



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 — 29 de abril
de 2023

Tutora: Alicia Olivares Gil
Tutor: Álgar Arnaiz González



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



Dña. Alicia Olivares Gil y D. Álgvar 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 29 de abril de 2023

Vº. Bº. de la Tutora:

Vº. Bº. del co-tutor:

Dña. Alicia Olivares Gil

D. Álgvar 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 general	III
Índice de figuras	IV
Índice de tablas	V
1. Introducción	1
2. Objetivos del proyecto	3
3. Conceptos teóricos	5
4. Técnicas y herraminetas	7
4.1. Herramientas de desarrollo	7
5. Aspectos relevantes del desarrollo del proyecto	17
6. Trabajos relacionados	19
6.1. A computer vision framework for finger-tapping evaluation .	19
6.2. The discerning eye of computer vision	20
6.3. Supervised classification of bradykinesia	21
7. Conclusiones y Líneas de trabajo futuras	23
Bibliografía	25

Índice de figuras

4.1. MediaPipe Hands	11
--------------------------------	----

Índice de tablas

1. Introducción

2. Objetivos del proyecto

3. Conceptos teóricos

4. Técnicas y herraminetas

4.1. Herramientas de desarrollo

Este capítulo detalla las distintas herramientas, librerías y frameworks que fueron utilizadas durante el desarrollo de este proyecto.

Fedora Linux

Aunque al iniciar el proyecto se utilizó Windows 10 como sistema operativo principal, la mayor parte del desarrollo ha sido realizado en la distribución de Linux **Fedora**, en concreto versiones 37 y 38.

Fedora se caracteriza por que sus repositorios se actualizan regularmente para incluir las últimas versiones estables de la mayoría de programas, al contrario de lo que ocurriría en una distribución como Debian estable. Pese a esto, Fedora es considerada como una distribución bastante estable y, durante el desarrollo de este proyecto, no se han encontrado problemas fuera de lo común.

Han sido instalados de forma nativa Python 3.9 para la fase de investigación, **T_EX Live** para la compilación de esta documentación y **Docker** para el desarrollo de la aplicación web.

Python

Python es un lenguaje de programación diseñado con el objetivo de ser fácil de escribir, para lograr esto existe una gran cantidad de abstracciones y detalles de los que se encarga, mientras que en otros lenguajes debería encargarse el programador. Esto tiene muchas ventajas, pero también algunas

desventajas, como, por ejemplo, que se pierde el control sobre la ejecución del programa que podría dar un lenguaje como C o C++, lo que significa que, en general, un mismo programa escrito completamente en Python va a ser más lento que su contrapartida escrita en un lenguaje de programación de más bajo nivel en el que se pueden especificar detalles como, por ejemplo, la gestión de memoria.

Pero Python tiene otra cualidad muy importante, y es su facilidad para interactuar con código escrito en C o C++ mediante lo que se conoce como *bindings* o *wrappers*. Esto permite al programador tener mayor control sobre las partes más críticas de un programa, escribiéndolas en C por ejemplo, mientras que las partes de menor importancia se pueden escribir rápidamente en Python.

Un ejemplo perfecto de lo anterior es la librería **NumPy**, utilizada para la gestión de matrices, entre muchas otras cosas. Esta librería implementa su funcionalidad principal en C y expone una interfaz en Python para facilitar su uso.

Este patrón se repite una y otra vez en el campo del aprendizaje automático, donde los algoritmos de aprendizaje se implementan a bajo nivel para que su ejecución sea lo más rápida posible, mientras que se expone una interfaz a alto nivel en Python para interactuar con los mismos. Ejemplos de esto son las librerías **PyTorch** y **TensorFlow**.

Jupyter Notebook

Jupyter Notebook es una herramienta que permite crear un archivo con extensión `.ipynb`, en el que el código Python se divide en “celdas” que se pueden ejecutar individualmente, pero manteniendo y compartiendo las variables establecidas de forma global entre todas estas celdas.

Esta herramienta fue de especial utilidad durante la fase de investigación, ya que permite observar detenidamente los cambios que se producen entre celdas y, así, encontrar problemas sin tener que ejecutar el código completo desde el principio cada vez que se realice un cambio, como ocurriría con un *debugger*.

Además de lo anterior Jupyter Notebook permite documentar de forma clara el código mediante otro tipo de celda, en el que se puede escribir código en Markdown, que es, en cierto modo, una alternativa menos verbosa a HTML.

Docker

Docker permite la creación y gestión de contenedores, un contenedor se puede ver como un sistema operativo dentro del anfitrión, pero sin llegar a ser una máquina virtual, debido a que se utiliza el *kernel* de Linux del sistema anfitrión y los archivos del contenedor existen en el sistema de archivos del anfitrión. Esto hace que la diferencia de rendimiento entre utilizar un contenedor y ejecutar programas de forma nativa sea mínima.

Un contenedor de Docker se basa en lo que se conoce como una imagen, que es un sistema de archivos preconfigurado para el propósito del contenedor. Se puede encontrar una gran cantidad de imágenes en [Docker Hub](#), como, por ejemplo, postgres, node o python que ya incluyen los ejecutables para utilizar cada herramienta respectiva. Pero, es posible que se necesite mayor control sobre la configuración de la imagen sobre la que se va a construir el contenedor, en estos casos se pueden utilizar lo que se conoce como *Dockerfile* para crear una imagen tomando como base otra imagen ya existente.

Un *Dockerfile* es un archivo que determina la imagen de base a utilizar (cualquier imagen de [Docker Hub](#) u otros sitios) y los comandos a ejecutar para crear la imagen que se desea obtener. Si se deseara empezar de cero se puede utilizar una imagen de Ubuntu o Debian como base.

Además de *Dockerfile* existe otro tipo de archivo denominado *docker-compose.yml* utilizado para coordinar sistemas compuestos por múltiples contenedores que interactúan entre sí. La aplicación web de este proyecto utiliza *docker-compose.yml* para determinar las interacciones entre los cuatro contenedores que existe (web, API, proxy inverso y base de datos).

PyCharm

PyCharm es un Entorno de Desarrollo Integrado (IDE) 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 utilizan Python y da soporte amplio tanto para los habituales ficheros `.py` como para `.ipynb` (Jupyter Notebook).

En este proyecto PyCharm fue utilizado para desarrollar la librería PADDEL durante la fase de investigación, que contiene el código de Python común a los Notebooks de Jupyter utilizados durante la fase de investigación.

Visual Studio Code

Visual Studio Code (VS Code) es un editor de texto orientado al desarrollo creado por Microsoft. Gracias a su gran extensibilidad es posible utilizarlo cómodamente en casi cualquier escenario.

Fue utilizado en la creación de esta documentación en conjunto con la extensión **L^AT_EX Workshop** que integra distintos compiladores de L^AT_EX (pdfTeX, XeTeX, LuaTeX, etc.) dentro del programa y añade otras funcionalidades que hacen trabajar L^AT_EX mucho más conveniente.

Además, gracias a la extensión **Dev Containers**, VS Code puede ser lanzado desde contenedores de Docker, permitiendo interactuar con las utilidades del contenedor. Gracias a esto VS Code ha sido la herramienta utilizada para desarrollar por completo la aplicación web.

Git

Actualmente Git es el sistema de control de versiones más extendido en proyectos de código abierto [3], 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 utilizó directamente el comando `git` desde una terminal de Linux.

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 interfaces desde Python.

En este proyecto se usa para leer y decodificar los archivos de vídeo utilizados durante la fase de investigación.

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[6], cuyo objetivo es extraer varios puntos que definen un esqueleto ligeramente simplificado de la mano humana.

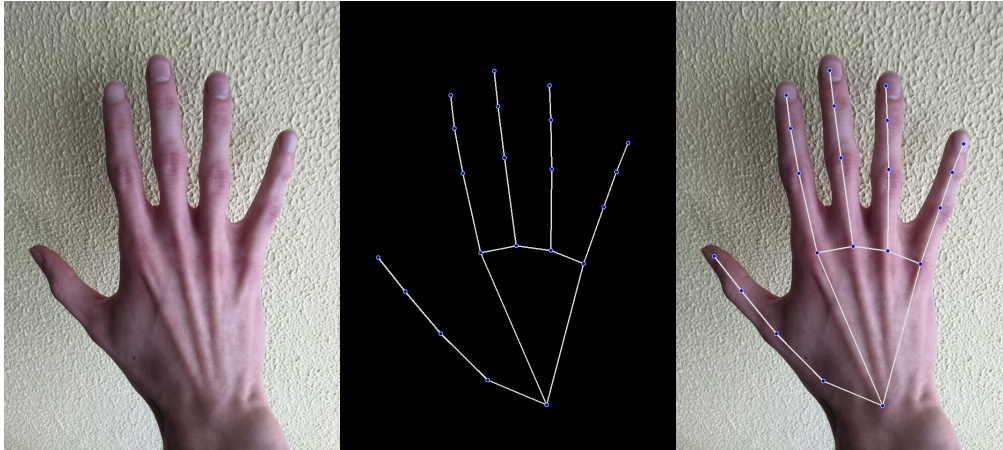


Figura 4.1: MediaPipe Hands

Como se puede ver el resultado es bastante bueno, además, al utilizar vídeos MediaPipe utiliza la información de los fotogramas anteriores para determinar de forma más exacta la pose en los fotogramas posteriores. Esto es muy útil para mejorar la precisión de las poses obtenidas, pero tiene la pega de que, al necesitar información de pasos previos, el proceso de extracción de poses no se puede paralelizar.

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. Se podría, por ejemplo, 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

Para el frontend de la aplicación, es decir, todo lo que va a ver un usuario de la web (HTML, CSS y JavaScript), se ha utilizado SvelteKit, que es un framework de JavaScript basado en Svelte, es similar a otros frameworks más populares como React o Angular, y se caracteriza por tener una sintaxis muy clara y simple además de reducir el tamaño de los archivos que se envían al navegador del cliente, haciendo que, en general, las páginas creadas con SvelteKit sean más rápidas que sus contrapartidas.

Typescript

JavaScript es un lenguaje de programación que no dispone de tipado estático, lo hace muy difícil detectar problemas fuera del tiempo de ejecución.

TypeScript es un lenguaje de programación desarrollado por Microsoft que utiliza una sintaxis de base muy similar a la de JavaScript, pero añadiendo tipado estático. Los navegadores no “entienden” TypeScript, por lo que éste debe ser compilado a JavaScript antes de ser enviado al usuario. Durante esta compilación se pueden detectar multitud de errores gracias al tipado.

Typesafe i18n

Typesafe i18n es una librería escrita en TypeScript que se puede utilizar para internacionalizar aplicaciones web, es decir, añadir soporte para más idiomas, de forma simple y mediante unas convenciones predefinidas.

Para ello se crea un archivo para cada idioma al que se le da soporte en el que se definen las variables con las cadenas de texto utilizadas a través de la web en el idioma respectivo. Cada uno de estos archivos debe contener las mismas variables.

A continuación en los lugares donde se usen cadenas de texto se debe importar la variable `LL` que es un objeto con todas las traducciones que se puede usar en la página respectiva para establecer las cadenas de texto dependiendo del idioma.

Axios

Axios es una librería de JavaScript que envuelve la función nativa `fetch` para hacerla más amigable y fácil de usar. Se caracteriza por soportar tipos de TypeScript y simplificar la realización de peticiones HTTP.

En este proyecto se ha utilizado esta librería para realizar peticiones REST a la API desde el navegador del usuario. Para ello se han creado multitud de funciones que reflejan los *endpoints* de la API y, así, permitir su uso desde el resto del sitio web.

Tailwind CSS

Tailwind CSS es una librería de clases de CSS su funcionamiento es similar a **Bootstrap** en cuanto a que existen multitud de clases que realizan diferentes acciones, la diferencia está en que Tailwind CSS es mucho más modular que Bootstrap, es decir, las clases realizan cambios muy pequeños sobre los objetos a los que afectan. Esto hace que Tailwind sea más lento de escribir que su contrapartida, pero al mismo tiempo permite un nivel mucho mayor de configuración sobre el diseño de una página web.

Prettier

Prettier es un formateador de código, creado concretamente para dar un formato consistente a archivos de HTML, CSS, JavaScript, TypeScript, etc.

Caddy

En las fases iniciales de desarrollo se decidió que la API debería estar alojada en el subdominio `api.`, mientras el dominio principal debería llevar a la web (SvelteKit). Esto significa que van a existir dos servicios funcionando de forma paralela (Web y API), y que dependiendo del dominio al que se realice 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, e incluso permite el uso de certificados *self-signed* (firmados por uno mismo, en lugar de por una entidad certificadora). Esto 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 (ya que hay ligeras diferencias dependiendo de si un navegador utiliza HTTP o HTTPS).

Es por todo lo anterior que, en última instancia, se decidió usar Caddy.

Formateado de código Python

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 (*e.g.* 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.

TSFresh

TSFresh^[1] es una librería de Python que se encarga del preprocesado y la extracción de características de series temporales.

Una serie temporal es una secuencia ordenada de valores numéricos con un valor temporal asociado. Estas secuencias por sí solas no son de mucha utilidad para entrenar un modelo, por lo que en la fase de preprocesado se debe realizar una extracción de características sobre las mismas y así obtener características numéricas que sí se pueden utilizar para entrenar un modelo.

TSFresh establece una forma estándar para extraer estas características, permitiendo extraer características propias además de las, aproximadamente, 700 características que extrae por defecto.

5. Aspectos relevantes del desarrollo del proyecto

6. 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 [5] 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 [4] 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.

7. Conclusiones y Líneas de trabajo futuras

Bibliografía

- [1] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh—a python package). *Neurocomputing*, 307:72–77, 2018.
- [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] OpenHub. MS Windows NT kernel description. <https://www.openhub.net/repositories/compare>. Accessed: 2023-03-13.
- [4] 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.
- [5] 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.
- [6] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.