



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 — 24 de noviembre
de 2016

Tutor: D. Álvar Arnaiz González y Dr. José
Francisco Diez Pastor

Índice general

Índice general	I
Índice de figuras	II
Apéndice A Manuales	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	8
Apéndice B Especificación de Requisitos	10
B.1. Introducción	10
B.2. Objetivos generales	10
B.3. Catalogo de requisitos	10
B.4. Especificación de requisitos	10
Apéndice C Especificación de diseño	11
C.1. Introducción	11
C.2. Diseño de datos	11
C.3. Diseño procedural	18
C.4. Diseño arquitectónico	18
Apéndice D Documentación técnica de programación	19
D.1. Introducción	19
D.2. Estructura de directorios	19
D.3. Manual del programador	19
D.4. Compilación, instalación y ejecución del proyecto	19
D.5. Pruebas del sistema	19
Apéndice E Documentación de usuario	20
E.1. Introducción	20

E.2. Requisitos de usuarios	20
E.3. Instalación	20
E.4. Manual del usuario	20
Apéndice F Procesado automático de la imagen	21
F.1. Por procesado de imagen	21
F.2. Por deep learning	26

Índice de figuras

A.1. Burndown de la semana 1	3
A.2. Burndown de la semana 2	4
A.3. Burndown de la semana 3	5
A.4. Burndown de la semana 4	6
A.5. Burndown de la semana 5	7
A.6. Burndown de la semana 6	9
C.1. Diagrama de clases inicial	11
C.2. Diagrama de paquetes inicial	14
C.3. Diseño de la interfaz de usuario	15
C.4. Diseño de la interfaz de usuario	16
C.5. Barra de herramientas de la aplicación	16
C.6. Barra de herramientas de la imagen	17
C.7. FigureCanvas de la imagen	17
C.8. Pestañas	18
F.1. Resumen visual filtros usados.	27
F.2. Pasos del procesado.	28

Apéndice A

Manuales

A.1. Introducción

Para la planificación temporal, hemos usado la metodología Scrum [?], que consiste en ir haciendo iteraciones o sprints que van dando valor añadido al proyecto, siempre tenemos versiones, de forma incremental nuestro proyecto va teniendo mas 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 burdownchar que nos indica el progreso perfecto frente al progreso real.

A.2. Planificación temporal

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

Se ha hablado del problema a resolver.

Se va a hacer un mini prototipo para evaluar las herramientas, librerías y algoritmos necesarios. Posteriormente a la reunión con el cliente (Rebeca) 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 LaTex
- 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.

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.

Gráfico del sprint 1 [A.1](#).



Figura A.1: Burndown de la semana 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:

- 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 cumplido 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) 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, 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. Gráfico del sprint 2 [A.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.

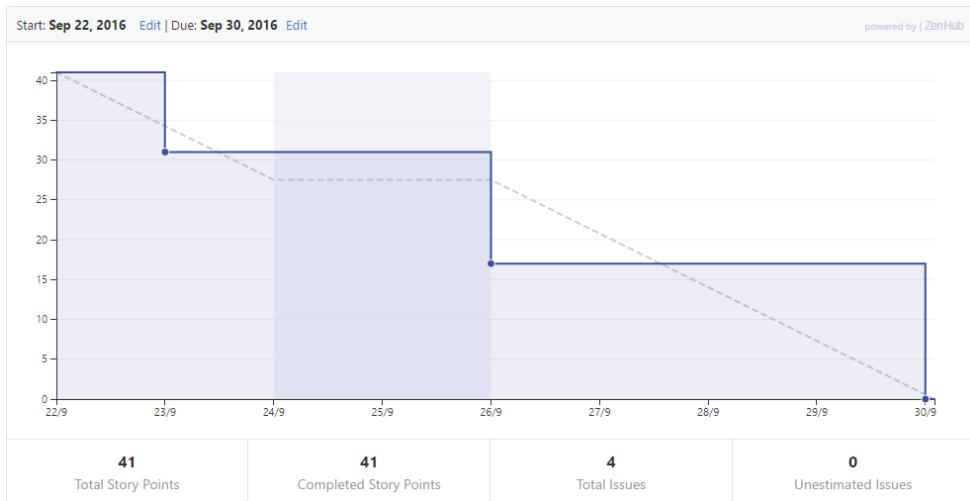


Figura A.2: Burndown de la semana 2

Cumplido:

Hemos cumplido 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 latex 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 version y no ha sido muy difícil.

Gráfico del sprint 3 [A.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.



Figura A.3: Burndown de la semana 3

- Implementación del código.
- Corrección de las memorias.
- Herramienta SonarQube.
 - Ejecutar la aplicación.
 - Corregir las horas de débito.

Cumplido:

Hemos cumplido los objetivos de la semana. 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.

Gráfico del sprint 4 [A.4](#).



Figura A.4: Burndown de la semana 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 cumplido con 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.

Gráfico del sprint 5 [A.5](#).



Figura A.5: Burndown de la semana 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.
 - Teses paquete código.
 - Teses paquete gui.

Cumplido:

Esta semana, hemos cumplido con 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.

Gráfico del sprint 6 **A.6.**



Figura A.6: Burndown de la semana 6

Sprint 7 (13/10/2016 - 22/10/2016)

A.3. Estudio de viabilidad

Viabilidad económica

Viabilidad legal

Apéndice B

Especificación de Requisitos

B.1. Introducción

Como introducción y para no repetirnos voy a explicar de forma mas general y sin entrar en detalles que contiene el proyecto.

El anexo contiene:

- Plan de proyecto: Consiste en la planificación temporal que hemos seguido adjuntando los gráficos de todas las semanas de trabajo.
- Requisitos: Contendrá el que consiste el proyecto y las definiciones de los requisitos.
- Diseño: Va a contener los diagramas de clases y la distribución en paquetes que hemos elegido. También contendrá que tiene cada clase.
- Manual programador:
- Manual del usuario: Este contendrá un tutorial de como usar la aplicación de forma correcta para la extracción de características de las imágenes.
- Investigación: Este apartado contendrá que hemos probado y como hemos investigado para el desarrollo del modo automático de la aplicación.

B.2. Objetivos generales

En este apartado lo que vamos a enumerar son los objetivos que nos hemos propuesto y hemos cumplido de todo el proyecto.

El proyecto consiste en:

- Construcción del prototipo de procesado de las imágenes.
 - Lectura de imágenen.
 - Binarización para detectar las 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.3. Catalogo de requisitos

Este apartado se refiere a las tareas que hemos realizado en cada uno de los respectivos Sprints.

Sprint 0

- Probar LaTex
- 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

Sprint 1

- 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.

Sprint 2

- 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.

Sprint 3

- 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.

Sprint 4

- 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.

Sprint 5

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

Sprint 6

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

- Pruebas unitarias.
 - Teses paquete código.
 - Teses paquete gui.

B.4. Especificación de requisitos

Apéndice C

Especificación de diseño

C.1. Introducción

C.2. Diseño de datos

Introducción

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.

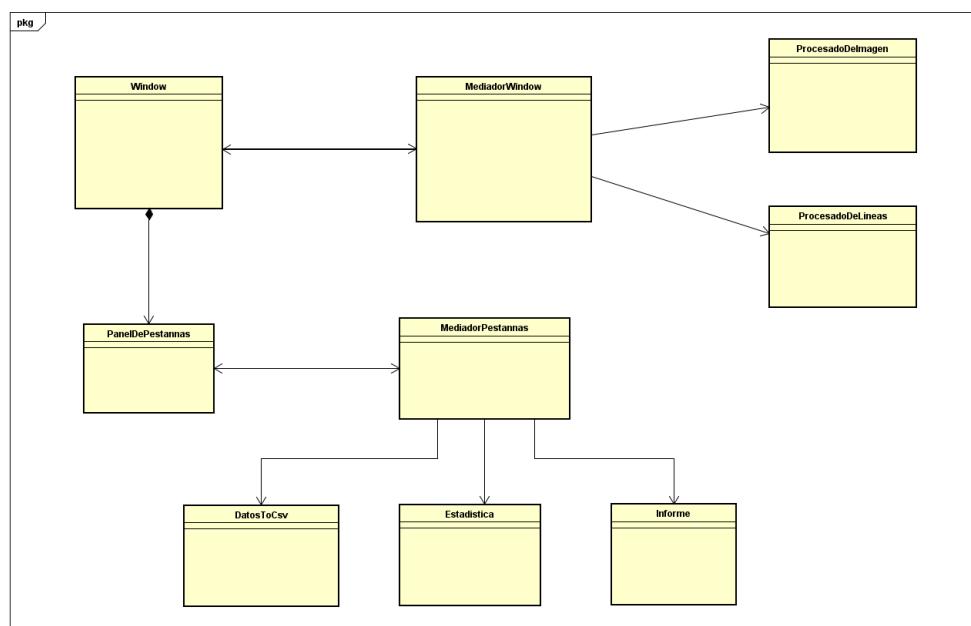


Figura C.1: Diagrama de clases inicial

Clases

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

- 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 LaTex 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.

Paquetes

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.

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 su libro más significativo 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 saber que hay por debajo, pero si poder acceder a las funciones que hay.

Obtenemos:

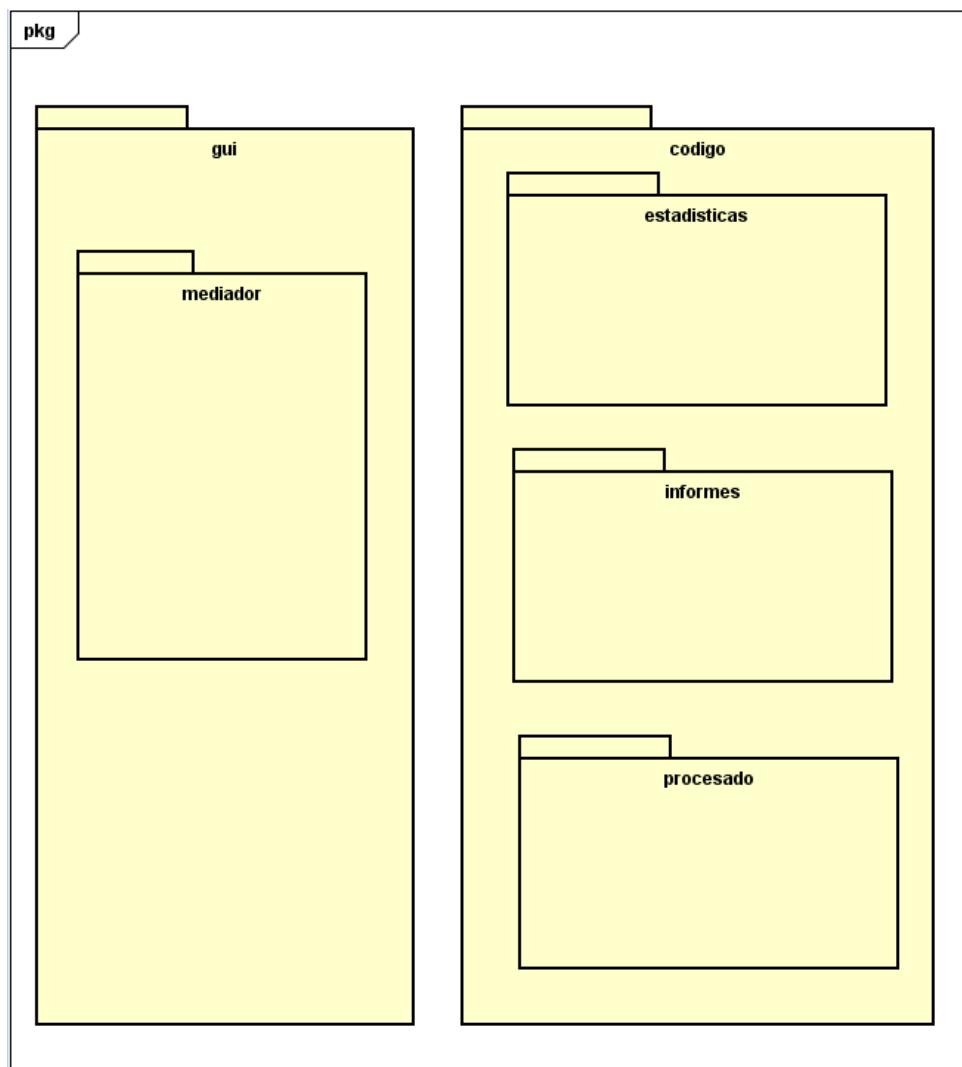


Figura C.2: Diagrama de paquetes inicial

- Permitir la parametrización.
 - Visualizar que ordenes se ejecutan en cada paso.
 - Construir operaciones complicadas a partir de otras más sencillas.
 - Permitir saber que habría que 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.

Diseño de la Interfaz

Primera versión

Para el desarrollo de la interfaz gráfica pensé en una planificación espacial acorde con los elementos que esta contendría. Un layout que seria 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 [C.3](#).

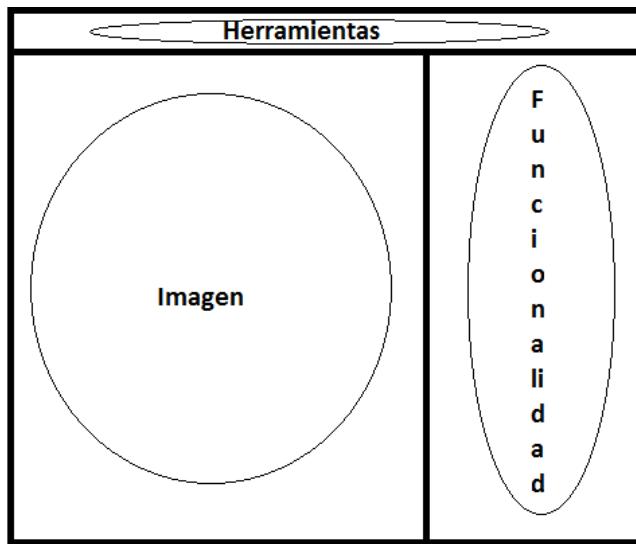


Figura C.3: Diseño de la interfaz de usuario

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 [C.4](#).

- Detección líneas rojas.

- Corrección y pintar líneas.
- Automático.

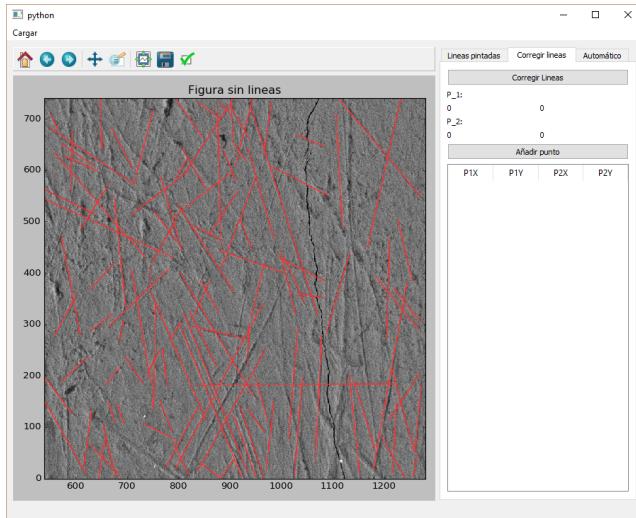


Figura C.4: Diseño de la interfaz de usuario

Barra de herramientas

Es una sección de la interfaz que nos va permitir de momento la carga de imágenes para su procesado [C.5](#).

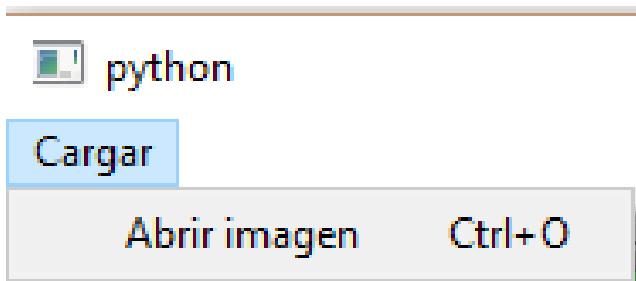


Figura C.5: Barra de herramientas de la aplicación

Barra de herramientas de la imagen

Es una sección de la interfaz que nos permitirá manipular la imagen para realizar estas acciones [C.6](#).

- 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.6: 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.7.

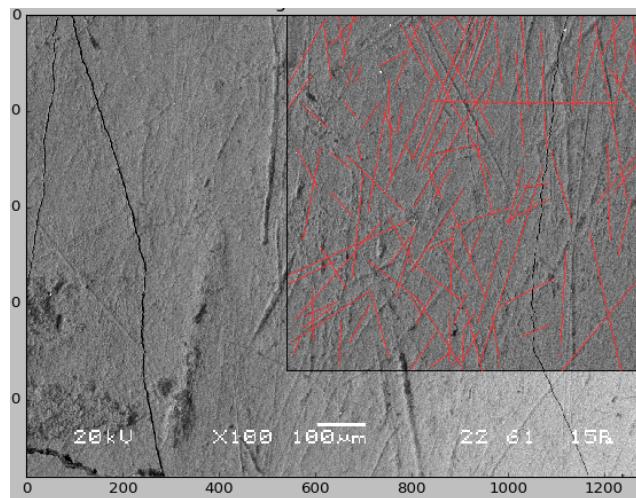


Figura C.7: FigureCanvas de la imagen

Pestañas

Sección de la imagen donde estarán implementadas las funcionalidades para visualizar las líneas que son detectadas por el algoritmo en la imagen [C.8](#).

- 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.

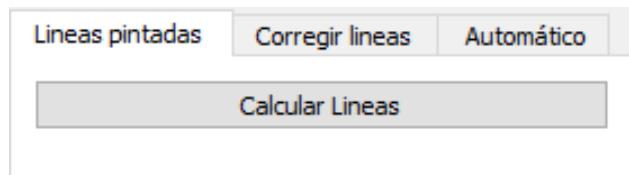


Figura C.8: Pestañas

C.3. Diseño procedimental

C.4. Diseño arquitectónico

Apéndice D

Documentación técnica de programación

- D.1. Introducción
- D.2. Estructura de directorios
- D.3. Manual del programador
- D.4. Compilación, instalación y ejecución del proyecto
- D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción**
- E.2. Requisitos de usuarios**
- E.3. Instalación**
- E.4. Manual del usuario**

Apéndice F

Procesado automático de la imagen

F.1. Por procesado de imagen

Todas las imágenes van hacer 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.
- Normalización.
- Desenfoques y difuminado.
- Eliminar ruido.

Laplace:

Esta función busca bordes usando el operador de Laplace. En nuestro caso de tamaño 3 ya que se puede especificar el tamaño.

Kernel Laplace [F.1](#).

```
from skimage.filters import laplace,prewitt  
bord=laplace(img,ksize=3)
```

Cuadro F.1: Kernel Laplace

0	1	0
1	-4	1
0	1	0

Cuadro F.2: Kernel Prewitt

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

Cuadro F.3: Kernel HScharr

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

Observaciones: Como podemos observar no es un método factible para nuestro propósito por lo que no vamos a continuar usándolo.

Prewitt:

Encuentra los bordes usando la transformada de Prewitt.

Kernel Prewitt [F.2](#).

Observaciones: Como podemos ver las grietas en la imagen 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.

Kernel HScharr [F.3](#). todo ello partido de 16 y para las verticales es la matriz transpuesta.

Observaciones: Como podemos ver en la imagen 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.

Sobel:

Este método busca bordes usando la transformada de Sobel.

Kernel HSobel [F.4](#). Todo ello partido de 4 y para las verticales es la matriz

Cuadro F.4: Kernel HSobel

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

Cuadro F.5: Kernel Roberts

0	1
-1	0

Cuadro F.6: Kernel g1

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

transpuesta.

Observaciones: Con este método tal y como podemos observar en la imagen crea más ruido que los anteriores pero también podemos observar la silueta de las estrías de dieta.

Roberts:

Esta función encuentra los bordes usando la operación cruzada de Robert Kernel Roberts [F.5](#).

Observaciones: 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.

Kernel g1 [F.6](#).

Kernel g2 [F.7](#).

Kernel g3 [F.8](#).

kernel g4 [F.9](#).

Observaciones: 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.7: Kernel g2

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

Cuadro F.8: Kernel g3

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

Cuadro F.9: Kernel g4

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

Autovectores matriz Hessiana:

Este método consiste en obtener la matriz hessiana y después sus autovectores de la matriz Hessiana y 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: Como podemos observar en la imagen 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. Los parámetros del canny en como sigma un valor intermedio de 1.4 junto con un umbral bajo, muy bajo, y un umbral alto, también bajo. Obtenemos una imagen resaltando algunos bordes pero no todo los que queremos ya que no están demasiado marcados.

Observaciones: Desde esta imagen 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: Esta imagen 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 seria por este camino.

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: Este método lo eh probado porque en la obtención de las líneas de sangre en los ojos 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

Comparativa de todos los filtros [F.1](#).

Procesado:

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

Pasos del algoritmo [F.2](#) y ??.

- Ecualizar la imagen original: Así conseguimos que sus histograma este mas repartido por toda la gama de grises y no centrado en un pequeño rango.
- Autovectores de la Hessiana: Obtenemos los autovectores de la matriz Hessiana y nos quedamos con los largos, ya que los autovectores pueden ser los cortos o los largos.
- Sustraemos el fondo a la imagen: Como la imagen no tiene de nuevo demasiada gama cromática le sustraemos el fondo para así quedarnos únicamente con las líneas, al tener ruido aparte de lo que queremos, tendremos mucho ruido.

- Binarizamos la imagen: Como la imagen aun habiendo sustraído el fondo no es binaria hay que aplicarle un threshold o umbral para pasar a blanco o negro, es decir, binarizarla.
- Eliminamos el ruido: Este 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, esto elimina la mayoría de los puntos de ruido.
- Erosionar con operador diamante: 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: 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 mascara binaria.
- Eliminar mas ruido: 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: 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 contiguos y muy cercanos, para formar segmentos mas grandes.
- Unir segmentos cercanos: 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.2. Por deep learning

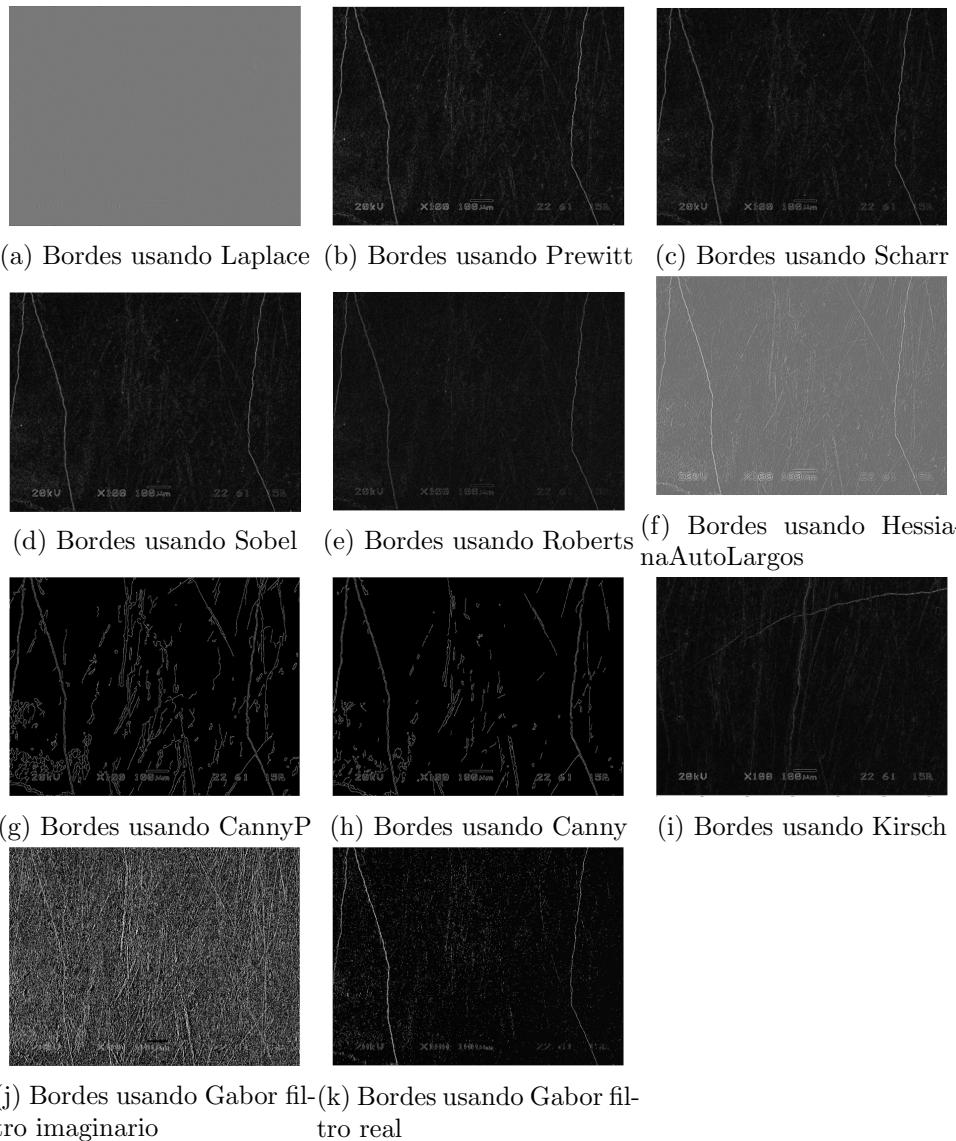


Figura F.1: Resumen visual filtros usados.

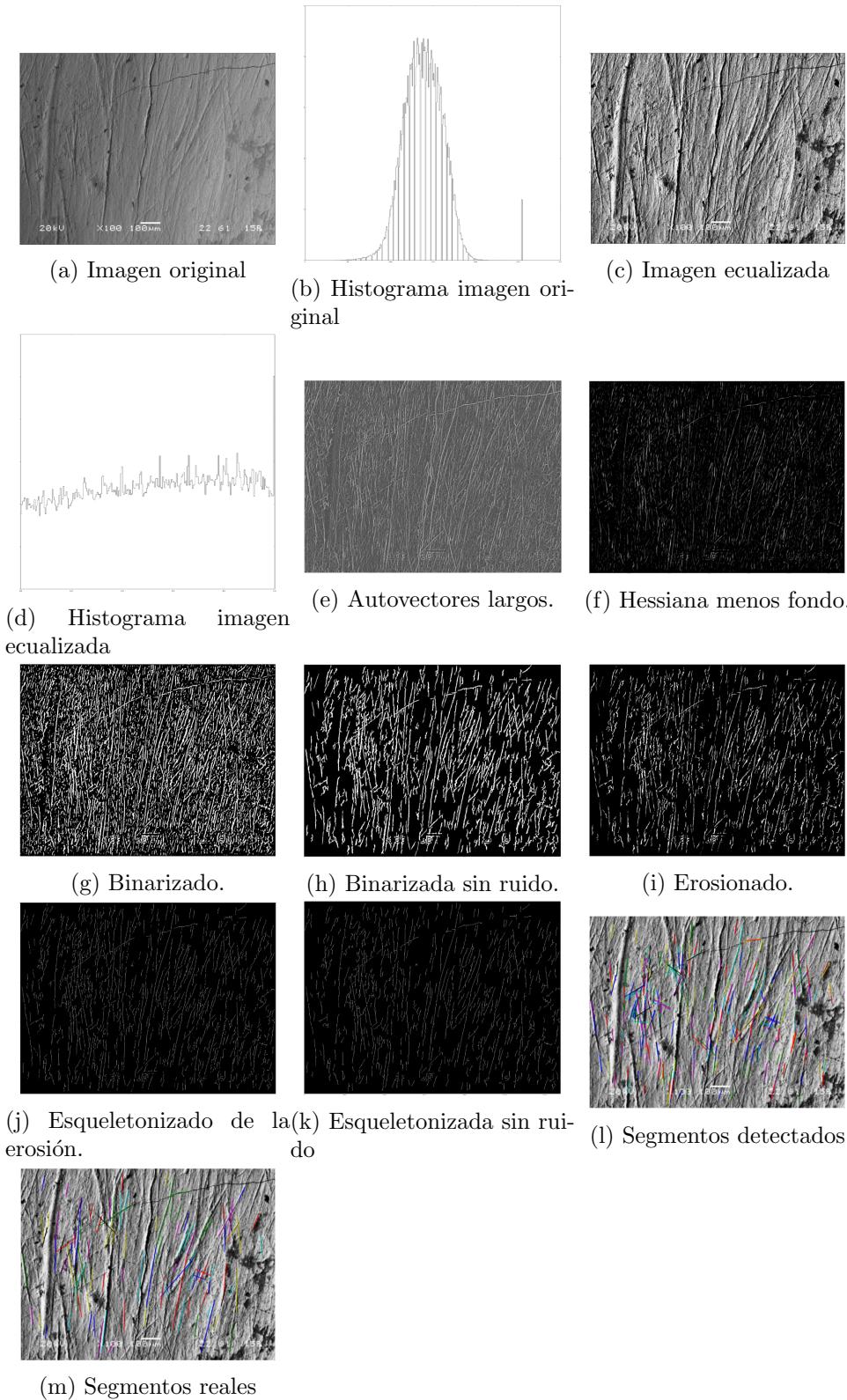


Figura F.2: Pasos del procesado.