UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Sistema de reconocimiento automático en arqueobotánica
Documentación Técnica

Presentado por Jaime Sagüillo Revilla
en Universidad de Burgos — 29 de junio de 2017
Tutor: D. Álvar Arnaiz González, Dr. José Francisco Díez Pastor y D.ª Virginia Ahedo García

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Planificación

A.1. Introducción

Para llevar a cabo este proyecto vamos a aplicar una metodología llamada *Scrum*. *Scrum* es una metodología de desarrollo software ágil, es decir, durante cada *sprint*[1], generalmente cada semana, se asignarán unas determinadas tareas a cumplimentar, con un producto como consecuencia de estas tareas. Al final de cada *sprint* se realizará una reunión junto a los tutores para validar los avances realizados y determinar las tareas a realizar durante el siguiente *sprint*.

Además, toda la comunicación se realizará a través de las *issues* de *GitHub* en `www.github.com/jasag/Phytoliths-recognition-system`.

A.2. Planificación temporal

En esta sección podremos ver la planificación del proyecto subdividida en *sprints*, como previamente comentaba. En cada uno de los sprints se detalla las tareas a realizar, algunos detalles descriptivos y un gráfico *burndown*.

Sprint 0

Estas son las tareas a realizar durante este *sprint* 0:

- Probar LaTeX.
- Gestor de tareas/versiones: *Github* y *Zenhub*.
- Instalar *Anaconda* y *Jupyter*.

[1] *Sprint*: es el período en el cual se lleva a cabo el trabajo en sí [5].

Figura A.1: Burndown del *sprint* 0

- Leer los artículos propuestos por los tutores.
- Comenzar a probar algunos algoritmos de binarización.

Como se puede ver las tareas a realizar son básicas, puesto que es el *sprint* 0 y es un *sprint* de mera adaptación al entorno de trabajo. La única tarea que supone un esfuerzo de comprensión mayor es la lectura de los artículos propuestos sobre trabajos relacionados o con una problemática similar a la nuestra. A continuación, en la figura A.1, se muestra el diagrama *burndown* de este *sprint*.

Sprint 1

Estas son las tareas a realizar durante esta *sprint* 1:

- Documentar lo realizado durante el *sprint* 0.
- Documentar lo que se irá realizando durante este *sprint* 1.
- Continuar probando con algoritmos de procesamiento de imágenes.
- Probar una aproximación con clasificadores al problema.

Puesto que en el *sprint* anterior no se documentó lo realizado, durante éste se pretende documentar todo lo realizado durante el *sprint* anterior y éste. Además de continuar probando con algoritmos de procesamiento de imágenes y comenzar a probar con la aproximación al problema mediante clasificadores.

En este *sprint* me vi desbordado de trabajo debido a la subestimación del esfuerzo a empeñar en las distintas tareas. No siendo capaz de comenzar

Figura A.2: Burndown del *sprint* 1

a probar una aproximación con clasificadores. Por ello la tarea «Probar una aproximación con clasificadores al problema» se vio movida al siguiente *sprint*.

A continuación, en la figura A.2, se muestra el diagrama *burndown* de este *sprint*. El cual tiene dicho aspecto debido a que muchas de las tareas se trabajaron de manera paralela, no siendo acabadas hasta el final del *sprint*. Y, además, algunas de las tareas no fueron cerradas cuando se debió, aspecto que se corregirá en los siguientes *sprints*.

Sprint 2

Estas son las tareas a realizar durante este *sprint* 2:

- Probar una aproximación con clasificadores al problema.
- Aplicación del método «Non-maximum suppression» sobre el clasificado.

Puesto que la aproximación mediante reconocimiento de imágenes no reflejaba unos resultados muy positivos, durante la reunión mantenida con los tutores se decidió el uso de una técnica distinta. Nos referimos a la utilización de un clasificador, junto a un descriptor visual.

Debido a que todavía no se poseían suficientes imágenes para el estudio del problema mediante esta técnica, lo que se decidió es aplicarla sobre otro problema de características similares, como es el reconocimiento de caras en imágenes. Con unos resultados bastante positivos debido a distintos razonamientos explicados en la Memoria, sección de Aspectos relevantes del proyecto.

A continuación, en la figura A.3, se muestra el diagrama *burndown* de este *sprint*.

Figura A.3: Burndown del *sprint* 2

Sprint 3

Estas son las tareas a realizar durante este *sprint* 3:

- Reorganizar los *Jupyter Notebooks*.
- Probar distintos clasificadores y métricas.
- Enviar fotos rotadas al clasificador.

Durante este *sprint*, primero, se reorganizó la estructura del proyecto. Aportando mucho más orden y claridad a nuestro proyecto. Después, se introdujeron múltiples clasificadores y métricas, los cuales introduciré en mayor medida en la memoria, como *Random Forest* [1] o *Gradient tree boosting* [2]. Por último, se enviaron imágenes rotadas al clasificador, con el fin de poder analizar una posible problemática.

A continuación, en la figura A.4, se muestra el diagrama *burndown* de este *sprint*.

Sprint 4

Estas son las tareas a realizar durante este *sprint* 4:

- Implementación de *Data Augmentation* en nuestro conjunto de entrenamiento.
- Implementación de controles de usuario.

Figura A.4: Burndown del *sprint* 3

Figura A.5: Burndown del *sprint* 4

Durante este *sprint* se aplicó en nuestro conjunto de entrenamiento la técnica *Data augmentation*. Esta técnica nos permitió aumentar el tamaño de nuestro conjunto de entrenamiento enormemente.

Además, se realizó un *notebook*[2], con controles de usuario, los cuales nos permiten escoger entre clasificadores, imágenes y probabilidades. Permitiendo la continua interacción entre el usuario y la clasificación de una imagen, sin la necesidad de modificar el código por parte del usuario del notebook para cambiar entre las distintas opciones.

A continuación, en la figura A.5, se muestra el diagrama *burndown* de este *sprint*.

[2]Siempre que nos referimos a un *notebook*, a lo que nos referimos es a un *Jupyter Notebook*

Figura A.6: Burndown del *sprint* 5

Sprint 5

Estas son las tareas a realizar durante este *sprint* 5:

- Implementar un *file chooser*
- Añadir más clasificadores.
- Correciones en la documentación.
- Estudiar como implementar un etiquetador de imágenes.

Durante este *sprint* se añadieron los clasificadores que deseábamos, es decir, un clasificador bayesiano y un clasificador mediante regresión logística. Además, se añadió un *file chooser* que nos permitiría, desde ese momento, escoger la imagen que deseemos dentro de nuestro sistema operativo. En cuanto a la documentación, se corrigió toda la realizada hasta ese momento. Y, por último, se hizo un estudio básico sobre como implementar un etiquetador de imágenes mediante un *Widget* de *Python*. Aunque, esta última tarea no tuviese ningún producto resultante en este *sprint*.

A continuación, en la figura A.6, se muestra el diagrama *burndown* de este *sprint*.

Sprint 6

Estas son las tareas a realizar durante este *sprint* 6:

- Estudiar los *Widgets* personalizados de *Jupyter Notebook* e *Ipython*.

Figura A.7: Burndown del *sprint* 6

Aunque este *sprint* se encuentre compuesto por una única tarea, no era menos complejo por ello. El objetivo de este *sprint* era obtener un *Widget* capaz de etiquetar imágenes. Pero en la realización de este se encontraron multiples problemas. Obteniendo como producto resultante tres posibles alternativas con aspectos a corregir.

Por lo tanto, en la figura A.7 mostramos el diagrama *burndown*, poco esclarecedor, de este *sprint*.

Sprint 7

Estas son las tareas a realizar durante este *sprint* 7:

- Estudiar *Bag of Words*.
- Añadir mayor parametrización al *Jupyter Notebook UI*.
- Corregir *bugs* del Widget previamente implementado.

Durante este *sprint* se consiguió, en primer lugar, corregir una de las alternativas del etiquetador de imágenes, o *Widget*, desarrolladas durante el *sprint* anterior. Además, se corrigieron y añadieron múltiples parámetros en el *Jupyter Notebook UI* y en las clases utilizadas por este *Notebook*. Y, por ultimo, se realizo un estudio sobre un modelo ampliamente usado para tareas de clasificación, llamado *Bag of Words*.

A continuación, en la figura A.8, se muestra el diagrama *burndown* de este *sprint*.

Figura A.8: Burndown del *sprint* 7

Sprint 8

Estas son las tareas a realizar durante este *sprint* 8:

- Implementar la funcionalidad de obtención de imágenes en el etiquetador de imágenes, o *Widget*.
- Mejorar la interfaz del etiquetador de imágenes.
- Crear un primer prototipo de interfaz de usuario.

Durante este *sprint* se realizó un primer prototipo de interfaz de usuario. Partiendo de este prototipo, se mejoró la interfaz del etiquetador de imágenes. Consiguiendo, así, una interfaz adecuada para el cliente. Además, se implemento la funcionalidad que nos permitiría obtener una imágen resultante de cada etiqueta realizada en las distintas imágenes.

A continuación, en la figura A.9, se muestra el diagrama *burndown* de este *sprint*.

Sprint 9

Este *sprint* tendrá una duración de dos semanas. Debido a la carga de trabajo asociada a este *sprint* y al ser días no lectivos por las vacaciones de Semana Santa.

Estas son las tareas a realizar durante este *sprint* 9:

- Añadir un texto a cada etiqueta que realizamos en una imagen.
- Añadir notificaciones al usuario en la carga y guardado de imágenes.

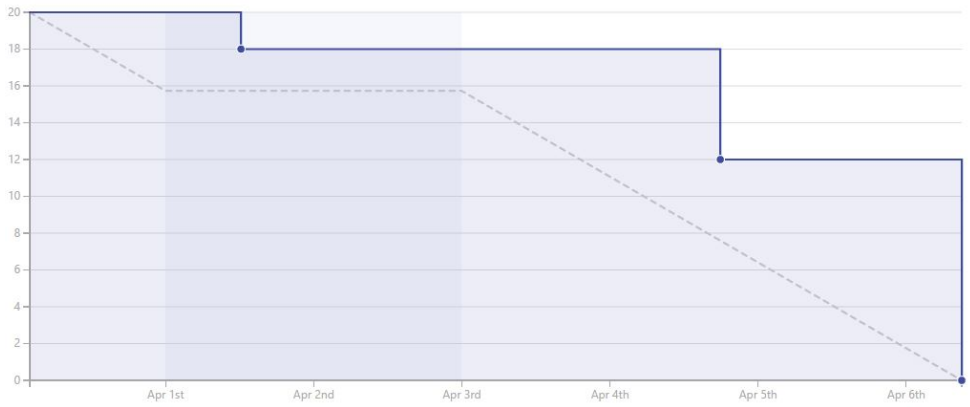

Figura A.9: Burndown del *sprint* 8

- Guardar las coordenadas de las etiquetas de cada imagen.
- Cargar las etiquetas de una imagen que haya sido previamente etiquetada.
- Controlar que el usuario no cree etiquetas en el SVG pero fuera de la imagen.
- Añadir la posibilidad de eliminar etiquetas previamente realizadas.
- Añadir un botón que permita guardar imágenes como negativos.
- Corregir los *notebooks* creados para la técnica *Bag of Words*.

Durante este *sprint* se completaron todas las tareas asignadas, excepto la corrección de los *notebooks* para la técnica *Bag of Words*. Los cuales no se revisaron porque no se utilizarían por el momento.

A continuación, en la figura A.10, se muestra el diagrama *burndown* de este *sprint*.

Sprint 10

Estas son las tareas a realizar durante este *sprint* 10:

- Tratar de reentrenar a *YOLO*.
- Corregir errores en la carga de etiquetas.
- Mejorar la interfaz del etiquetador de imágenes.
- Documentación: manual de usuario del etiquetador de imágenes.

Figura A.10: Burndown del *sprint* 9

Figura A.11: Burndown del *sprint* 10

- Documentación: diseño (prototipo).

Durante este *sprint* se comenzaron las pruebas con el detector automático de objetos, para su futura adaptación a la detección de fitolitos. Además, se trataron de solucionar algunos errores presentes en el etiquetador. Y, finalmente, se documentó en la medida de lo posible el manual del etiquetador. Tratando de facilitar su uso por parte de los usuarios, en breves momentos.

A continuación, en la figura A.11, se muestra el diagrama *burndown* de este *sprint*.

Sprint 11

Estas son las tareas a realizar durante este *sprint* 11:

- Documentación: aspectos relevantes

Figura A.12: Burndown del *sprint* 11

- Documentación: conceptos *YOLO* y *BoW*
- Documentación: técnicas y herramientas.
- Documentación: manual del programador.
- Poner correctamente los separadores de archivo en Linux Y Windows.
- Crear script para la lectura de las coordenadas desde *YOLO*.
- Documentación: *README*.

Durante este *sprint*, principalmente, se trato de documentar algunos de los aspectos de la memoria. Y, por último, se creó un *script* para la lectura y conversión de las coordenadas para el entrenamiento de *YOLO*.

A continuación, en la figura A.12, se muestra el diagrama *burndown* de este *sprint*.

Sprint 12

Estas son las tareas a realizar durante este *sprint* 12:

- *Data augmentation*: rotar imágenes y coordenadas en ángulos de 90 grados.
- *Data augmentation*: espejar imágenes.
- *Data augmentation*: aplicar ruidos a las imágenes.
- *Data augmentation*: realizar cambios de tamaño en las imágenes.
- Documentación: introducción.

Figura A.13: Burndown del *sprint* 12

- Documentación: objetivos del proyecto.
- Documentación: trabajos relacionados.
- Documentación: Documentación: diseño arquitectónico y procedimental.

Durante este *sprint* se trato de continuar escribiendo algunos de los aspectos todavía no documentados hasta el momento. Y se creo una herramienta para la aplicación de técnicas de *data augmentation* sobre un conjunto de imágenes. Además, la última de las tareas planteadas para este *sprint* fue completada en un sprint más adelante por ser muy subestimada en cuanto al esfuerzo requerido.

A continuación, en la figura A.13, se muestra el diagrama *burndown* de este *sprint*.

Sprint 13

Estas son las tareas a realizar durante este *sprint* 13:

- *Data augmentation*: generador de imágenes.
- *Data augmentation*: aplicar filtros en las imágenes (clarecer,oscurecer).
- Entrenar por primera vez a YOLO con un primer dataset de fitolitos.

Durante este *sprint* se finalizo el generador de imágenes que aplicaba las técnicas de *data augmentation*, implementadas durante el *sprint* anterior, para generar un conjunto de imágenes mucho mayor. Y, además, dado que nuestros clientes nos enviaron un conjunto de imágenes etiquetadas de fitolitos, comenzamos con los primeros entrenamientos a *YOLO*.

Figura A.14: Burndown del *sprint* 13

A continuación, en la figura A.14, se muestra el diagrama *burndown* de este *sprint*.

Sprint 14

Estas son las tareas a realizar durante este *sprint* 14:

- Correcciones menores en fichero *README*.
- Evaluar los resultados de *YOLO* con un primer dataset.
- Entrenar a YOLO con el dataset aplicando *data augmentation*.

Durante este *sprint* se entrenó a YOLO, con un dataset de un tamaño mayor. Además, se completaron otras tareas pendientes de las semanas anteriores, relativas a la documentación y a la evaluación del modelo generado por el entrenamiento.

A continuación, en la figura A.15, se muestra el diagrama *burndown* de este *sprint*.

Sprint 15

Estas son las tareas a realizar durante este *sprint* 15:

- Documentación: lineas futuras.
- Reetiquetar un único tipo de fitolitos.
- Documentación: completar el anexo de planificación del proyecto.

Figura A.15: Burndown del *sprint* 14

Figura A.16: Burndown del *sprint* 15

- Documentación: completar los aspectos restantes del anexo de diseño.
- Documentación: Preparar el anexo de requisitos.
- Entrenar el clasificador con los fitolitos reetiquetados.

Durante esta semana, principalmente, se trato de continuar con aspectos de documentación. Y, a su vez, se continuo entrenando a *YOLO* con el fin de conseguir algún resultado.

A continuación, en la figura A.16, se muestra el diagrama *burndown* de este *sprint*.

Sprint 16

Estas son las tareas a realizar durante este *sprint* 16:

Figura A.17: Burndown del *sprint* 16

- Documentación: Complejidad de la tarea.
- Unificar separadores en todos los fuentes.
- Documentación: conceptos teóricos restantes.
- Desarrollar teses para los distintos módulos.

Durante esta semana se unificaron los separadores del sistemas operativo, se documento y se realizaron teses para los módulos más relevantes del sistema.

A continuación, en la figura A.17, se muestra el diagrama *burndown* de este *sprint*.

Sprint 17

Estas son las tareas a realizar durante este *sprint* 17:

- Añadir la funcionalidad de clasificar fitolitos.

Durante esta semana se volvió al enfoque mediante clasificadores. Principalmente, se trato de encontrar las distintas opciones para la clasificación de fitolitos y evaluar los resultados. Para, más tarde, aplicarlo a la detección de estos mediante ventana deslizante.

A continuación, en la figura A.18, se muestra el diagrama *burndown* de este *sprint*.

Figura A.18: Burndown del *sprint* 17

Figura A.19: Burndown del *sprint* 18

Sprint 18

Estas son las tareas a realizar durante este *sprint* 18:

- Hallar el mejor clasificador posible.
- Aplicar la ventana deslizante a las imágenes de fitolitos.

Durante esta semana se obtuvo el clasificador que mejor se adaptaba a la problemática mediante un *script* que evaluaba las distintas opciones. Y, finalmente, se aplico a la detección de fitolitos en imágenes.

A continuación, en la figura A.19, se muestra el diagrama *burndown* de este *sprint*.

Costes de personal	
Salario mensual neto	748,8€
Retención IRPF (15 %)	240€
Seguridad social (38,2 %)	611,2€
Salario mensual bruto	1600€
Salario total(5 meses)	8000€

Tabla A.1: Costes de personal

A.3. Estudio de viabilidad

Viabilidad económica

En esta sección se realiza un análisis de los costes económicos que hubiera supuesto el desarrollo de este proyecto en un entorno empresarial.

Coste del personal

Este proyecto ha sido desarrollado por un único desarrollador a tiempo parcial. En la tabla A.1 muestro el desglose de los costes ocasionados por el salario que hubiese recibido en una situación real.

Para realizar los cálculos referentes a la seguridad social debemos de tener en cuenta que tanto la empresa como el empleado participan en su pago. Pero que entre ambas suman el 38,2 %. Por parte de la empresa se desglosa de la siguiente manera:

- Contingencias comunes: 23,6 %
- AT\EP: 1,95 %
- Desempleo: 5,5 %
- Formación profesional: 0,6 %
- Fondo garantía profesional: 0,2 %

Por otra parte, se encuentran las aportaciones realizadas por el empleado:

- Contingencias comunes: 4,7 %
- Desempleo/formación profesional: 1,65 %

Costes *hardware*	Coste total	Coste amortizado
Ordenador portatil	1300€	108,33€€
JetBrains PyCharm	87,26€	87,26
Total	1387,36	195,59€

Tabla A.2: Costes informáticos

Costes totales	
Coste del personal	8000€
Costes informáticos	195,59€
Total	8195,59€

Tabla A.3: Costes totales

Costes informáticos

En cuanto al *hardware*, para el desarrollo de este proyecto únicamente ha sido necesario un ordenador portátil, con una amortización de 5 años. El cual incluía todo lo necesario para llevarlo a cabo, aunque con algunas limitaciones a nivel de rendimiento. Por otro lado, los costes a nivel *software* son los referentes al *IDE* utilizado: *JetBrains PyCharm*. En la tabla A.2 se muestran ambos costes.

Costes totales

Por lo tanto, la suma total de los cálculos se recoge en la tabla A.3

Viabilidad legal

En este apartado enuncio las distintas librerías utilizadas en este proyecto junto a sus licencias, en todos sus casos de código abierto. Véase la tabla A.4. Si se desea indagar más sobre las distintas librerías se recomienda ver la sección de herramientas y técnicas dentro de la memoria.

Finalmente, este proyecto esta publicado bajo la licencia *BSD 3-clause* con la que se permite un libre uso, modificación, distribución y uso privado de este. Sin embargo, con la condición de que el código debe ser suministrado en todas las ocasiones junto a la licencia que expone las distintas garantías de uso. Para finalizar, esta licencia introduce una muy limitada responsabilidad sobre la utilización de este proyecto y ningún tipo de garantía.

Librería	**Versión**	**Licencia**
Numpy	1.12	BSD
scikit-learn	0.18	BSD
scikit-image	0.13	BSD
Matplotlib	1.12	PSF
Jupyter Notebook	1	BSD
Jupyter Dashboards	0.6	BSD
Ipython File Upload	0.1.2	MIT
darkflow	0	GPL 3

Tabla A.4: Licencias de las librerías

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este anexo se pretenden indicar los distintos objetivos propuestos en el desarrollo de este proyecto. Así, como el conjunto de requisitos y sus especificaciones.

B.2. Objetivos generales

Los objetivos generales de este proyecto son los siguientes:

- Realizar un estudio de las técnicas del estado del arte que solucionen el reconocimiento automático de fitolitos con la mejor precisión y eficiencia posible.
- Crear un sistema de etiquetación de fitolitos. Con el que podamos crear un conjunto de imágenes etiquetadas para llevar a cabo el sistema automático de reconocimiento de fitolitos.
- Crear un sistema que nos permita multiplicar en cantidad nuestro conjunto de imágenes. Debido al diminuto conjunto de imágenes que nos ha sido proporcionado.
- Crear un sistema capaz de reconocer fitolitos automáticamente en una imagen.

B.3. Catálogo de requisitos

Derivados de los objetivos anteriores, poseemos un conjunto de requisitos para el conjunto de aplicaciones resultantes de este proyecto.

Requisitos funcionales

- **RF-1** Crear un sistema capaz de reconocer fitolitos automáticamente.
  - **RF-1.1** Subir una nueva imagen y predecir donde se encuentran los distintos fitolitos.
- **RF-2** Crear una aplicación que permita el etiquetado de fitolitos. Con las siguientes funcionalidades:
  - **RF-2.1** Escoger una imagen dentro de nuestro sistema operativo.
  - **RF-2.2** Elegir entre los distintos tipos de fitolitos.
  - **RF-2.3** Eliminar etiquetas realizadas sobre una imagen.
  - **RF-2.5** Cargar una imagen previamente etiquetada junto a sus etiquetas.
  - **RF-2.6** Guardar las coordenadas de las etiquetas y las imágenes de manera persistente.
  - **RF-2.7** Notificar al usuario de los distintos eventos.
  - **RF-2.8** Mostrar el tipo de fitolito etiquetado encima de cada etiqueta.
- **RF-3** Crear una herramienta capaz de multiplicar el conjunto de imágenes etiquetadas de fitolitos mediante la aplicación de técnicas de *data augmentation*.
  - **RF-3.1** Elegir el número de imágenes resultantes.
  - **RF-3.2** Reescalarlas.
  - **RF-3.3** Rotarlas.
  - **RF-3.4** Espejarlas.
  - **RF-3.5** Aplicarlas ruidos.
  - **RF-3.6** Aplicarlas filtros de oscurecimiento y aclaramiento.
  - **RF-3.7** Aplicarlas aleatoriamente las distintas modificaciones anteriores, permitiendo el mayor número de combinaciones posible.

Requisitos no funcionales

- **RNF-1** Facilidad de uso: las herramientas diseñadas para el usuario final deben de ser intuitivas y fáciles de utilizar con una mínima formación.

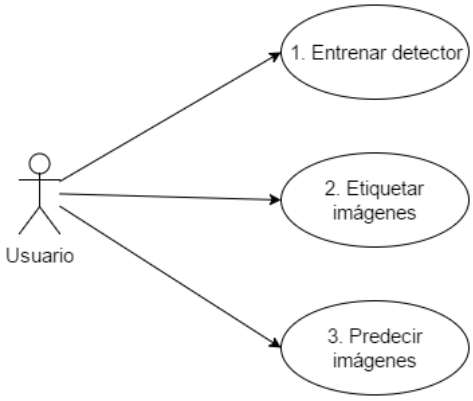

Figura B.1: Diagrama general de casos de uso.

Caso de uso 1: Entrenar detector		
Descripción	Permite al usuario entrenar el detector automático de fitolitos.	
Requisitos	RF-1	
	RF-1.1	
Precondiciones	Tener instalado *darkflow*.	
Secuencia normal	Paso	Acción
	1	El usuario lanza el comando de entrenamiento.
	2	Se carga el modelo.
	3	Se cargan los pesos del modelo.
	4	Se convierten las coordenadas de las etiquetas.
	5	Se comienza el entrenamiento.
Postcondiciones	Se guardan unos ficheros con los pesos resultantes del entrenamiento.	
Excepciones	La elección de opciones no compatibles entre sí.	
Importancia	Alta	
Urgencia	Media	

Tabla B.1: Caso de uso 1: Entrenar detector

B.4. Especificación de requisitos

La especificación de requisitos será explicada mediante diagramas de casos de uso en notación UML y una tabla por cada caso de uso, la cual los explica en mayor detalle.

Diagramas de casos de uso

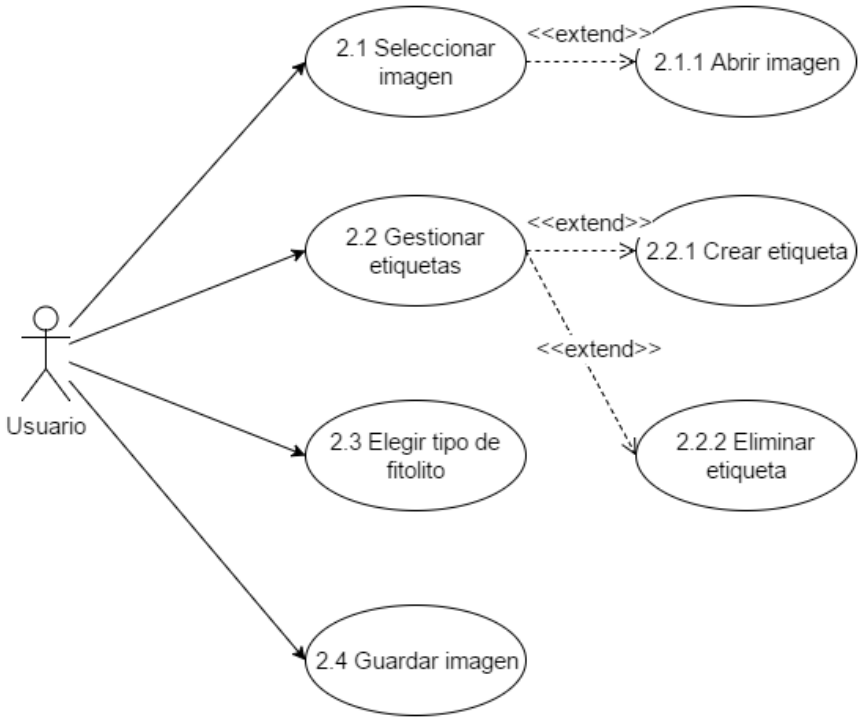

Figura B.2: Diagrama extendido del segundo caso de uso del anterior.

Caso de uso 2: Etiquetar imágenes		
Descripción	Permite al usuario identificar los múltiples tipos de fitolitos.	
Requisitos	RF-2	
	RF-2.1	
	RF-2.2	
	RF-2.3	
	RF-2.4	
	RF-2.5	
	RF-2.6	
	RF-2.7	
	RF-2.8	
Precondiciones	Ejecutar el *Jupyter Notebook*.	
Secuencia normal	Paso	Acción
	1	El usuario selecciona una imagen.
	2	La imagen se carga.
	3	El usuario etiqueta los fitolitos.
	4	El usuario pulsa en el botón guardar.
	5	Se guardan las imágenes y coordenadas.
Postcondiciones	Se guardan los ficheros que almacenan las coordenadas de las etiquetas y las imágenes.	
Excepciones	Tratar de cargar un documento distinto a una imagen.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.2: Caso de uso 2: Etiquetar imágenes

Caso de uso 3: Predecir imágenes		
Descripción	Permite al usuario identificar los fitolitos en una imagen.	
Requisitos	RF-1	
	RF-1.1	
Precondiciones		
Secuencia normal	Paso	Acción
	1	El usuario escoge una imagen.
	2	El usuario pulsa en el botón para predecir una nueva imagen.
	5	Se predice la imagen.
	6	Se muestran los resultados.
Postcondiciones		
Excepciones	Tratar de cargar un documento distinto a una imagen. Escoger erróneamente las configuraciones.	
Importancia	Alta	
Urgencia	Alta	

Tabla B.3: Caso de uso 3: Predecir imágenes

Caso de uso 2.1: Seleccionar imagen		
Descripción	El usuario puede seleccionar una imagen desde su sistema operativo. La cual se carga, muestra y finalmente se le permite etiquetarla.	
Requisitos	RF-2	
	RF-2.1	
	RF-2.5	
	RF-2.7	
	RF-2.8	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa el botón de subida de imágenes.
	2	El usuario selecciona una imagen.
	3	El usuario pulsa el botón de abrir imagen.
	4	Se carga la nueva imagen.
	5	Se notifica al usuario.
Postcondiciones	La imagen se muestra por pantalla.	
Excepciones	No se ha seleccionado una imagen, sino un documento u otro tipo de fichero.	
Importancia	Alta	
Urgencia	Alta	
Comentarios		

Tabla B.4: Caso de uso 2.1: Seleccionar imagen

Caso de uso 2.2: Gestionar etiquetas		
Descripción	El usuario podrá crear y eliminar etiquetas.	
Requisitos	RF-2	
	RF-2.3	
	RF-2.8	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	Si el usuario clica sobre una «x».
	1.2	Se borra etiqueta.
	2	Sino:
	2.2	Se crea una etiqueta.
Postcondiciones	Se cambia el estado de las etiquetas.	
Excepciones	Ninguna.	
Importancia	Alta	
Urgencia	Alta	
Comentarios	Se permite crear etiquetas fuera de la imagen y sin haber cargado una imagen. Pero estas no se guardarán.	

Tabla B.5: Caso de uso 2.2: Gestionar etiquetas

Caso de uso 2.3: Elegir tipo de fitolito		
Descripción	El usuario puede elegir entre los distintos tipos de fitolito antes de realizar una etiqueta.	
Requisitos	RF-2	
	RF-2.2	
	RF-2.6	
	RF-2.8	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el botón del tipo de fitolito que desee.
	2	Se realiza un cambio en las distintas variables que incorporan información dependiente del tipo de fitolito. Como el directorio en el que guardar el recorte.
Postcondiciones	Cambios en las variables dependientes del contexto de este.	
Excepciones	Ninguna.	
Importancia	Media	
Urgencia	Media	
Comentarios	La selección del tipo de fitolito permite distinguir donde o como guardar la información. Por lo tanto, es fundamental.	

Tabla B.6: Caso de uso 2.3: Elegir tipo de fitolito

Caso de uso 2.4: Guardar imagen		
Descripción	El usuario puede guardar las coordenadas y la imagen que haya sido etiquetada.	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa en el boton guardar de la imagen.
	2	Se guardan las coordenadas y la imagen de manera persistente.
Postcondiciones	La imagen y coordenadas se almacenan localmente.	
Excepciones	Ninguna.	
Importancia	Alta	
Urgencia	Alta	
Comentarios		

Tabla B.7: Caso de uso 2.4: Guardar imagen

Caso de uso 2.1.1: Abrir imagen		
Descripción	El usuario puede abrir una imagen.	
Precondiciones	El usuario ha seleccionado una imagen.	
Requisitos	RF-2	
	RF-2.1	
Secuencia normal	Paso	Acción
	1	El usuario selecciona la imagen
	2	Se carga la información de la imagen.
	3	Se guarda persistentemente.
Postcondiciones	Se guarda la imagen en el almacenamiento.	
Excepciones	No se ha seleccionado una imagen, sino un documento u otro tipo de fichero.	
Importancia	Alta	
Urgencia	Alta	
Comentarios		

Tabla B.8: Caso de uso 2.1.1: Abrir imagen

Caso de uso 2.2.2: Crear Etiqueta		
Descripción	El usuario puede etiquetar los distintos tipos de fitolitos en la imagen.	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa un click sobre la imagen.
	2	El usuario mueve el ratón por la imagen.
	3	El usuario vuelve a pulsar un click para realizar una etiqueta.
	4	Se escribe por encima de la etiqueta el tipo de fitolito.
	5	Se añade una cruz que permite la eliminación de la etiqueta.
Postcondiciones	La etiqueta se define encima de la imagen.	
Excepciones	Ninguna.	
Importancia	Alta	
Urgencia	Alta	
Comentarios	Se permite crear etiquetas fuera de la imagen y sin haber cargado una imagen. Pero estas no se guardarán.	

Tabla B.9: Caso de uso 2.2.2: Crear Etiqueta

Caso de uso 2.2.3: Eliminar Etiqueta		
Descripción	El usuario puede eliminar las distintas etiquetas realizadas en una imagen.	
Precondiciones	Ninguna	
Secuencia normal	Paso	Acción
	1	El usuario pulsa un click sobre «x» de una etiqueta.
	2	Se elimina la representación de esa etiqueta.
	3	Se eliminan las coordenadas asociadas a la etiqueta.
Postcondiciones	La etiqueta se elimina visual e internamente.	
Excepciones	Ninguna.	
Importancia	Baja	
Urgencia	Baja	
Comentarios		

Tabla B.10: Caso de uso 2.2.3: Eliminar Etiqueta

Apéndice C

Especificación de diseño

C.1. Introducción

En este anexo se introducirá el diseño de los distintos aspectos de este proyecto a todos los niveles *software*.

C.2. Diseño de datos

En este trabajo se manejan dos tipos de datos principales:

- Imágenes con formato JPG.
- Ficheros JSON que contienen las coordenadas de las etiquetas realizadas con el etiquetador de imágenes.

Tanto las imágenes como las etiquetas de las imágenes se almacenan de manera persistente en nuestro disco duro. Para observarlas, deberemos navegar, dentro del proyecto, por la siguiente ruta de carpetas: *code*, *rsc*, *img*. Una vez situados en la carpeta *img*, nos encontraremos una serie de carpetas con el nombre de cada tipo de fitolito y una carpeta *Default*[1].

La razón de la existencia de una carpeta por cada tipo de fitolito es el almacenamiento organizado de los recortes generados por el etiquetador. Almacenando en cada una de estas carpetas los recortes o imágenes correspondientes a un tipo de fitolito.

Respecto a la carpeta *Default*, en ella se almacenan las imágenes completas junto a un fichero *JSON* por imagen. En cada uno de estos ficheros se almacenan las coordenadas realizadas con el etiquetador en dicha imagen. Un ejemplo del contenido de un fichero JSON sería el mostrado en la figura C.1.

[1]Siempre y cuando hayamos utilizado el etiquetador anteriormente.

```
{"2017_5_17_17_57Image_7344.jpg":
    {"Bilobate": [[865, 1110, 1183, 1402]],
    "Spherical": [[1132, 1282, 2207, 2357], [1238, 1414, 368, 533]]}}
```


Figura C.1: Ejemplo de fichero JSON.

Siendo el formato: nombre de la imagen y nombre de cada tipo de fitolito, solo apareciendo los existentes en la imagen. Teniendo cada tipo de fitolito una lista de etiquetas, donde cada etiqueta tiene cuatro coordenadas. Las cuatro coordenadas se almacenan con el siguiente orden y significado:

- Desplazamiento en el eje y de la esquina superior izquierda de una etiqueta.
- Desplazamiento en el eje y de la esquina inferior derecha de una etiqueta.
- Desplazamiento en el eje x de la esquina superior izquierda de una etiqueta.
- Desplazamiento en el eje x de la esquina inferior derecha de una etiqueta.

C.3. Diseño procedimental

En este proyecto hay que destacar tres procedimientos principales, los cuales explicaré, brevemente, a continuación.

Procedimiento que realiza el etiquetador de imágenes

El procedimiento muy simplificado del funcionamiento del etiquetador cuando se carga una nueva imagen es el mostrado en el procedimiento 1.

Algoritmo 1: Procedimiento de funcionamiento del etiquetador

```
si Cambio de imagen entonces
    Cargamos imagen.
    si La imagen se encuentra en el directorio por defecto y tiene un
     fichero JSON correspondiente entonces
        Cargamos las etiquetas previamente realizadas.
        Mostramos las etiquetas junto a la imagen.
    fin
    Escuchamos los eventos del ratón sobre el SVG.
fin
```

Una vez cargada la imagen, como indico finalmente, se escuchan los eventos del ratón. En concreto, los *clicks* y movimientos de este sobre el SVG. Ya que

la creación de los rectangulos, o etiquetas, sobre la imagen se hacen a partir de dichos eventos. Para ello, se poseen dos *listeners*, u observadores de eventos, uno para cada evento. Los *clicks* se utilizan para la creación de un nuevo rectángulo o su finalización. Y los movimientos del ratón, una vez hecho un primer click[2], permiten redimensionar el rectángulo de manera totalmente intuitiva.

Nótese que existen más variables a tener en cuenta, como el tipo de fitolito, el cual determina el directorio donde se guardará el recorte obtenido de la etiqueta y como se almacenan las coordenadas en el fichero *JSON*, entre otras. Pero lo que se intenta realizar en este apartado es una explicación simplificada del procedimiento.

Procedimiento en el *notebook* para la detección de caras

El procedimiento simplificado del funcionamiento del *notebook* para el reconocimiento de caras en una imagen es el mostrado en el procedimiento 2.

Algoritmo 2: Procedimiento de funcionamiento del etiquetador

1 Realizamos la ventana deslizante sobre la imagen.
2 Obtenemos las características del histograma de los gradientes.
3 Clasificamos la imagen.
4 Eliminamos las ventanas redundantes con non-maximum suppression.
5 Mostramos la imagen con los rectángulos.

C.4. Diseño arquitectónico

En este apartado se representan los aspectos más relevantes a nivel de diseño arquitectónico mediante la notación UML. UML, según sus siglas Lenguaje de Modelado Unificado, es una notación comúnmente utilizada para representar y documentar un sistema con una mayor abstracción [4].

Este proyecto no contiene un diseño muy elaborado a nivel arquitectónico por dos razones fundamentales. La primera ha sido el esfuerzo requerido en la investigación y el aprendizaje de técnicas muy variadas para afrontar el problema. Hasta dar con la más adecuada. Por otro lado, la mayoría del código ha sido desarrollado en los *Jupyter Notebooks*. Los cuales nos permiten interaccionar fácilmente con el código y documentarlo a su vez, aportando una fácil introducción a otros usuarios.

Aun así existen tres módulos con código empaquetado: módulo utilizado para la gestión de carpetas del etiquetador, módulo utilizado para la predicción

[2]De manera que hayamos creado el rectángulo.

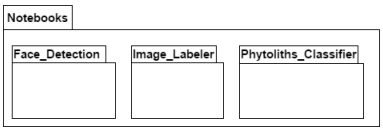

Figura C.2: Diagrama de paquetes

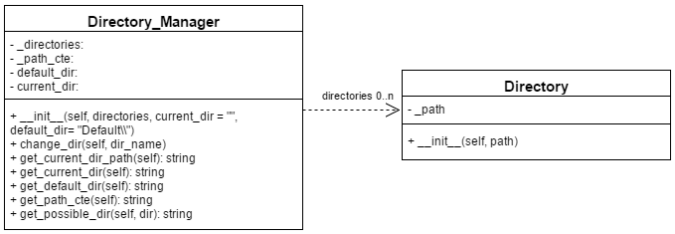

Figura C.3: Diagrama de clases de las clases encargadas de la gestión de carpetas

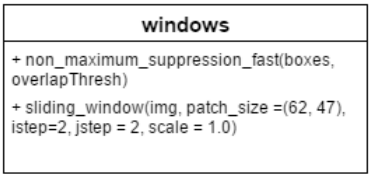

Figura C.4: Clase estática encargada de las tareas de non-maximum suppression y ventana deslizante

de caras y el modulo para la predicción de fitolitos. Véase el diagrama de paquetes C.2.

Modulo para la gestión de carpetas

Por un lado, tenemos el módulo que se encarga de la gestión de carpetas donde el etiquetador almacena las imágenes. Véase el diagrama de clases C.3.

Modulo para la predicción de caras

Por otro lado, tenemos las clases encargadas de la predicción de caras en una nueva imagen mediante el *notebook* explicado anteriormente. En este caso existen 3 clases, las cuales interaccionan entre sí. La primera es una clase estática con 2 funciones para los cálculos del non-maximum suppression y la ventana deslizante. Véase la figura C.4

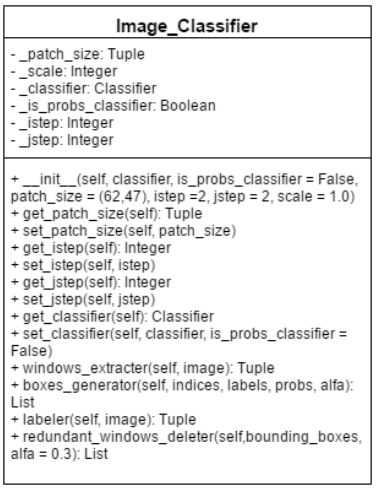

Figura C.5: Clase encargada de las tareas de clasificación de una imagen

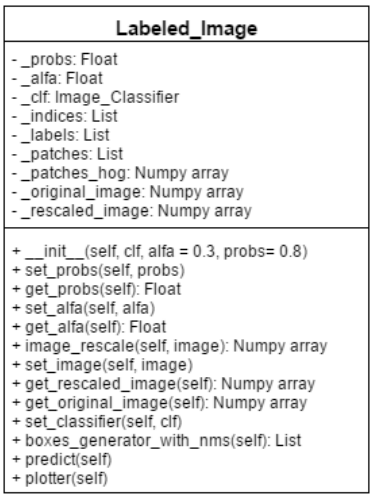

Figura C.6: Clase encargada de de guardar la imagen, mostrarla y poder clasificarla.

Además, tenemos la clase encargada de envolver al clasificador y realizar las tareas de clasificación de una imagen. Véase la figura C.5. Y, finalmente, tenemos la clase encargada de guardar la imagen, mostrarla y poder clasificarla con el apoyo del resto de clases. Véase la figura C.6. Para finalizar este modulo, podemos apreciar en la figura C.7 como interaccionan las tres clases.

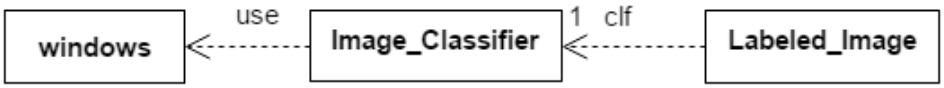

Figura C.7: Diagrama de clases del *Jupyter Notebook* para la detección de caras.

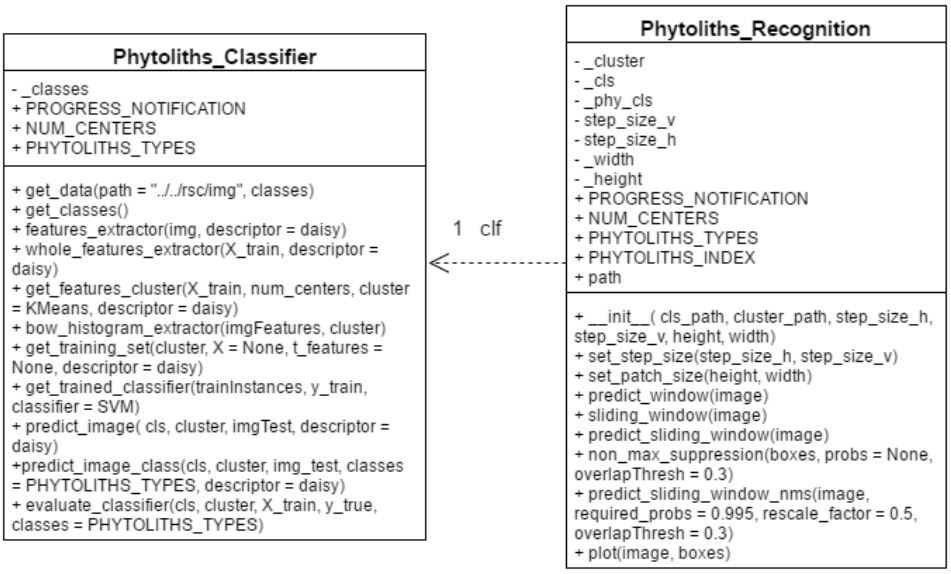

Figura C.8: Diagrama de clases para la detección de fitolitos.

Modulo para la predicción de fitolitos

Por último, el modulo más importante que consta de unicamente 2 clases: la clase encargada de la clasificación de fitolitos y la clase encargada de el reconocimiento de objetos en imágenes de fitolitos, representadas en la figura C.8.

C.5. Diseño de las interfaces

En esta sección se explicarán las diferentes interfaces de los productos realizados en este trabajo fin de grado.

Jupyter Notebook para la detección de caras

Durante el primer mes se trabajó en un *Jupyter Notebook* que trataba de reconocer caras mediante un clasificador junto a un extractor de características, como ya hemos explicado en secciones anteriores. Con este *Jupyter Notebook*

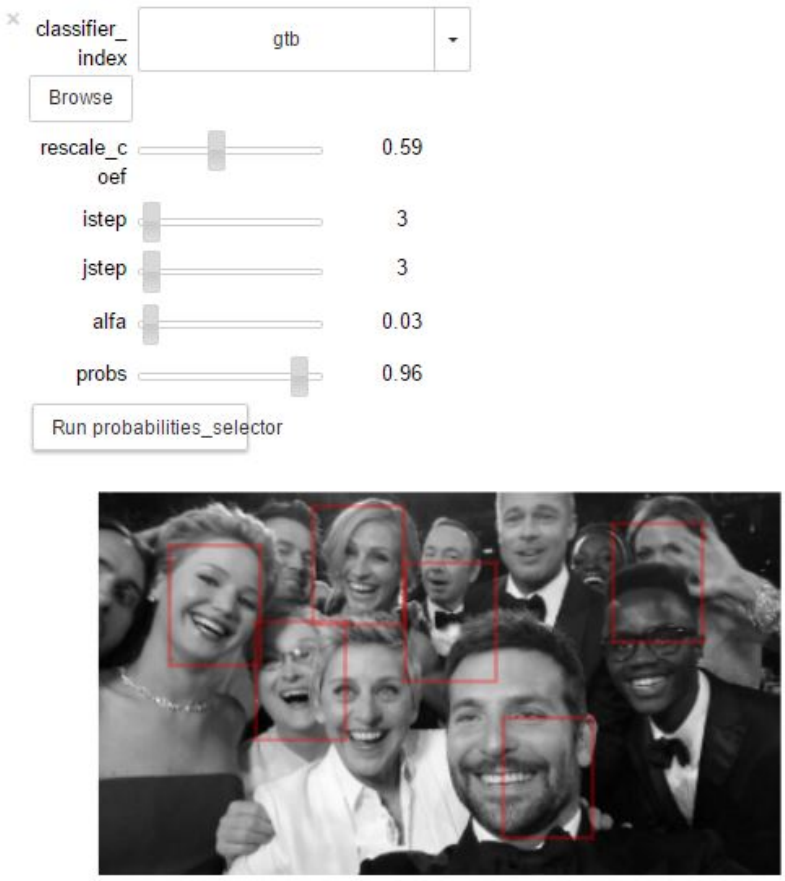

Figura C.9: *Jupyter Notebook* para el reconomiento de caras

se trataba de facilitar la evaluación del rendimiento de los clasificadores y el cambio de las distintas variables de manera interactiva.

Por lo tanto, la interfaz de este *Jupyter Notebook* no ha sido trabajada para su uso por parte del usuario. Sino que fue creada para un uso más de experimentación. Es por ello que la interfaz no tiene un buen grado de usabilidad como podemos observar en la figura C.9.

Etiquetador de imágenes

La interfaz del etiquetador de imágenes ha sido desarrollada partiendo del prototipo mostrado en la ilustración C.10. Tratando de crear una interfaz lo más simple e intuitiva para el usuario.

El resultado tras la implementación de este producto es el mostrado en la C.11. Obteniendo una interfaz muy similar a la prototipada en un principio.

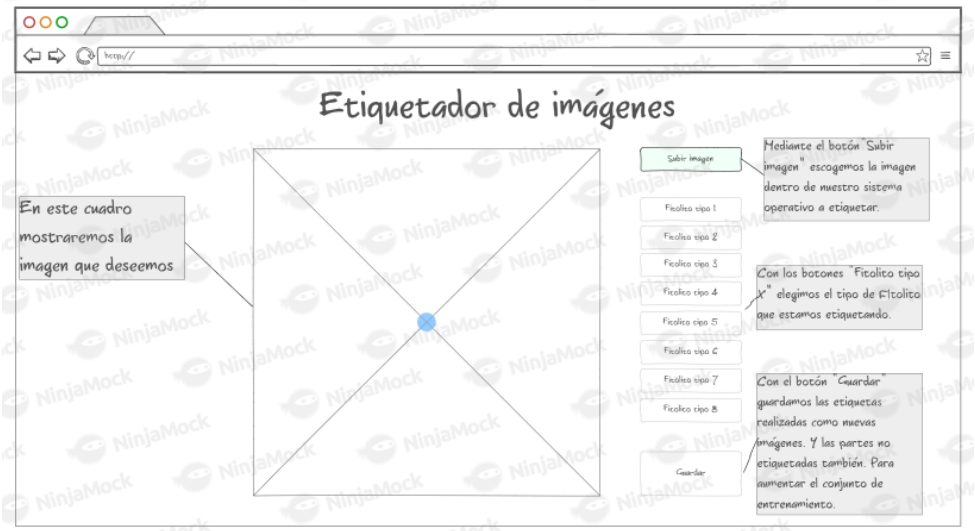

Figura C.10: Prototipo del etiquetador de imágenes

Pero añadiendo algún elemento más, para facilitar su uso por parte del usuario.

Reconocimiento automático de fitolitos

La interfaz para el reconocimiento automático de fitolitos consta de un conjunto de *sliders*, los cuales nos permiten cambiar diversos parámetros para experimentar con este y dos botones: botón para subir una imagen y botón para ejecutar el reconocimiento. Véase la figura C.12. Cada uno de los botones y *sliders* se encuentran explicados en el manual de usuario E.

Figura C.11: Etiquetador de imágenes

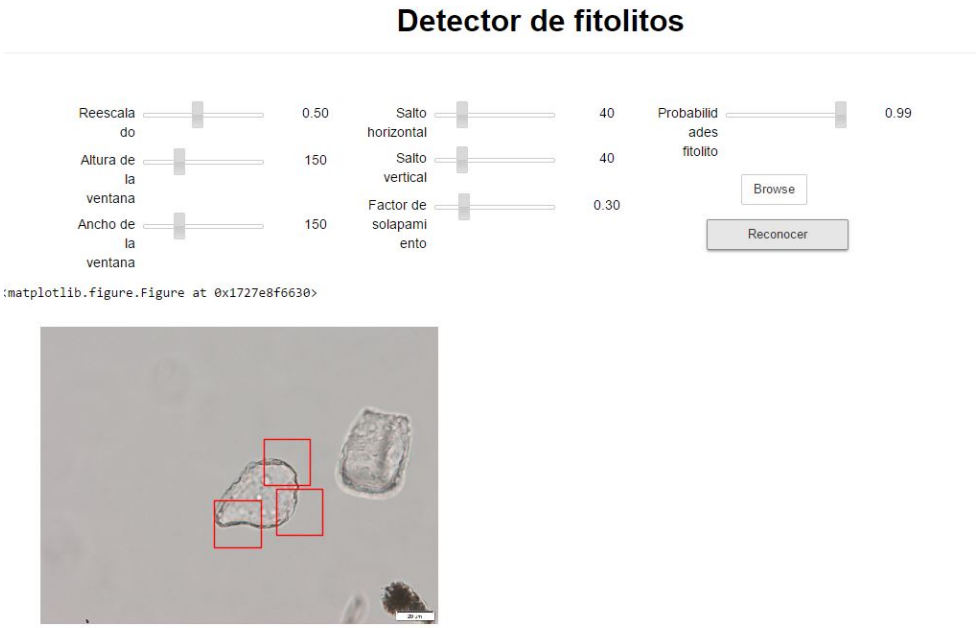

Figura C.12: Reconocimiento automático de fitolitos

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este anexo se introducirá todos los directorios, *notebooks* y documentación útil necesaria para las personas que deseen continuar o trabajar en el proyecto.

D.2. Estructura de directorios

La estructura de directorios del proyecto, en estructura de árbol, es la siguiente:

- */*, es decir, el directorio raíz. En el se encuentra el fichero de la licencia, el *README*, el *.gitignore* y las siguientes carpetas:
  - ***code***: contiene toda la lógica de la aplicación.
    - ***imgaug***: repositorio para realizar *Data Augmentation* con *Python 2*.
    - ***notebooks***: contiene todos los notebooks creados para este proyecto.
    - ***rsc***: contiene los recursos necesarios por las distintas aplicaciones.
    - ***src***: contiene el notebook principal para tareas de reconocimiento. Esta carpeta aun se encuentra en desarrollo.
  - ***doc***: contiene la documentación del proyecto.
    - ***img***: contiene todas las imágenes de la memoria y anexos del proyecto.

- ***tex***: contiene los ficheros correspondientes a cada uno de los anexos.

D.3. Manual del programador

Etiquetador de imágenes

En esta sección explicare más en detalle como funciona internamente el etiquetador.

Base del etiquetador: *Widgets* de *Python*

El etiquetador está creado mediante los *Widgets* de *Python*. Un *Widget* es un objeto de Python con representación en navegadores. Este nos permite la comunicación entre *JavaScript* y *Python*. Facilitando, así, crear interfaces *Web* interactivas, como es nuestro caso [3].

El etiquetador de imágenes es un *Widget* personalizado, el cual ha sido creado por nosotros. Pero, además, este utiliza otros *Widgets* prefedefinidos por los *ipywidgets* de los *Jupyter Notebook* para los botones.

Javascript

La parte de código *JavaScript* se ocupa de representar todos los elementos visuales y capturar los eventos.

En cuanto a elementos visuales, nos referimos al SVG, la imagen, rectangulos o etiquetas y textos. Los cuales son elementos *HTML*. Y, en cuanto a eventos, nos referimos a los clicks o movimientos del ratón sobre nuestros elementos *HTML*.

Python

La parte de *Python* controla toda la lógica de la aplicación. Desde que imagen se muestra, hasta las conversiones de las coordenadas de la imagen entre la vista y la imagen real.

Además, para los botones, los cuales son *Widgets* de *Python*, se controlan tambien sus eventos desde el propio *Python*.

D.4. Compilación, instalación y ejecución del proyecto

Compilación, instalación y ejecución del etiquetador de imágenes

El etiquetador no requiere de ninguna compilación. Simplemente es necesario llevar a cabo los pasos indicados en el *Manual del usuario etiquetador de imágenes* para su instalación y ejecución. Y, en ese momento, estaremos listos para modificar o mejorar el código todo lo que deseemos.

Procedimiento de entrenamiento de *YOLO*

Para realizar el entrenamiento de *YOLO*, *darkflow* nos provee de un *script* que nos facilita dicha tarea mediante la linea de comandos[1], llamado *flow*. Además, existen dos posibilidades principales para realizar el entrenamiento sobre nuestro *dataset*:

- Utilizar unos pesos[2] pre-entrenados para nuestro modelo.
- Crear un modelo que se ajuste a nuestras necesidades.

La ventaja de utilizar unos pesos pre-entrenados reside en que la red neuronal ya habrá aprendido ciertas características, como bordes o formas. Por lo tanto, el tiempo de entrenamiento será mucho menor. Pero puede que la reutilización de dicho modelo no se adecue a nuestras necesidades ya sea porque tengamos un número distinto de clases al modelo inicial o porque el contexto sea totalmente distinto.

Por otro lado, se encuentra el entrenamiento desde cero de un nuevo modelo que se adecue a nuestras necesidades. El cual suple las desventajas presentadas por la anterior opción, pero introduce una mayor complejidad a la hora del entrenamiento.

En cualquier caso, los dos siguientes ejemplos, correspondientes a las opciones planteadas, nos permitirían entrenar el modelo:

```
flow --train --model cfg/yolo-tiny.cfg
 --load bin/yolo-tiny.weights
 --dataset "Fitolitos" --annotation "Fitolitos"
```

o

[1] Solo ha sido probado desde un sistema Linux.

[2] Pesos: son cada uno de los valores asignados a una neurona en una red neuronal.

```
flow --model cfg/yolo-new.cfg --train
 --dataset "Fitolitos" --annotation "Fitolitos"
```

Donde la opción *–train* indica que se desea entrenar un modelo, la opción *–load* indica donde se encuentran los pesos pre-entrenados, la opción *–dataset* indica la ruta de la carpeta donde se encuentran las imagenes de los fitolitos y la opción *–annotation* indica donde se encuentran los ficheros con las coordenadas[3].

A modo de resumen, voy a enumerar todas las opciones que nos permite escoger la herramienta *flow*:

- *imgdir*: ruta que contiene las imágenes con las que realizar pruebas.
- *binary*: ruta que contiene los pesos.
- *config*: ruta que contiene los ficheros de configuración.
- *dataset*: ruta que contiene las imágenes.
- *backup*: ruta que contiene las copias de seguridad de los pesos.
- *summary*: ruta que contiene los resumenes *TensorBoard*, los cuales se pueden abrir con una herramienta de *TensorFlow*.
- *annotation*: ruta que contiene los ficheros con las anotaciones o coordenadas de las etiquetas dentro de las imágenes.
- *threshold*: umbral que determina las probabilidades mínimas que debe cumplir una detección o caja para mostrarse.
- *model*: ruta al fichero de configuración deseado.
- *trainer*: algoritmo de entrenamiento.
- *momentum*: optimizador aplicable a los algoritmos de entrenamiento: *rmsprop* y *momentum*.
- *verbalise*: mostrar como se forma el grafo o red neuronal.
- *train*: entrenar la red neuronal.
- *load*: inicialización de la red.

[3]La modificaciones realizadas sobre *darkflow* para permitir la lectura y conversión de mis coordenadas están pensados para que el valor de las opciones *dataset* y *annotation* sean el mismo. De otra manera el modelo fallará en el entrenamiento.

- *savepb*: guardar configuración y pesos a un fichero *protobuf*.
- *gpu*: cantidad de la capacidad de los procesadores gráficos a utilizar.
- *gpuName*: nombre de la tarjeta gráfica.
- *lr*: tasa de aprendizaje para el entrenamiento.
- *keep*: número de últimos resultados que se mantienen guardados.
- *batch*: tamaño del lote de imágenes.
- *epoch*: número de *epochs*.
- *save*: número de ejemplos tras los cuales se realiza una copia de seguridad.
- *demo*: para realizar una demo con *webcam*.
- *queue*: tamaño del lote para la demo.
- *json*: salida de las cajas en formato *json*.
- *saveVideo*: guardar video de la entrada de video o camara.
- *pbLoad*: ruta del fichero *protobuf*.
- *metaLoad*: ruta del fichero .meta generado junto al fichero *protobuf*.

Apéndice E

Documentación de usuario

E.1. Introducción

En este anexo se indicarán los distintos requisitos necesarios para ejecutar las diversas aplicaciones desarrolladas durante el proyecto. Así como, toda la información necesaria para la correcta utilización de la herramienta. Y la solución ante posibles problemas en su utilización.

E.2. Requisitos de usuarios e instalación

Los requisitos *software* necesarios para poder ejecutar las aplicaciones son:

- Anaconda. Descargando e instalando la versión para Python 3.6[1].
- IPython File Upload. Siguiendo los pasos de instalación en `https://github.com/peteut/ipython-file-upload`.
- *Jupyter Dashboards*. Siguiendo los pasos de instalación en `https://github.com/jupyter/dashboards`.
- Este proyecto se puede descargar o clonar desde el siguiente enlace al repositorio: `https://github.com/jasag/Phytoliths-recognition-system`. Entrando, preferiblemente, en la pestaña de *Releases* y descargando el zip *v0.1.0*. Véase la figura E.1.

[1]Es altamente recomendable utilizar *Anaconda* para la instalación de *Python* junto a algunos paquetes necesarios. Puesto que la instalación aislada de *Python* llevaría al usuario a tener que instalar manualmente múltiples paquetes que ya vienen preinstálados con *Anaconda*.

Figura E.1: Primera versión del etiquetador

E.3. Manual del usuario

En esta sección se explicarán los distintos aspectos a tener en cuenta en el uso de los distintos productos.

Manual del usuario: etiquetador de imágenes

Una vez completados satisfactoriamente los pasos anteriores, ejecutamos *Jupyter Notebook* desde *Anaconda*[2]. Y desde esta aplicación, abrimos el *notebook Image_Labeler.ipynb*, en la carpeta *code/notebooks* dentro de este proyecto previamente descargado.

Con *Image_Labeler.ipynb* ya abierto, tendremos que navegar por la barra de navegación del *notebook* para llevar a cabo los dos siguientes pasos:

1. Ejecutar todas las celdas del *notebook*. Para ello, navegamos por *Cell* y clicamos en *Run All*. Como se puede observar en la figura E.2.

2. Activar *Dashboard Preview*. Para ello, navegamos por *View* y clicamos en *Dashboard Preview* E.3.

Una vez realizados los pasos anteriores, tendremos como resultado la pantalla inicial del etiquetador, tal y como se observa en la figura E.4.

[2] Si es la primera vez que utilizas Anaconda es recomendable utilizar Anaconda Navigator. Ya que este nos facilitará ejecutar *Jupyter Notebook*, sin el uso de la linea comandos.

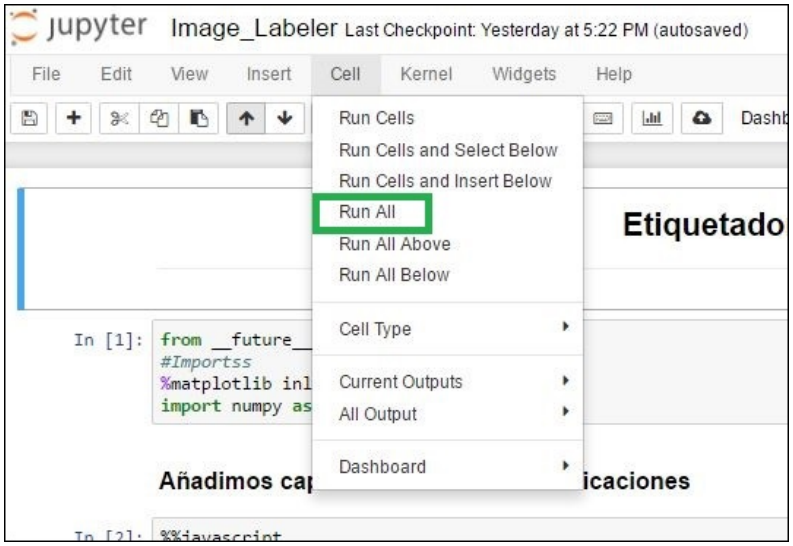

Figura E.2: Ejecución de todos los pasos del *notebook*

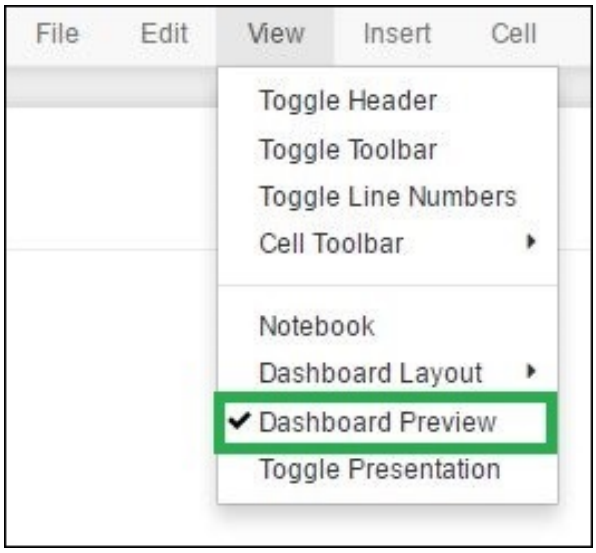

Figura E.3: Selección de la vista del *notebook*

Uso del etiquetador

Como resultado de los pasos previos, el etiquetador de imágenes estará listo para su funcionamiento.

El etiquetador está compuesto por dos partes principales:

- Parte izquierda: imagen donde etiquetar los fitolitos.

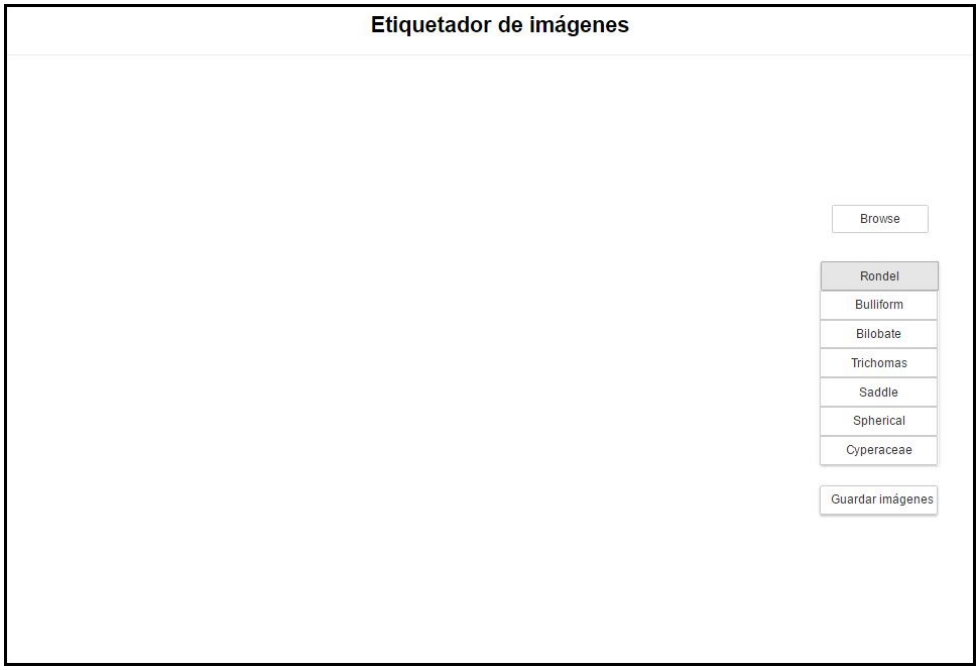

Figura E.4: Etiquetador de imágenes

- Parte derecha: «botones».

La parte izquierda mostrará la imagen a ser etiquetada. Y la parte derecha mostrará los botones que nos permitirán seleccionar los distintos tipos de fitolitos presentes en la imagen. Como podemos ver en la figura E.4.

Parte derecha del etiquetador: botonera

La parte derecha del etiquetador esta compuesta por tres grupos de botones. El botón para la carga de una imagen, los botones de selección de fitolito y el botón de guardado. Como podemos obsevar en la figura E.5. En las secciones siguientes explicaré sus funciones.

Cargar imagen

Para etiquetar una imagen, el primer paso será escoger una imagen en nuestro ordenador mediante el botón *Browse*, situado en la esquina superior derecha del etiquetador. Como se puede ver en la figura E.4.

Una vez pulsado, se mostrará una ventana, véase figura E.6, en la que escogeremos la imagen que deseemos.

Tras escoger la imagen, esta se cargará en la parte izquierda del etiquetador, dando lugar a algo similar a lo mostrado en la figura E.7.

Figura E.5: Parte derecha del etiquetador

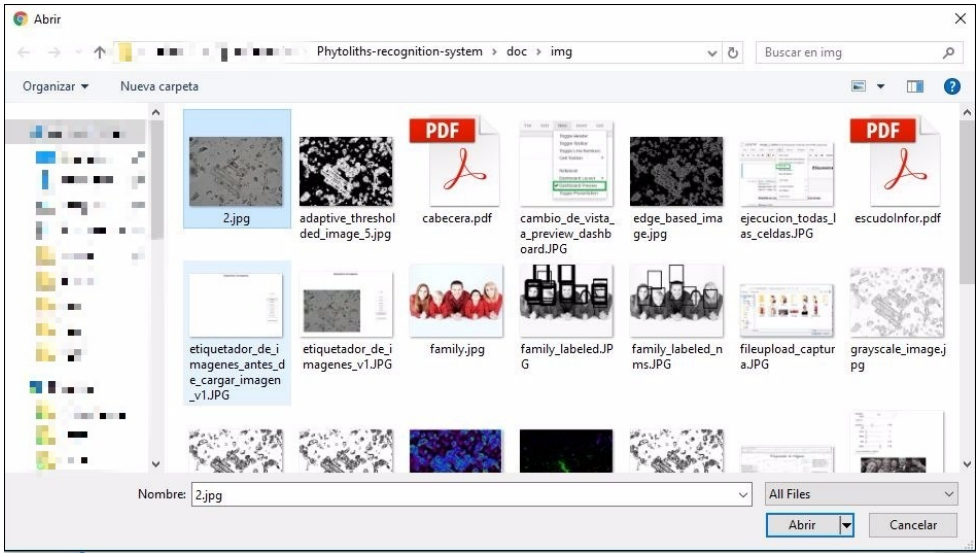

Figura E.6: Ventana de subida de ficheros

Etiquetar distintos tipos de fitolitos

Antes de explicar como se crean las etiquetas, debemos de tener en cuenta los botones para la selección de tipos de fitolito.

Cada vez que se quiera etiquetar un tipo de fitolito distinto tendremos que pulsar en el botón correspondiente al tipo de fitolito objetivo.

A su vez, y para facilitarnos distinguir a que tipo de fitolito pertenece cada etiqueta, se escribe encima de cada etiqueta el tipo de fitolito que ha

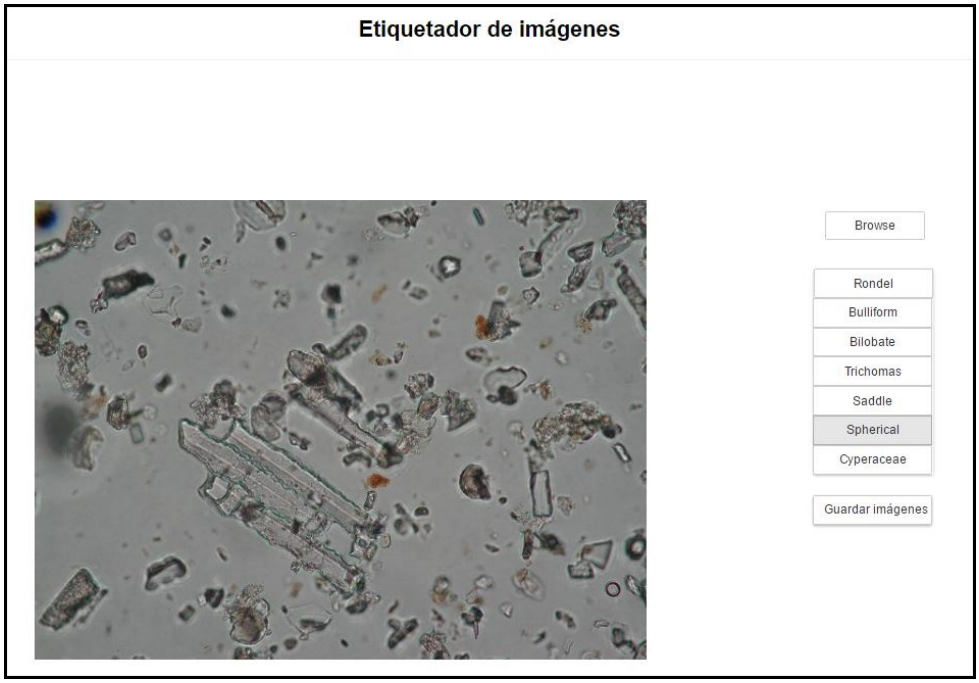

Figura E.7: Etiquetador de imágenes con una imagen cargada

sido etiquetado, véase figura E.8.

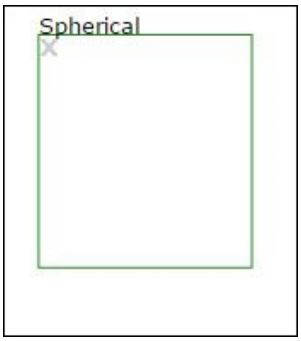

Figura E.8: Ejemplo de etiqueta

Realizar etiquetas

Una vez cargada la imagen en el etiquetador, y comprendido el uso de los botones de selección de tipo de fitolito, podemos comenzar a etiquetar fitolitos. Para etiquetar, basta con clicar en la imagen, definiendo una esquina del rectangulo, mover el ratón hacía donde deseemos y volver a clicar, definiendo la otra esquina de este. Para, así, definir una etiqueta como la que se puede

ver en la figura E.8.

Eliminar etiquetas

Como es normal, un usuario puede equivocarse dibujando una etiqueta que no deseaba. Por ello, en la esquina superior izquierda de cada etiqueta, disponemos de una «X», como se observa en la figura E.8, que nos permitirá eliminar etiquetas previamente realizadas. Posibilitando así revertir un error del usuario.

Guardar etiquetas

Después de realizar todas las etiquetas, tan solo queda guardarlas. Para ello basta con clicar el botón «Guardar imágenes» en la esquina inferior derecha del etiquetador, véase la figura E.4.

Es **importante** destacar que no sirve de nada etiquetar una imagen, si no pulsamos más tarde en el botón guardar. Debido a que las etiquetas realizadas no habrán sido guardadas. Y, por lo tanto, todas las etiquetas realizadas sobre una imagen se perderán.

Etiquetas realizadas fuera de la imagen

En la versión actual del etiquetador, este te permite realizar etiquetas fuera de la imagen. Pero estas etiquetas no serán guardadas.

Por lo tanto, debemos de tener en cuenta que las etiquetas que no tengan todas sus aristas dentro de la imagen no se guardan.

Aun así, en la siguiente sección explicaré como comprobar que las etiquetas previamente realizadas han sido guardadas.

Carga de etiquetas previamente realizadas

Las imágenes etiquetadas se guardan en la carpeta */code/rsc/img* dentro de la carpeta contenedora del proyecto. Al cargar imágenes que se encuentren dentro de esta carpeta, las etiquetas previamente realizadas sobre estas imágenes se cargarán con ellas[3]. Esto nos permitirá poder añadir nuevas etiquetas a imágenes ya procesadas, modificarlas o eliminarlas.

[3]Nótese que los nombres de las imágenes no son los cargados originalmente, ya que todas las imágenes etiquetadas son renombradas para no sobrescribir imágenes previamente etiquetadas con nombres iguales.

Notificaciones

Se han incluido en el etiquetador una serie de notificaciones, las cuales nos permitirán cercionarnos de distintos eventos. Las notificaciones que se muestran son las siguientes:

1. Notificación en la carga de una imagen. Véase la figura E.9.
2. Notificación en la carga incorrecta de un fichero que no sea una imagen. Véase la figura E.10.
3. Notificación en el guardado de etiquetas. Véase la figura E.11.

Gracias a estas notificaciones podremos saber si la imagen que hemos seleccionado era correcta, o no, y si las etiquetas ya han sido guardadas.

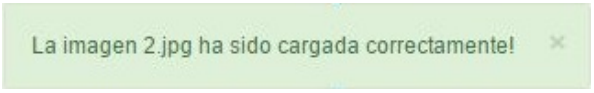

Figura E.9: Notificación en la carga de una imagen

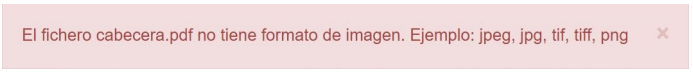

Figura E.10: Notificación en la carga incorrecta de un fichero que no sea una imagen

Figura E.11: Notificación en el guardado de etiquetas

Bloqueo del etiquetador

En algunas ocasiones, y por razones que desconozco hasta el momento, el etiquetador puede bloquearse al cargar imágenes previamente etiquetadas. Para solucionar este problema, basta con recargar el etiquetador, yendo a la barra de navegación, navegando por *Cell* y finalmente clicando en *Run All*. Como hicimos en la puesta en marcha del etiquetador.

Bibliografía

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. 2001. [Online; accedido 30-Mayo-2017].

[3] Jupyter Team. Ipywidgets and jupyter-js-widgets, 2017. [Online; accedido 30-Abril-2017].

[4] Wikipedia. Lenguaje unificado de modelado — wikipedia, la enciclopedia libre, 2017. [Online; accedido 27-Junio-2017].

[5] Wikipedia. Scrum — Wikipedia, the free encyclopedia, 2017. [Online; accedido 12-Marzo-2017].