UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

X-Teeth
Documentación Técnica

Presentado por Ismael Franco Hernando
en Universidad de Burgos — 6 de julio de 2022
Tutores: César Ignacio García Osorio y
José Miguel Ramírez Sanz

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

Dentro de este apartado se van a tratar dos puntos acerca del proyecto:

- La Planificación temporal.
- La Viabilidad del mismo.

A.2. Planificación temporal

Para llevar a cabo el proyecto se ha seguido la metodología de trabajo *Scrum*, dónde cada semana se llevaba a cabo un *sprint* distinto. En cada uno de estos *sprints* se consensuaban unos objetivos y tareas a llevar a cabo para la siguiente semana.

En situaciones especiales, las reuniones se podrían atrasar unas semanas con el fin de desempeñar correctamente los objetivos marcados.

Para poder reflejar el trabajo que se iba realizando semana tras semana se utilizó la plataforma de *GitHub* cuyo repositorio que recoge el trabajo del proyecto es https://github.com/ifh1001/TFG_X-Teeth.

Sprint 1 - 02/02/2022

Este primer *sprint* se basó principalmente en la explicación por parte de mis tutores sobre el trabajo a realizar en el TFG X-Teeth. Además, se

concretó la fecha de las reuniones de cada *sprint*, siendo esta los viernes a las 11:30 de la mañana a través de *Microsoft Teams*.

Como primeras tareas a llevar a cabo fueron:

- Crear cuenta en *Overleaf*, para hacer la documentación con LaTeX.
- Crear el repositorio en *GitHub*.
- Leer diversos artículos sobre el *Deep Learning* en la rama de la odontología.
- Buscar información acerca de la librería de *Keras*, ya que se pensó que podría ser una solución para llevar a cabo el proyecto.

Sprint 2 - 11/02/2022

En esta segunda reunión se habló sobre los pocos casos de dientes que había para poder trabajar y que mi tutor José Miguel había estado desarrollando un *notebook* que hacía uso del *data augmentation* para así disponer de un mayor número de casos distintos.

También, debido al problema de tener pocas radiografías, se decidió usar *Detectron2*, ya que es una biblioteca de *Python* que permite detectar objetos con muy pocos casos. Como resultado de esta reunión, las tareas a desarrollar para la siguiente semana fueron:

- Terminar el *notebook* de mi tutor.
- Intentar instalar *Detectron2* en mi ordenador.

Sprint 3 - 18/02/2022

Para este *sprint* solo puede terminar el *notebook*, ya que la parte de instalación de *Detectron2* me resultó imposible, debido a que no existe una versión oficial para trabajar con Windows 10 y generalmente surgen incompatibilidades entre varios paquetes en este sistema operativo. Como resultado de esto, la única tarea a realizar para el siguiente *sprint* fue seguir intentando instalar *Detectron2*.

Sprint 4 - 23/02/2022

Esta reunión fue distinta a las anteriores, ya que inicialmente mantuvimos una charla con el odontólogo que da origen a este proyecto. Se le comentó que había un error en algunos de los casos que nos había pasado y se acordó que nos pasaría un total de 150 radiografías para poder trabajar correctamente con *Detectron2*.

La segunda parte de la reunión fue ya sin el odontólogo, en la cual comenté que no conseguí instalar *Detectron2*. Debido a esta situación, se decidió trabajar por el momento con *Google Colab*, ya que era muy sencillo usar la librería de detección de objetos, y en el futuro utilizar alguna de las máquinas a disposición de la Universidad de Burgos, como Gamma o Beta.

Las tareas resultantes este *sprint* fueron:

- Aprender a utilizar *Detectron2* en *Google Colab*.
- Crear una función que transformara los archivos JSON del dentista a la forma COCO que necesita *Detectron2* para poder trabajar.

Sprint 5 - 04/03/2022

Para este *sprint* tras haber realizado con éxito la segunda tarea de la reunión pasada, creé el primer sistema entrenador de *Detectron2*, dónde, pese a trabajar con tan solo 10 imágenes (7 para entrenar y 3 para testear) se obtuvieron unos resultados bastante buenos en la parte de detección del diente y regulares en la parte del nervio.

Tras obtener estos resultados tan esperanzadores, las siguientes tareas a hacer fueron:

- Aplicar la técnica *Convex Hull* a las imágenes rotadas obtenidas por el *notebook* terminado del *sprint* 2, con el fin de obtener los puntos de la máscara y generar los JSON correspondientes y poder trabajar con más casos.
- Aplicar la técnica IoU (*Intersection over Union*), para comprobar si las predicciones del *Detectron2* están siendo buenas.

Sprint 6 - 11/03/2022

Los resultados obtenidos por el *Convex Hull* no fueron los correctos, ya que esta técnica no funciona correctamente en las zonas curvas de los dientes y nervios.

Además, detecte un problema en el *notebook* del *sprint* 2, puesto que las máscaras de los dientes y nervios no se encontraban en el tamaño correcto con respecto a la imagen de la radiografía correspondiente.

Por último, se creyó conveniente aplicar otra estrategia con el *Detectron2*, ya que inicialmente teníamos un modelo para entrenar a la vez dientes y nervios, por lo que se pensó que con la siguiente estrategia se conseguirían mejores resultados:

1. Crear un modelo que solo entrenara dientes.
2. Obtener las predicciones de los dientes.
3. Recortar la caja que contenía el diente de la predicción.
4. Crear un modelo que únicamente entrenara nervios.
5. Obtener las predicciones de los nervios.

Por tanto, para el siguiente *sprint* tendría que realizar las siguientes tareas:

- Buscar una técnica que realice correctamente la detección de los puntos de las máscaras de los dientes y nervios.
- Solucionar el problema de tamaños entre las máscaras y la radiografía.
- Llevar a cabo la nueva estrategia del *Detectron2*.
- Mostrar una gráfica con la *pérdida total* durante el entrenamiento con el fin de ver a partir de que iteración se produce un *overfitting*.

Sprint 7 - 25/03/2022

En el anterior *sprint* se completaron con éxito todas las tareas. Pese a ello, la técnica aplicada para detectar los bordes gracias a la librería CV2 de *Python* obtuvo unos buenos resultados aunque no del todo precisos.

Por tanto, sería útil buscar alguna otra alternativa con el fin de ver si se puede conseguir una mayor precisión en la detección de los puntos de los bordes de las máscaras.

Como resultado de esta reunión, las tareas a realizar para la siguiente semana son:

- Buscar alguna alternativa en la detección de bordes, con el fin de conseguir mejores resultados.
- Entrenar *Detectron2* con las radiografías originales y no con las que tienen el diente resaltado, ya que seguramente se estén consiguiendo buenos resultados gracias a la "ayuda" que dispone en las imágenes.
- Buscar información acerca de la convención PEP8, y empezar a aplicarlo, con el fin de conseguir un código más limpio.
- Añadir un límite en la parte de validación para que una vez se encuentre el punto en el que se produce el *overfitting* no siga entrenando.

Sprint 8 - 01/04/2022

En esta semana se ha conseguido mejorar la forma de obtener los puntos de las máscaras de los dientes y nervios, para poder trabajar correctamente con ellas.

Esto se ha conseguido juntando dos librerías: inicialmente se usa *scikit-image* para obtener únicamente el borde de cada una de las máscaras y, a continuación, se utiliza CV2 para conseguir todos los puntos que dan lugar a ese borde. De esta manera se consigue una precisión mucho mayor que aplicando únicamente CV2 sobre las imágenes.

Las próximas tareas a realizar son:

- Instalar *Detectron2* en la máquina de la Universidad de Burgos, Gamma, para poder trabajar en ella.
- Automatizar el proceso de repartición del conjunto de datos de modo que el 60 % sea de entrenamiento, un 20 % para test y el último 20 % para validación.

Sprint 9 - 22/04/2022

Para este *sprint* no se consiguió instalar *Detectron2* en la máquina Gamma debido a alguna incompatibilidad entre versiones. Pese a ello, se siguió trabajando en *Google Colab*, y se crearon las funciones necesarias para

poder distribuir las imágenes de manera aleatoria entre el entrenamiento, validación y test.

Adicionalmente, se añadió de forma correcta el cálculo de la precisión de las predicciones a través del IoU ya mencionado anteriormente. Dicho valor se calculó tanto en el *notebook* de dos modelos entrenadores cómo en el de un modelo entrenador, con el fin de ver cuál de ellos muestra mejores resultados, que por el momento es el segundo.

Por último, se mejoró el código en uno de los *notebook* para facilitar la comprensión del mismo a través del estándar PEP8.

La tarea a realizar para la próxima semana es:

- Conseguir instalar *Detectron2* en Gamma mediante versiones más antiguas de los paquetes con el fin de evitar incompatibilidades.

Sprint 10 - 29/04/2022

Para esta semana se ha conseguido instalar *Detectron2* en Gamma. Para ello, se han tenido que utilizar versiones antiguas tanto de *CUDA Toolkit* cómo de *PyTorch* para poder evitar las distintas incompatibilidades entre versiones.

Se han instalado la versión 10.1 de *CUDA Toolkit* y la versión 1.4 de *PyTorch*, junto con la versión del *Detectron2* compatible para soportar dichas librerías.

Las dos tareas próximas a realizar son:

- Implementar de forma correcta el método para poder medir el tamaño del nervio del diente de la imagen.
- Investigar las distintas GUI de *Python* para elegir una de ellas y poder montar la aplicación final.

Sprint 11 - 06/05/2022

Para este *sprint* se ha implementado una técnica que permite calcular la longitud del nervio. Dicha técnica se basa en obtener la distancia máxima entre el nervio y el diente. Pese a que se ha conseguido que funcione correctamente, estamos a la espera del dentista para poder transformar la distancia en píxeles a una longitud real y poder analizar los resultados que da.

La segunda técnica que se quería implementar se basaba en usar *Skeletonize* de la librería *scikit-image*. Dicha técnica no se ha implementado de forma correcta y, por tanto, aún no se sabe si puede dar buenos resultados.

Con respecto a las GUI de *Python*, tras analizar varias se ha decidido trabajar con *EasyGUI* y con *PySimpleGUI*. Y una vez se hayan probado ambas elegir una de las dos.

Como resultado, las tareas a realizar para la semana que viene son:

- Corregir el uso de *Skeletonize* para que funcione correctamente y ver que resultados da.
- Instalar *EasyGui* y *PySimpleGui* en el entorno de *conda* de Gamma para ver que funcionen correctamente y seleccionar la mejor.

Sprint 12 - 13/05/2022

Para esta reunión han ocurrido dos hechos importantes. El primero es que a mitad del *sprint* anterior el odontólogo que da origen a este proyecto había cambiado de idea. Inicialmente, lo principal era obtener la medida del nervio, pero ahora el odontólogo quiere la longitud del diente.

El segundo hecho relevante, ha sido que el propio odontólogo ha asistido a la reunión para ver los resultados que se están consiguiendo. También, nos comentó el proceso que realiza para poder medir el tamaño del diente y al mostrarle las técnicas que utilizamos para calcular la longitud nos indicó que la correcta sería la que mide el diente por los puntos centrales.

Como última cosa a comentar sería la imposibilidad de poder ejecutar un GUI desde el *notebook* en Gamma, debido a que al ser un servidor no tiene ningún *display* asignado. Por ello, la aplicación final se llevará a cabo en un *notebook*.

Las próximas tareas a hacer son:

- Investigar y probar *ipywidgets* con el fin de poder tener una pequeña interfaz dentro del *notebook* que contendrá la aplicación final.
- Para poder medir la longitud del diente por la parte central, se ha pensado en emplear *Skeletonize* sobre el nervio y a la lineal central obtenida alargarla hasta el diente, para posteriormente calcular su longitud.

Sprint 13 - 20/05/2022

Para este *sprint* se ha implementado la técnica para medir la longitud del diente a través de la línea central del nervio. Con dicha técnica se han implementado un total de tres técnicas distintas para poder calcular el tamaño del diente.

Pese a tener varias técnicas, ninguna de ellas parece que da buenos resultados por el momento, ya que los errores relativos son demasiados altos para que el odontólogo pueda usar algunos de estos métodos.

Con respecto a *ipywidgets*, se ha conseguido instalar, además de investigar y probar los distintos tipos de *widgets* que hay con el fin de ver cuáles utilizar en la aplicación final.

Las tareas a realizar para la próxima semana son:

- Investigar porque los resultados obtenidos no son los esperados, ya que el proceso de segmentación es bastante bueno.
- Desarrollar la aplicación final con las distintas funcionalidades de *ipywidgets*.

Sprint 14 - 27/05/2022

Para esta semana se ha conseguido desarrollar una simple aplicación que permita subir al odontólogo una radiografía y que le presente los resultados obtenidos. Actualmente, la aplicación muestra la radiografía junto con la recta correspondiente a la longitud del diente y un cuadro de texto con el tamaño del mismo.

Con respecto a los errores que se estaban obteniendo a la hora de calcular la longitud de los dientes, tras analizar los métodos implementados y la segmentación de dientes y nervios por parte de *Detectron2*, se ha llegado a la conclusión de que el problema probablemente venga dado o por la resolución que nos pasó el odontólogo o por los tamaños de los dientes que nos indicó en las distintas radiografías.

Para la siguiente semana las tareas a realizar son:

- Permitir subir en la aplicación varias radiografías a la vez y mostrar para cada una de ellas los resultados obtenidos.
- En la aplicación, mostrar la segmentación del diente y el nervio obtenida en cada caso.

- Probar a exportar el modelo obtenido del *Detectron2* a un *notebook* en *Google Colab* con el fin de facilitar las conexiones a la aplicación.
- En caso de que la tarea anterior no se pueda efectuar, usar *tmux* junto con *Ngrok* para permitir conexiones al *notebook* que tiene la aplicación en Gamma.

Sprint 15 - 03/06/2022

Para esa semana se han conseguido implementar las distintas mejoras de la aplicación final.

Con respecto al modelo, al ser un archivo pesado de unos 330 MB aproximadamente, hay problemas a la hora de subirlo tanto a *Google Colab* como a *GitHub*. Por ello, no se ha podido probar si el modelo funcionará en la aplicación subida a *Google Colab*.

Por último, mencionar que el dentista nos ha indicado que la última técnica que usamos para medir la longitud de los dientes no es la correcta en aquellos casos donde el nervio tenga una cierta curvatura. Por tanto, hay que implementar una cuarta técnica que mida la longitud del diente a través del nervio, calculando las longitudes en pequeñas porciones para poder medir correctamente las zonas curvas de los dientes.

Las próximas tareas a realizar son:

- Buscar si existe algún *widget* que permita hacer zum a las imágenes, para poder observar mejor los resultados obtenidos en la aplicación.
- Con el fin de dar más *feedback* al usuario en la aplicación a la hora de cargar las imágenes, mostrar un mensaje con las distintas imágenes cargadas.
- Usar alguna técnica que permita dividir el modelo en pequeños archivos para poder trabajar con ello de forma cómoda.
- Implementar la nueva técnica de cálculo de longitud del diente mencionada anteriormente.

Sprint 16 - 10/06/2022

Para este *sprint* se ha buscado algún *widget* que permita hacer zum a las imágenes en los resultados de la aplicación, pero no se ha encontrado

nada útil. También, se ha añadido que cada vez que se cargue una imagen se muestre un mensaje con el nombre de todas las imágenes cargadas.

Con respecto a la técnica a implementar, se ha desarrollado correctamente. Pero la parte de buscar la distancia máxima desde el extremo del nervio hasta el diente no es la correcta, por lo que se realizará de igual forma que la técnica anterior, alargar como si fuera una línea recta hasta el borde del diente.

Por otro lado, con respecto a dividir el modelo para poder subirlo a *Google Colab* y a *GitHub*, se ha encontrado una forma más sencilla. Dicho modo se basa en tener el modelo subido en *Google Drive* y gracias al paquete *gdown* descargarlo.

Tras poder subirlo a *Google Colab*, se ha corroborado que se puede utilizar el modelo obtenido en Gamma en los *notebooks* de *Google*.

Por último, la aplicación en Gamma se ha lanzado también con *Ngrok*, de manera que, actualmente, se puede acceder a la aplicación a través de un enlace. Pese a ello, también se va a realizar la aplicación en *Google Colab*, de modo que haya dos alternativas para poder usarla.

Por tanto, las tareas a hacer para la semana que viene son:

- Corregir el problema de la cuarta técnica implementada para calcular la longitud del diente.
- Una vez esté corregido el problema de la tarea anterior, actualizar la aplicación para que trabaje con dicha estrategia para calcular la longitud deseada.
- Implementar la aplicación en *Google Colab*.
- Comentar las diversas celdas de los diferentes *notebooks* para facilitar su comprensión.

Sprint 17 - 16/06/2022

Para esta semana se ha solucionado el problema de la última técnica implementada. También se ha actualizado la aplicación para que trabaje con dicha técnica.

La aplicación también se ha desarrollado y probado en *Google Colab*, donde funciona correctamente.

Con respecto a comentar las celdas de todos los *notebooks*, se ha empezado, pero no se ha terminado.

Las próximas tareas a realizar son:

- Terminar de comentar todos los *notebooks*.
- Retocar la aplicación, añadiendo manual de usuario y más descripciones y efectuando alguna pequeña mejora.

Sprint 18 - 24/06/2022

Este fue el último *sprint* del proyecto. En él se revisó la aplicación final y se revisó parte de la documentación que se llevaba hecha. Con respecto a la documentación de los diferentes *notebooks* no se consiguió comentar todos los subidos a lo largo del proyecto a *GitHub*.

Por tanto, las últimas tareas a realizar son:

- Terminar de comentar todos los *notebooks*.
- Terminar de escribir toda la documentación del proyecto.

A.3. Estudio de viabilidad

En los siguientes apartados se procederá a estudiar, tanto la viabilidad económica como la viabilidad legal del proyecto.

Viabilidad económica

Para estimar la viabilidad económica se analizarán dos elementos importantes, primero el coste de mantener al personal de la aplicación y segundo el coste del hardware necesario para tener siempre activa la aplicación.

Coste del personal

Para llevar a cabo la aplicación y mantenerla a lo lago del tiempo se va a suponer que se contrata a un programador cuyo sueldo medio en España es de unos 29.800€ brutos al año, lo que supone unos 1.610€ netos al mes (valores obtenidos en Jobted[1]).

[1]Jobted: https://www.jobted.es/salario/programador

Coste estimado por mes	1 610€
Coste estimado año	19 320€

Tabla A.1: Costes estimados personal

Hardware

La aplicación puede usarse de dos formas distintas:

- Gamma: accediendo al *notebook* alojado en la máquina de la Universidad de Burgos.
- Google Colab: accediendo al *notebook* compartido de Colab.

Debido a las dos opciones que hay para usar la aplicación se podrían distinguir dos costes distintos.

Por un lado tenemos *Google Colab* donde se puede compartir *notebooks* de forma gratuita a otros usuarios a través de un enlace que les da acceso.

Por el otro lado, tenemos el *notebook* en Gamma, por lo que la máquina tiene que estar siempre activa para no dejar a los clientes sin servicio. Debido a ello, tendrá que encontrase la máquina siempre conectada a la red eléctrica.

Coste estimado por mes	100€
Coste estimado año	1 200€

Tabla A.2: Costes estimados hardware

Costes Totales

Tras analizar los costes del personal y los costes del hardware, los costes estimado al mes y al año son:

	Mes	**Año**
Coste Personal	1 610€	19 320€
Coste Hardware	100€	1 200€
Total	1 710€	20 520€

Tabla A.3: Costes estimados totales

Viabilidad legal

Con respecto a la viabilidad legal, se han empleado bibliotecas de *Python* que permiten su uso y comercialización gratuita.

El principal problema viene con las radiografías, ya que nos las ha prestado el odontólogo y por ello no se pueden subir con este proyecto para que puedan probarse. Esto podría suponer un problema, puesto que cualquier usuario que quiera comprobar como funciona la aplicación necesitará poseer alguna radiografía dental.

Incluso para la gente que quiera replicar este proyecto, por su parte, deberá de tener un número razonable de radiografías para poder entrenar su modelo de manera correcta.

Las bibliotecas de *Python* utilizadas en la aplicación final junto con la licencia asociada a cada una de ellas se puede ver en la tabla A.4.

Biblioteca	**Licencia**
NumPY	BSD
OpenCV	BSD
Matplotlib	PSF
Jupyter Widgets	BSD
Detectron2	Apache 2.0
SciPy	BSD
PyTorch	BSD
Scikit-Image	BSD
gdown	MIT

Tabla A.4: Licencias Bibliotecas Python

Todas estas licencias, junto con la licencia BSD de *Jupyter Notebook* permiten que la aplicación final pueda ser distribuida y comercializada sin ninguna limitación.

Tras analizar todas las licencias, la licencia colocada al proyecto es GPL v3 que permite la comercialización del *software* desarrollado, su modificación, distribución, realizar patentes y su uso de forma privada.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado se habla acerca de los diferentes objetivos generales del proyecto, junto con los diferentes requisitos y casos de uso.

B.2. Objetivos generales

Los objetivos generales del proyecto son:

- Investigar y aplicar nuevas funcionalidades del *deep learning* en la rama de la odontología.
- Desarrollar una aplicación web, sencilla de usar y con un fácil acceso para todo el mundo.
- Ayudar en el proceso de la endodoncia, de forma que la obtención de la longitud, actualmente hecha de forma manual, se pueda automatizar con la aplicación.
- Intentar conseguir que el error de la aplicación en la estimación de la longitud del diente no sea superior a 0.5 milímetros.

B.3. Catalogo de requisitos

En esta sección se hablará acerca de los distintos requisitos que tiene la aplicación final. Adicionalmente, se indicará quien es el actor de dicha aplicación.

Actor de la aplicación:

Odontólogo: Será la persona que una vez haya realizado la radiografía del diente del paciente, la introduzca en la aplicación para obtener los resultados de la longitud de dicho diente.

Los requisitos funcionales de la aplicación son:

- **REQ.1:** Llevar a cabo una aplicación capaz de calcular la longitud del diente de una radiografía.
  - **REQ.1.1:** El odontólogo podrá cargar la/las radiografías deseadas.
  - **REQ.1.2:** El odontólogo podrá ejecutar tantas veces como quiera la aplicación con las radiografías que desee.
  - **REQ.1.3:** El odontólogo podrá ver que las imágenes se han subido correctamente.
- **REQ.2:** Llevar a cabo una aplicación capaz de mostrar la segmentación de la radiografía junto con la recta de cálculo de distancia.
  - **REQ.2.1:** El odontólogo podrá ver la segmentación del diente y del nervio.
  - **REQ.2.2:** El odontólogo podrá ver la recta sobre la que se obtiene la longitud del diente.
  - **REQ.2.3:** El odontólogo podrá ver adicionalmente la radiografía original.

B.4. Especificación de requisitos

Diagrama de casos de uso

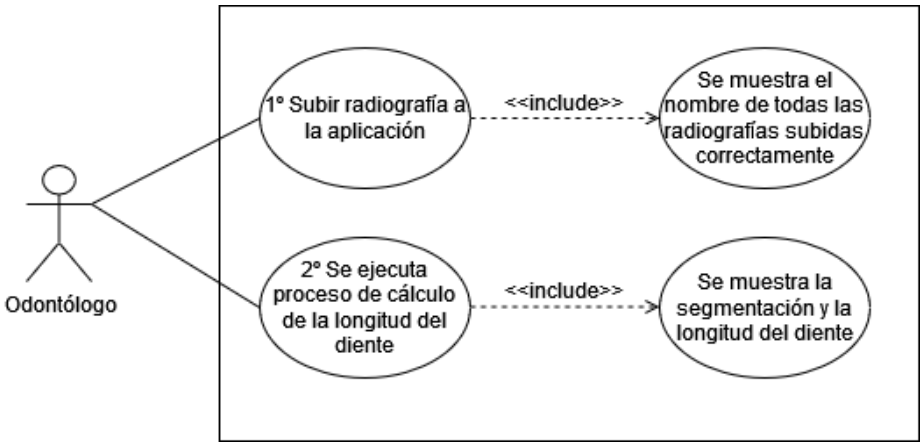

Figura B.1: Diagrama de Casos de Uso de la Aplicación

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado se describen los datos utilizados en el proyecto, como está formada la aplicación final y el diseño procedimental de la misma.

C.2. Diseño de datos

Este punto contiene de forma detallada los datos usados para entrenar con *Detectron2*. También se habla de la aplicación y los ficheros que la forman.

Radiografías Dentales

Para poder entrenar el modelo de *Detectron2* es necesario tener previamente imágenes con sus correspondientes JSON sobre las zonas que interesan al modelo que aprenda.

Es por ello, que el odontólogo Álvaro Zubizarreta Macho nos ha cedido un total de 10 radiografías junto con sus correspondientes JSON del diente y del nervio del que se desea conocer su longitud.

Aplicación

Las funciones de la aplicación se encuentran todas ellas en el fichero `funciones.py`. En su interior nos encontramos únicamente con una clase,

la clase Predictor, la cual implementa un patrón *Singleton* de modo que solo exista un objeto que apunto al modelo de la aplicación.

El resto del fichero son todo funciones encargadas de llevar a cabo las distintas operaciones necesarias en la aplicación.

C.3. Diseño procedimental

Este punto contiene el diagrama de secuencias que simula la iteración del usuario con la aplicación y los diferentes pasos que se dan, como se puede ver en la Figura C.1.

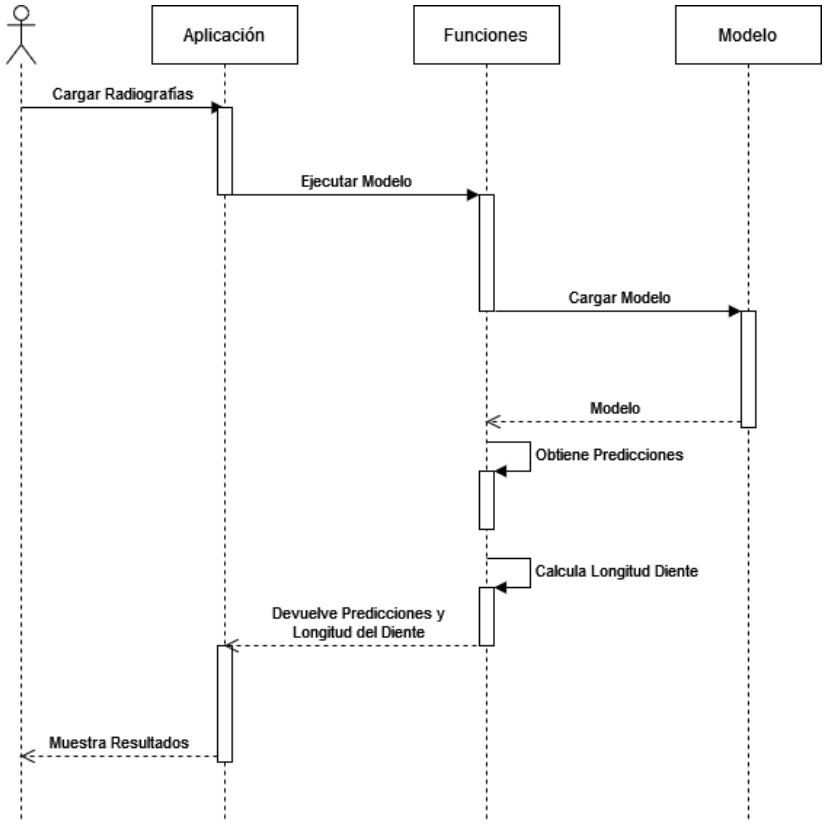

Figura C.1: Diagrama de Secuencias de la Aplicación

C.4. Diseño arquitectónico

El diseño de la aplicación se podría dividir en tres partes principales: aplicación, funciones y modelo.

La aplicación es un *notebook* encargado principalmente de cargar radiografías y mostrar los diferentes resultados de las mismas.

Las funciones, cuyo fichero tiene que estar en el mismo directorio que la aplicación, contiene todos los elementos necesarios para cargar el modelo, obtener las predicciones y calcular la longitud de los dientes. Gestiona todo el control de secuencia de código.

Por último, tenemos el modelo, que tiene que encontrarse en una carpeta con nombre `/output` dentro del directorio de la aplicación. Este modelo es el obtenido a lo largo proyecto con *Detectron2*.

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este apartado va dedicado a aquellas personas del ámbito informático que quisieran seguir con el proyecto y que necesitan conocer los elementos básicos para su correcta comprensión. En ese apartado se hablará de:

- **Estructura de directorios:** contiene una explicación básica de como están distribuidos los directorios, y los ficheros que contiene cada uno de ellos. Dicha estructura se puede comprobar en el repositorio de *GitHub* del proyecto.

- **Manual del programador:** se explica la funcionalidad de cada fichero, de forma que puedan ser comprendidos en caso de que se quiera trabajar con ellos.

- **Compilación, instalación y ejecución del proyecto:** contiene los pasos necesarios para la instalación, compilación y ejecución de la aplicación del proyecto.

- **Pruebas del sistema:** recoge las diversas pruebas que se han realizado a lo largo del proyecto.

D.2. Estructura de directorios

En este punto se va a halar acerca de la distribución de directorios y ficheros que hay en el repositorio de *GitHub*[1] llevado a cabo a lo largo del proyecto.

```
/
├── doc
├── src
└── README.md
```

La distribución principal del directorio está formada por tres elementos:

- `doc`: dierctorio que contiene toda la documentación asociada al poryecto.
- `src`: directorio que contiene todos los ficheros con código hechos en el proyecto.
- `README.md`: fichero que contiene una descripción del repositorio en *GitHub*

Ahora analizaremos internamente los dos directorios del proyecto, para ver que contiene cada uno de ellos.

Documentación

La documentación se encuentra en el directorio `/doc` y cuya estructura es la siguiente:

```
/doc/
└── latex
```

El directorio `/doc/latex` contiene en su interior toda la documentación realizada en LaTeX, y que sigue la estructura dictaminada por el tribunal de la Universidad de Burgos.

Código

Todo el código realizado a lo largo del proyecto se encuentra en el directorio `/src`. En su interior se encuentran todas las pruebas llevadas a cabo junto con la aplicación final y todo lo necesario para que pueda funcionar correctamente. La estructura de directorios es la siguiente:

[1] Repositorio: https://github.com/ifh1001/TFG_X-Teeth

```
/src/
├── aplicacion
│   ├── output
│   │   └── descargaModelo.py
│   ├── Aplicacion.ipynb
│   └── funciones.py
└── pruebas
    ├── ConvexHull.ipynb
    ├── DataAugmentation.ipynb
    ├── DeteccionBordes.ipynb
    ├── Detectron2_1Entrenador.ipynb
    ├── Detectron2_2Entrenadores.ipynb
    └── Skimage+CV2_DetecciónBordes.ipynb
```

El directorio `/src` está formado por dos directorios principales:

- `aplicacion`: está formado por el *notebook* que permite usar la aplicación final del proyecto, junto con las diversas funciones que tiene en el archivo `funciones.py` para que la aplicación pueda funcionar correctamente. Además, contiene el directorio `/output` donde en su interior se encuentra un ejecutable de *Python* que permite descargar el modelo desde *Google Drive*.
- `pruebas`: este directorio contiene todas las pruebas del proyecto. Las pruebas se pueden dividir en tres categorías:
  - **Data augmentation de las radiografías:** el fichero encargado de obtener las modifcicaciones de las radiografías originales a través de *data augmentation* es `DataAugmentation.ipynb`.
  - **Detección de bordes:** correspondiente a las tres aproximaciones para obtener los puntos del borde de dientes y nervios, cuyos ficheros son `ConvexHull.ipynb`, `DeteccionBordes.ipynb` y `Skimage+CV2_DetecciónBordes.ipynb`.
  - **Puesta a punto de Detectron2 y la obtención de su modelo:** los ficheros `Detectron2_1Entrenador.ipynb` (trabaja con un modelo) y `Detectron2_2Entrenadores.ipynb` (trabaja con dos modelos) son pruebas con *Detectron2* con el fin de aprender su uso y diversas pruebas para obtener el modelo final. Además, el fichero `Detectron2_1Entrenador.ipynb` contiene las diversas pruebas con las técnicas implementadas para obtener la longitud del diente.

D.3. Manual del programador

Previamente, ya se ha indicado que todo el código del proyecto se encuentra en el directorio `/src`. Dicho directorio está formado por otros dos directorios correspondientes a las pruebas y a la aplicación final.

El contenido de ambos directorios ya se ha explicado previamente, por lo que si en el futuro alguien desea actualizar la aplicación necesitará actualizar el fichero `funciones.py` añadiendo las nuevas implementaciones.

En caso de querer continuar con las pruebas o investigar nuevas técnicas de cálculo de longitud del diente, se deberán de añadir al directorio de pruebas. Otra posibilidad es transformar las pruebas a archivos `.py`. ya que actualmente las pruebas se encuentran en *notebooks*, de forma que pueda ser más sencillo trabajar con ellos

D.4. Compilación, instalación y ejecución del proyecto

Para poder utilizar la aplicación del proyecto no es necesaria su descarga e instalación en ningún dispositivo, puesto que tiene dos opciones para poder utilizarla, y ambas son *online*.

Como única instalación que habría que hacer es la de *Detectron2* junto con la descarga del modelo y de las funciones de la aplicación en *Google Colab*. Esto se debe a que los *notebooks* en *Google Colab* pierden todos sus ficheros tras la finalización de su sesión. Además, *Detectron2* no se encuentra como una de las bibliotecas ya instaladas, por lo que siempre será necesaria su descarga de su repositorio de *GitHub* con su posterior instalación.

Todo este proceso se encuentra automatizado en la aplicación, tan solo sería necesaria la ejecución de las celdas encargadas de dichas tareas y esperar a que terminen.

Con respecto al uso de la aplicación en Gamma, no sería necesaria ninguna instalación de nada adicional, ya que el servidor cuenta con todo lo necesario para que la aplicación pueda funcionar al completo.

Por último, en caso de que alguien quiera instalar en su ordenador el entorno necesario para usar la aplicación tiene que tener cuidado con las versiones que instala, ya que las versiones de *Detectron2*, *PyTorch* y *CUDA Toolkit* tienen que ser compatibles entre ellas. En el caso de este proyecto las versiones son las mostradas en la tabla D.1.

Biblioteca	**Versión**
Detectron2	0.2.1
PyTorch	1.4.0
CUDA Toolkit	10.1

Tabla D.1: Versiones Detectron2, PyTorch y CUDA Toolkit

D.5. Pruebas del sistema

La base principal de este proyecto ha sido la investigación del *deep learning* en la rama de la odontología para finalmente realizar una aplicación. Inicialmente, se trabajó en la obtención de más radiografías para el proyecto, mediante la transformación de las originales que nos habían sido prestadas por el odontólogo.

Lo siguiente fueron diversas pruebas de técnicas para obtener los puntos de los dientes y nervios de las radiografías modificadas para poder usarlas en la creación de un modelo.

Con el modelo obtenido se efectuaron pruebas de las predicciones, más concretamente se utilizó la técnica IoU *(Intersection over Union)* para evaluar como de buenas eran los resultados del modelo.

Finalmente, se hicieron pruebas de los diversos métodos para obtener la longitud del diente, aunque como ya se ha comentado previamente, los valores obtenidos no eran del todo reales al no tener tamaños iguales en las radiografías ni la longitud del diente radiografiado.

Como se puede apreciar, se han llevado a cabo diversas pruebas en las técnicas y modelos obtenidos, pero no se han efectuado como tal pruebas del sistema. Ya que la aplicación era algo más secundario en el proyecto, y los mayores esfuerzos se centraron en la investigación.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apartado se hablará acerca de los diferentes requisitos de usuario, el proceso de instalación para la aplicación del proyecto y el manual de usuario con el fin de que quede explicado cómo usar la aplicación.

E.2. Requisitos de usuarios

Los requisitos de usuario necesarios son muy sencillos, tan solo es necesario que el usuario disponga de una conexión a internet con la que poder conectarse al *notebook* de Gamma o de *Google Colab*.

Como requisito adicional tendrá que conocer el enlace que le permitiría acceder a dichos *notebooks*.

E.3. Instalación

Para poder disfrutar de la aplicación no es necesaria ninguna instalación en el ordenador del usuario. Tan solo será necesario instalar *Detectron2* en el *notebook* de *Google Colab*, ya que no tiene por defecto la librería necesaria y cada sesión será necesaria su instalación al borrar todos los directorios de la aplicación al finalizar la sesión de *Colab*.

De forma contraria, en Gamma se podrá utilizar la aplicación sin instalar ningún paquete adicional, porque la máquina siempre cuenta con todas las librerías necesarias en el entorno instaladas.

E.4. Manual del usuario

Al poder usar la aplicación desde dos métodos distintos, según donde nos encontremos tendremos un manual de usuario u otro.

Google Colab

Una vez hayamos accedido al *notebook* de *Google Colab* es turno de instalar y descargar todo lo necesario antes de poder utilizar la aplicación. Esto se debe principalmente a que *Google* borra todo el contenido al finalizar sesión y no está instalado *Detectron2* por lo que es necesaria su instalación.

Para poder ejecutar las celdas del *notebook* hay que pulsar al botón circular (Figura E.1) que se encuentra al lado izquierdo de cada celda.

Figura E.1: Botón Ejecución Google Colab

El primer paso será descargar e instalar *Detectron2* que es un proceso que suele tomar unos minutos. A continuación, se descargan las funciones de la aplicación, subidas al repositorio de *GitHub* del proyecto.

Lo siguiente será importar todas las bibliotecas necesarias para el correcto funcionamiento de la aplicación. También será necesario descargarse el modelo de *Google Drive*, que aunque es pesado, tarda apenas unos segundos en descargarse.

Ahora ya se dispone de todo lo necesario para poder usar la aplicación. Es turno de subir las radiografías, para ello se ejecutará la celda **Cargar Imagen**. Ahora habrá aparecido un botón (Figura E.4) que permite subir las radiografías a la aplicación.

Finalmente, se ejecutará la celda **Calcular Longitud**. Tras esperar unos segundos empezarán a aparecer los resultados, formados por la radiografía original, la radiografía segmentada (detección del diente y nervio) y la radiografía con la recta por la cual se ha obtenido su longitud. También, aparecerá un recuadro para cada radiografía con su longitud (Figura E.2).

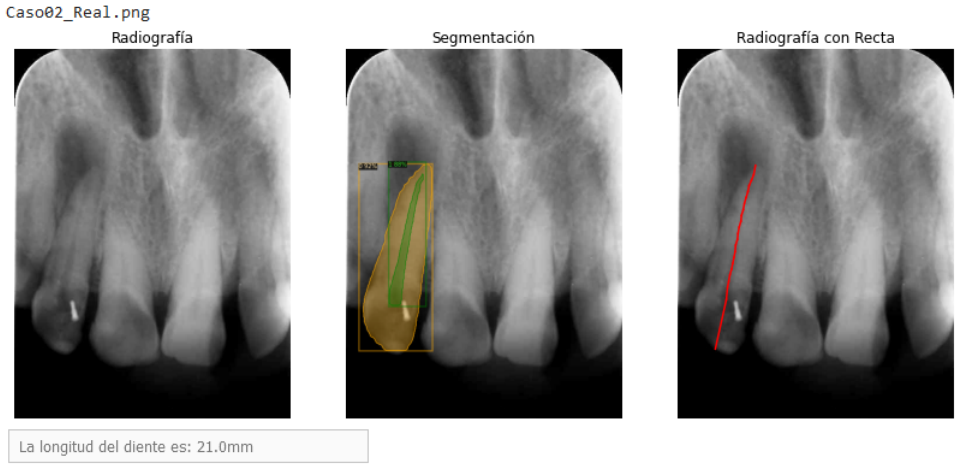

Figura E.2: Resultado Aplicación Final.

Gamma

Una vez hayamos entrado a *Jupyer Notebook* a través de la herramienta *Ngrok*, hay que dirigirse a la aplicación, que se corresponde al fichero `Aplicacion.ipynb`.

Una vez nos encontremos en el interior del *notebook* hay que importar todos los paquetes necesarios en la aplicación. Para ello nos colocamos en la primera celda y usaremos el botón del menú superior **Run**. Una vez se haya terminado de ejecutar, aparecerá a su izquierda el número 1, como se aprecia en la Figura E.3.

```
In [1]: import ipywidgets as widgets
        from funciones import aplicacion
```

Figura E.3: Ejemplo Ejecución Celda 1 Aplicación

A continuación, es turno de preparar la subida de las radiografías a la aplicación. Para ello, nos colocamos en la siguiente celda y la ejecutamos de igual manera. Ahora aparecerá un botón, llamado **Upload** (Figura E.4), que al pulsarlo nos permitirá subir todas las radiografías deseadas.

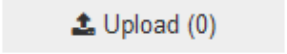

Figura E.4: Botón Upload

Para poder subir varias radiografías a la vez, es tan sencillo como mantener pulsada la tecla *Ctrl* e ir seleccionando todas las radiografías deseadas.

Una vez se hayan subido las imágenes, aparecerá un mensaje en la celda con los nombres de las radiografías cargadas. Ahora es momento de ejecutar el proceso de cálculo. Para ello, nos vamos a la tercera celda y la ejecutamos. Tras esperar unos segundos empezarán a aparecer los resultados, formados por la radiografía original, la radiografía segmentada (detección del diente y nervio) y la radiografía con la recta por la cual se ha obtenido su longitud.

Finalmente, aparecerá una recuadro indicando la longitud del diente correspondiente a cada radiografía (Figura E.2).

Por último, sería recomendable cerrar la sesión del *notebook* que acabamos de usar. Para ello, cerramos la pestaña de la aplicación y en el botón de *Running* (Figura E.5) saldrá la aplicación. Es tan sencillo como pulsar al botón *Shutdown* (Figura E.6) y habremos finalizado toda ejecución en la aplicación. De esta forma se evita que surja cualquier posible problema.

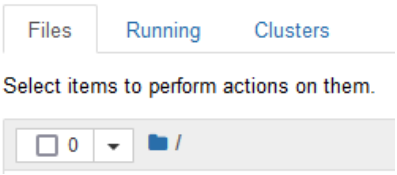

Figura E.5: Botón Running

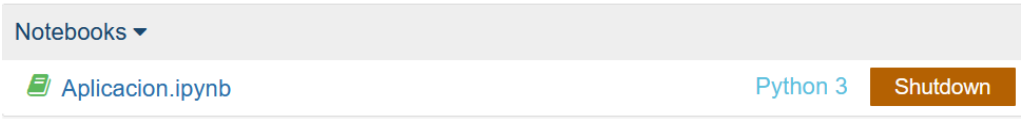

Figura E.6: Botón Shutdown

Bibliografía