive (TP)	False negative (FN)
	Negative class	False positive (FP)	True negative (TN)

Cuadro D.1: Estructura de la matriz de confusión.

Cuadro D.2: Tabla de resultados matriz de confusión

Prueba	True positive	False negative	False positive	True negative
Figura D.1	36136	3840	3685	1185139
Figura D.2	31670	844	6514	1189772
Figura D.3	23114	8238	7103	1190345
Figura D.4	30284	1773	14753	1181990

Pruebas

Aquí van a aparecer una muestra de las pruebas que hemos realizado sobre la aplicación para medir la calidad de la detección.

Para las pruebas vamos a optar por construir una matriz de confusión sobre los resultados, la tabla mostrada a continuación, sera el modelo que vamos a seguir [D.1](#).

Prueba 1: La matriz de confusión obtenida para esta prueba la podemos observar en la tabla [D.2](#). Las imágenes original, como muestra la figura [D.1a](#), y calculada, como muestra la figura [D.1b](#), las podemos observar en la figura [D.1](#). Los resultados de las medidas de calidad se muestran en la tabla [D.3](#).

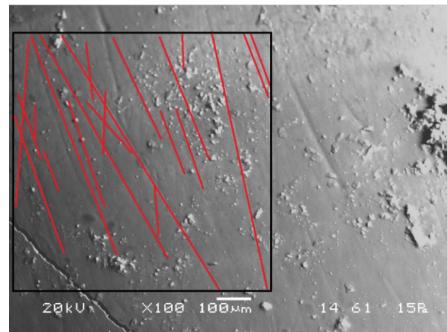
Prueba 2: La matriz de confusión obtenida para esta prueba la podemos observar en la tabla [D.2](#). Las imágenes original, como muestra la figura [D.2a](#), y calculada, como muestra la figura [D.2b](#), las podemos observar en la figura [D.2](#). Los resultados de las medidas de calidad se muestran en la tabla [D.3](#).

Prueba 3: La matriz de confusión obtenida para esta prueba la podemos observar en la tabla [D.2](#). Las imágenes original, como muestra la figura [D.3a](#), y calculada, como muestra la figura [D.3b](#), las podemos observar en la figura [D.3](#). Los resultados de las medidas de calidad se muestran en la tabla [D.3](#).

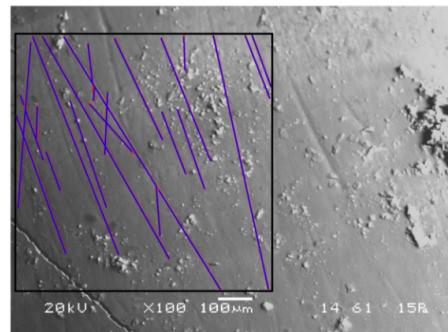
Prueba 4: La matriz de confusión obtenida para esta prueba la podemos observar en la tabla [D.2](#). Las imágenes original, como muestra la figura [D.4a](#), y calculada, como muestra la figura [D.4b](#), las podemos observar en la figura [D.4](#). Los resultados de las medidas de calidad se muestran en la tabla [D.3](#).

Cuadro D.3: Resultados medidas de calidad

Prueba	TPR	TNR	PPV	NPV	FPR	FDR	FNR	ACC	F1	MCC	BM	MK
Figura D.1	0.90	0.99	0.99	0.99	0.003	0.09	0.09	0.99	0.91	0.90	0.9	0.90
Figura D.2	0.97	0.99	0.83	0.99	0.17	0.17	0.02	0.99	0.90	0.90	0.83	0.82
Figura D.3	0.74	0.99	0.76	0.99	0.01	0.24	0.26	0.99	0.75	0.74	0.73	0.76
Figura D.4	0.94	0.99	0.67	0.99	0.01	0.33	0.05	0.99	0.78	0.79	0.93	0.67

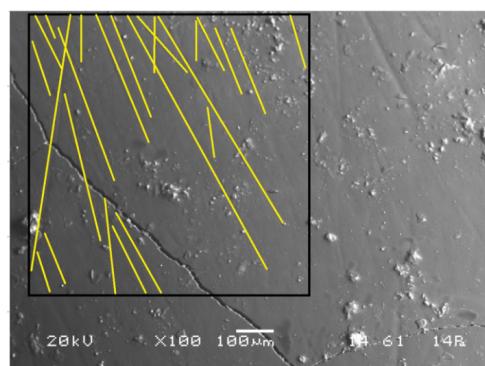


(a) Imagen original prueba 1.

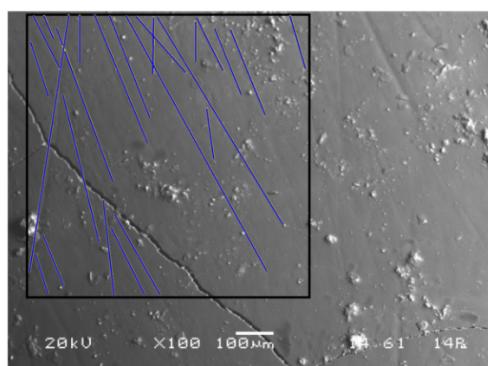


(b) Imagen calculada prueba 1.

Figura D.1: Imágenes prueba 1

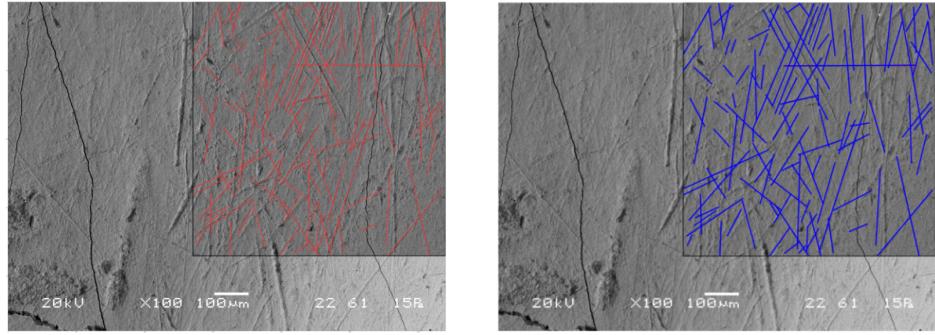


(a) Imagen original prueba 2.



(b) Imagen calculada prueba 2.

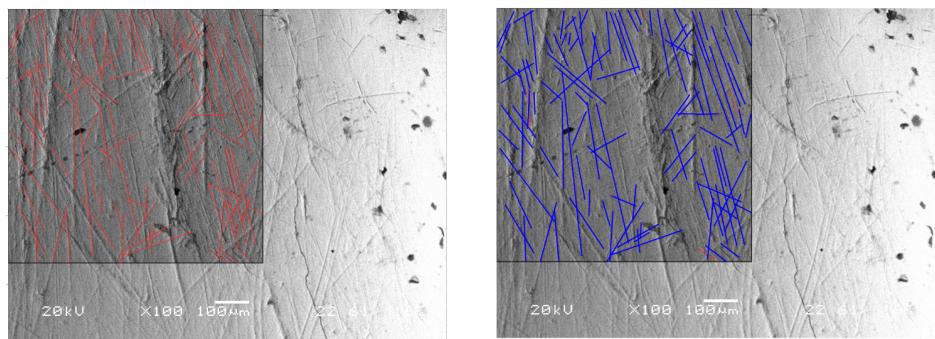
Figura D.2: Imágenes prueba 2



(a) Imagen original prueba 3.

(b) Imagen calculada prueba 3.

Figura D.3: Imágenes prueba 3



(a) Imagen original prueba 4.

(b) Imagen calculada prueba 4.

Figura D.4: Imágenes prueba 4

Apéndice E

Documentación de usuario

E.1. Introducción

Esta sección se centra en la explicación y demostración de cómo se debe utilizar la aplicación, de forma intuitiva y sencilla.

E.2. Requisitos de usuarios

Para que un usuario pueda ejecutar y usar nuestra aplicación, deberá tener instalado Python 3.5 en su máquina, aunque para facilitar al usuario no tener que saber como descargar las librerías o dependencias, hemos implementado un script que funcionará teniendo instalado Miniconda para Python 3.5, explicaremos su uso en la instalación.

- Tener una distribución Windows instalada.
- Python 3.5: Para poder ejecutar necesitaremos tener Python 3.5.
 - Miniconda para Python 3.5:
Se puede descargar a través del siguiente enlace <http://conda.pydata.org/miniconda.html> y siguiendo las instrucciones a través de este otro enlace <http://conda.pydata.org/docs/install/quick.html#windows-miniconda-install>.
Es una distribución que incluye Python y facilita el uso de este lenguaje y la instalación de multitud de librerías.
Sera necesario tenerlo instalado en la carpeta principal de nuestro usuario C:\Users\NombreTuUsuario
- Tener el proyecto descargado o clonado con los fuentes:
Se puede descargar a través del siguiente enlace https://github.com/Itg0001/TFG_DietaPorDientes.git

E.3. Instalación

Una vez que tengamos Miniconda para Python 3.5 y el proyecto descargado o clonado, podremos proceder a la instalación del mismo, siguiendo los pasos descritos a continuación:

- Primero, si hemos descargado el proyecto lo tendremos en formato .zip deberemos descomprimirlo en la carpeta deseada.
 - Una vez descomprimido dentro del directorio del proyecto estará ubicado un ejecutable para Windows .bat llamado «EjecutarGui.bat».
- Procederemos a hacer doble clic sobre este fichero ejecutable y se abrirá una terminal, la primera vez que lo hagamos, tardará un rato, porque descarga las librerías necesarias para la ejecución del mismo.
- Una vez terminado el proceso anterior, se abrirá la aplicación.

Possibles errores

En caso de que al instalar o descargar los paquetes salte algún error esto, puede deberse a que se ha perdido la conexión a Internet y no puede descargar los paquetes necesarios. Para solucionarlo volvemos a ejecutar y comenzará donde se quedó al caerse la red. No perdemos el tiempo usado hasta ese momento.

Otro fallo común es que no se haya instalado la distribución de Miniconda sobre el directorio indicado: C:\Users\NombreTuUsuario. Para solucionar este problema deberemos instalarlo sobre este directorio.

En caso de que surja algún otro problema contactar con el desarrollador a través del siguiente e-mail: itg0001@alu.ubu.es

E.4. Manual del usuario

Una vez tengamos todo bien configurado y la aplicación se ejecute correctamente, deberíamos tener esta ventana inicial, como se ve en la figura E.1.

Una vez llegados a este punto tendremos varias opciones para editar o calcular las estrías de una imagen, dependiendo, si la imagen está en blanco o si las estrías ya están pintadas sobre ella, tendremos las siguientes opciones:

- Abrir imagen E.4:
- Cargar proyecto E.4:

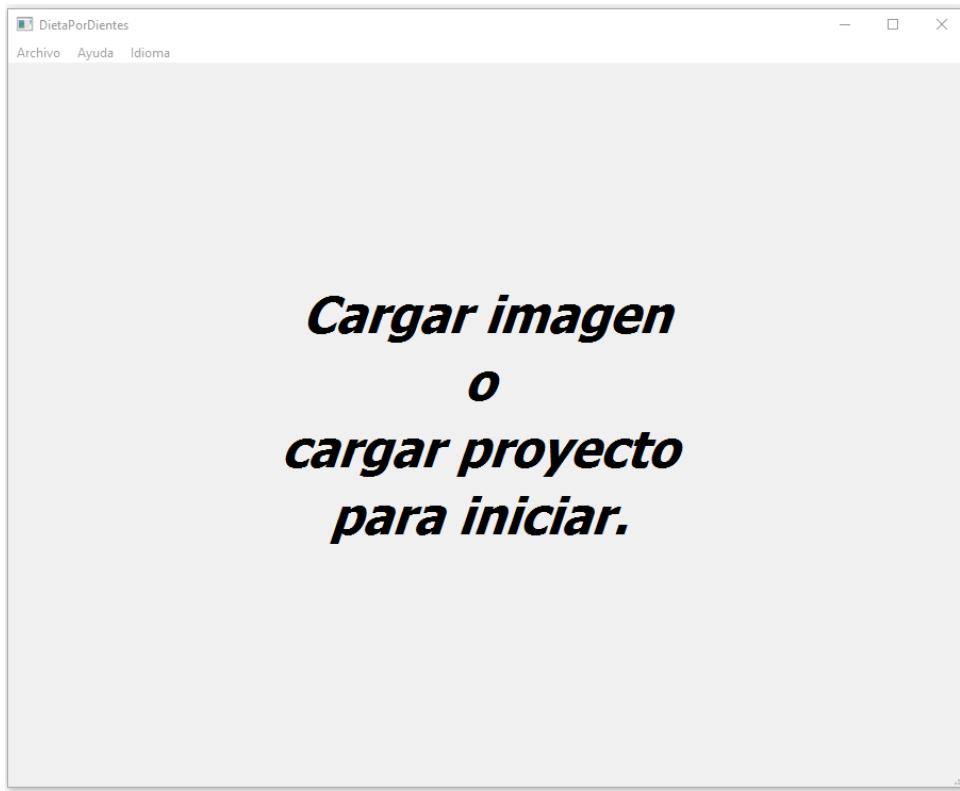


Figura E.1: Ventana inicial de la aplicación

Abrir imagen

En esta sección se explicará como cargar una imagen en nuestra aplicación, para empezar un proyecto desde cero.

Deberemos seleccionar la opción de Archivo > Nuevo Proyecto. Como se puede ver en la imagen E.2

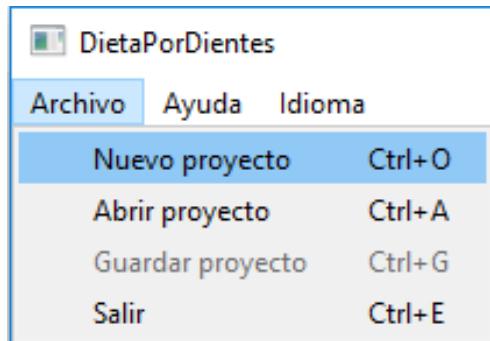


Figura E.2: Opción para cargar una imagen

Una vez que hagamos click en este punto, se nos muestra una ventana para elegir ficheros, en el cual debemos explorar hasta la carpeta donde estén las imágenes que queremos analizar. Como podemos observar en la figura E.3

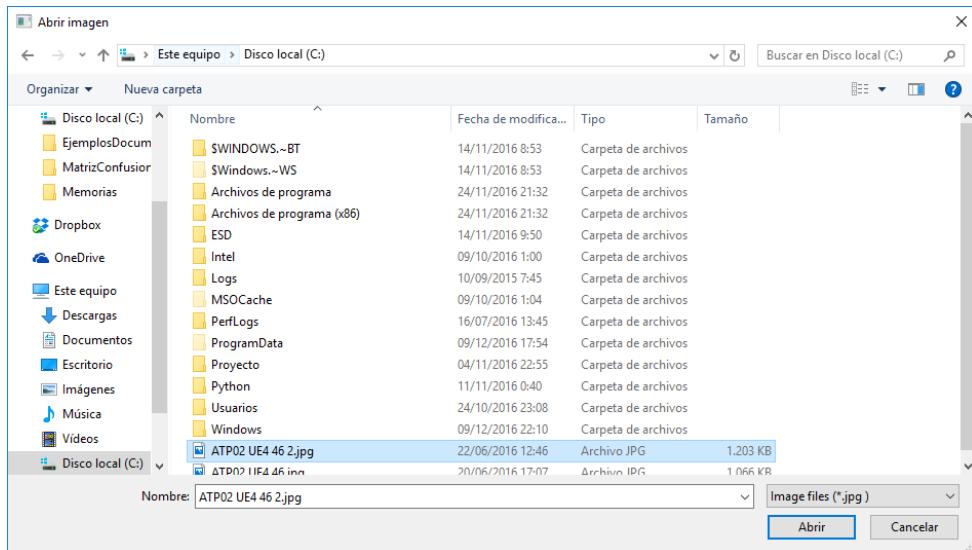


Figura E.3: Seleccionador de ficheros

Una vez seleccionado deberemos dar a aceptar si hemos seleccionado la imagen que queremos evaluar. Dependiendo de la imagen que hemos abierto pueden pasar dos cosas: uno E.4a, que la imagen abierta este pintada, dos E.4b, que la imagen abierta no este pintada. Dependiendo de esto tendremos estas dos ventanas, como se muestra en la figura ??.

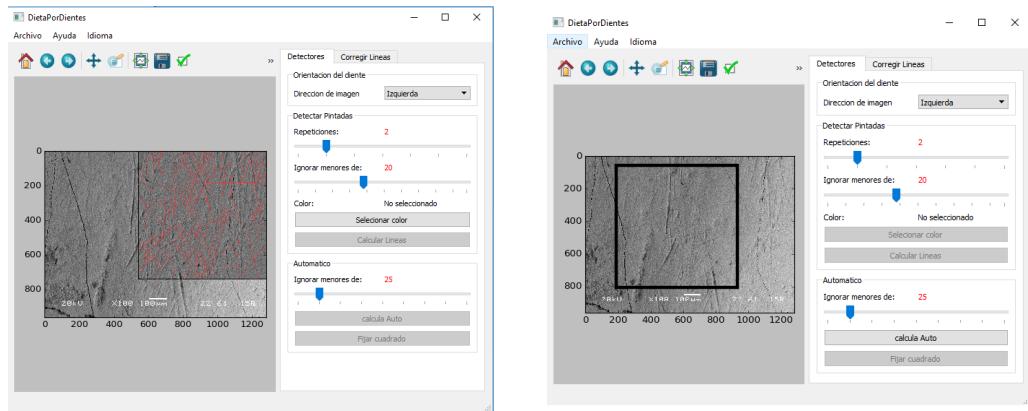
Cargar proyecto

En esta sección se explicará como cargar un proyecto en nuestra aplicación, para continuar un proyecto ya empezado.

Deberemos seleccionar la opción de Archivo > Abrir Proyecto. Como se puede ver en la figura E.5

Una vez que hacemos clic sobre la opción de abrir proyecto, ahora debemos explorar, como podemos observar en la siguiente figura E.6, hasta el directorio que contiene los ficheros que se generan al guardar un proyecto «Pintada.jpg, Original.jpg, Proyecto.xml, Salida-Estadísticas.csv, Salida-Líneas.csv y Tabla.tex» y una vez seleccionado el directorio que contiene los ficheros del proyecto, damos a «Abrir carpeta» para cargar el proyecto.

Una vez abierto el proyecto tendremos la imagen junto con sus estrías guardadas que estarán pintadas sobre la imagen, como podemos observar en la figura E.7.



(a) Opcion con lineas pintadas.

(b) Opcion con lineas sin pintar.

Figura E.4: Opciones de apertura de imagen.

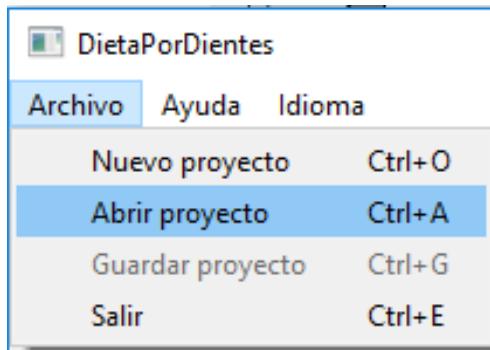


Figura E.5: Opción para abrir un proyecto.

Cambiar idioma

Para cambiar el idioma de la aplicación, deberemos seleccionar en el menú principal, «Idioma > Una de las dos opciones» como podemos observar en la figura E.8.

Hemos optado por cambiar el idioma de la aplicación para que perduren los cambios en futuras ejecuciones, para realizar esto, debemos reiniciar la aplicación. Si tenemos datos sin guardar, nos preguntará si queremos guardar los cambios o no antes de cerrar la aplicación.

Modo semi-automático para lineas pintadas

Este modo solamente podrá ser usado cuando dispongamos de una imagen que tenga las estrías pintadas sobre ella y estas estén contenidas dentro de un cuadrado que delimita su área, como podemos observar en la figura E.9.

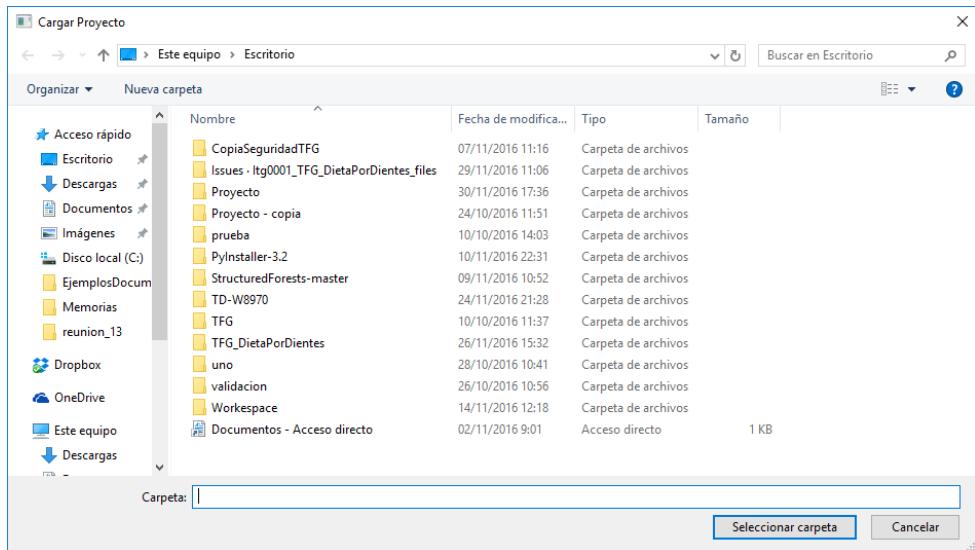


Figura E.6: Opción para abrir un proyecto.

Una vez que tengamos una imagen con las estrías pintadas cargada y válida como mostramos en la figura, procederemos a seleccionar su orientación, como muestra en la figura E.10.

A continuación deberemos seleccionar tanto el número de repeticiones del algoritmo para la obtención de los segmentos pintados «Cuanto mayor número de ellos mas tiempo tardara», la longitud mínima que debe ignorar el algoritmo para no detectar segmentos muy pequeños. Como podemos observar en la figura E.11.

Finalmente, nos quedaría la opción mas importante que sin ella el algoritmo no podrá ser ejecutado. Es seleccionar el color del que estén pintadas las líneas, para ello deberemos hacer click sobre el botón «Seleccionar color», como podemos observar en la figura E.12a y después de hacer click sobre el, este se desactivará, como podemos observar en la figura E.12b. Las estrías pintadas a veces pueden ser muy finas y no seremos capaces de clicar sobre esos pixeles, por lo que en este caso deberemos ampliar la imagen en una región que contenga estrías pintadas. Para ampliar debemos seleccionar el botón de ampliar y dibujar un rectángulo sobre la imagen con el clic derecho pulsado, como podemos observar en la figura E.12c. Una vez seleccionado el color correctamente, cuando este pintado el label de color actual con el color de las estrías pintadas, se activará el botón de calcular las líneas, como podemos observar en la figura E.12d.

Para finalizar, una vez que hagamos click en el botón de calcular líneas ya podremos añadir estrías nuevas, eliminar algunos segmentos, borrar todos o guardar los datos del proyecto para calcular las estadísticas de las estrías.

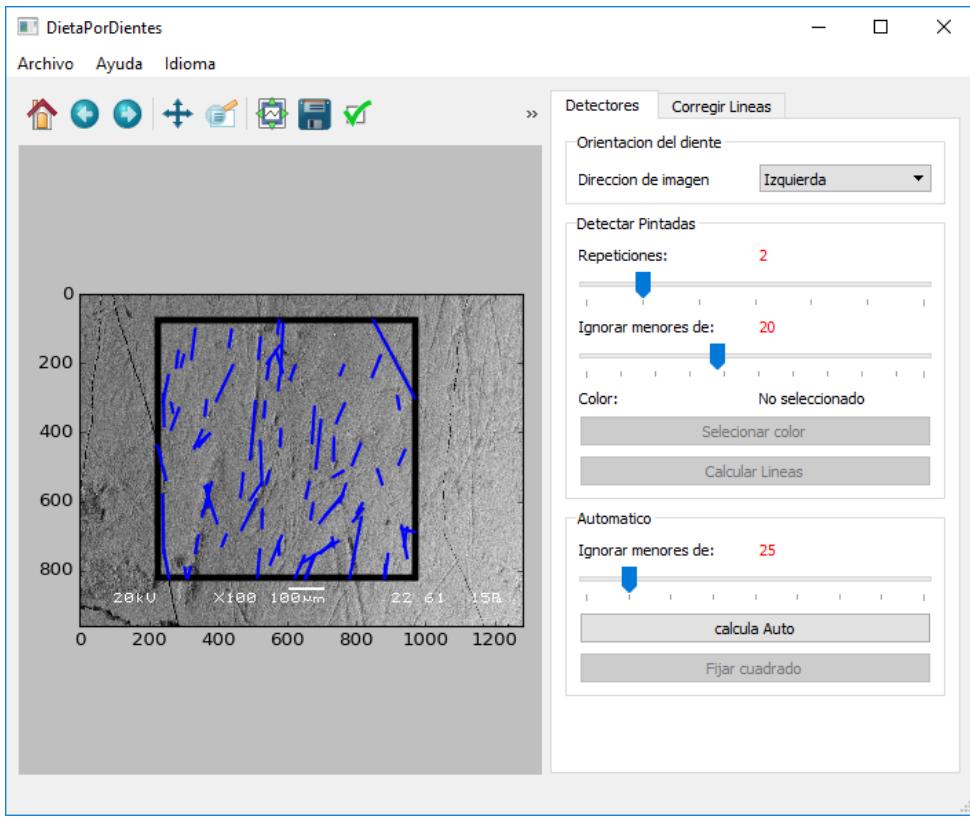


Figura E.7: Como se muestra un proyecto cargado o abierto.

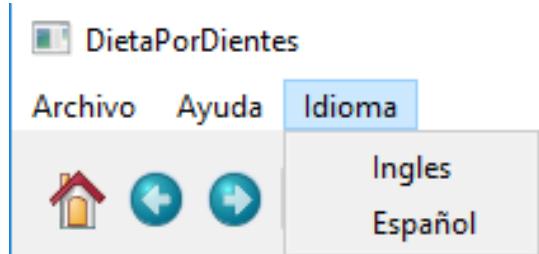


Figura E.8: Como cambiar el idioma.

Modo manual

Después de cargar una imagen. Ya sea pintada o sin pintar. Deberemos abrir una imagen como indica el apartado E.4 o abrir un proyecto tal y como indican el apartado E.4. Una vez que tengamos la imagen o el proyecto cargado podremos pintar nuevas estrías en la imagen siguiendo los siguientes pasos.

- Pulsaremos el botón de corregir lineas, como muestra la figura E.13a.

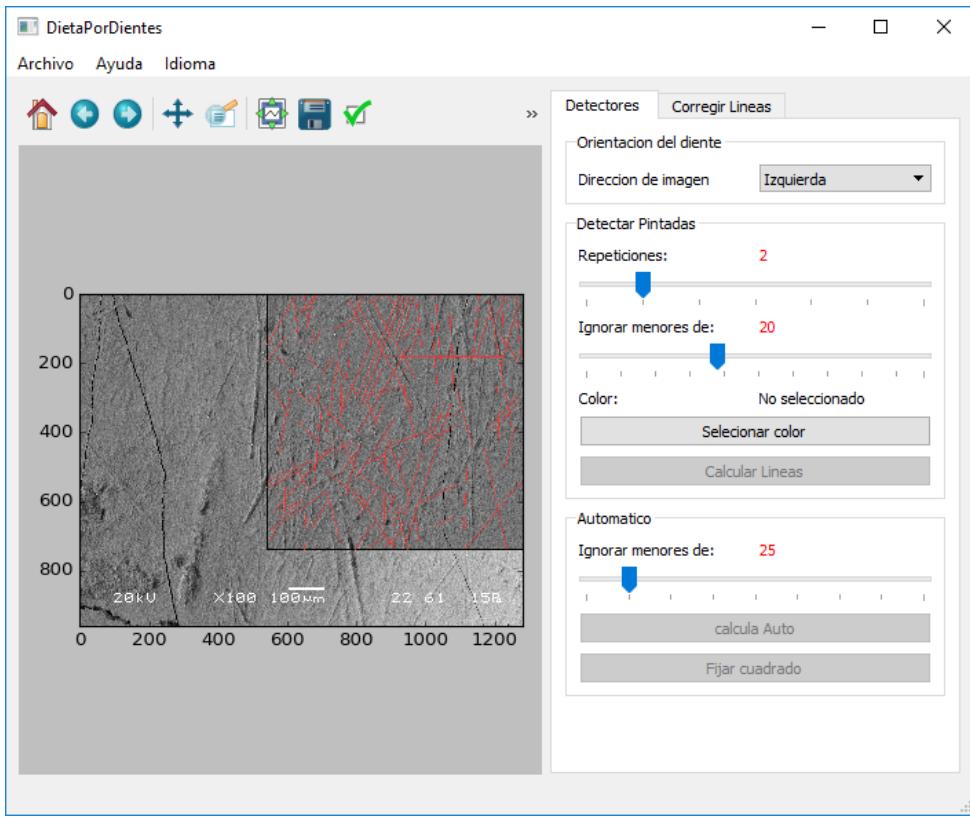


Figura E.9: Imagen válida para detección de estrías modo semiautomático.

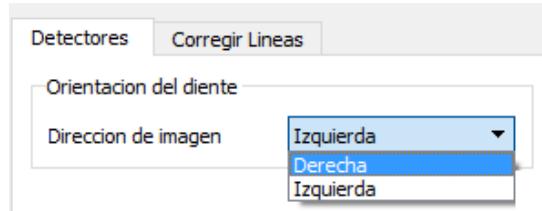


Figura E.10: Selección de la orientación de la imagen.

- Una vez pulsado deberemos seleccionar dos puntos: inicio y final del segmento que queremos añadir. Dentro del recuadro de la región pintable que esta delimitada por un recuadro negro, como se muestra en las figuras E.13b y E.13c, ahora solo nos quedara añadir el segmento a la tabla como muestra la figura E.13d.

Para los demás pasos no hace falta explicación ya que sus respectivos botones dicen claramente que funciones desempeñan.



Figura E.11: Selección de las opciones disponibles para este modo.



(a) Selección del color en el que estén las estrías pintadas.

(b) Despues de hacer click sobre él.



(c) Ampliar región.

(d) Despues de seleccionar correctamente el color.

Figura E.12: Pasos para seleccionar el color correctamente.

Modo automático

Después de cargar una imagen sin pintar deberemos abrir una imagen como indica el apartado E.4 o abrir un proyecto tal y como indica el apartado E.4.

Una vez que tengamos la imagen o el proyecto cargado, podremos configurar el modo automático, lo primero será asignar la orientación de la imagen, como podemos ver en la figura E.10, también deberemos seleccionar la longitud mínima de corte que ignorará el algoritmo automático, como podemos observar en la figura E.14.

Una vez seleccionado el tamaño mínimo de corte podremos ejecutar el algoritmo haremos click sobre el botón de calcular automático, obteniendo el resultado mostrado en la figura E.15 y podremos mover el recuadro del área con el que nos queremos quedar a la región de interés que consideremos apropiada.

Una vez que la situemos donde mayor densidad de estrías haya detectado, la fijaremos e ignorará las que se queden por fuera de la región, como podemos observar en la figura E.16.

Corregir líneas

P_1:	0	0
P_2:	0	0

Corregir Líneas

(a) Activar el modo de corregir o añadir un segmento.

Corregir líneas

P_1:	716.0	76.0
P_2:	0	0

Corregir Líneas

(b) Primer punto seleccionado.

Corregir líneas

P_1:	716.0	76.0
P_2:	1035.0	478.0

Corregir Líneas

(c) Segundo punto seleccionado.

Tabla

Anadir punto			
Borrar seleccionado			
Limpiar tabla			
P1X P1Y P2X P2Y			
Anadir segmentos			
Guardar tabla			

(d) Añadir segmento.

Figura E.13: Pasos para añadir un segmentos manualmente.

Automatico

Ignorar menores de: **25**

calcula Auto

Fijar cuadrado

Figura E.14: Selección de la longitud mínima de corte.

Pasado este punto, en la pestaña de corregir líneas quedarán añadidas todas aquellas que ha detectado el algoritmo y hemos encuadrado en la región factible, de estas líneas podremos obtener tanto estadísticas como un proyecto para editar mas adelante. Guardando el proyecto con el botón para dicha función.

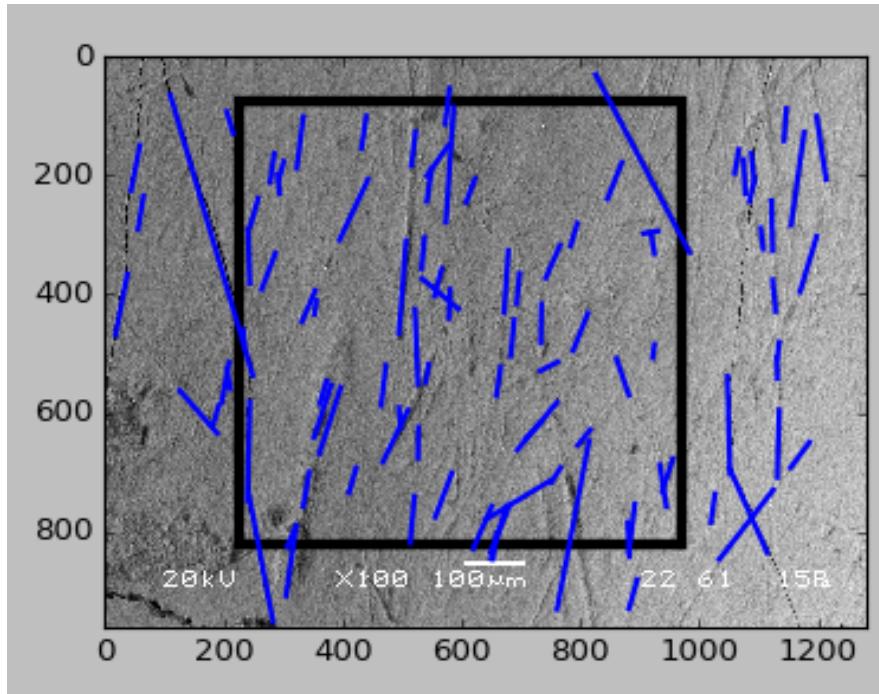


Figura E.15: Segmentos calculados por el modo automático.

Guardar proyecto

En este apartado vamos a explicar como guardar un proyecto una vez que hayamos abierto o cargado una imagen, calculado las estrías con el modo semiautomático o por el automático, y la tabla este rellenada con estas. Si el proyecto es cargado a partir de uno ya existente, no tener abiertas las estadísticas en Excel o cualquier otro editor, porque sino no funcionará la operación de guardar. Esto sera transparente al usuario porque dejara los ficheros como estaban antes, no los modificará, e informará al usuario de que no han sido guardados.

Podemos usar tanto el botón de guardar que aparece en la pestaña dos, la pestaña de corregir lineas, [E.17a](#) como el botón del menú principal que aparece en la figura [E.17b](#)

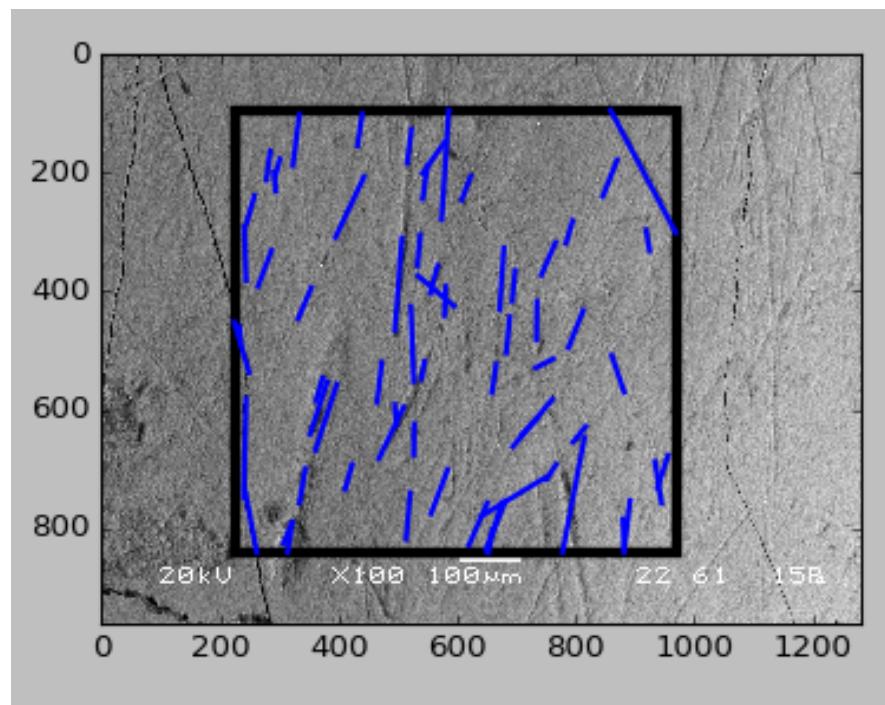
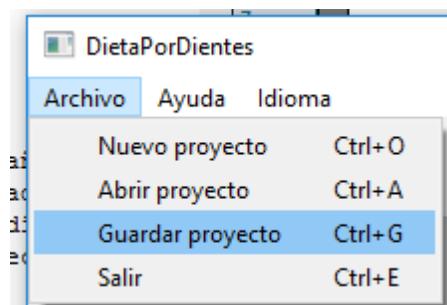


Figura E.16: Segmentos englobados en el cuadrado.



(b) Botón de guardar en el menú principal.

(a) Botón de guardar en la pestaña, corregir líneas.

Figura E.17: Opciones para guardar tabla.

Apéndice F

Procesado automático de la imagen

F.1. Introducción

En este apartado vamos a explicar lo que hemos investigado sobre la detección de bordes para el modo automático.

Esto lo vamos afrontar desde dos puntos de vista, el primero será por el procesado de imágenes, esto consistirá en binarizar la imagen usando un kernel conocido y observar si los resultados nos muestran algún resultado factible. Como segunda opción vamos a intentar ejecutar algoritmos de Deep Learning para ver si mejora el resultado usando estos algoritmos.

Para entender este anexo y este procesado, por lo menos es necesario haberse leído los conceptos teóricos y herramientas mencionadas en la memoria ya que no se ha duplicado dicha información en varios sitios.

F.2. Extracción de bordes mediante procesado de imagen

Esta sección consistirá, partiendo de una imagen en extraer las características, en este caso será la detección de los bordes contenidos, se corresponderán con las estrías que queremos detectar. Lo vamos a detectar a partir de los métodos explicados a continuación.

A todas las imágenes se las va a aplicar un proceso de convolución, con el Kernel [?] específico en cada caso.

Una convolución es una operación matemática, que no es la multiplicación de matrices tradicional, es el proceso de agregar cada porción de la imagen a

sus vecinos locales ponderado por el Kernel. Este proceso no solo vale para detectar bordes, tiene estas otras aplicaciones.

- Detección de bordes: Detectar los puntos donde acaba un objeto y empieza otro, también las aristas o bordes de las imágenes.
- Desenfoques y difuminado: Desenfocar o difuminar la imagen a partir de hacer la operación de convolución con una máscara diseñada para ello.
- Eliminar ruido: Sirve para moderar los valores de la imagen y hacer que estén mas centrados en un rango y así eliminar el ruido aleatorio.

Laplace:

Esta función busca bordes usando el operador de Laplace [?]. En nuestro caso de tamaño 3 (ya que se puede especificar el tamaño). Se ilustra el Kernel de Laplace en la figura F.6.

Observaciones: Como se ilustra en la figura F.1a, no es un método válido para nuestro propósito, porque no detecta los bordes, simplemente deja algunas partes difuminadas, pero nada sobre lo que podamos trabajar para obtener la imagen binarizada. Por lo que no vamos a continuar usándolo.

Prewitt:

Encuentra los bordes usando la transformada de Prewitt [?].

Se ilustra el Kernel de Prewitt en la figura F.7.

Observaciones: Como podemos ver las grietas, en la figura F.1b, las detecta bien pero las estrías de dieta son siluetas muy tenues y contiene mucho ruido.

Scharr:

Con este método encontramos los bordes usando la transformada de Scharr [?]. Se ilustra el Kernel de Scharr en la figura F.8 todo ello partido de 16 y para las verticales es la matriz transpuesta.

Observaciones: Como podemos ver en la figura F.1c, las estrías de dieta son muy tenues pero ligeramente mejores que en la anterior, tampoco demasiado pero ligeramente, el método es algo más rápido también pero no obtenemos algo tangible, en cuanto a bordes detectados ya que se muestran de forma muy tenue y no podrán ser extraídos en la binarización necesitaríamos mayor diferencia, entre el fondo el y borde.

Sobel:

Este método busca bordes usando la transformada de Sobel [?].

Se ilustra el Kernel de Sobel en la figura F.9.

El kernel de Sobel, partido de 4 para los bordes horizontales. Para los bordes verticales, es el kernel de Sobel, partido de 4 y transpuesto.

Observaciones: Con este método tal y como podemos ver en la figura F.1d, crea más ruido que los anteriores pero también podemos vislumbrar las siluetas de las estrías de dieta.

Roberts:

Esta función encuentra los bordes usando la operación cruzada de Robert [?].

Se ilustra el Kernel de Roberts en la figura F.10.

Observaciones: Como podemos ver en la figura F.1e, este método produce mucho ruido y las estrías son líneas demasiado tenues.

Kirsch:

Esta función encuentra los bordes usando el kernel de Kirsch [?]. Para cada dirección. No estaba implementado en Skimage por lo que he implementado el método. Como podemos ver en las figuras F.1, F.2, F.3 y F.4, están los kernel para los bordes Horizontal, vertical diagonal ascendente y diagonal descendente.

Cuadro F.1: Kernel g1

5	5	5
-3	0	-3
-3	-3	-3

Cuadro F.3: Kernel g3

Cuadro F.2: Kernel g2

5	5	-3
5	0	-3
-3	-3	-3

Cuadro F.4: Kernel g4

5	-3	-3
5	0	-3
5	-3	-3

Cuadro F.5: Distintos kernels de Kirsch

-3	-3	-3
5	0	-3
5	5	-3

Observaciones: Como podemos observar en la figura F.1i. Este método produce mucho ruido y las estrías son líneas demasiado tenues. No obstante de los métodos hasta ahora usados es en el que mas aprecian.

Cuadro F.6: Kernel Laplace

0	1	0
1	-4	1
0	1	0

Cuadro F.8: Kernel HScharr

Cuadro F.7: Kernel Prewitt

1	1	1
0	0	0
-1	-1	-1

Cuadro F.9: Kernel HSobel

3	10	3
0	0	0
-3	-10	-3

Cuadro F.10: Kernel Roberts

1	2	1
0	0	0
-1	-2	-1

Cuadro F.11: Kernels de los métodos analizados.

0	1
-1	0

Cuadro F.11: Kernels de los métodos analizados.

Autovectores matriz Hessiana:

Este método consiste en obtener la matriz hessiana [?] y después sus autovectores, esto nos devuelve dos matrices la matriz i1 es la matriz con el autovector más largo y la i2 es la matriz con autovector más corto.

Observaciones: Antes de aplicar el método, podemos leer en la documentación de Skimage, que es apropiado para detectar bordes continuos, entre otras formas, por lo que podemos pensar que en nuestro caso, cumple los requisitos para una buena detección ya que estos también son continuos. Como podemos observar en la figura F.1f en escala de grises del autovector de los valores más largos las siluetas de las estrías de dieta son las que más se remarcán sobre un tenue fondo gris pero pudiendo ser observadas por lo que este podría ser un punto de partida.

Canny:

Eliminando ruido:

Primero, obtener los bordes, llamar a una función que elimina el ruido y después al detector de bordes Canny [?], para obtener los bordes. Para los parámetros usados en la función de Canny utilizaremos:

- Sigma: Un valor intermedio de 1.4. Este parámetro afecta a la desviación estándar del filtro Gausiano.
- Umbral mínimo: Un valor muy bajo. Este parámetro nos indica el valor mínimo para ser considerado un borde.

- Umbral Máximo: Un valor bajo pero mayor que el umbral mínimo. Este parámetro nos indica el valor máximo para ser considerado un borde.

Gracias a estos parámetros usados obtenemos una imagen resaltando algunos bordes pero no todo los que queremos ya que no están demasiado marcados.

Observaciones: Desde esta figura F.1g con bastante ruido ya podemos observar que las más marcadas son detectadas pero no se consiguen diferenciar demasiado bien, pero en comparación con los demás métodos tiene de las mejores salidas aun no siendo buena de ir en esta línea tendíramos que usar esto.

Modificando parámetros Canny:

La segunda opción ha sido usar un detector de bordes Canny solamente modificando sus parámetros pero en esta opción los valores de los umbrales deben ir sin normalizar entre 0 y 1, sino entre 0 y 255.

Observaciones: La figura F.1h detecta menos ruido que con la otra tentativa pero sigue siendo deficiente en cuanto a las líneas ya que aparte de detectar pocas detecta las que realmente no son estrías de dieta. Pero de ir en alguna línea sería por este camino.

Gabor:

Gabor [?] es un filtro linear con un kernel gausiano que es modulado por un onda sinusoidal plana. Principalmente se usa en visión artificial de clasificación y detección de bordes. Obtenemos un par de imágenes.

Observaciones: Como podemos observar en la imagen usando Gabor con filtro imaginario F.1j. Como podemos observar en la imagen usando Gabor con filtro real F.1k. Este método lo he probado porque en la obtención, de las líneas correspondientes a vasos sanguíneos en los ojos «blood vessels detection [?]» es lo que se usa para ello pero al no ser líneas continuas y no seguir un patrón no he conseguido buenos resultados. En el filtro real no es tan malo pero en el filtro imaginario es ruido puro.

Comparativa Filtros

Como podemos ver en la figura F.1 hemos usado casi todos los filtros conocidos para hacer esta comparativa, a simple vista ninguno de los métodos mencionados o probados no parece que funcione, ya que el problema no es fácil. Manualmente no observamos algunas de las estrías que los expertos marcan por lo tanto, detectar algunas sera algo efectivo.

Pero dentro de la comparativa la que mejor detecta algunos de los bordes parece el método de la matriz Hessiana con autovectores largos. Aunque en

la imagen se presenta poca diferencia entre fondo y las estrías por lo que nos centraremos en ese procesado haber si conseguimos diferenciarlo y extraer las características.

Procesado:

Partiendo del análisis anterior vamos a juntar los mejores resultados y añadir pasos adicionales para obtener la máscara binaria que mas podamos ajustar a nuestras necesidades.

La detección de los bordes en la imagen es solo uno de los pasos en nuestro algoritmo, este se compone por una serie de pasos divididos en tres categorías, preprocesado para mejorar la calidad de la imagen, Detección de bordes y postprocesado de la imagen de bordes. Todos los pasos del algoritmo se muestran en la figura F.2.

■ Preprocesado:

- Ecualizar la imagen original F.2a, consistirá en extender el histograma de la imagen original para que no este centrado en un rango pequeño como podemos observar en su histograma F.2b, pasado este paso obtendremos la imagen ecualizada F.2c. Así conseguimos que su histograma F.2d este mas repartido por toda la gama de grises y no centrado en un pequeño rango.

■ Detección de bordes:

- Autovectores de la Hessiana F.3a: Para detectar los bordes utilizaremos el método antes mencionado de la matriz Hessiana del cual elegiremos solo los largos, ya que los autovectores pueden ser los cortos o los largos.
- Sustraemos el fondo a la figura F.3b: Como la imagen no tiene de nuevo demasiada diferencia entre el fondo y los bordes, sustraeremos el fondo, para quedarnos únicamente con las estrías detectadas, al tener ruido aparte de los elementos buscados, tendremos que eliminar dicho ruido mas adelante.
- Binarizamos la figura F.3c: Como la imagen después de sustraer el fondo no es binaria, cualidad necesaria para ser una máscara. Hay que aplicar un threshold o umbral para pasar a blanco o negro, binarizar, es decir los pixeles que no pasen el umbral serán negros y los que lo pasen formaran parte de los elementos que queremos detectar y serán blancos. Entonces la imagen quedara binarizada.

■ Postprocesado de la imagen y bordes:

- Eliminamos el ruido **F.3d**: El paso anterior, en estas condiciones de trabajo genera mucho ruido, por lo que tendremos que intentar reducirlo y para ello eliminamos los trozos que son pequeños, ya que el ruido parece ser aleatorio de fragmentos muy pequeños, esto elimina la mayoría de los puntos de ruido.
- Erosionar con operador diamante **F.3e**: Para suavizar la imagen y evitar que queden líneas finas a modo de antenas, erosionamos la imagen con un operador estructurante en forma de diamante, así conseguimos que pase por la mayoría de las figuras.
- Esqueletonizar la imagen **F.3f**: Una vez tengamos la imagen sin tanto ruido, reducimos el grosor de las líneas para que la función de Hough funcione, así nos detecta las líneas que corresponden a esta máscara binaria.
- Eliminar mas ruido **F.3g**: Este paso anterior vuelve a generar ruido porque algunos trozos se dividen en pequeños segmentos y con la segunda reducción de ruido conseguimos hacerlos desaparecer y quedarnos con las líneas grandes.
- Detectar los segmentos **F.4a**: Una vez que tenemos todos los segmentos detectados mediante Hough, que se corresponden con las líneas que hay en la imagen, tenemos que unir los que sean continuos y muy cercanos, para formar segmentos mas grandes.
- Unir segmentos cercanos **F.4b**: Para este paso vamos a usar lo mismo que hemos utilizado dentro de la parte, detección de las líneas pintadas. La imágenes tienen mucho ruido y aunque hemos conseguido procesar y reducirlo, aun no vamos a detectar un alto por ciento de las estrías.

F.3. Extracción de bordes mediante Deep Learning

En esta sección vamos a intentar ejecutar algoritmos ya existentes de Deep Learning y si funcionan re-entrenarlos para que detecten los bordes en nuestras imágenes, pero para ello tendremos que probar todos lo posible y de los que funcionen y detecten decentemente probar que pasa si re-entrenamos para nuestras imágenes.

Algoritmos probados, en Matlab, en Python 3.5 y en Python 2.7:

- Tensorpack: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/ppwwyyxx/tensorpack>

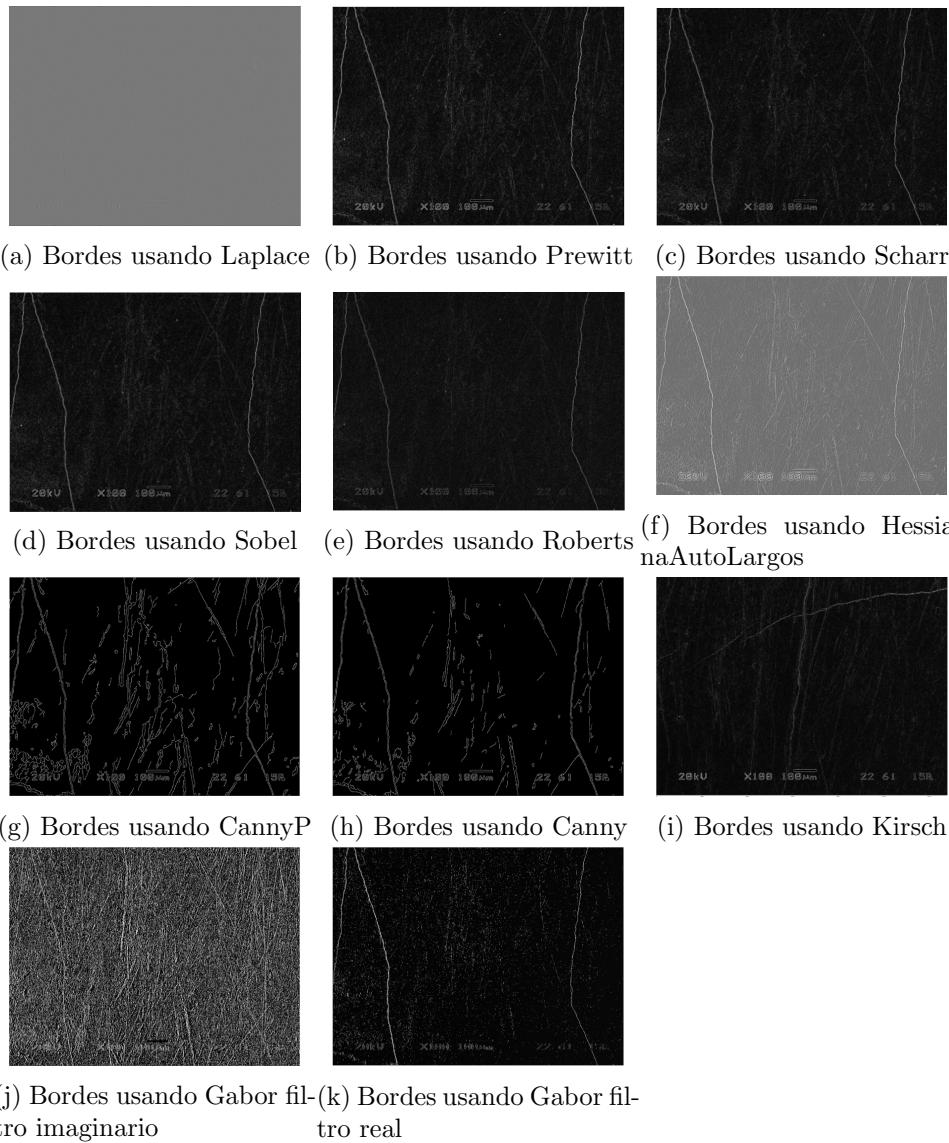
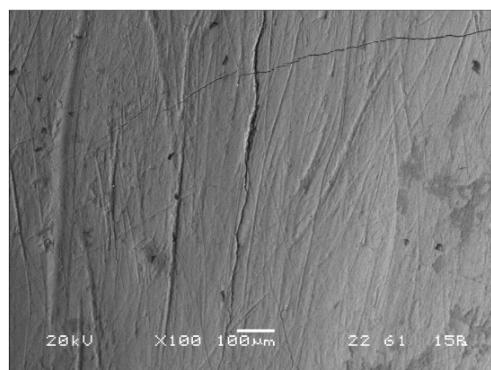
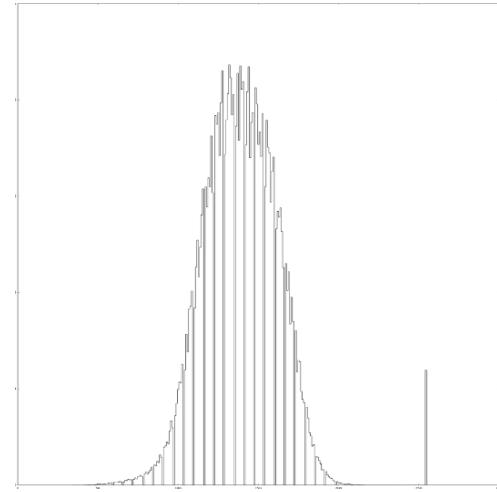


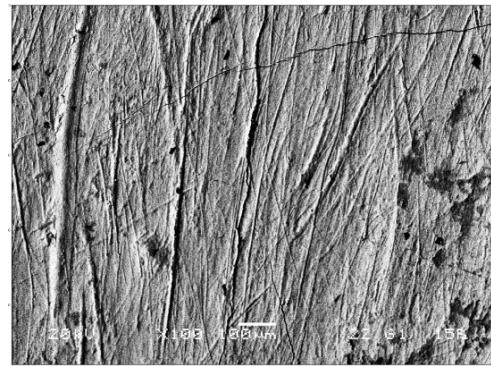
Figura F.1: Resumen visual filtros usados.



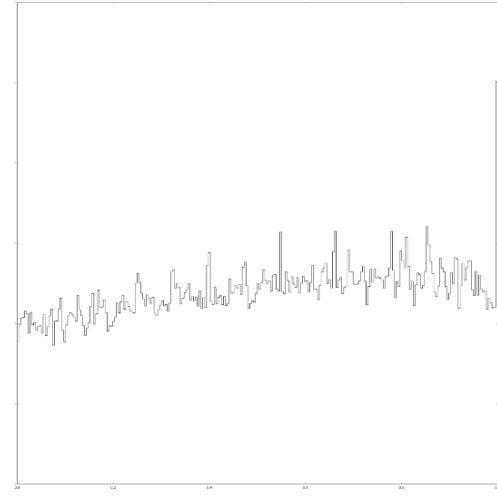
(a) Imagen original



(b) Histograma imagen original

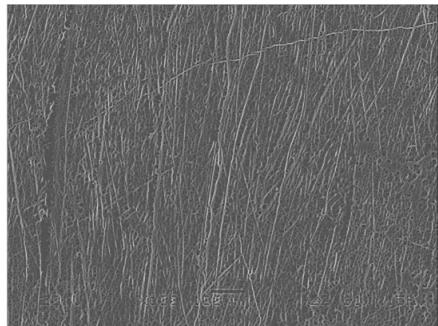


(c) Imagen ecualizada



(d) Histograma imagen ecualizada

Figura F.2: Pasos del procesado parte uno.



(a) Autovectores largos.



(b) Hessiana menos fondo.



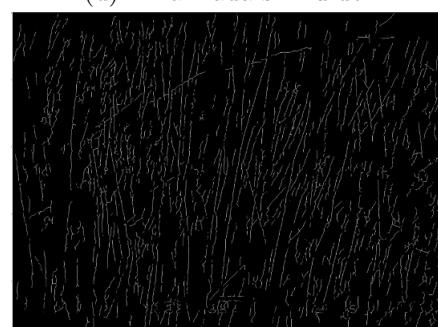
(c) Binarizado.



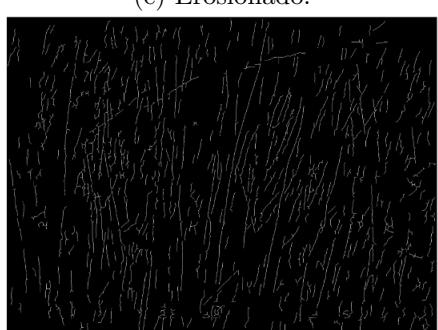
(d) Binarizada sin ruido.



(e) Erosionado.

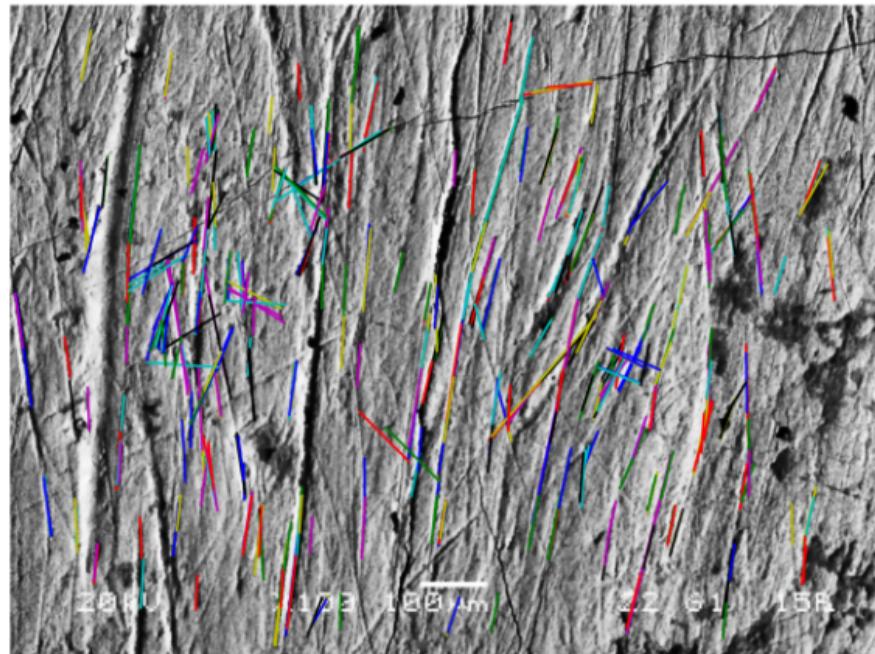


(f) Esqueletonizado de la erosión.

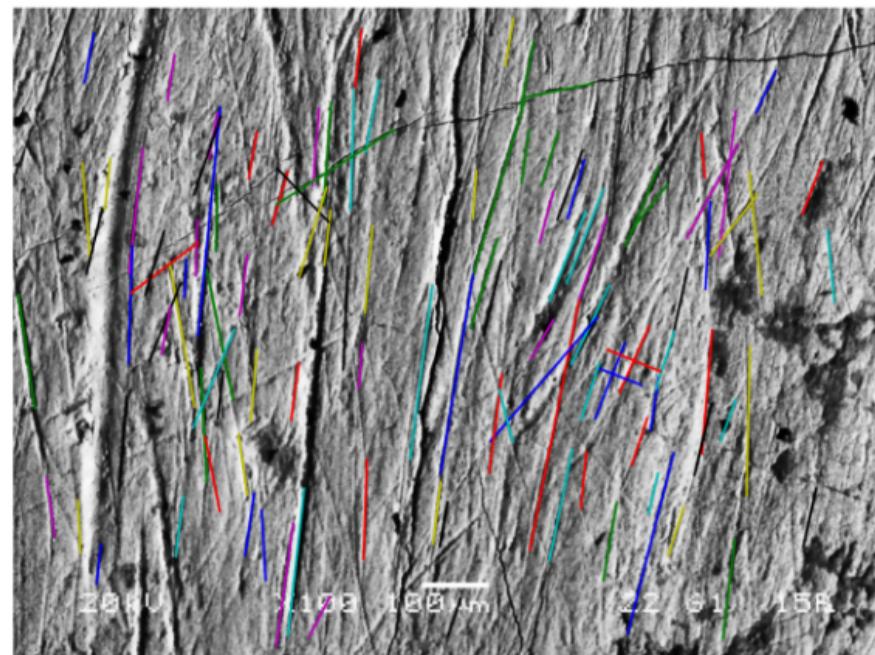


(g) Esqueletonizada sin ruido

Figura F.3: Pasos del procesado parte dos.



(a) Segmentos detectados



(b) Segmentos reales

Figura F.4: Pasos del procesado parte tres.

- Structured edge forest: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/pdollar.edges>
- Sketch Tokens: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/gitlim/SketchTokens>
- Oriented Edge Forests: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/RongchangZhao/oef>
- Retina blood vessel: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/orobix/retina-unet>
- StrudturedForest Python2: Podemos encontrar el proyecto que contiene dicho algoritmo en GitHub atraves de este enlace: <https://github.com/ArtanisCV/StructuredForests>

Una vez probados todos ellos, no hemos sido capaces de ejecutarlos, ya que por alguna razón en todos ellos, debía de faltar alguna clase, referencias o simplemente no estaban actualizados, en sus repositorios de GitHub.