UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Identificación de Parkinson por visión artificial

Presentado por Catalin Andrei Cacuci
en Universidad de Burgos — 12 de marzo de 2023
Tutor: Álvar Arnaiz González

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dña. Alicia Olivares Gil y D. Álvar Arnaiz González, profesores del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Catalin Andrei Cacuci, con DNI X7451927L, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado Identificación de Parkinson por visión artificial.

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

En Burgos, a 12 de marzo de 2023

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
Dña. Alicia Olivares Gil	D. Álvar Arnaiz González

Resumen

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

Descriptores

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

Abstract

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

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Objetivos del proyecto

Conceptos teóricos

Técnicas y herraminetas

4.1. Herramientas de desarrollo

Windows Subsystem for Linux

Windows Subsystem for Linux (WSL) es una utilidad creada principalmente por Microsoft y Canonical (Mantenedores de Ubuntu) que permite utilizar un sistema operativo Linux desde Windows sin grandes pérdidas de rendimiento. Existen varias opciones de distribución que se pueden utilizar (Kali Linux, OpenSUSE, etc.), pero la mas utilizada y mejor mantenida para WSL es Ubuntu, por lo que es la que se ha utilizado.

La mayor parte del desarrollo ha sido realizada desde WSL, donde se ha instalado tanto LaTeX como Python 3.9 y algunas dependencias, proceso mucho más simple de realizar en Linux que en Windows.

Visual Studio Code

Visual Studio Code (VS Code) es un popular editor de texto altamente extensible orientado al desarrollo creado por Microsoft. Gracias a su gran extensibilidad es posible utilizarlo cómodamente en casi cualquier escenario. Mediante una de estas extensiones (denominada literalmente WSL) se puede conectar con WSL y utilizar los programas que tiene instalados como si hubiese sido ejecutado diréctamente desde un sistema operativo Linux.

En este caso se ha utilizado VS Code para:

- Crear tanto esta memoria como los anexos mediante la extensión LaTeX Workshop que integra distintos compiladores de LaTeX (pdfTeX, XeTeX,

LuaTeX, etc.) dentro del programa y añade otras funcionalidades que hacen la edición de código LaTeX mucho más conveniente.

- Crear la aplicación web, que utiliza el framework de JavaScript SvelteKit, mediante multitud de extensiones para integrar la multitud de librerías utilizadas (Svelte, TypeScript, Prettier, etc.).

- Editar archivos de configuración utilizados para la creación de los contenedores de Docker que componen la totalidad de la aplicación web (web, API y proxy inverso).

PyCharm

PyCharm es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) creado y mantenido por JetBrains, una compañía que se dedica a crear software y cuyos productos más representativos son multitud de IDEs para diferentes lenguajes de programación (IntelliJ (Java), CLion (C y C++), GoLand (Go), etc.).

El objetivo de PyCharm es facilitar el desarrollo de aplicaciones que utilizad Python y da soporte amplio tanto para los habituales ficheros `.py` como para `.ipynb` (Notebooks de Jupyter), que han sido utilizados extensamente en el proyecto.

Además, PyCharm por defecto permite la integración con entornos WSL de forma muy fácil, simplemente se le debe indicar la localización del ejecutable de Python en el sistema de archivos de la distribución de WSL.

En este proyecto PyCharm se ha utilizado para:

1. Desarrollar la librería PADDEL, que contiene el código de Python común a los Notebooks de Jupyter utilizados durante la fase de investigación y a la aplicación web.

2. Desarrollar la API que utiliza la aplicación web.

Git

Actualmente Git es el sistema de control de versiones más extendido en proyectos de código abierto [1], en este proyecto se ha usado para gestionar y guardar de forma segura los cambios que se han ido realizado con el tiempo en la plataforma GitHub.

Aunque existen varias herramientas con interfaces gráficas que envuelven el funcionamiento de Git (GitKraken, GitHub Desktop, etc.) para este proyecto se ha utilizado directamente el comando `git` desde una terminal de WSL.

OpenCV

OpenCV es una librería de visión artificial que contine varias utilidades para interactuar con archivos de vídeo e imágenes de forma similar a como se trabajaría en un programa como MATLAB. OpenCV está escrito en C y C++ pero tiene *bindings* que permiten interactuar con sus APIs desde Python.

En este proyecto se usa para leer y decodificar los archivos de vídeo utilizados.

MediaPipe

MediaPipe es una librería de aprendizaje automático mantenida por Google que da acceso a varios modelos preentrenados que se centran en la detección de características corporales a partir de flujos de imágenes, en este caso se utilizó MediaPipe Hands, cuyo objetivo es extraer varios puntos que definen un esqueleto ligeramente simplificado de la mano humana.

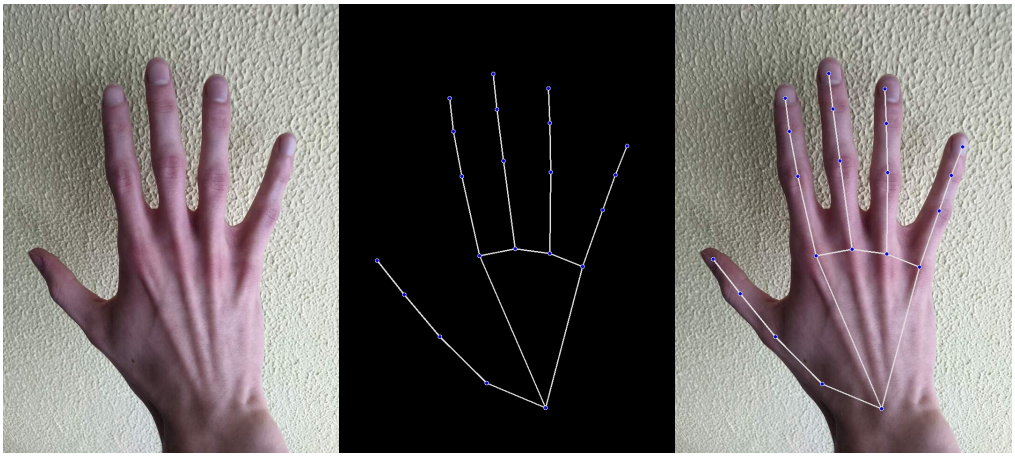

Figura 4.1: MediaPipe Hands

FastAPI

FastAPI es un *framework* de Python utilizado para crear APIs de tipo REST que se caracteriza por permitir al programador centrar sus esfuerzos en el código relacionado con la lógica de negocio al realizar la mayoría de acciones comunes entre diferentes APIs por defecto (como la creación de documentación, validación de tipos, serialización y deserialización de peticiones, etc.).

En este proyecto se ha creado una API REST para permitir la interacción con el código de Python utilizado para crear, entrenar y utilizar los modelos desde cualquier dispositivo y lenguaje de programación (siempre y cuando permita realizar peticiones HTTP). Con esto se consigue que en el futuro se puedan crear nuevas formas para interactuar con los modelos sin necesidad de alterar el código ya existente, por ejemplo, se podría crear una aplicación móvil para facilitar la toma y subida de vídeos. Además estas nuevas aplicaciones podrían estar creadas por personas ajenas al proyecto que necesiten una implementación diferente para su caso de uso específico.

SvelteKit

SvelteKit es el *framework* de JavaScript utilizado para la creación de la página web del proyecto. Para crear una aplicación web con esta herramienta se utilizan archivos `.svelte` que se utilizan para definir las diferentes páginas y los componentes reutilizables de la web.

SvelteKit se caracteriza por estar compilado, es decir, cuando se construye la aplicación los ficheros `.svelte` (no ejecutables por el navegador) son compilados a únicamente ficheros de HTML, CSS y JavaScript. En este paso de compilación se realizan varias optimizaciones sobre el tamaño y velocidad del código, resultando en una página muchos más ligera y rápida.

En este proyecto se ha creado una aplicación es SvelteKit para servir como *frontend* a la API creada con FastAPI.

Caddy

En las fases iniciales de desarrollo se decidió que la API debería estar alojada en la ruta `/api`, mientras el resto de rutas debería llevar a la web (SvelteKit). Esto significa que van a existir dos servicios funcionando de forma paralela (Web y API), y que dependiendo de la ruta a la que se realize una petición, ésta debería ser redirigida a un servicio u otro.

Esta situación es uno de los casos de uso idóneos para un proxy inverso, que es un servicio que recibe peticiones HTTP, las altera si es necesario, y las redirige al servicio de destino (el cual se determina mediante ciertas reglas de redirección).

En un principio se optó por utilizar Nginx para implementar este proxy inverso con muy buenos resultados gracias a su fácil configuración, rendimiento y simplicidad. Pero surgieron problemas al añadir soporte para utilizar certificados SSL y realizar peticiones seguras mediante HTTPS, ya que era deseable una solución que gestionase de forma automática la petición y renovación de estos certificados y Nginx, debido a su simplicidad no dispone de esta funcionalidad.

Se probaron dos alternativas que solucionaban este problema, Traefik y Caddy, ambas escritas en GoLang. Traefik dispone de una gran cantidad de funcionalidades incluyendo gestión de certificados SSL, entre muchas otras, como un panel de control desde el que visualizar estadísticas sobre el nivel de uso del servicio.

La otra alternativa, Caddy, es mucho más simple pero realiza la gestión de certificados de forma automática por defecto, sin necesidad de configuración específica, e incluso permite el uso de certificados *self-signed* (firmados por uno mismo, en lugar de por una entidad certificadora) que es de gran utilidad durante el desarrollo para asegurar que el funcionamiento de la aplicación va a seguir siendo el mismo en el entorno final de producción (hay ligeras diferencias dependiendo de si un navegador utiliza HTTP o HTTPS). Es por esto que al final se decidió usar Caddy.

Docker

Docker es una herramienta que permita la creación y gestión de contenedores de sistemas operativos aislados del sistema anfitrión y entre sí con un *overhead* my pequeño con respecto a una ejecución nativa.

En este caso se ha utilizado Docker para facilitar el despliegue de la aplicación web, con tres contenedores (API, web y proxy inverso), en cualquier máquina que disponga de Docker. La única configuración que se debe realizar para desplegar la aplicación es la creación de un archivo de variables de entorno (`.env`) en el que se definen valores específicos al entorno (nombre de dominio, puertos en los que escuchar las peticiones).

Docker permite la creación de redes locales de contenedores, en este caso se utiliza una de estas redes para permitir la conexión entre los tres

contenedores como si fuesen máquinas distintas conectadas a una misma red local donde la única máquina (contenedor) con conexión al exterior es la que está ejecutando el servicio del proxy inverso, por el que tienen que pasar todas las peticiones antes de llegar al servicio de API o al de web.

Formateado del código

Se han utilizado las siguientes herramientas para realizar el formateado del código escrito en Python de forma automática:

- **Autoflake**: herramienta utilizada para la eliminación de sentencias `import` y variables innecesarias.
- **Isort**: herramienta para la reordenación de sentencias `import`.
- **Black**: formateador de código caracterizado por un enfoque en convenciones, por esto permite una configuración muy limitada por parte del usuario.

Make

A través del proyecto hay comandos de cierta complejidad que se repiten con bastante frecuencia (p. ej. formatear el código, lanzar contenedores, limpiar contenedores, etc.), para facilitar el uso de estos comandos se ha utilizado la utilidad GNU Make, que viene por defecto en la mayoría de sistemas operativos de tipo Linux y se puede instalar tanto en Windows como en MacOS.

El funcionamiento de Make consiste en la creación de un archivo denominado `Makefile` que contiene diferentes comandos y secuencias de comandos con las dependencias que existen entre los mismos.

Por ejemplo:

```
format:
    autoflake src
    isort src
    black src
```

Un fichero `Makefile` con el contenido anterior permite ejecutar el comando `make format` (siempre desde el mismo directorio en el que se encuentra `Makefile`) para ejecutar la secuencia de comandos denominada `format`, que, en este caso ejecutará varios comandos para realizar un formato del código.

Aspectos relevantes del desarrollo del proyecto

Trabajos relacionados

Durante la última década se ha intentado utilizar la visión por computador para evaluar la Enfermedad de Parkinson en múltiples ocasiones con diferentes resultados.

Este capítulo recopila de forma resumida algunos de los trabajos más relevantes.

6.1. A computer vision framework for finger-tapping evaluation

Este artículo [2] documenta el uso de visión por computador para determinar el nivel de severidad de la Enfermedad de Parkinson y distinguir entre individuos con esta enfermedad e individuos sin ella.

El método empleado se caracteriza por utilizar vídeos con la cara de la persona y ambas manos a los lados de la cabeza, apuntando las puntas de los dedos hacia la misma. Esto se utiliza para poder normalizar las distancias en base a características faciales.

El estudio se realizó sobre 13 pacientes con la Enfermedad de Parkinson, tomando 17 vídeos de cada paciente durante un día, además de un grupo de control de 6 individuos, tomando 2 vídeos al día durante una semana por cada uno. Aunque algunos de estos vídeos fueron descartados. En total se utilizaron 471 vídeos.

Metodología

1. Se detecta la cara del individuo para la normalización. Esto se basa en que la longitud de la mano de una persona adulta es aproximadamente igual a la altura de su cara.

2. Se obtiene una serie temporal que representa la amplitud del movimiento de los dedos índice y pulgar de la mano dominante del individuo.

3. Se extraen un total de 15 características de esta serie temporal, por ejemplo:

   - Correlación cruzada media entre los máximos locales de dos intervalos distintos de tiempo de la serie temporal. Esto mismo se realiza también sobre los mínimos locales.
   - Número total de toques de dedos durante la grabación.
   - Velocidad media de la apertura de dedos.
   - Velocidad media del cierre de dedos.
   - ...

4. Se realiza una selección de características eliminando aquellas redundantes y usando el algoritmo chi-cuadrado.

5. Se entrena una máquina de vectores de soporte (SVM) mediante las características obtenidas para realizar la clasificación.

Resultados En cuanto a la distinción entre pacientes de la Enfermedad de Parkinson y el grupo de control se obtuvo una precisión del 95 %, que es una cifra que se debería tomar con precaución debido que, aunque se han utilizado 471 vídeos, estos provienen de únicamente 19 personas.

6.2. The discerning eye of computer vision

En este estudio [4] realizado sobre 39 pacientes con la Enfermedad de Parkinson y sobre un grupo de control de 30 individuos se tomaron vídeos de ambas manos de cada individuo mientras realizan toques de los dedos índice y pulgar (de forma similar a cómo se han tomado las muestras para este trabajo). Dando un total de 133 vídeos (se descartó uno).

De estos vídeos se han extraído diferentes características y comprobado la relación que existe entre éstas y diferentes escalas que clasifican el nivel de gravedad de la Enfermedad en un paciente, como la *Modified Bradykinesia Rating Scale* (MBRS) que categoriza el movimiento de los individuos en 5

niveles, del 0 al 4, siendo 0 un movimiento normal y 4 el nivel de mayor gravedad.

Metodología

1. Se utiliza una librería de visión por computador, en concreto DeepCutLab, para obtener una serie temporal de la amplitud entre las puntas de los dedos pulgar e índice.
2. Se normaliza esta serie temporal utilizando la amplitud máxima detectada, que va a convertirse en el valor 1, siendo todos los demás valores escalados proporcionalmente.
3. Se extraen las siguientes características:
   - Velocidad, calculada como la tasa media de cambio.
   - Variabilidad de la amplitud, calculada como el coeficiente de variación de la diferencia media entre máximos y mínimos de diferentes intervalos de 1 segundo de la serie temporal.
   - Regularidad del ritmo, calculada utilizando la Transformada Rápida de Fourier y, a continuación, midiendo la potencia de la frecuencia dominante más la potencia de las frecuencias en un intervalo de 0.4 Hz alrededor de ésta (un ritmo más regular concentra una mayor potencia en una única frecuencia).

Resultados Se observó una correlación bastante alta entre las características utilizadas y la categoría del individuo dentro de las escalas de medición de la Enfermedad de Parkinson utilizadas medida por un experto en el campo.

6.3. Supervised classification of bradykinesia

Este estudio [3] es muy similar al anteriormente explicado, y está realizado por un equipo compuesto por casi los mismos participantes. En este caso se utilizaron 70 vídeos, de ambas manos de 20 pacientes con la Enfermedad de Parkinson y de un grupo de control de 15 individuos.

Metodología La metodología es prácticamente igual que antes, la diferencia principal está en las características que se extraen de la serie temporal correspondiente con la amplitud, se ha obtenido:

- Frecuencia, medida como la frecuencia máxima de la Transformada Rápida de Fourier de la serie temporal.
- Amplitud, calculada como la densidad espectral, que se ha obtenido mediante la integral cuadrada del espectro de la Transformada Rápida de Fourier.

Con estas características se ha realizado clasificación binaria mediante clasificación bayesiana ingenua (naive bayes), regresión logística y máquina de vectores de soporte, tanto con función lineal como con función de base radial.

Resultados Los mejores resultados se obtuvieron con máquina de vectores de soporte con función de base radial, que coincide en un 73 % de los casos con la clasificación de expertos en el campo.

Conclusiones y Líneas de trabajo futuras

Bibliografía

[1] MS Windows NT kernel description. https://www.openhub.net/repositories/compare. Accessed: 2023-03-13.

[2] Taha Khan, Dag Nyholm, Jerker Westin, and Mark Dougherty. A computer vision framework for finger-tapping evaluation in parkinson's disease. *Artificial intelligence in medicine*, 60(1):27–40, 2014.

[3] Stefan Williams, Samuel D Relton, Hui Fang, Jane Alty, Rami Qahwaji, Christopher D Graham, and David C Wong. Supervised classification of bradykinesia in parkinson's disease from smartphone videos. *Artificial Intelligence in Medicine*, 110:101966, 2020.

[4] Stefan Williams, Zhibin Zhao, Awais Hafeez, David C Wong, Samuel D Relton, Hui Fang, and Jane E Alty. The discerning eye of computer vision: Can it measure parkinson's finger tap bradykinesia? *Journal of the Neurological Sciences*, 416:117003, 2020.