UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Dieta por Dientes

Presentado por Ismael Tobar García
en Universidad de Burgos — 23 de noviembre de 2016
Tutor: Álvar Arnaiz González y José Francisco Diez Pastor

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

Dr. José Francisco Diez Pastor y D. Álvar Arnaiz González, profesores del departamento de Departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Ismael Tobar García, con DNI 71286542-C, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 23 de noviembre de 2016

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Introducción

Para ponernos en contexto sobre que va a hacer y en que va a consistir el proyecto, vamos a explicar brevemente que hará.

El problema planteado, consiste en crear una herramienta o aplicación para detectar las estrías que se producen en los dientes, al masticar distintos tipos de comida.

Para ser mas exactos, todos los materiales tienen una dureza tangible, los dientes tienen una dureza superior a la de los alimentos, sin embargo algunas partes de los alimentos tienen mas dureza que los dientes, por lo que los consiguen rayar, creando unas estrías que dependiendo de los alimentos siguen distintos patrones.

Vamos a detectar esas estrías, pero en el contexto de la prehistoria, para después poder clasificar los individuos en función de la dieta que llevaban.

Nuestro proyecto va a tener una parte de visión artificial, que detecte las lineas que han pintado, sobre las imágenes de microscopio de electrones a 100 aumentos, otra parte de diseño e implementación, del software a usar y distintos análisis de las herramientas que usaremos para ello.

La herramienta a usar tendrá implementados 3 modos claramente diferenciados.

- Detección automática de las lineas pintadas sobre la imagen.
- Pintar lineas manualmente sobre la imagen o corregir las posibles lineas que el anterior modo no consiga detectar.
- Automático que buscara las estrías por la imagen.Este modo quedara como extra, ya que hemos tenido una gran agilidad realizando el proyecto, vamos a tratar esta parte para completar e investigar sobre detección automática de bordes.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

2.1. Objetivos:

Esta parte se corresponde con el esquema de todos los puntos.

- Analizar el problema planteado por Rebeca y buscar una solución:
  - Primero tenemos que documentarnos y saber que son las lineas a detectar.
  - Buscar una solución para detectar dichas estrías pintadas, de forma automática, ya que hasta ahora era un problema manual que tenían que medir a mano.
  - Permitir que se pinten através de la aplicación para futuras imágenes.
  - Iniciar la forma de detección completamente automática.
- Crear un notebook para el procesado:
  - Crear un prototipo interactivo atrevés de los notebooks de jupyter.
  - Procesar la imagen para obtener la mascara o imagen binarizada con las líneas.
  - Detectar los segmentos que se corresponden con las líneas.
  - Juntar los segmentos para obtener líneas reales.
- Crear la aplicación:

- Crear una interfaz gráfica con la imagen,y botonería.
- Añadir a la gui el modo 1: Buscar estrías pintadas.
- Añadir a la gui el modo 2: Permitir pintar estrías.
- Añadir a la gui el modo 3 o a un notebook: Completamente automático.

2.2. Resumen

Analizar y resolver el problema:

El problema va a consistir en detectar las lineas que tienen ya pintadas, en las imagenes ,para poder automatizar dicho proceso, ya que los pasos anteriores eran:

- Pintar las estrías encima de la imagen.
- Obtener de las estrías ,de forma manual, su tamaño, angulo,dirección.

Y los pasos através de nuestra aplicación son:

- Abrir la imagen y seleccionar el color de las lineas
- Dar al botón de calcular las lineas y guardarlo: Tendíamos el CSV con los estadísticos y atributos de ellas.

Como podemos observar nuestros pasos nos devuelven las estrías de forma mucho mas cómoda y mas rápida que buscándolas a mano.

2.3. Crear un prototipo

En esta parte hemos pensado que seria mas cómodo, antes de ponernos a diseñar o implementar, comprobar que lo que tenemos pensado para resolver el problema que funcione.

Crear un notebook de jupyter no nos exige que programemos nada de la GUI, al ser interactivo, fuimos haciendo todos los pasos necesarios para resolver el problema y una vez conseguido. Como producto tenemos el núcleo de calculo y pasamos a hacer el diseño.

2.4. Crear la aplicación

Llegados a este punto ya tenemos todo lo necesario para hacer nuestras clases y nuestra aplicación.

Así que ya tenemos todo lo necesario y solo queda el producto entregable que sera la aplicación enlazada con la GUI junto con, la creación de botones, pestañas , ventanas con sus respectivas implementaciones y funcionalidades.

Conceptos teóricos

En esta sección vamos a centrarnos en explicar todos los conceptos necesarios para poder entender los pasos de los procesos realizados en la aplicación y saber en que se sustentan.

- Espacios de color: Las imágenes nos son mas que matrices de numeros que expresan, valores o poderaciones, distintos para representar los colores, la suma de todas ellas transmitido atrabes de un medio de reproducion, pantalla o similar, nos muestran las imagenes. Cada espacio de color tiene unas propiedades y unas ventajas, para obtener las característica que buscamos, con mayor o menor dificultad, por eso es necesario explicar los distintos espacios de colores.
- Transformada de Hough: Es un procedimiento matemático para buscar segmentos dentro de las imágenes binarizadas, dado que es complejo y lo usamos, es necesario explicar porque funciona y en que consiste.
- La reducción de grosor: Es un paso clave en el procesado porque gracias a esto obtenemos los resultados que mas se ajustan a la realidad. No es complejo pero si que es muy útil y sin este paso no serviría nada de nuestro procesado, su función es eliminar el ruido.
- Grafos: Gracias a la teoría de grafos podemos procesar muchos segmentos similares en uno solo, de forma sencilla y eficiente, por lo que es necesaria su explicación.
- Bordes: Al final las estrías que detectamos no son mas que bordes dentro de la imagen y todo lo usado es para la detección de estos por lo que es necesaria su explicación. Como dato curioso, para la resolución de nuestro problema lo que hemos hecho es reducirlo a un problema de detección de bordes.

3.1. Espacios de color

El color que percibimos en los objetos que nos rodean depende de la radiación reflejada en ellos. Según los estudios, nosotros como humanos tenemos un rango "de luz visible", ese rango son en verdad tres frecuencias diferentes dentro del rango 769THz a 384THz [9].
Por lo que en verdad una imagen que percibimos es la unión de las tres frecuencias diferentes y para poder simular este hecho las maquinas simulan esta capacidad innata de los humanos creando los espacios de color que son modelos matemáticos para representar en una maquina lo que se observa en la figura 3.1.

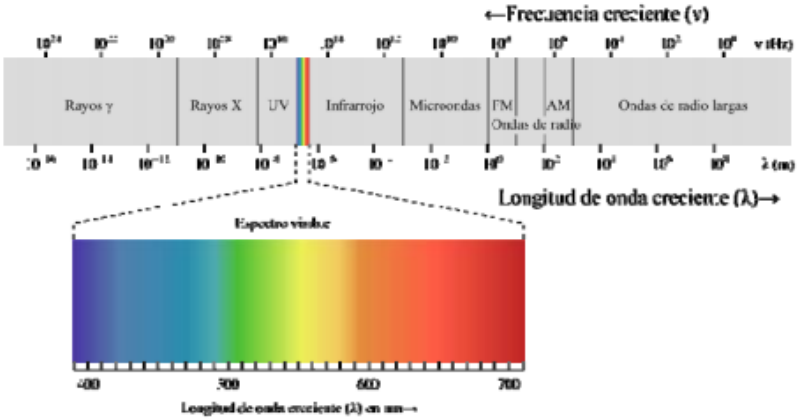


Figura 3.1: Frecuencias de luz visible [9]

RGB

El modelo RGB es usado por todos los sistemas digitales para la representación y captura de imágenes.

Se divide en tres canales como se muesra en la figura 3.2.

- R: canal del rojo (RED) contiene la intensidad de rojo de cada pixel
- G: canal del verde (GREEN) contiene la intensidad de verde de cada pixel
- B: canal del azul (BLUE) contiene la intensidad de azul de cada pixel

La combinación de estos colores crea todas la gama de colores representable. El valor de la intensidad de cada canal depende de la codificación usada para su representación (Ocho bits dan Dieciséis millones de colores) como se muestra en la figura 3.3.

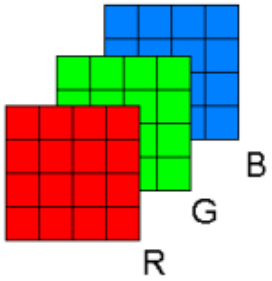


Figura 3.2: Los tres canales del espacio RGB [9]

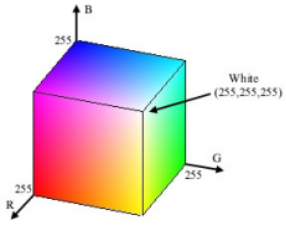


Figura 3.3: Representación del modelo RGB[9]

HSV

El modelo HSV [12] esta orientado a la descripción de los colores en términos mas prácticos para el ser humano que el RGB, los canales significan algo mas que la cantidad de cada color, por lo que es mas practico para el ser humano.Lo que se observa en la figura 3.4.

- H: (Matiz) que representa el tono o color.
- S: (Saturación) representa el nivel de saturación de un color.
- V: (Brillo) representa la intensidad lumínica.

Una ventaja con otros espacios de color parecidos es que este permite representar todas las combinaciones del espacio RGB.

CIELAB

El modelo lab [21],mas conocido como CIELAB,es otra forma de representar los colores. Este en concreto, se basa en como representa los colores el ojo

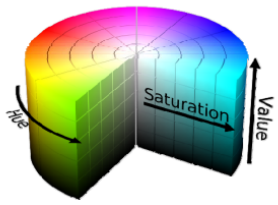


Figura 3.4: Representación del modelo HSV[9]

humano, la diferencia entre LAB y CIELAB, es que CIELAB utiliza raíces cubicas, mientras que LAB usa raíces cuadradas.

El modelo lo podemos ver en la siguiente figura 3.5.

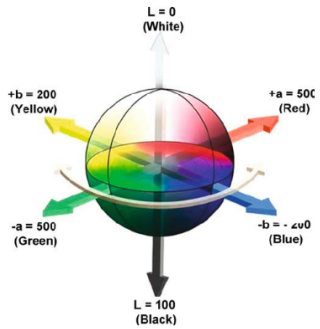


Figura 3.5: Representación del modelo CIELAB[10]

Componentes del modelo:

- L: Luminosidad de negro a blanco.
- A: Desde el color rojo al verde.
- B: Es la gradiente del color azul.

Ventajas:

- Corregir colores: Es mas rápido hacer una corrección del color que en otros espacios de color.
- Mas cantidad de colores: Con esta representación del color conseguimos representar mas numero de colores, incluso colores imaginarios.
- La perdida es mínima al cambiar a otro espacio de color.

3.2. Transformada de Hough

Uno de los puntos relevantes del proyecto es la detección de las líneas pintadas o detectadas por el algoritmo (modo automático) para ello vamos a usar una técnica que sirve para detectar formas, expresadas de forma matemática, dentro de imágenes.

Esta técnica fue inventada por Richard Duda y Peter Hart en 1972 pero diez años antes, Paul Hough propuso y patentó [3] la idea inicial de detectar líneas en la imagen. Mas tarde se generalizó para detectar cualquier figura.

Teoría

Normalmente para detectar figuras sencillas en una imagen primero hay que usar algún algoritmo de detección de bordes o una binarización de la imagen, quedándonos con la región de interés apropiada (los píxeles que forman las rectas) pero normalmente faltan píxeles por el ruido en la imagen.

Para ello el método de Hough propone solucionar el problema detectando grupos de puntos que forman los bordes de la misma figura y así conseguir unirlos creando la recta real a la que pertenecen.

Pseudocódigo Transformada de Hough

Como podemos ver en el siguiente pseudocódigo 7.

Algoritmo 1: Pseudocódigo de la transformada

```
Input: Imagen
Output: (list) de segmentos encontrados
foreach punto en la imagen do
    if punto (x,y) esta en un borde: then
        foreach angulo en ángulos Θ do
            Calcular ρ para el punto (x,y) con angulo Θ
            Incrementar la posición (ρ , Θ ) en el acumulador
Buscar las posiciones con mayores valores en el acumulador
return rectas Las rectas cuyos valores son los mayores en el
  acumulador
```

Limitaciones

Para que este proceso sea exitoso, los bordes del objeto deben ser detectados correctamente con un buen pre-procesado de la imagen y aparecer claramente las nubes de puntos que forman las rectas. Como se muestra en el figura 3.6.

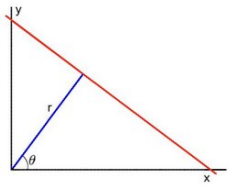


Figura 3.6: Líneas de Hough[2]

Transformada probabilística de Hough

Tal y como se explica en [6], la transformada probabilística de Hough es una version que se basa en que la detección de bordes o la producción de la imagen binaria que contiene el objeto, podría tener ruido y por lo tanto los píxeles que corresponden al ruido con la transformada normal podrían ser considerados como una recta, cuando en verdad es ruido.

Para que unos puntos sean considerados recta en la transformada probabilística, es necesario menos puntos que en la transformada normal de Hough. Pero este método penaliza a los puntos que se encuentran aleatoriamente por toda la imagen (ruido) frente a los que se localizan perfectamente agrupados formando las rectas. Un exceso de ruido en la imagen también haría este método inservible, pero para pequeñas cantidades lo hacen mas preciso que el método normal. Otra ventaja es que con este método obtenemos el segmento que necesitamos, no la prolongación de él hasta el infinito.

3.3. Skeletonize

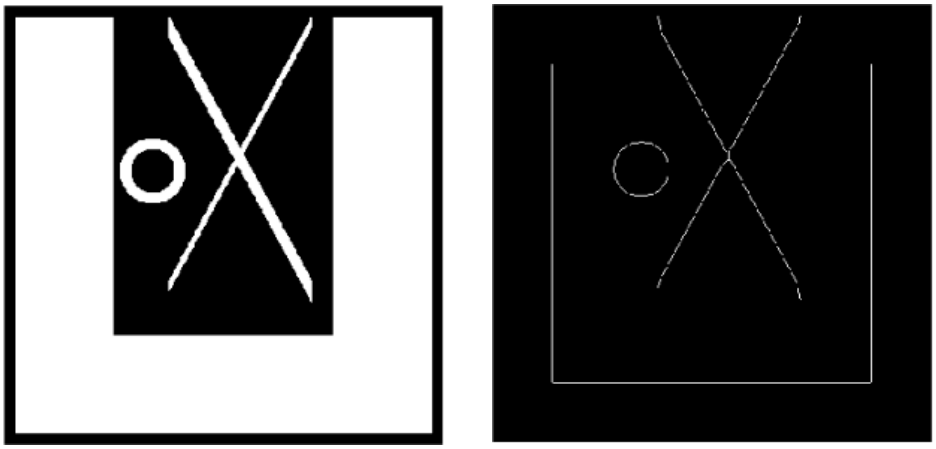
Tal y como se explica en [16], dentro del pre-procesado de la imagen, uno de los puntos clave para que nuestro método funcione. Después de su binarización y la detección de los bordes de la imagen a procesar, debemos reducir la región sobre la que aplicar la transformada de Hough, así conseguiremos que esta sea mas rápida, y detecte menos número de líneas imaginarias por cada línea real.

Esto lo conseguiremos usando una función de esqueletonizado que nos devuelve lo que su nombre indica el esqueleto de los bordes de la imagen reducidos a 1 pixel, Como podemos observar en la imagen 3.7.

3.4. Grafos

Introducción

La teoría de grafos[18] en un campo dentro de la computación y de las matemáticas, estudia las propiedades de los grafos que están compuestos por



(a) Original (b) Skeletonizada

Figura 3.7: Ejemplo de eskeletonize.

vértices y aristas, que comunican estos los vértices. Es una rama muy amplia pero solo vamos a centrarnos en uno de los problemas que podemos resolver gracias a estas teorías.

K-componentes

Una de las propiedades del grafo con sus componentes que nos indica cómo de fuertemente conexos están sus vértices, gracias a esta teoría nos aprovechamos que cuando un grafo esta dividido en clusters, agrupaciones fuertemente conexas de parte de sus vértices, por el problema de los k-componentes podemos saber que conjuntos de vértices forman los clusters por lo tanto, aplicado a nuestro problema los vértices que tengan aristas que los comuniquen pertenecerán a las mismas rectas y de cada conjunto de vértices sacaremos una recta.

3.5. Bordes

En lo que nos estamos basando, para poder resolver el problema, es en la detección de bordes mediante kernels conocidos. Para poder detectar donde esta el cambio de color en el histograma y esos cambios bruscos de pigmentación son los que nos indican que eso es un borde. Para ello se han desarrollado numerosos métodos matemáticos que pasando una mascara, por toda la imagen, nos devolverían otra imagen con los bordes llamada mascara.

Como en los anexos donde usamos toda la variedad de ellos que hemos encontrado aquí simplemente los nombraremos.

Kernel [26]: Un kernel es una matriz que se operara con una porcion de pixeles para suavizar o detectar algún elemento que sea de nuestro interés.

- Laplace [28].
- Prewitt [29].
- Scharr [5].
- Sobel [31].
- Roberts [30].
- Kirsch [27].
- Gabor [24].

Aparte de los métodos antes mencionados hemos combinado distintos formas para detectar bordes através de los autovectores [23] largos, de la matriz Hessiana [25].

Técnicas y herramientas

4.1. Gestores de tareas:

Trello

Es una pizarra virtual también conocida como canvas en la cual podemos organizar nuestros proyectos a través de su aplicación web de forma fácil e intuitiva desde cualquier equipo en el que introduzcamos nuestra cuenta ya que al estar en la nube no se pierde nuestra información ni aunque se degrade el sistema.
Podemos acceder a él através del siguiente enlace: https://trello.com/

Ventajas:

- Es muy rápido su aprendizaje y su uso así como su simplicidad.
- La hemos usado en el transcurso de la carrera por lo que ya estamos familiarizados con el entorno.

Desventajas:

- No esta integrado dentro de nuestro repositorio por lo que lo tendríamos que usar como una herramienta mas en la que al final duplicaríamos trabajo.

Version One

Es un gestor de tareas online en el cual podemos gestionar todas las tareas de nuestros proyectos así como realizar un seguimiento de forma visual del estado del proyecto y de las características del mismo.
Podemos Acceder a el através del siguiente enlace: https://www.versionone.com/

Ventajas:

- Podemos incluir código, por lo que es mas completo que otras herramientas de gestión de tareas.

Desventajas:

- Como antes hemos indicado en este caso también sería algo que nos duplicaría trabajo al no estar integrado en nuestro repositorio. Sería también una herramienta a parte que no se comunicaría con nuestro repositorio.

ZenHub

Es una herramienta similar a Trello que también es a modo de pizarra donde ver los cambios y el estado de un vistazo pero con algunas diferencias. Podemos Acceder a el através del siguiente enlace: https://www.zenhub.com/

Ventajas:

- La ventaja principal es que podemos integrarlo desde GitHub por lo que ya no sería necesario duplicar trabajo con el uso de una aplicación externa.

Desventajas:

- No podremos añadir código pero tampoco es algo catastrófico, ya que justo en el repositorio donde se integra la herramienta podemos visualizar dicho código.
- Por este motivo he decidido usar esta herramienta.

4.2. Gestores de versiones:

GitHub

Es un repositorio de versiones donde el código queda organizado por tareas (issues) y las versiones cada vez que hacemos un commit se actualiza las clases de código mostrando lo que ha cambiado. El software esta escrito en Ruby usando el framework Ruby on Rails.
Podemos Acceder a el através del siguiente enlace: https://github.com/

Ventajas:

- Es de los repositorios mas usados y esta basado en Git.
- El código es público y cualquiera que le interese te puede proponer cambios en el mismo, seguirte y ver el proyecto.
- Las distintas versiones del código están en la nube por lo que si se nos borra el contenido del disco duro aun así podremos recuperarlo.

Desventajas:

- También puedes tener proyectos privados solo con el modo de pago pero al no influir sobre este proyecto no pasa nada.
- Nos hemos decidido por este repositorio porque es uno de los mas usados y como se basa en Git al igual que muchos de sus competidores guarda mucha similitud con ellos. Al utilizar dicho repositorio de forma apropiada no deberíamos tener problemas si el día de mañana queremos usar otro.

Bitbucket

Este repositorio web también guarda nuestro código y nuestras iteraciones sobre el proyecto para así tener una visión mas completa sobre nuestro trabajo. Este software esta escrito en Python.
Podemos acceder a el através del siguiente enlace: https://bitbucket.org/

Ventajas:

- Es un repositorio muy usado y que esta basado en Git que es casi la referencia en este tipo de proyectos.
- Tiene cuentas gratuitas para proyectos privados y públicos.

Desventajas:

- No se puede incluir mas de 5 personas en los proyectos gratuitos.

4.3. Interfaz gráfica de usuario:

Tkinter

Es una librería que proporciona el diseño y visualización de interfaces de usuario en Python. Esta a su vez, esta basada en librerías de TK/TCL que están incluidas por en la propia instalación.

Ventajas:

- Es fácil de usar y es recomendable para el aprendizaje del lenguaje.
- Viene preinstalado con la distribución por lo que su uso es inmediato
- Al servir para aprendices y venir preinstalado podemos encontrar multitud de tutoriales y de documentación sobre ello.

Desventajas:

- Pocos elementos gráficos. Escaso control de las ventanas y bastante lento.

WxPython

Es una librería basada en otra importante que veremos mas adelante, también es multiplataforma y esta programada en C/C++, es mas nueva que Tkinter.

Ventajas:

- Es más difícil de usar que Tkinter pero aun así hay mucha documentación sobre ella.
- Dispone de gran cantidad de elementos gráficos por lo que es bastante potente aunque con alguna limitación respecto a otras.
- Permite hacer una barrera o separación entre el código Python y lo que es la interfaz.
- Cuenta con una gran comunidad de gente que lo usa y publica ejemplos y tutoriales.

Desventajas:

- La principal desventaja es que se actualizan las versiones mucho y para mantener una aplicación durante largo tiempo perdemos tiempo de.
- Es más complejo de usar que el anterior.

PyQt

Es más difícil de usar que WxPython y WxWidget pero da más control sobre los elementos gráficos y muchas librerías se basan en ello, por lo que se puede encontrar bastante información sobre esta librería.

Ventajas:

- Al ser tan usada, si instalamos Python desde Anaconda, que es un conjunto de librerías y aplicaciones de Python, ya tendríamos PyQt4 por defecto instalado.
- Podemos usar un IDE con el que estamos muy familiarizados en la carrera y que funciona muy bien (Eclipse) pero tenemos que instalar un plugin para Python (PyDev), ya que es un IDE basado en java también deberíamos instalar Java 8.

Desventajas:

- Es mas difícil de entender y comprender que los anteriores pero quedan mas limpias las interfaces.
- Si somos puristas e instalamos Python solo, sin IDE ni nada no vendría instalado, pero si usamos Anaconda si que vendría instalado.

WxWidgets:

Como hemos mencionado anteriormente, es muy parecido a WxPython por lo que no vamos a entrar en detalle. También está programado en C/C++ y es multiplataforma, por lo que da un aspecto de comportamiento nativo.

4.4. Plantillas

Las plantillas sirven para generar código con la sustitución de los valores dentro de sus variables y así tener siempre un código con los valores en la ejecución.
Nosotros las vamos a usar en combinación con LaTeX para generar el pdf con el informe de las estadísticas de la ejecución del algoritmo que detecta las líneas.
Como podemos observar hay muchos frameworks para usar las plantillas pero hemos elegido comparar varios usados otros años por compañeros.

Mustache

Pystache es una implementación de Mustache en Python para el diseño de plantillas, está inspirado en Ctemplate [7] y et [14]. En estos tipos de lenguaje no se puede aplicar lógica de aplicación simplemente son para una presentación más fluida.

Cuadro 4.1: Tabla comparativa de herramientas

Biblioteca	Lenguaje	Licencia	Variables	Funciones	Include	Inclusiones condicionales	Bucles	Evaluacion	Asignaciones	Errores y excepciones	Plantillas naturales	Herencia
Jinja2	Python	BSD	SI	SI	SI	SI	SI	SI	SI	SI	SI	SI
Mustache	+30	MIT	SI	SI	SI	SI	SI	NO	NO	SI	SI	NO

Ventajas:

- Tiene documentación online.
- Está disponible para una gran variedad de lenguajes de programación.
- Si necesitásemos hacer documentos XML seria la herramienta perfecta.

Desventajas:

- Carece de ejemplos o tutoriales de cómo usarlo con LaTeX en su versión de Python.

Jinja2

El nombre de la librería viene del templo Japonés Jinja.
Es una librería para escribir plantillas, para Python, funciona con las últimas versiones de Python, también es una de las más usadas librerías está inspirada en Django.

Ventajas:

- Viene preinstalado con Anaconda ya que es uno de los más utilizados.
- Fácil de usar y de pasar los parámetros.
- Fue creada para Python2 pero también funciona en Python3

Desventajas:

- No esta implementado más que para Python.

Comparación de las herramientas

Esta comparación son dos filas, como se observa en la Tabla 4.1 sacada de la tabla comparativa de herramientas obtenida en wikipedia [22].

4.5. IDE:

Un IDE es una aplicación para poder desarrollar código y ejecutar pero facilita la navegabilidad de por los paquetes y poder debugear de forma mas eficiente que sin él.
Es una herramienta indispensable para el desarrollo de software, un IDE se compone normalmente de un editor de código fuente, un depurador, la mayoría de ellos tienen también autocompletado, compilador, indentador de código e interprete.

Ventajas:

- Maximizar la productividad.
- Unificar todo el ciclo de desarrollo en una herramienta.
- Algunos soportan múltiples lenguajes.
- Sin ello es difícil leer código y editarlo.

Eclipse:

Es una aplicación software con muchos plugins y herramientas para el desarrollo de software, esta basado en Java, para su ejecución e instalación es necesario tener Java instalado.
Fue desarrollado por IMB en un principio, pero ahora esta desarrollado por la Fundación Eclipse, una fundación independiente y sin animo de lucro.

Podemos acceder a la pagina atraves del siguiente enlace `https://eclipse.org/home/index.php`

Ventajas:

- Lo hemos usado en múltiples asignaturas durante la carrera.
- Es gratuito por lo que podemos descargarlo sin problemas.
- Es un programa muy completo y que es sencillo.

PyDev:

Es un plugin para poder usar el IDE de Eclipse para programar en Python, Jython e IronPython. Se instala desde eclipse desde: Help, Install New Software.

Podemos encontrarlo a atraves del siguiente enlace [15]

4.6. Modelado

El modelado es la creación de diagramas que nos explican que apariencia y comunicación van a tener los objetos entre ellos dentro de nuestra aplicación.

Es necesaria una herramienta especializada para ello, hay muchas en el mercado, pero en las asignaturas de la carrera hemos usado una con la que ya estamos familiarizados.

Aun así vamos a analizar todas las disponible y comparar algunas de ellas.

Astah

Es una herramienta de modelado UML para creación de diagramas orientados a objetos. Podríamos hacerlos dibujando pero es mas preciso y mas profesional usar una herramienta que esta pensado para ello. Podemos encontrar la herramienta através del siguiente enlace [17].

XML

Para guardar los nombres de los ficheros que se generan al guardar un proyecto, vamos a usar un archivo XML, que contenga los nombres de los ficheros que se generan.
XML significa Extensible Marking Language, también otra de sus propiedades es que es un lenguaje sencillo que estructura el contenido por medio de:

- Cuerpo: Es obligatorio en su sintaxis, normalmente contiene un elemento raíz del que cuelgan todos los demás.
- Elementos: Pueden tener mas elementos, cadenas de caracteres o nada.
- Atributos: Los contienen los elementos y son sus características o propiedades.
- Secciones CDATA: Para especificar caracteres especiales sin que rompan la estructura XML.

Para mas informacion respecto a ello podemos encontrarlo en este enlace: [19].

En nuestro caso lo vamos a usar como un fichero donde se guardan todo lo que contendría un proyecto.

Conclusión

Después de analizar dos opciones de herramientas y viendo que todas ellas tienen gran similitud ,vamos a usar la que ya conocemos como funciona, que nos va a servir para todos los diagramas.

4.7. OCR

Para entrar en materia adecuadamente para explicar la librería usada mas adelante vamos a introducir lo que es un OCR [20] y en que consiste.
Un OCR es un proceso dirigido a buscar de forma óptica un reconocimiento de caracteres en imágenes, es decir hacer una extracción de características de la imagen en cuestión, devolviendo el texto que contiene la imagen.

Ventajas:

- Poder pasar documentos manuscritos a archivos de texto a ordenador.
- Facilitar trabajo que antes hacían humanos de forma mas rápida al realizarlo un ordenador.
- Poder automatizar muchos sistemas.

Desventajas:

- Depende de que dispositivo tome la imagen puede meter ruido en ella complicando o inutilizando el posterior reconocimiento.
- La distinta distancia entre caracteres puede variar en el reconocimiento produciendo errores.
- En un escenario idílico funcionaria perfectamente pero ya sabemos que dichas situaciones nunca ocurren.

Pasos del algoritmo:

- Binarizar la imagen.
- Fragmentación y/o segmentación.
- Skeletonizar los componentes.
- clasificar lo detectado.

Tesseract

Como podemos ver Tesseract [13] es un OCR de software libre.
Originalmente estaba hecho en C pero se migró a C++ en 1998, mas tarde fue concebido como Open Source en el 2005 al retomar el proyecto la universidad de Nevada (Las Vegas) en EEUU. Finalmente en 2006 fue retomado por Google.

Lo potente de esta librería es que si tus necesidades fuesen otras, podrías reentrenar la red para que reconozca nuevos idiomas o nuevos estilos de letras.

4.8. Logger

Es una herramienta que crea un fichero de registro, el cual almacena los mensajes de error que aparecen en la aplicación, facilitando el mantenimiento de esta.

Es una herramienta muy útil, ya que controlar todas las posibles excepciones es muy complicado, porque no sabemos por donde podría romper el código, de esta forma sabemos donde ha roto o que lo ha causado.

Logging

Es la implementacion de un logger para python [8] en este logger como en casi todos, podemos clasificar los mensajes de error en función de su gravedad en distintos niveles.

- Info: Para mensajes importante de información.
- Warning: Para niveles que son avisos de que no se ha hecho lo que se quería pero no son graves.
- Error: Para errores que han echo un mal funcionamiento del programa.
- Critical: Para mensajes que son críticos en nuestro programa.

4.9. Ejecutable

Para distribuir el software, deberíamos poder ser capaces de crear un .exe, pero no lo hemos sido. Después de probar todas las herramientas que hacen dicha función, por que han salido las ultimas versiones hace relativamente poco tiempo, estas que hay disponibles quedan muy desactualizadas, hemos tenido que resolver el problema de otra forma.

Py2exe

Esta herramienta [4], es una extension que se encarga de empaquetar un script de Python en un ejecutable .exe, para poder distribuir la aplicación de forma que solo necesitemos el .exe para ejecutar todo, sin necesidad de tener Python instalado.

No la hemos podido usar ya que su última actualización es para Python 2.7.

Py2app

Esta herramienta es casi idéntica a la anterior pero solo esta disponible para MAC OS.

No la hemos podido usar ya que el cliente no va a usar MAC OS.

Pyinstaller

Esta herramienta [1], si que hemos podido ser capaces de usarla, pero su última version esta disponible para Python 3.4.

Indican que para Python 3.5 podría funcionar pero no es así, al ejecutarlo salían errores pero buscándolos uno a uno, en su repositorio de GitHub, los podíamos corregir, después de sucesivos errores terminamos en un bucle, corregíamos uno y salia el anterior y así sucesivamente de forma recursiva. Finalmente algunos de los que salían, no se pudieron corregir, no la hemos usado y no hemos sido capaces de solucionarlos.

Porque según lo investigado y visto en todo tipo de foros, llegamos a la conclusión que con 3.5 no le funciona a nadie. No obstante en su siguiente actualización puede que lo corrijan y funcione para 3.5.

cxFreeze

Esta herramienta, es muy parecida a la de Py2exe, pero su última versión sigue estando únicamente estable para Python 3.4. Si buscamos mas en su repositorio vemos que menciona una version 3.5, pero no funciona ya que directamente no lo instala, y no solo a mi, encontré mas gente en la misma situación.

Conclusión:

Hemos seguido otras formas de conseguir lo que queremos, Python es un lenguaje interpretado y necesitamos tener Python 3.5.

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.

Aspectos relevantes del desarrollo del proyecto

5.1. Entorno de desarrollo

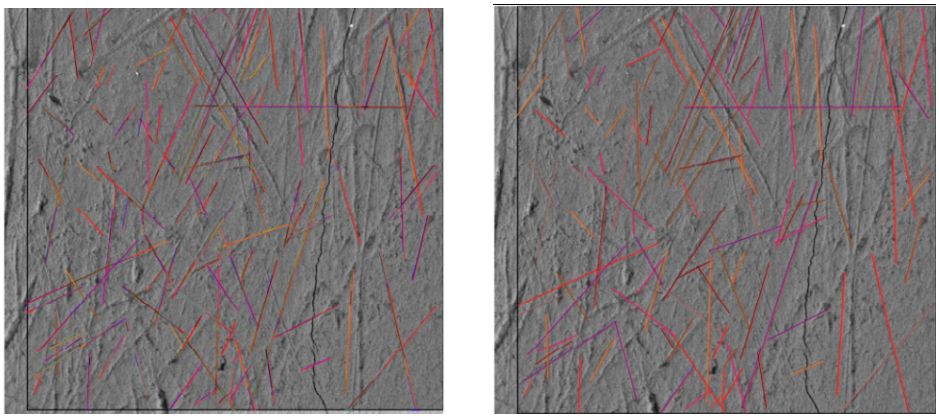
Como entorno de desarrollo de los prototipos hemos utilizado Jupyter, ya que en sus notebooks interactivos puedes ejecutar directamente código Python como si fuese un intérprete. Y Eclipse mas PyDev, como IDE, para programar en clases y paquetes de forma mas cómoda.

Ventajas

- He podido añadir widgets para calibrar en buen grado las funciones que hemos utilizado. Gracias a estos widgets podemos dar valores e ir viendo como cambia la salida de la función de forma interactiva. Como podemos observar en la imagen 5.8.

Figura 5.8: Ejemplo de un widget sobre la función de Hough

- Su rápida visualización sin tener grandes conocimientos de interfaz gráfica ha sido un gran apoyo para poder visualizar desde el principio las imágenes procesadas y como quedaban. Como podemos observar en la imagen 5.9.
- Desde el propio entorno puedes ejecutar, no solo código estructurado en script, sino también código estructurado en clases y llamadas a métodos



(a) Líneas antes de unir (b) Líneas después de unir

Figura 5.9: Ejemplo de una visualización del resultado intermedio de las funciones.

es como un IDE pero con limitaciones. Como podemos observar en la imagen 5.10.

```
In [32]: img=leerImagen()
         distance_red=distanciaAlRojo(img)
         imgBin=binarizar(distance_red)
         imgBinCrop,imgCrop=cropImg(imgBin,img)
         sinRuido=reducirGrosor(imgBinCrop)
         lines=proHough(10,5,11,sinRuido)
         G=nx.Graph()
         G=combina(8,4,lines,G)
         k_components = apxa.k_components(G)
         segmentosDeVerdad=segmentosVerdad(k_components,lines)
         segmentosDeVerdad

Out[32]: [((434, 12), (310, 323)),
          ((485, 467), (434, 643)),
          ((469, 251), (337, 15)),
          ((531, 293), (513, 79)),
          ((722, 182), (337, 184)),
          ((463, 340), (294, 19)),
```

Figura 5.10: Ejemplo de una ejecución.

- Multitud de librerías y funciones que en entornos parecidos como Matlab serian de pago y aquí al ser software libre el ejemplo anterior lo resume en una librería Numpy [11].

5.2. Procesado imagen

Para llegar a conseguir calcular las líneas que había pintadas en las imágenes, tube que realizar una serie de pasos que vamos a resumir en tres etapas.

Binarización

Partiendo de una imagen que solo tenía líneas en rojo pintadas encima de las estrías producidas por el desgaste y lo demás de la imagen en escala de grises. Lo primero fue leer la imagen a trabes de las funciones ya programadas en la librería de Scikit-Image(skimage) [16]. Una vez que tenemos la imagen guardada en el espacio de color RGB podemos empezar el procesado quedándonos con el canal Rojo.

Calculamos la distancia de cada pixel de la imagen al color deseado, pasamos la imagen de distancias a blanco y negro y así tendremos un valor entre 0 y 256 en cada pixel correspondiente a la distancia al color. Cuanto mas alejado mas negro y los que sean del color, en blanco.

Para que la diferencia sea blanco o negro binarizamos, la imagen con un valor umbral así los valores de la distancia que sean mayores que el umbral pasaran a valer máximo y los que no consigan pasar el umbral serán los bordes blancos 5.11.

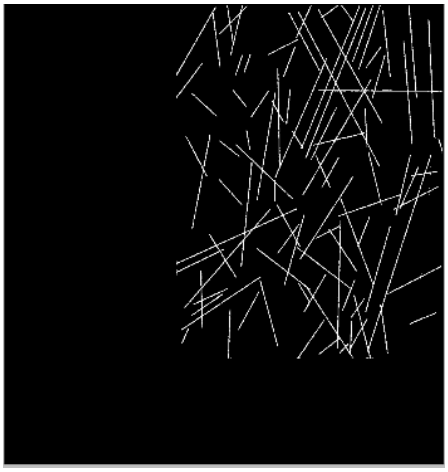
Obtener segmentos

Partimos de la imagen binarizada y lo primero es reducir el grosor de las líneas detectadas a un pixel, eso lo conseguimos llamando a la función skeletonize que nos devuelve la imagen con las líneas de un pixel (así no acumulamos errores y es mas rápida la búsqueda de rectas).
Seguidamente llamamos a la función «probabilistic hough line» que nos va a encontrar segmentos que formaran las líneas el funcionamiento ha sido explicado en el apartado (conceptos teóricos). Como podemos observar en la figura 5.12.

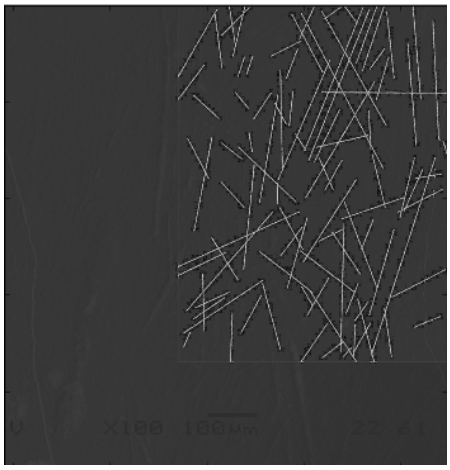
Procesado de segmentos

Llegados a este punto, lo que tenemos son:

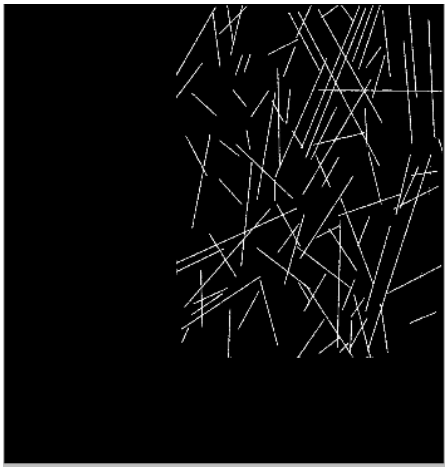
- Muchos segmentos que forman las líneas reales y tenemos que unirlos. Para ello vamos a usar la teoría de grafos añadiendo los segmentos a un grafo.
- Para unir dos segmentos tiene que cumplirse que la distancia mínima entre sus extremos sea menor que un umbral y si pasa este punto comprobaremos que el angulo que forman entre ellas sea menor a otro umbral.
- Si cumplen las dos condiciones añadiremos un camino al grafo desde la recta uno a la recta dos como se ve en la figura 5.13.



(a) Original



(b) Distancia al rojo

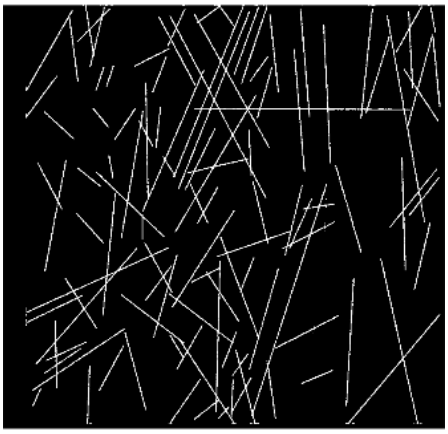


(c) Imagen binarizada

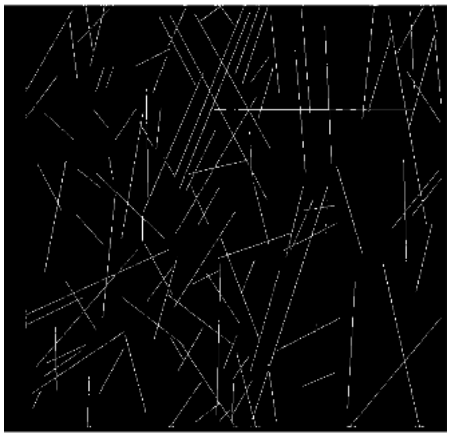
Figura 5.11: Resumen visual binarización.

Recuperación de líneas

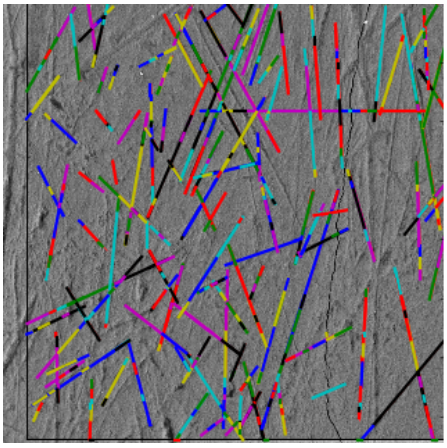
Ahora lo que tenemos es un grafo con clusters, ya que cada cluster se identifica con únicamente una recta y tendremos tantos como rectas. Un problema de grafos es el problema de las k-componentes pero a nosotros solo nos interesan las 1-componentes del grafo ya que cada grupo de estos segmentos cercanos se corresponde con una recta real. Devolvemos la combinación de los segmentos mas relevantes de cada cluster y estos se convierten en nuestra buscada linea real 5.14.



(a) Binarizada

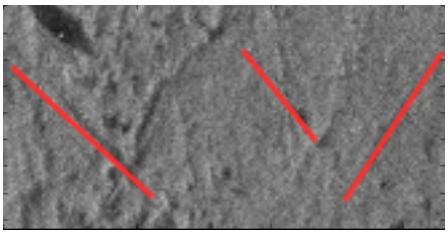


(b) líneas a 1 pixel

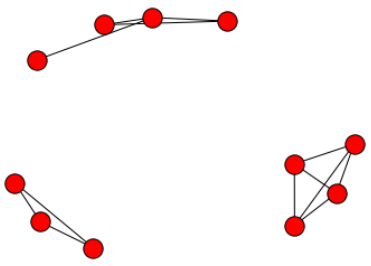


(c) Imagen original con segmentos

Figura 5.12: Resumen visual Obtener Segmentos.



(a) Imagen con segmentos en rojo.



(b) Grafo de clusters de segmentos.

Figura 5.13: Resumen visual procesado de Segmentos.

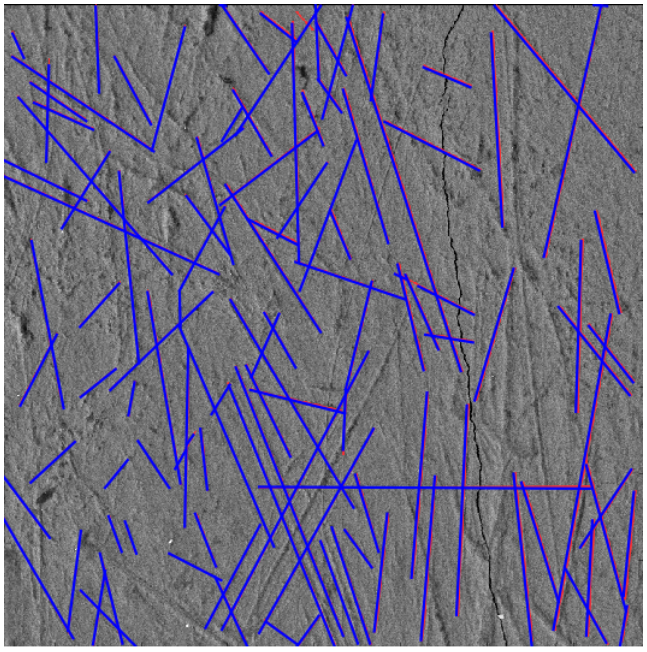


Figura 5.14: Líneas obtenidas después de procesar el grafo

Resumen pasos

- Binarizar la imagen para solo quedarnos con los objetos de interés
- Obtener segmentos que forman trozos de las líneas.
- Añadir caminos entre las líneas cercanas en un grafo.
- Obtener los grupos de líneas próximas y devolver la recta que las una.

5.3. 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 5.15.

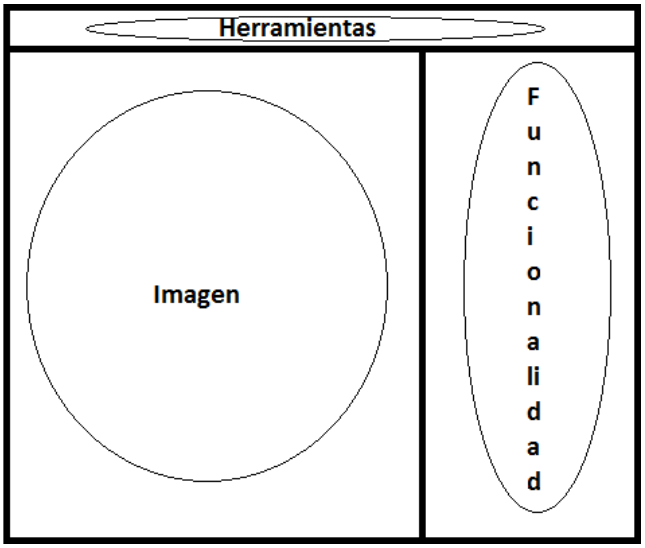


Figura 5.15: Diseño de la interfaz de usuario

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 5.16.

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

Barra de herramientas

Es una sección de la interfaz que nos va permitir de momento la carga de imágenes para su procesado 5.17.

Barra de herramientas de la imagen

Es una sección de la interfaz que nos permitirá manipular la imagen para realizar estas acciones 5.18.

- Volver al principio.
- Retroceder un paso.
- Avanzar un paso.

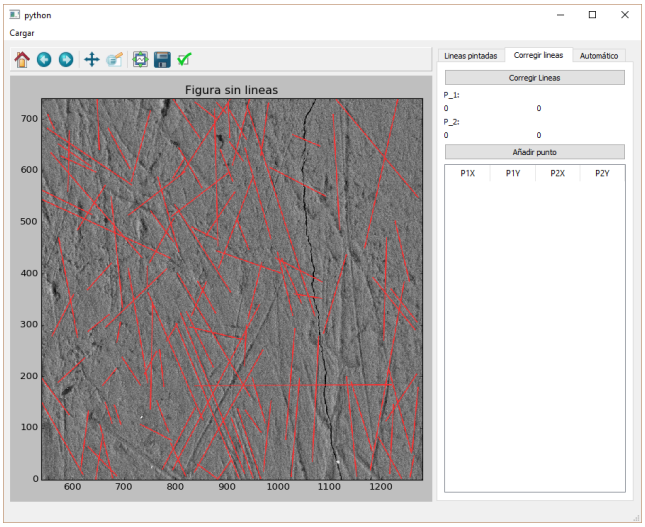


Figura 5.16: Diseño de la interfaz de usuario

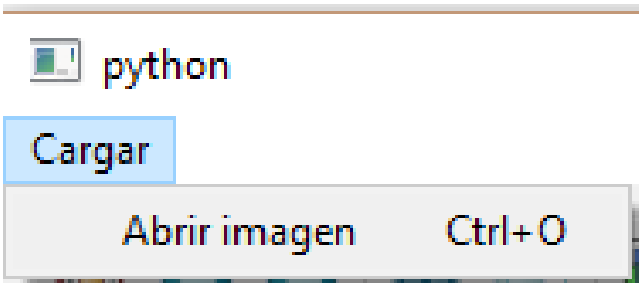


Figura 5.17: Barra de herramientas de la aplicación

- 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.

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 5.19.

Figura 5.18: Barra de herramientas de la imagen

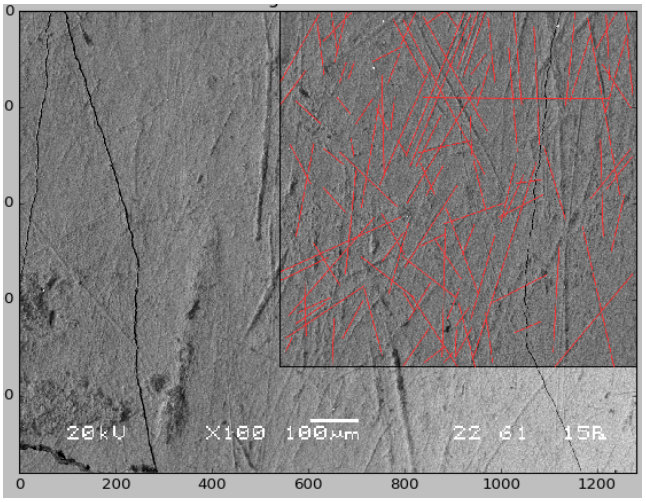


Figura 5.19: FigureCanvas de la imagen

Pestañas

Sección de la imagen donde estarán implementadas las funcionalidades para visualizar las líneas que son detectadas por el algoritmo en la imagen 5.20.

- 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.

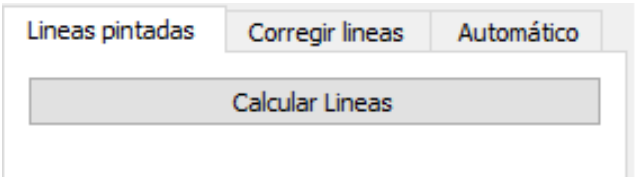


Figura 5.20: Pestañas

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] Giovanni Bajo. Python script to exe, 2015,. [Ejecutable].

[2] G. Bradski. Transformada de hough, 2000. [imagen de la documentacion de OpenCv].

[3] Paul Hough. Patente de paul hough line transform, 1962. [Internet; descargado 26-septiembre-2016].

[4] TijnGommans JimmyRetzlaff, PeggyJoyc. Python script to exe, 2014,. [Ejecutable].

[5] Johncostela. Scharr, 2016. [Kernel de Scharr].

[6] Nahum Kiryati, Heikki Kälviäinen, and Satu Alaoutinen. Randomized or probabilistic hough transform: unified performance evaluation. *Pattern Recognition Letters*, 21(13–14):1157 – 1164, 2000. Selected Papers from The 11th Scandinavian Conference on Image.

[7] dragotin OlafvdSpek. Template, 2005. [Herramienta].

[8] Python. Registrador de errores, 2015. [logger].

[9] César Represa. Manual de hardware de aplicación especifica, 2015. [imágenes del manual].

[10] D V Parwate S K Shukla. Radiosterilization of fluoroquinolones and cephalosporins:, Feb 2009. [AAPS PharmSciTech].

[11] Scypi.org. Biblioteca de funciones python, 2013. [Biblioteca de funciones].

[12] Alvy Ray Smith. Modelo hsv, 1978. [Modelo de color].

[13] Ray Smith. Ocr, 2006,. [Herramienta].

[14] suyu34. Et-template, 2015. [Herramienta].

[15] Aleks Totic. Plugin de eclipse para desarrollo en python, July 2003,. [Plugin].

[16] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python, 2014. [Internet; descargado 26-septiembre-2016].

[17] Change Vision. Modelado UML, 2006,. [Herramienta].

[18] Wikipedia. Teoría de grafos, -,. [Modelo matemático].

[19] Wikipedia. Xml, -,. [XML].

[20] Wikipedia. Reconocimiento óptico de caracteres. [Herramienta].

[21] wikipedia. Modelo lab, 1978. [Modelo de color].

[22] Wikipedia. Comparativa de template engines, 2007,. [Tabla].

[23] Wikipedia. Eigenvectors, 2016. [Matrix eigenvectors].

[24] Wikipedia. Gabor, 2016. [Kernel de Gabor].

[25] Wikipedia. Hessiana, 2016. [Matriz Hessiana].

[26] Wikipedia. Kernels, 2016. [Tipos de Kernels].

[27] Wikipedia. Kirsch, 2016. [Kernel de Kirsch].

[28] Wikipedia. Laplace, 2016. [Kernel de Laplace].

[29] Wikipedia. Prewitt, 2016. [Kernel de Prewitt].

[30] Wikipedia. Roberts, 2016. [Kernel de Roberts].

[31] Wikipedia. Sobel, 2016. [Kernel de Sobel].