UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Comparativa de APIs para el uso de las manos como dispositivos para la interacción con el computador

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Aragó Eliche, Esther

Tutor/a: Agustí Melchor, Manuel

CURSO ACADÉMICO: 2021/2022

Resumen

En 1938 se terminó de construir la Z1, considerada como la primera computadora mecánica programable del mundo, desde entonces nuestros computadores han avanzado a un ritmo vertiginoso y también la manera de interactuar con ellos.

Hemos pasado de tener que conectar y cambiar diversos cables, a solo tocar un botón o mover un ratón para que el computador realice lo que necesitamos. Nuestros deseos de mejorar la interacción nos ha llevado a conseguir interactuar mediante pantallas táctiles o incluso poder utilizar nuestra huella dactilar como mecanismo de seguridad. Desde hace mucho tiempo nos hemos imaginado un futuro en el que poder interactuar con el computador sin necesidad de contacto y ya hay nuevos dispositivos que se han atrevido a implementar esta tecnología, ya sea mediante sensores o captación de imagen.

La comunicación a través de gestos proporciona una forma de comunicación más sencilla e intuitiva, además de aportar avances para las personas con algún tipo de problema de movilidad o dermatológico e incluso evitar propagar enfermedades por el contacto de diferentes personas con un mismo dispositivo.

El proyecto aborda la creación de una aplicación, compatible con cualquier computador, que detecte los gestos que realice una persona con la mano para poder interactuar con el dispositivo, con el objetivo de ofrecer una forma distinta de comunicación al teclado y ratón.

Para su realización se han estudiado las diferentes posibilidades de detectar los gestos y explorado el software de código abierto que hemos encontrado en la web, comparando así las diversas librerías que utilizan.

La comparación se ha realizado mediante una serie de pruebas que han medido la bondad de los modelos y el código utilizado, observando el tiempo que se emplea en su ejecución así como los recursos utilizados. A continuación se ha determinado la API más adecuada que servirá como base de nuestra aplicación.

Finalmente, mediante diferentes aportaciones al código base se ha obtenido lo que forma nuestra aplicación final, capaz de reconocer gestos en tiempo real y realizar acciones en el computador utilizando únicamente la cámara para capturar los gestos realizados.

Palabras clave: *Detección de gestos, visión por computador, machine learning, tiempo real, interacción, redes profundas*

Abstract

In 1938 the construction of the Z1 was completed, considered to be the first programmable mechanical computer in the world, since then our computers have advanced at a dizzying pace and also the way of interacting with them.

We have gone from having to connect and change various cables, to just touching a button or moving a mouse for the computer to do what we need. Our desire to improve interaction has led us to manage to interact through touch screens or even to be able to use our fingerprint as a security mechanism. For a long time we have imagined a future in which we can interact with the computer without the need for contact and there are already new devices that have dared to implement this technology, either through sensors or image capture.

Communication through gestures provides a simpler and more intuitive form of communication, in addition to providing advances for people with some type of mobility or dermatological problem and even avoiding the spread of diseases due to the contact of different people with the same device.

The project tries to create an application, compatible with any computer that detects the gestures that a person makes with his hand to be able to interact with the device, with the aim of offering a different form of communication than keyboard and mouse.

For its realization, the different possibilities of detecting gestures have been studied and the open source software that we have found on the web has been explored, comparing the various libraries that they use.

The comparison has been made through a series of tests that have measured the goodness of the models and the code used, observing the time spent in their execution as well as the resources used. Next, the most suitable API that will serve as the basis of our application has been determined.

Finally, through different modifications of the base code, what forms our final application has been obtained, capable of recognizing gestures in real time and performing actions on the computer using only the camera to capture the gestures.

Palabras clave: *Gesture detection, computer vision, machine learning, real time, interaction, deep neural networks*

Resum

En 1938 es va acabar de construir la Z1, considerada com la primera computadora mecànica programable del món, des d'aquell moment els nostres computadors han avançat a un ritme vertiginós i també ho ha fet la manera d'interactuar amb ells.

Hem passat d'haver de connectar i canviar diversos cables, a només tocar un botó o moure un ratolí perquè el computador realitze el que necessitem. Els nostres desitjos de millorar la interacció ens ha portat a aconseguir interactuar mitjançant pantalles tàctils o fins i tot poder utilitzar la nostra empremta dactilar com a mecanisme de seguretat. Des de fa molt de temps ens hem imaginat un futur en el qual poder interactuar amb el computador sense necessitat de contacte i ja hi ha nous dispositius que s'han atrevit a implementar aquesta tecnologia, ja siga mitjançant sensors o captació d'imatge.

La comunicació a través de gestos proporciona una forma de comunicació més senzilla i intuïtiva, a més d'aportar avanços per a les persones amb algún tipus de problema de mobilitat o dermatològic i fins i tot evitar propagar malalties pel contacte de diferents persones amb un mateix dispositiu.

El projecte aborda la creació d'una aplicació, compatible amb qualsevol computador, que detecte els gestos que realitze una persona amb la mà per a poder interactuar amb el dispositiu, amb l'objectiu d'oferir una forma diferent de comunicació al teclat i ratolí.

Per a la seua realització s'han estudiat les diferents possibilitats de detectar els gestos i s'ha explorat el software de codi obert que hem trobat a la web, comparant així les diverses llibreries que utilitzen.

La comparació s'ha realitzat mitjançant una sèrie de proves que han mesurat la bondat dels models i el codi utilitzat, observant el temps que s'empra en la seua execució així com els recursos utilitzats. A continuació s'ha determinat la més adequada que servirà com a base de la nostra aplicació.

Finalment, mitjançant diferents aportacions al codi base s'ha obtingut el que forma la nostra aplicació final, capaç de reconéixer gestos en temps real i realitzar accions en el computador utilitzant únicament la càmera per a capturar els gestos realitzats.

Paraules clau: *Detecció de gestos, visió per computador, machine learning, temps real, interacció, xarxes neuronals*

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1 Motivación

Hoy en día siempre que utilizamos algún dispositivo electrónico necesitamos alguna herramienta como intermediaria para interactuar con este, lo que es conocido como periférico de entrada.

En el día a día nos encontramos con todo tipo de dispositivos electrónicos para los que se debe hacer uso de un intermediario, por ejemplo para ver la televisión se emplea un mando a distancia donde se deben de pulsar los botones. También nos encontramos en un mundo donde los "smartphones" nos acompañan a cualquier lugar, anteriormente eran los teléfonos móviles con los que interactuábamos mediante botones, hoy en día basta con una pantalla táctil. Por otro lado, muchas personas trabajan con ordenadores o los usan en su tiempo libre, utilizando con ellos los teclados y ratones. Igual que pasa con las televisiones, las personas que juegan a videojuegos con alguna consola tienen un mando con el que interactuar.

Todos estos dispositivos requieren de una manipulación manual, ya sea tocando un botón o moviendo el dedo por la pantalla. Es por esto que si nos paramos a pensar no podríamos hacer nada de nuestro día a día sin tener el contacto de nuestras manos con algún dispositivo.

Bien es cierto que la comunicación con los dispositivos ha ido cambiando, hemos observado cómo la industria ha evolucionado la interacción persona-máquina. En el caso de los teléfonos móviles hemos pasado de solo utilizar botones a utilizar una pantalla táctil e incluso poder desbloquear el teléfono según nuestra "id facial" o nuestra huella dactilar. Incluso hoy en día, en ciertas televisiones, podemos cambiar de canal solo con nuestra voz, gracias a los asistentes de voz. Además hemos visto como las consolas han mejorado esta interacción a distancia mediante sensores, que se empezó a popularizar con la Wii (Nintendo) y continuado con Move (Playstation) y Kinect (Xbox).

En el año 2020, con la llegada de la Covid-19 a nuestras vidas, nos hemos dado cuenta de todo lo que manipulamos con nuestras manos y lo incómodo que puede llegar a resultarnos el tocar todo con ellas. Además existe un pequeño sector que tienen problemas de piel que les impiden tocar elementos o simplemente que les es complicado utilizar un ratón o un teclado, por ejemplo.

Es por todo esto, que el uso de los gestos y movimientos de nuestras manos podría ser una buena solución para estas personas a la hora de comunicarse con un dispositivo electrónico. En nuestro caso nos vamos a centrar en utilizar las manos como dispositivos para interacción con el computador, olvidándonos así de usar ratones y teclados.

1.2 Objetivos

El objetivo principal del presente Trabajo Fin de Grado es **evaluar distintas librerías** que servirán como base para el **desarrollo** de una aplicación que permita al usuario **interactuar con el computador mediante unos gestos simples e intuitivos** realizados con la mano. De forma más concreta, se trata de hacer un prototipo de aplicación que funcione a modo de script. Al realizar un script se trata de no añadir una interfaz gráfica, se trata de abordar las implicaciones de un uso transparente de los gestos como entrada de datos de usuario. Así la aplicación debe servir para encontrar

la librería que nos permita reconocer gestos. Es importante recalcar que debe de funcionar a tiempo real, es decir, al momento que se haga un gesto debe de responder con una acción por parte del computador.

Para poder llevar a cabo el objetivo principal será necesario alcanzar los siguientes subobjetivos:

- Establecer gestos distintos entre sí que no creen conflictos y tengan una mayor usabilidad.
- Realizar pruebas de las distintas librerías y compararlas.
- Regular el volumen del equipo, ya sea para subirlo o bajarlo.
- Realizar movimiento del cursor y acción de clic izquierdo.
- Pasar transparencias en una presentación, tanto hacia delante como hacia atrás.

1.3 Estructura de la memoria

La estructura de esta memoria se puede dividir en tres partes. La primera de ellas es un estudio sobre el significado y aplicaciones de las técnicas que se utilizan en la actualidad, así como los dispositivos que interactúan mediante gestos y tenemos disponibles en el mercado. La segunda parte tendrá como base los prerrequisitos desde los cuales queremos comenzar la aplicación, examinando, analizando y comparando las diferentes librerías sobre detección de manos que podemos encontrar de código abierto. Por último, la tercera parte, se explicará en detalle el desarrollo de la aplicación y su puesta en marcha, así como las dificultades encontradas y las pruebas finales realizadas por distintos usuarios.

Por último, cabe destacar que las imágenes utilizadas para las pruebas de las librerías y los ficheros resultantes de la aplicación son accesibles en el siguiente enlace: [https://github.com/esarel1998/TFG-GestureInteraction].

2. Estado del arte

En este segundo apartado se realizará una investigación de la evolución en la detección de gestos y los productos que se han ido encontrando en el mercado a lo largo de la historia. Además se definirán tres conceptos que hoy en día están muy de moda, como inteligencia artificial, machine learning y Deep learning. Conocer estos conceptos nos ayudarán a comprender mejor el proyecto.

2.1 Inteligencia artificial, Machine Learning y Deep Learning

La **Inteligencia Artificial (AI)** y el **Machine Learning (ML)** son dos conceptos muy de moda hoy en día, pero es importante matizar que no se trata de lo mismo, tienen grandes diferencias.

En 1955 fue la primera vez que se acuñó el término de Inteligencia Artificial, fue *John McCarthy*, profesor de matemáticas en la universidad de Dartmouth, donde lo definió como un proceso capaz de *"hacer que una máquina se comporte de formas que serían llamadas inteligentes si un ser humano hiciera eso".* Esta primera definición fue muy acertada aunque actualmente se ha ampliado, ya no solo se basa en el comportamiento humano sino también a razonar como las personas, es decir, razonar de una forma racional y actuar como tal.

Al hablar de Machine Learning realmente se trata de un subconjunto de la AI, se trata del conjunto de algoritmos capaces de encontrar patrones en datos masivos y elaborar predicciones, mejorando cada vez su respuesta y aprendiendo de ellos de forma autónoma, sin necesidad de intervención humana.

A su vez podemos hablar del **Deep Learning (DL)**, un subconjunto de ML, que podemos definir como la parte que hace referencia al aprendizaje profundo o automático. Lo que se realiza en este aprendizaje profundo es un proceso por capas que simula el funcionamiento de nuestro cerebro a través de las redes neuronales. En otras palabras, el DL utiliza modelos estadísticos mediante los cuales localiza patrones. Esta técnica se utiliza en la detección de imágenes, su clasificación o el reconocimiento facial entre otros.

En la figura 1 podemos encontrar de forma más clara como relacionamos los conceptos de AI, ML y DL.

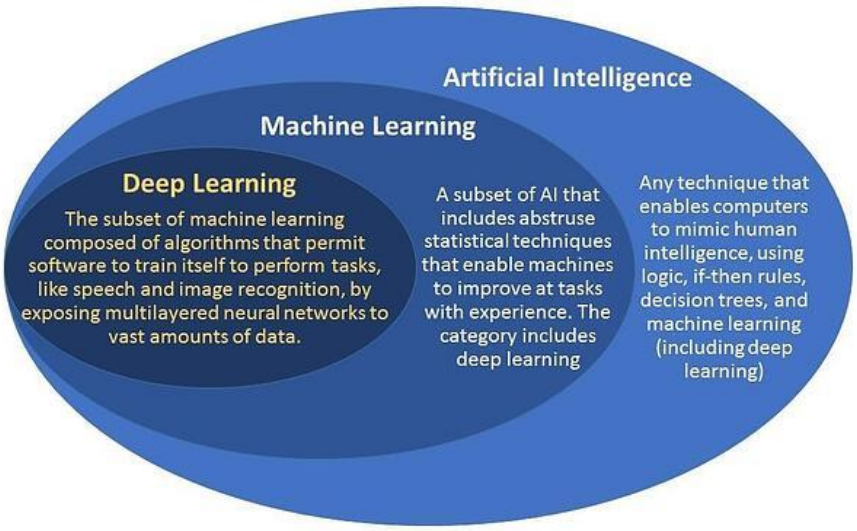


Figura 1 *Relación entre los conceptos AI, ML y DL*

Fuente: [1]

También es importante destacar que el ML tiene dos tipos de aprendizaje el supervisado y el no supervisado. El primero de estos se caracteriza porque los datos entrantes tienen una etiqueta proporcionada por el ser humano, es decir, está en supervisión del ser humano que el algoritmo encuentre en los datos a los elementos etiquetados. El aprendizaje no supervisado es el cual los datos no vienen con etiquetas previas, lo que hace que la intervención del ser humano sea innecesaria.

Es importante notar la diferencia que existe entre ML y DL: DL se trata de un tipo de algoritmo para obtener un modelo, ML puede utilizar árboles de decisión basados en umbrales obtenidos de forma estadística o DL basado en redes neuronales.

A continuación vamos a profundizar en tres clasificadores distintos, los árboles de decisión y las máquinas de vectores de soporte que se tratan de unos métodos clásicos de ML. El otro clasificador que nombramos es el de aprendizaje profundo, una evolución de las redes clásicas y que permiten un gran número de datos de entrada y también un gran número de capas intermedias.

2.1.1 Árboles de decisión

El algoritmo de árbol de decisión suele utilizarse en el ML, se trata de un modelo de predicción que mediante construcciones lógicas y siguiendo unas reglas llega a una conclusión. Como vemos en la figura 2 se parte de un nodo inicial, el nodo raíz, desde el cual se pasa según cumpla o no cumpla una condición a otro nodo, el cual puede ser final, donde ya tendríamos una solución al problema, o nodo interno donde se volverá a evaluar según una condición y pasará al siguiente nodo. Este proceso se repetirá hasta que se llegue a una solución final que sea lo más cercana a una solución óptima al algoritmo.

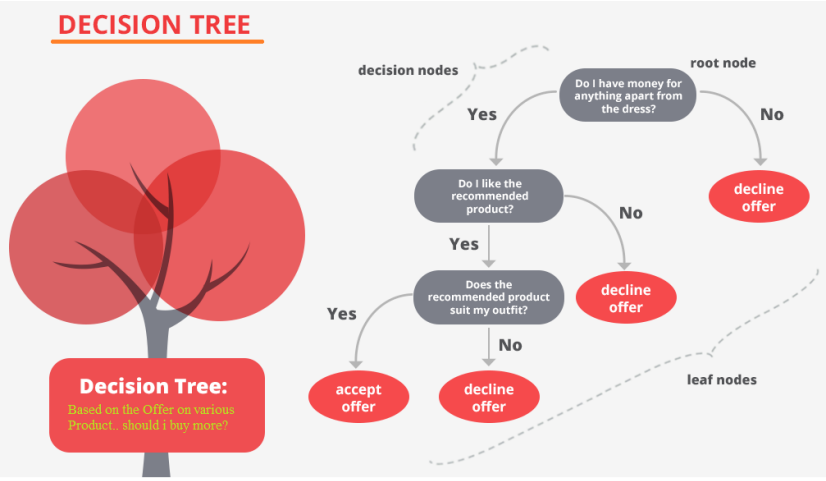


Figura 2 *Ejemplo Árbol de Decisión*

Fuente: [2]

Este algoritmo es muy fácil de entender y al ser simplemente clasificar siguiendo una condición el cálculo es sencillo. El problema es que los árboles de decisión se adaptan muy bien a los datos introducidos, buscando mucho los detalles, lo que hace que no se pueda tener un árbol generalizado para datos similares. Es por esta razón que en ML se usan más de un árbol de decisión, lo que llamamos Random Forest, el cual realiza con distintas muestras de los datos de entrenamiento distintos árboles donde se pueden llegar a diferentes resultados cogiendo como válido aquel resultado que más veces se repita. De esta forma se conseguirá un modelo más generalizado y fiable.

2.1.2 Aprendizaje profundo

Como bien se ha nombrado anteriormente el “Deep Learning” se caracteriza por tener un comportamiento similar a las redes neuronales humanas. El modelo que se emplea en “Deep Learning” suele ser el de redes neuronales artificiales, a pesar de ser un modelo mucho más complicado que el de árboles de decisión, podemos encontrar muchas librerías que con pocas líneas de código son capaces de crear redes neuronales.

El funcionamiento de las redes neuronales se dividiría en distintas capas, tal y como vemos en la figura 3, cada capa es un conjunto de neuronas donde la entrada suele venir de una capa anterior y la salida será la entrada a otra capa posterior. Cada neurona contará con un valor numérico, denominado peso, con el que modificará la entrada recibida. Podemos identificar tres tipos de capas:

- **Capa de entrada**, es la primera capa que compone la red neuronal y recibe los datos reales, por lo tanto son las encargadas de asimilar los archivos que se les proporcione, ya sean imágenes, datos numéricos, etc.
- **Capa oculta**, puede haber una o varias capas de este tipo, son las que se ocupan del procesamiento de la información y realizan los cálculos oportunos. Cuantas más neuronas tiene esta capa más complejos pueden ser los cálculos que efectúan. Se les denomina capas ocultas porque no se sabe qué valores tienen de entrada ni de salida.

- **Capa de salida,** es el resultado visible de la red neuronal, es la información que obtiene la red como conclusión o la toma de decisión que realiza. Esta capa también puede tener uno o varios resultados, depende lo que se esté analizando y que se busque obtener.

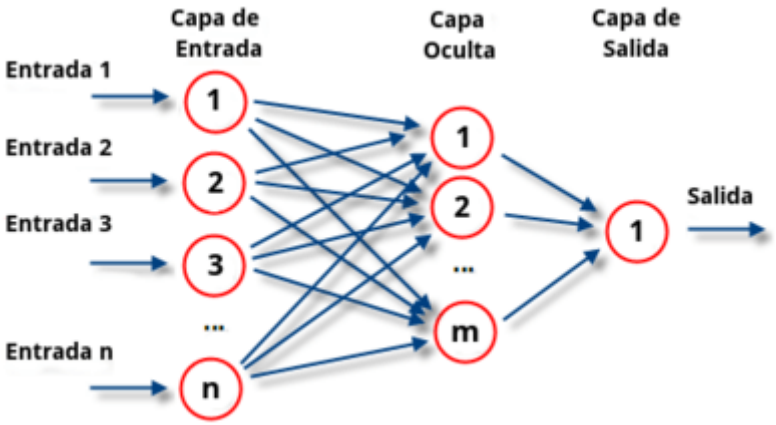


Figura 3 *Redes neuronales*

Fuente: [3]

Un aspecto importante de la red neuronal son los enlaces de las neuronas, es decir, la conexión entre una neurona y la siguiente. Los valores de salida de cada neurona también son multiplicados por un peso del enlace, el cual puede activar o desactivar a las neuronas consecutivas. Igual que también puede tener una función que se utilice como límite que no se debe de propasar antes de darle salida a la información hacia otra neurona, llamado función de activación. El objetivo de esta función viene a ser obtener relaciones no lineales, así hay muchos tipos de funciones que pueden ser utilizados como función de activación.

Para que la red neuronal funcione como nosotros pretendemos, lo primero que hay que hacer es entrenarla, esto se consigue mediante cambios en los valores de los pesos de cada neurona. Le brindaremos unos datos de entrada y según se consigan los resultados obtenidos, se modificarán los pesos para obtener menos errores, también habrá que tener en cuenta el grado de participación de cada neurona, este modelo se le denomina *"backpropagation"*. Con este modelo se parte desde la salida hacia el resto de capas hasta llegar a las de entrada modificando los pesos de una forma iterativa y recursiva.

2.1.3 Máquina de vectores de soporte (SVM)

Las máquinas de vectores de soporte, SVM por sus siglas en inglés, son un conjunto de algoritmos de aprendizaje supervisado relacionados con los problemas de clasificación y de regresión. Mediante las muestras, es decir, los datos de entrenamiento, el SVM genera un hiperplano de separación, pudiendo así clasificar en dos partes distintas, partiendo de un vector de soporte en cada uno de los lados, calculando el margen máximo, tal y como vemos en la figura 4.

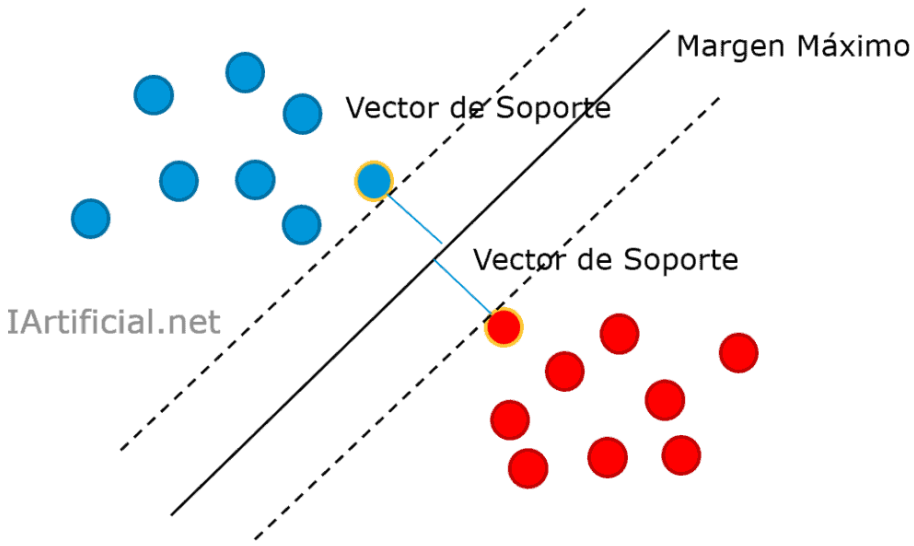


Figura 4 *Vector de soporte*

Fuente: [4]

En el ejemplo anterior se ve una separación lineal, no siempre es posible una separación perfecta, así que se intenta buscar un modelo generalizado. Para controlar la compensación de errores se maneja un parámetro C, con el que se tiene un margen de pequeños errores permitidos.

No siempre es posible una separación lineal, a veces para poder separarlos correctamente utilizaremos una nueva dimensión con lo que poder realizar la separación mediante una superficie de decisión. Para poder hacer una separación no lineal utilizaremos las funciones kernel, que nos permitirán la entrada de nuevas dimensiones.

Es importante destacar que las SVM lineales son mucho más veloces y eficientes, pero no son utilizables en todos los problemas, así que se podrán utilizar los SVM con diferentes funciones kernel.

2.2 Historia de la evolución tecnológica

El reconocimiento de gestos se basa en ser capaces de interpretar un gesto, por ejemplo, cuando una persona cierra el puño y extiende hacia arriba su dedo pulgar, depende el contexto cultural de esa persona, como por ejemplo en España, sabríamos que nos está queriendo trasmitir que algo está bien o correcto. Así podemos entender que las computadoras mediante el reconocimiento de gestos comprenden como se expresa un ser humano con su cuerpo. De esta forma un gesto de un ser humano para una máquina puede contener información.

El procesamiento de imágenes y su captura por visión suelen ser las técnicas más utilizadas para la detección de gestos, aunque también existen otras técnicas mediante hardware, como por ejemplo el uso de un mando interactivo o unos guantes.

En el mercado actual existen una gran variedad de productos con los que se interactúan mediante gestos, algunos utilizando dispositivos periféricos que tienen en cuenta los movimientos y otros con la utilización de la cámara para captarlos como en nuestro proyecto. A continuación se explicarán los más populares e interesantes:

Nintendo Wii

Nintendo fue la primera empresa que demostró al mundo la posibilidad de jugar con movimiento, mediante su producto estrella en el 2006, la consola Wii, la primera que mediante un mando inalámbrico podía detectar movimientos en un plano tridimensional.

Desde su debut en el mercado solo hacía que romper record de ventas, al principio fue diseñado más para niños y adolescentes, pero acabó siendo un producto que toda la familia podía usar, algo muy intuitivo.

El mecanismo de la consola es muy sencillo, simplemente tienes que colocar una barra de infrarrojos cerca del televisor con el que vas a jugar y sujetar un mando, bien es cierto que el mando tiene distintos botones que para algunos juegos es necesario utilizar o para comenzarlos al menos, luego con los gestos y movimientos del mando ya puedes interactuar con la consola, gracias a la detección de infrarrojos cuando se apunta hacia la barra de leds. En la figura 5 podemos ver la videoconsola, el mando y la barra de leds.

Desde 2019 ya no se fabrican más modelos de esta consola y fue un año más tarde cuando sacaron el último videojuego disponible en el mercado, si bien es cierto que marcó un antes y un después en la forma de jugar a los videojuegos. *[8]*

Figura 5 *Nintendo Wii*

Fuente: [8]

PlayStation Move

Tras el éxito de Wii, el resto de empresas de videoconsolas comenzaron con la investigación sobre la detección de gestos. De esta forma Sony sacó al mercado Move, un sistema de control de videojuegos mediante el movimiento. Captura los movimientos mediante un mando que el jugador lleva en su mano, llamado Motion Controller, y una cámara, PlayStation Eye, que se coloca enfrente del jugador cerca del televisor.

Motion Controller es el mando diseñado para poder jugar con Move, es necesario que se utilice este mando puesto que no es como otros mandos, tiene una forma alargada y en uno de los extremos tiene una esfera que se ilumina de diferentes colores a los que podemos encontrar en la habitación. En la figura 6 se puede ver la diferencia de los mandos que utiliza de normal Playstation y el Motion Controller.

PlayStation Eye es una cámara digital que posteriormente fue sustituida por PlayStation Camara. Son los mandos los que cuentan con sensores de movimiento, la cámara detecta el color del mando y su posición para así determinar la interacción con el videojuego. PlayStation Camara, es mucho más novedosa pero está muy ligada a la realidad virtual, pues se debe de conectar a los auriculares VR de PlayStation.

Volviendo a PlayStation Eye se puede destacar su buen funcionamiento incluso cuando la luz es casi nula, por ejemplo solo con la luz del televisor es capaz de detectar la imagen y todos sus movimientos. Según Sony, es capaz de reconocer rasgos faciales y determinar la orientación y posición de la cabeza del usuario. Además también dispone de un micrófono. *[7]*

Figura 6 *PlayStation Move*

Fuente: [7]

XBOX Kinect

Microsoft, con Xbox también investigó acerca del reconocimiento de gestos, y utilizó Kinect. Kinect fue uno de los controladores de juegos más vendidos en el momento, permite controlar el juego sin necesidad de contacto físico, a diferencia de los dos anteriores este fue el primero que no necesitaba ninguna referencia, como un mando, para poder reconocer los gestos.

Reconoce gestos, comandos de voz, objetos e imágenes, todo esto es capturado simplemente mediante la cámara. En la figura 7 observamos cómo es capaz de capturar un gesto de la mano. La cámara al principio estaba pensada para solo ser usada en Xbox y acabó saliendo al mercado para poder utilizarla en cualquier computador.

El instrumento principal es una cámara RGB y una segunda con un sensor CMOS de infrarrojos, el cual se utiliza para poder detectar los movimientos de las articulaciones. Un dato a destacar de la cámara es su buen funcionamiento a la hora

de detectar la profundidad de la imagen y que sin tener mucha luz es capaz de capturar los gestos. *[9]*

Figura 7 *Xbox Kinect*

Fuente: [9]

Manus (Guantes)

Con los avances en la realidad virtual, se avanzó en los estudios sobre la captación de gestos y fueron muchas las empresas que invirtieron en cómo capturar los gestos de las manos. Muchas de ellas pensaron que sería buena idea utilizar unos guantes que detectarán los gestos, evitando así la necesidad de utilizar una cámara.

Una de las empresas más populares en el mercado es HTC Vive que tiene a la venta varios guantes. El primero que creó fue Manus VR, se conectan mediante bluetooth y tiene 5 sensores en cada guante, es decir, uno por dedo de la mano. Esta fue la primera versión que sacaron, con demasiado cableado y teniendo que sujetarlo a la muñeca siendo esto un poco incómodo. Tenía una autonomía entre 6 y 8 horas y su funcionamiento se basaba en código abierto.

Posteriormente fueron mejorando los diseños y ahora disponen en su web de hasta cinco guantes distintos. Entre ellos Prime X, utilizado para interactuar directamente con tus manos en VR o para transmitir y grabar los movimientos en cualquier lugar, es compatible con todo el software estándar de VR y se pueden construir integraciones. Se destaca que es rápido y preciso, también a la hora de calibrar, con tres simples gestos estaría listo. Respecto a su autonomía, puede durar en pleno funcionamiento sobre unas 5 horas, aunque cuenta con batería extraíble e intercambiable. Además, un dato a destacar, es que con quitar el módulo electrónico se podrían lavar los guantes o reemplazarlos, algo muy útil. En la figura 8 se puede ver el diseño del guante Prime X. *[10]*

Figura 8 *Manus Prime-X*

Fuente: [10]

Leap Motion Controller

Leap Motion es un sensor realmente pequeño pero muy útil que detecta los gestos que se realizan con las manos en el aire sobre él, reconociendo las manos y una pequeña parte del antebrazo, se suele conectar mediante un USB y se coloca entre el teclado y nuestro cuerpo.

Se trata de un módulo óptico de seguimiento de manos que captura los movimientos que se realizan, con una gran precisión. Cuenta con dos cámaras y tres sensores infrarrojos que le permite hacer un seguimiento de todos los movimientos de las manos y los dedos, con una profundidad de entre 10 y 60cm.

Simplemente conectándolo con nuestro computador podremos utilizarlo como si se tratase de un ratón o un teclado sin necesidad de tocar nada, además se puede utilizar con cualquier aplicación existente y es capaz de detectar los movimientos de los diez dedos a la vez, dando así más posibilidades de gestos distintos. En la figura 9 se puede ver el dispositivo y como es capaz de detectar la posición de las manos. *[11]*

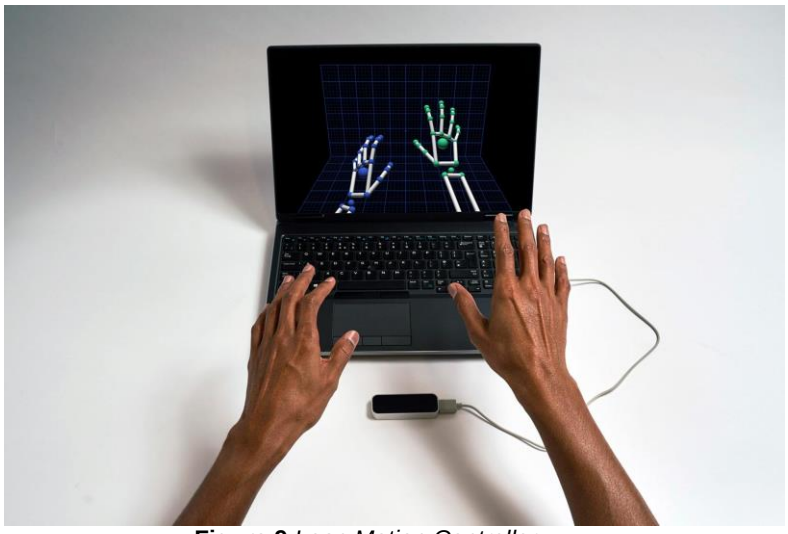


Figura 9 *Leap Motion Controller*

Fuente: [11]

"Smartphones"

Dentro del campo de los "smartphones" también se ha comenzado a implantar esta tecnología. Por ejemplo, la empresa tecnológica multinacional china, Huawei, es el segundo mayor fabricante de teléfonos móviles en el mundo, solo por detrás de Samsung, pese a no ser tan conocida como esta última desde hace unos años se sitúa en los puestos altos de las marcas de telefonía móvil.

Huawei siempre intenta estar al día en las novedades tecnológicas y es por esto que sacó al mercado el Huawei Mate 40 pro 5G, un "Smartphone" que incluye "AI Gesture Control", un controlador de gestos que te permiten subir el volumen, pausar un vídeo, contestar una llamada o deslizar arriba/abajo y a los lados. Es capaz de detectar los movimientos gracias a un sensor colocado en el "notch", el módulo situado en la parte de arriba de la pantalla delantera, el cual reconoce los gestos. *[12]*

Gestoos

Esta empresa vende tres aplicaciones diferentes, la primera se llama "activity monitoring", la cual mediante diferentes modelos es capaz de detectar las actividades humanas y su comportamiento, ayudando así a la seguridad de las personas o bajar el vandalismo entre otras. La segunda aplicación es "gesture device control", esta sería la más parecida a nuestra aplicación, en la cual se interactúa con un dispositivo mediante movimientos en el aire, así esta aplicación puede ser aplicada a diferentes pantallas digitales, robots o en el campo de la salud. Por último, nos encontramos con la aplicación "spatial awareness", la cual sirve para administrar activos, detectar anomalías y mejorar la competitividad mediante datos visuales.

Hay que destacar que disponen de una tecnología "air touch", que es parecida a la aplicación "gesture device control", pero en este caso se trata del dedo índice como un puntero que a la hora de aproximarlo a la pantalla reaccionará como si fuera un toque en ella, esto se consigue gracias a un sensor de profundidad.

Como hemos observado Gestoos lleva al mercado una aplicación muy similar a lo que queremos crear en este trabajo fin de grado. Además disponen de una demo de la

aplicación “activity monitoring” que se puede solicitar para poder probarlo en tu dispositivo. *[13]*

De todos estos ejemplos, en los que se muestra la aplicación de la tecnología de detección de gestos en diferentes dispositivos y sus distintas utilidades, es importante destacar las diferentes formas que hay para poder obtener los gestos que realiza el usuario utilizando simplemente una cámara o con necesidad de otro elemento hardware. Los primeros que utilizaron la tecnología de gestos lo empezaron a hacer con un dispositivo externo, posteriormente gracias a los distintos avances en las técnicas de visión por computador que han incorporado técnicas de “Deep Learning”, se pudo avanzar a utilizar únicamente una cámara o incluso usar un hardware más complejo. Estas técnicas de “Deep Learning” que se han incorporado han sido gracias a la investigación en el campo de la Inteligencia artificial, abordando su subconjunto de “Machine Learning” para acabar en el subconjunto de “Deep Learning”.

Por último, hay que destacar que nuestra aplicación vendida como producto no sería única en el mercado, uno de los productos que encontramos en el mercado que más se asemejaría a nuestra aplicación es el proporcionado por la empresa **Gestoos**.

3. Identificación y análisis de soluciones posibles

Como hemos visto en el punto anterior, hay diferentes formas para poder detectar gestos, podemos utilizar una cámara o un sensor de infrarrojos junto con un elemento hardware o unos guantes. Existen muchas formas diferentes de plantear nuestra aplicación por ello es importante determinar las bases desde las que comenzaremos a buscar soluciones.

3.1 Soluciones existentes

Tenemos dos posibles soluciones, por una parte podemos crear nuestra propia aplicación desde cero, lo que supondría tener que crear el modelo, entrenarlo y mejorarlo para por último poder implementarlo. Por otra parte podemos elegir un software de código abierto (OSS por sus siglas en inglés, Open Source Software) y ver si se puede mejorar e implementarlo.

El software de código abierto es donde el código fuente y sus derechos que suelen ser exclusivos ahora son publicados de forma que pertenecen al dominio público. Este tipo de código se suelen desarrollar de manera colaborativa y se encuentran en internet, una vez obtenidos estos códigos pueden ser utilizados para cualquier propósito de la misma forma que se encuentra o con modificaciones.

Antes de tomar una decisión se ha hecho una investigación acerca de los diferentes códigos abiertos que existen en la red, viendo la cantidad de gente que ha desarrollado su código de detección de gestos y lo ha publicado con buenos resultados, hemos decidido que lo mejor sería obtener distintos modelos y poder ver las diferencias y a partir de uno de estos comenzar a desarrollar nuestro proyecto. Además si no utilizáramos código abierto, para crear nuestro modelo y poder entrenarlo necesitaríamos tener acceso a un buen conjunto de datos y de calidad, además de etiquetarlo y transformarlos, algo que nos llevaría mucho tiempo y que muchas veces se convierte en un cuello de botella para la creación de un buen proyecto. Un modelo de código abierto ya está preentrenado, con lo que los datos ya tienen los pesos correspondientes, acelera el desarrollo y permite ejecutar más iteraciones para el perfeccionamiento del modelo final.

Tras investigar el software de código abierto que hemos encontrado en la red, nos hemos quedado con las siguientes opciones:

1) **GESTIA**

   GestIA es una aplicación basada en Deep learning, concretamente se trata de una red neuronal convolucional.

   Para poder entrenar a GestIA se seleccionaron 6 gestos que puede hacer la gran mayoría de la población con la mano, siendo estos puño, palma cerrada, palma abierta, pulgares hacia arriba y hacia abajo y dedo índice hacia arriba. Utilizaron un script de Python para capturar los fotogramas con diferentes iluminaciones y fondos, intentando poner la mano con los gestos de forma ideal y no ideal, para entrenarlo mejor, en total consiguieron más de 4000 imágenes. Estas imágenes fueron

etiquetadas para ser más fácil de detectar, además se distribuyeron el 90% para entrenamiento y el 10% formaron parte del test.

La idea principal era que primero fuera capaz de detectar si había una mano en la imagen y luego detectar el gesto que estaba haciendo entre los seis gestos procesados.

Ya que lo primero era si había mano o no en la imagen, se comenzó con la detección de objetos y luego la de gestos. Para la detección de objetos partían de dos modelos posibles Faster RCNN y SDD Mobilenet, el primero escanea una imagen y extrae sobre unas 2000 regiones de esta, en cada una de las regiones implementa una red neuronal convolucional y por último clasifica las regiones con una SVM. Por otra parte, SDD Mobilenet pasa la imagen directamente a través de distintas capas convolucionales proporcionando en ellas cuadros delimitadores, lo que hace que no sean necesarias las regiones. Es por esto último que se decantaron por el modelo de SDD Mobilenet, ya que es significativamente más rápido a la hora de detectar objetos.

Si directamente hubieran partido desde cero habrían necesitado muchísimas más imágenes, por lo que partieron de un modelo entrenado en detección de objetos llamado COCO, haciendo así que el proceso de aprendizaje pasase de una detección de objetos a gestos, llamado aprendizaje de transferencia, ya que se transfiere lo que se aprende, es decir, transfiere el detector de objetos a gestos. Para entrenar el modelo final tardaron unas 14 horas y tuvieron que instalar la API de Tensorflow en Amazon Web Service y cargar en la nube el modelo de SDD Mobilenet.

Una vez entrenado el modelo se necesitan hacer las pruebas de inferencia, en este caso se utilizó el kit de herramientas de software de Intel, **OpenVINO.** Este kit de herramientas es gratuito y sirve para optimizar los modelos de Deep Learning y la implementación y ejecución del modelo gracias a un motor de inferencia de Intel. Gracias a la herramienta Model Optimizer, mediante líneas de código se generaron los archivos a través del framework de TensorFlow para que el modelo fuera inferido en OpenVINO. Este paso ayudó a bajar la latencia del modelo y a poder utilizar la aplicación a tiempo real.

Por último, esto acaba siendo una aplicación para escritorio que deja la posibilidad a que tú indiques cada gesto a que entrada de teclado se refiere, tal y como podemos observar en la figura 10. Además acaba guardando la última referencia de entrada al teclado en local.

Figura 10 *Inicio de aplicación GestIA*

Fuente: [15]

2) DLIB

DLIB es una librería muy conocida por ser de las mejores cuando se habla de detección, ya sea objetos, caras, manos, etc. Para poder detectar los objetos DLIB contiene se basa en Histograma de Gradientes Orientados (HOG) y Máquina de Soporte de Vectores(SVM). En la figura 11 observamos cómo funciona el procesado de una imagen en SVM.

Los HOG son un tipo de descriptor de características, se tratan de vectores, es decir, una matriz de números, que codifican información útil sobre el contenido de una imagen. Estos descriptores de características son de los más potentes hoy en día, con estos vectores al añadirlos a un modelo de aprendizaje automático, en nuestro caso SVM, se podrá ver una predicción y ver las similitudes entre varias imágenes.

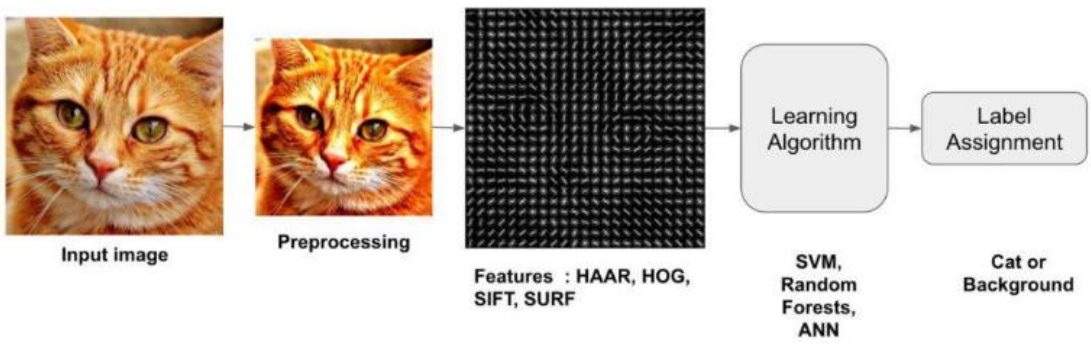


Figura 11 *Procesador de imagen en SVM*

Fuente: [18]

Simplemente con HOG y SVM se tendría un clasificador de imágenes de ML, pero lo que buscamos es un detector, el cual se realizará mediante una ventana deslizante que recorrerá la imagen de izquierda a derecha, para poner así un cuadro delimitador que diga que gesto se detecta, además irá acompañado de una pirámide de imagen, para que así la ventana deslizante detecte el objeto en cualquier tamaño.

En este caso hemos encontrado la librería y sabemos cómo hacerlo funcionar y el código, el ejemplo realizado es solo la detección de la mano, no de gestos por lo que habría que entrenarlo con nuestros gestos para poder comparar con el resto de opciones, algo que nos llevará un poco más de tiempo.

3) OPENCV

En esta opción, mediante la librería OpenCV, se trata de buscar los puntos claves de la mano, es decir las articulaciones de los dedos y sus puntas, estaríamos hablando de algo parecido a la detección de puntos de referencia faciales o a la postura del cuerpo humano.

Para poder desarrollar esta tecnología utilizaron unas imágenes de manos etiquetadas y una red neuronal, similar a las utilizadas para detectar la postura del cuerpo humano para así obtener los puntos claves de las manos. Consiguieron obtener imágenes en 3D gracias a 31 cámaras HD puestas en distintos ángulos, así la percepción de los puntos claves sería más adecuada y más precisa al pasarlo a una imagen en 2D. Esto último será importante para poder obtener los puntos claves en imágenes que son difíciles de predecir.

El modelo produce 22 puntos claves, 21 de ellos en la mano y el punto 22 refiriéndose al fondo de la imagen, tal como se ve en la figura 12.

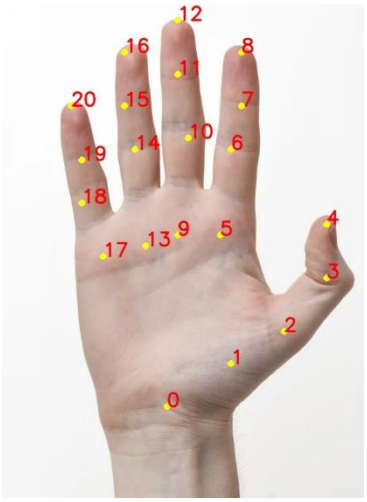


Figura 12 *Puntos clave OpenCV*

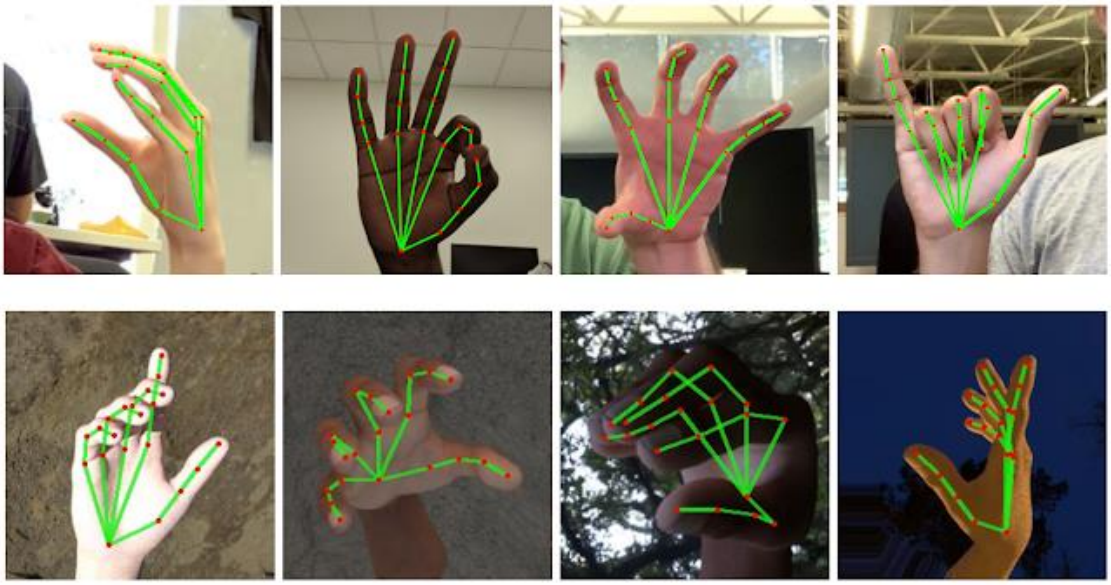
Fuente:[16]

Gracias a estos puntos clave y la ayuda de ser vistas las imágenes desde distintos ángulos, se consigue que pasándole unos datos de imágenes etiquetados se consiga tener un modelo de detección bastante mejorado.

Por último, vale la pena destacar que OpenCV es una librería de Intel y es una de las más populares dentro de la visión artificial. Además se encuentra dentro del kit de herramientas de OpenVINO, del cual hemos hablado recientemente.

4) MEDIAPIPE

MediaPipe es una multiplataforma de Google de código abierto para crear canalizaciones y así procesar los datos, ya sean de un origen de vídeo o de audio. Realiza un seguimiento de las manos y los dedos, mediante machine learning es capaz de detectar 21 puntos claves en una mano en imagen 3D, dibujando los puntos y sus conexiones como se muestra en la figura 13. Con este método se puede detectar los puntos en tiempo real.



Fuente 13 *Ejemplo de detección de manos de mediapipe*

Fuente: [19]

Para poder detectar los 21 puntos clave, la canalización se basa en varios modelos unidos que funcionan conjuntamente. El primer modelo es un detector de la palma de la mano, BlazePalm, que después de examinar toda la imagen obtiene un cuadro delimitador de la mano. El segundo modelo recoge el cuadro delimitador y establece los distintos puntos clave de la mano. Por último, se tiene un reconocedor de gestos que clasifica por gestos, teniendo en cuenta los puntos obtenidos en la imagen respecto a los gestos etiquetados que se tienen previamente.

Esta canalización resulta bastante eficiente sobre todo gracias a la detección de la palma, es algo que reduce mucho los datos, ya no son necesarias las rotaciones,

la escala o la traslación, así la mayor capacidad se dedica la precisión de las coordenadas de los puntos clave.

En el caso de esta solución, para poder detectar los puntos clave utilizan una predicción directa de coordenadas. Para poder enseñar al modelo se utilizan más o menos 30000 imágenes etiquetadas manualmente, entre ellas se incluyen manos sintéticas sobre diferentes fondos. Las manos sintéticas aumentaron el error esperado, pero una vez mezcladas estas y las del mundo real se consigue un resultado respecto a errores mucho mejor.

3.2 Materiales

Para la realización de este proyecto una de las cuestiones que debemos abordar es el elemento hardware que utilizaremos para detectar los gestos. Para ello hay que tener en cuenta varios factores, uno de ellos el presupuesto, que en nuestro caso al tratarse de un trabajo fin de grado, es nulo. Además del presupuesto es importante saber que el objetivo de este proyecto es poder ser utilizado por cualquier usuario en el entorno que desee, es por estas razones que utilizaremos la cámara del ordenador para poder determinar los gestos.

3.1.1 Equipamiento hardware

Al utilizar la cámara de un ordenador, deberemos de tener en cuenta que no todas las cámaras de los usuarios tendrán la mejor resolución ni calidad, por lo que intentaremos hacer un modelo que detecte correctamente los gestos en la mayoría de situaciones posibles.

Así pues, para la realización de todas las actividades de este proyecto, es decir, pruebas de detección de gestos e implementación del modelo, se realizarán en el portátil *LENOVO Ideapad 300-15SK de 15,6"* con un *procesador Intel i7-6500U CPU de cuádruple núcleo a 2.50GHz, 12GB de memoria RAM y tarjeta gráfica integrada Intel HD Graphics 520*. Dispone de una cámara *Lenovo EasyCamara* que cuenta con *0.92MP, resolución de 1280x720 16fps y vídeo en HD.*

Pese a que la realización del proyecto se origine desde el portátil comentado, posteriormente se utilizarán distintos computadores para analizar la aplicación final obtenida otras cámaras, así obtendremos una mejor visión global a la hora de sacar conclusiones del trabajo realizado.

3.1.2 Equipamiento software
Lenguaje de programación

El desarrollo del presente trabajo se ha llevado a cabo completamente con el lenguaje de programación Python, un lenguaje a alto nivel que facilita el trabajar con

inteligencia artificial, big data y machine learning, entre muchos otros campos. Elegimos Python, versión 3.6.5, principalmente por tres motivos:

- **Aprendizaje**: Python es un lenguaje de programación sencillo de aprender y que en pocas horas se puede comenzar a programar con ligera soltura. Además se trata de un lenguaje con muchos recursos que facilitan el proceso.
- **Dominante en IA**: Actualmente Python es un lenguaje que domina en el campo de la inteligencia artificial, gracias sobre todo a la gran cantidad de librerías que posee, entre otras cosas. La instalación y utilidad de las librerías es sencillo y rápido.
- **Operatividad**: Python resulta ser un lenguaje de programación que se puede utilizar en cualquier computador, independientemente del sistema operativo que tenga, así nos ayuda con la universalidad de la aplicación.

El entorno de desarrollo de la aplicación será Visual Studio Code, un IDE (Interactive Development Environment) desarrollado por Microsoft de código abierto y gratuito que soporta múltiples lenguajes de programación, incluido Python entre ellos. Además es personalizable, pudiendo cambiar los atajos del teclado, el tema del editor y las preferencias. Es importante destacar que incluye depuración de código y control integrado de Git, que nos ayudará a desarrollar las distintas versiones y ver nuestros avances.

Librerías empleadas

Como se ha comentado anteriormente, Python tiene la ventaja de disponer de una gran cantidad de librerías. Para este proyecto han sido empleadas algunas de ellas en distintos códigos, a continuación se enumeran y se realiza una breve explicación de las más importantes:

- **Keyboard**: Esta pequeña librería permite trabajar con Python todos los eventos que puedan tener lugar en el teclado, desde registrar teclas de acceso rápido hasta la simulación de pulsar una tecla. Además de enviar estos eventos también sirve para escucharlos y saber que tecla ha tocado el usuario. Además es compatible con cualquier teclado internacional. *[25]*

- **Json**: Esta librería simplemente se ha utilizado para trabajar en Python con archivos de tipo Json. La hemos utilizado para poder intercambiar datos ligeros. *[26]*

- **Numpy**: La librería Numpy es una de las grandes librerías que tiene Python, ofrece diversas funciones matemáticas como integrales, álgebra lineal o generación de números aleatorios. Es muy rápido y potente con las matrices. S sintaxis de alto nivel lo hace accesible para todo tipo de programadores y fácil de utilizar. *[27]*

- **Pyautogui**: Esta librería es muy parecida pero un poco más completa que la primera que hemos comentado pues permite controlar los eventos de ratón y teclado. Esta librería es más simple que Pyutogui, deja simular la pulsación de

una tecla, pero aumenta su funcionalidad ya que permite el movimiento de ratón y hacer clic. Además también se puede utilizar como búsqueda de imágenes. *[28]*

- **Dlib**: Como ya se ha comentado anteriormente, Python destaca como lenguaje para machine learning, de forma principal por las librerías que existen para ello. Una de ellas es Dlib que contiene algoritmos de machine learning y se utiliza en diferentes dominios como la robótica, teléfonos y grandes dispositivos informáticos, entre otros. Dentro de los algoritmos de machine learning destaca los modelos de clasificación y regresión, las máquinas de vectores de soporte y algunos algoritmos de clustering como k-means. *[29]*

- **OpenCV**: Esta librería es aún más conocida si cabe que la anterior. OpenCV se basa en inteligencia artificial, machine learning y reconocimiento de caras, manos, es decir, reconocimiento de imágenes. La mayoría de sus usos viene a ser este último, así es capaz de procesar una imagen y reconocerla o clasificarla según las características de sus pixeles. *[30]*

- **Mediapipe**: La librería Mediapipe todavía no es tan conocida y hay poca información en la web ya que hace poco que se creó. Se trata de una librería creada por google, que ofrece soluciones de machine learning para ser utilizadas a tiempo real. *[20]*

Muchas de estas librerías explicadas se utilizan durante el proyecto, ya sea en la comparación de las APIs o a la hora de crear nuestra aplicación final. Todas ellas son gratuitas al tratarse de software de código abierto, además disponen de una gran estabilidad y son simples y fáciles de usar.

3.3 Pruebas de las APIs

Una vez detectados los posibles softwares de código abierto que encontramos en la red, procedemos a descargarnos e instalar todos en nuestro entorno específico para realizar unas pruebas y ver cual podríamos mejorar para poder implementarlo en nuestro proyecto.

Las pruebas que llevamos a cabo se dividen en cuatro grandes bloques, los cuales se realizarán en las diferentes APIs:

BLOQUE 1

- Instalación de cada una de las opciones.
- Comprobar que se ejecutan correctamente en nuestro entorno.

BLOQUE 2

- Cerciorarse que se detecta una o ambas manos en unas imágenes estáticas y vídeo previamente grabado.
- Determinar el porcentaje de imágenes estáticas con un gesto concreto que se detecta correctamente.
- Cuantificar tiempo que tarda en detectar los gestos en las imágenes estáticas.

- Corroborar que se detecte los diferentes gestos en un vídeo previamente grabado.

BLOQUE 3

- Asegurar que se ejecutan las diferentes APIs en vídeo a tiempo real.
- Cerciorarse que se detectan los gestos en vídeo a tiempo real.
- Cuantificar tiempo que tarda en detectar los gestos en vídeo a tiempo real.
- Medir el consumo de recursos del computador.

BLOQUE 4

- Realizar distintos gestos en vídeo a tiempo real con una imagen no optima, es decir, con diferente profundidad, rapidez de realizar el gesto, hacerlo con poca iluminación o mucha, etc.

En las pruebas del primer bloque si se da el caso de que alguna de nuestras opciones no consiga pasarla será eliminada y no continuará con el resto de pruebas.

Sin embargo las pruebas de los tres bloques restantes se realizarán para todas las APIs capaces de superar la primera, para a la hora de elegir la mejor opción como base de nuestra aplicación se tenga en cuenta todos los resultados obtenidos.

Las pruebas del segundo bloque se llevan a cabo en las mejores condiciones posibles, unas condiciones óptimas donde en la imagen lo único que se observa es el brazo y la mano en un entorno con buena iluminación y a una distancia coherente de la cámara.

BLOQUE 1

En este primer bloque nos hemos encontrado con algunas dificultades, pues cada una de las APIs nos han llevado a diferentes problemas a la hora de descargarlas y poder ejecutarlas.

En el caso de **GestIA** tuvimos que instalar la versión de openvino 2020.1, la cual requiere que instales otros programas en nuestro caso instalamos VisualStudio 2017, CMake 3.19.4 y Python 3.6.5 la versión de 64 bits. Al principio probamos con una versión de Python más actualizada pero hasta esta versión no lo reconocía, es por esto que decidimos continuar con esta versión para llevar a cabo el proyecto. Una vez instalados todos los requerimientos que nos solicita, ejecutamos openvino estableciendo las variables de entorno y configuramos el modelo optimizador correctamente.

Una vez tenemos todo instalado tuvimos problemas a la hora de ejecutarlo, que se solucionaron borrando del archivo requeriments.txt la línea *"pkg-resources==0.0.0"*. Después de este ajuste para ejecutarlo hay que iniciar openvino en la misma terminal a la que se lanza la API.

Por lo que podemos concluir que **GestIA** se instala y se ejecuta correctamente, pasando así las pruebas del primer bloque.

DLIB fue más sencillo de instalar y dio menos problemas que el anterior, una vez descargado los códigos tiene distintos test que se pueden pasar para comprobar que se

ejecuta correctamente. Uno de los requerimientos de DLIB fue instalar las librerías de matplotlib y pyautogui. Además el código estaba escrito en jupyter, por lo que tuvimos que descargar la extensión de jupyter en nuestro VSCode y con ella su extensión en Python.

Es importante señalar que en el caso de GestIA una vez lanzabas el código ya podías utilizarla, sin embargo con Dlib se tuvo que ir ejecutando celda por celda del archivo jupyter. Además, en estas celdas se creaba la máquina de vectores de soporte mediante una ventana deslizante, siendo recomendable realizarlo dos veces para tener más imágenes que entrenase el modelo. Pese a esto fue sencillo de instalar pero se requirió más tiempo al entrenar el modelo.

También **DLIB** consiguió pasar las pruebas del primer bloque.

En el caso de **OpenCV** fue muy sencillo de instalar, simplemente hubo que descargar los códigos no tuvo ningún requerimiento, más allá de instalar la librería de OpenCV y fue realmente sencillo al no encontrar ningún problema, así pues fue el más rápido en pasar las pruebas de este primer bloque.

Por último tenemos **MediaPipe**, el cual también fue bastante sencillo, una vez descargado los códigos e importada la librería se pudo probar y comprobar que funcionaba correctamente, pasando así todas las pruebas de este primer bloque. Es importante destacar un hecho de esta y es que no se debe de poner el nombre “Mediapipe” a ningún archivo de la misma carpeta en la que está el código Python a ejecutar, pues sino da error ya que busca este archivo.

En conclusión, todas las opciones que tenemos en cuenta han pasado las pruebas de este primer bloque.

BLOQUE 2

En este bloque de pruebas es importante que todas pasen ese primer punto, es decir, que todas detecten al menos una mano. Las pruebas que se realizarán en este apartado serán todas con las mismas imágenes estáticas y el mismo vídeo, todo ello realizado con una buena iluminación y una distancia adecuada de la cámara, intentaremos que se realicen con las condiciones más favorables. Así podremos medir tanto la bondad de la librería a la hora de detectar el gesto como cuánto tarda al tratarse de una simple imagen y en buenas condiciones. Además como todas tendrán exactamente la misma imagen de entrada se podrá comparar con mayor exactitud.

Respecto a los gestos que se realizarán en las imágenes, se ha hecho una investigación alrededor de los gestos más comunes y habituales, aquellos que incluso una persona con algún problema de movilidad sea capaz de realizar. También hay que tener en cuenta que de estos se han elegido los gestos que menos confusión creen entre sí, es decir, que sean bastante distintos. Estos gestos serán los siguientes:

- Palma de la mano (5 con la mano)

Figura 14 *Gesto palma de la mano*

Fuente: [Elaboración propia]

- Dedo índice hacia arriba

Figura 15 *Gesto dedo índice hacia arriba*

Fuente: [Elaboración propia]

- Dedo pulgar hacia arriba (gesto de OK)

Figura 16 *Gesto de ok*

Fuente: [Elaboración propia]

- Puño cerrado

Figura 17 *Gesto del puño cerrado*

Fuente: [Elaboración propia]

- Dedo índice y corazón hacia arriba (gesto de la paz)

Figura 18 *Gesto de la paz*

Fuente: [Elaboración propia]

Todas las imágenes y vídeo que se han utilizado en este apartado también se encuentran en el enlace que se encuentra en este documento en el punto 1.3.

Es importante destacar que además de utilizar las mismas imágenes, también tendremos el menor número de programas abierto, así cada una de las librerías se ejecutarán en las mismas condiciones.

Para poder analizar los resultados conforme realizábamos las pruebas íbamos guardando en un libro de Excel los distintos tiempos en los que se conseguía detectar la mano y el gesto y cuántos de ellos estaba bien detectado y cuántos no.

En primer lugar, todas las librerías, excepto **DLIB**, han sido capaces de detectar al menos una mano. DLIB nos ha dado problemas en este bloque, sí que somos capaces de pasarle la imagen y el vídeo pero en ningún caso ha sido capaz de detectar correctamente dónde se encuentra la mano, por lo que ha sido imposible utilizar esta librería en este segundo bloque de comparación.

Al descartar una de las opciones para las pruebas de este bloque solo nos quedaban tres librerías a comparar. Las primeras imágenes que le hemos pasado a cada una de las librerías han sido las del gesto de la palma de la mano, es decir, realizar un cinco.

Como podemos observar en la parte izquierda de la figura 19 tanto OpenCV como Mediapipe obtienen un acierto del 80% habiendo solo fallado en dos imágenes a detectar, además se trata de las mismas dos imágenes en ambos casos por lo que puede que se trate de una imagen que sea más difícil de detectar. El caso de GestIA solo llega a un 20% de acierto siendo este un registro bastante bajo. En la parte derecha de la figura vemos en un gráfico de líneas la diferencia de cuántos segundos tardan en detectar el gesto cada librería, como vemos OpenCV es la que mayor tiempo tarda con diferencia. Sin embargo GestIA y Mediapipe son más similares en cuanto a tiempos, pero Mediapipe destaca mucho más por sus aciertos.

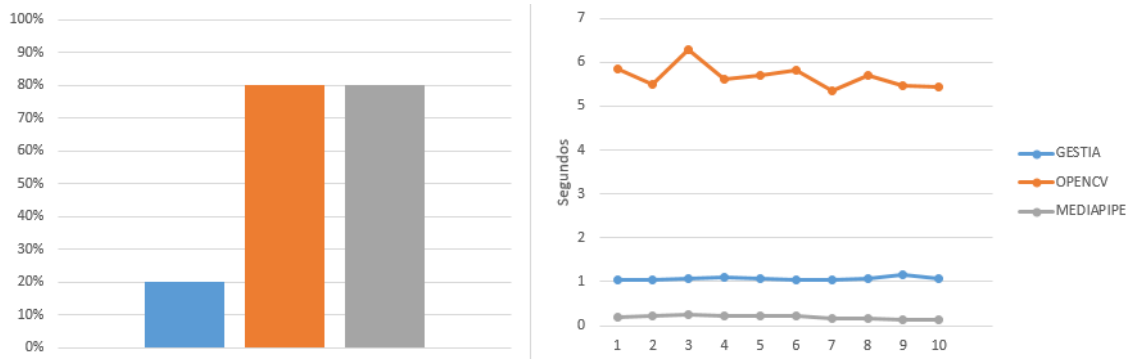


Figura 19 *Resultados detección palma en imagen estática*

Fuente: [Elaboración propia]

El siguiente gesto que se quiere comprobar es el del dedo índice hacia arriba, como hemos hecho anteriormente se ha pasado todas las imágenes a cada una de las librerías. En la figura 20, vemos los resultados obtenidos con las librerías GestIA y Mediapipe 100% de acierto, mientras que OpenCV un 80% de aciertos. En este caso, igual que en el anterior, los segundos que tardan en detectar el gesto OpenCV vuelve a tardar muchos segundos en cada imagen. Mediapipe sigue tardando muy poco para detectar el gesto, no se llega a acercar ni al segundo.

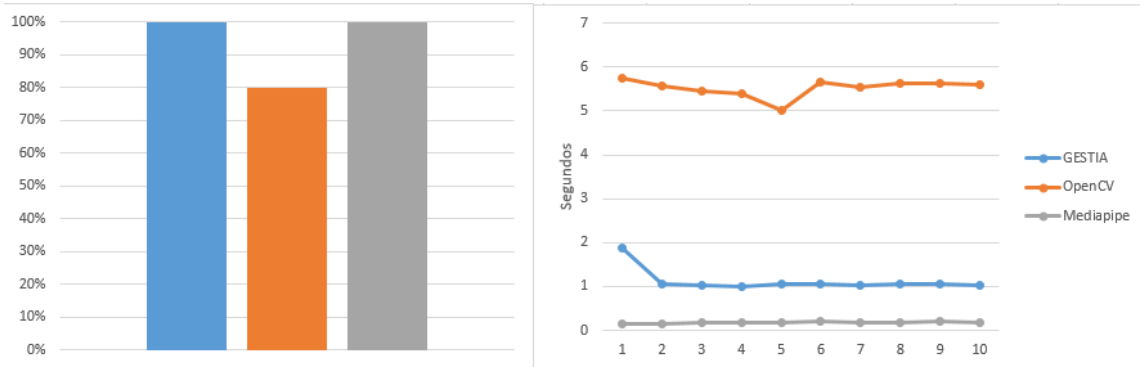


Figura 20 *Resultados detección índice en imagen estática*

Fuente: [Elaboración propia]

Las siguientes imágenes que comparamos son las del dedo pulgar hacia arriba, el conocido gesto como “Ok!”. En este caso como vemos en la figura 21 los aciertos son muy pocos, en este caso la librería con mayor porcentaje de aciertos es OpenCV y solo cuenta con un 30%, le sigue Mediapipe con 20% y GestIA en este caso no detecta el gesto en ninguna imagen. Tener tan poco acierto en estas imágenes se debe principalmente a la dificultad de detectarlo de la forma en la que tenemos el código en este momento. Como en los anteriores respecto al tiempo que tardan en detectar el gesto, OpenCV destaca por ser el que más tarda y Mediapipe el que menos. Pese a que GestIA no haya detectado un gesto se contabilizan los segundos en los cuales busca un gesto aunque la respuesta que nos da no es la esperada, es decir, el no detectar el gesto.

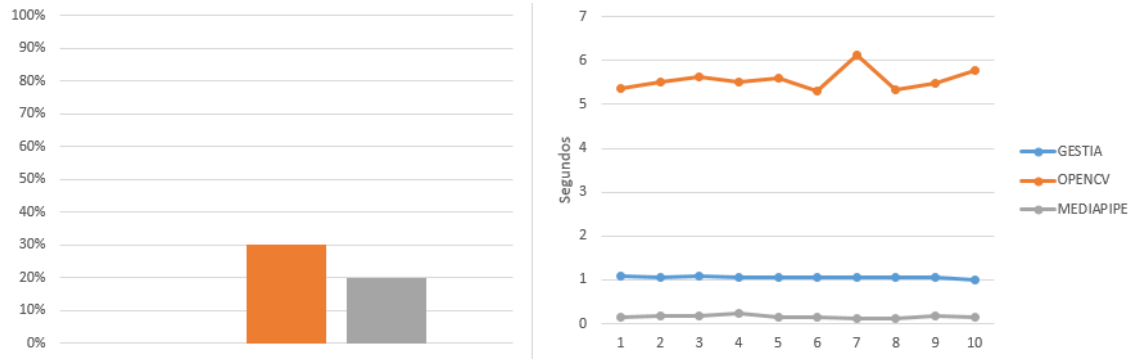


Figura 21 *Resultados detección ok en imagen estática*

Fuente: [Elaboración propia]

El siguiente gesto que comparamos es el puño, en este caso pasa algo parecido con GestIA que en el anterior, no es capaz de detectar el puño. En la figura 22 observamos que la detección del gesto de OpenCV y Mediapipe es más alto que el anterior, el primero llega a un 90% de acierto y el segundo llega a un 70% de acierto. Respecto al tiempo volvemos a lo mismo, Mediapipe es mucho más rápido en detectar el gesto y con mucha diferencia.

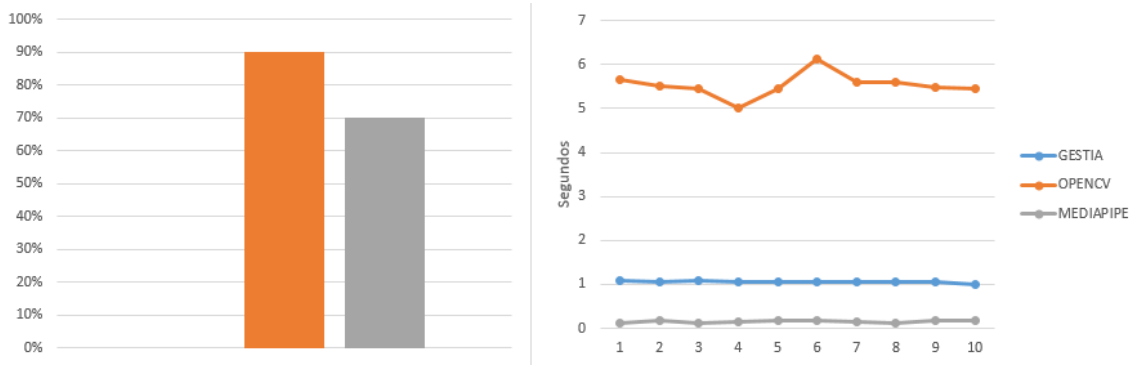


Figura 22 *Resultados detección puño en imagen estática*

Fuente: [Elaboración propia]

Por último tenemos el gesto de solo el dedo índice y anular levantado, es decir, el conocido gesto de la paz. En la figura 23 vemos como con este gesto es la librería Mediapipe la que consigue un mayor porcentaje de aciertos, un 90%, le sigue OpenCV con un 80% de aciertos y por último GestIA con un porcentaje mucho más bajo de un 40%. Respecto al tiempo es igual que anteriormente, OpenCV tarda demasiado respecto a los otros dos.

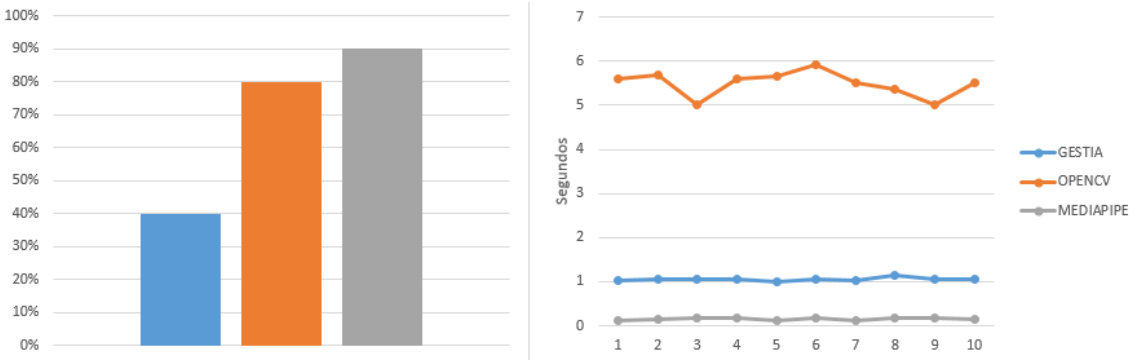


Figura 23 *Resultados detección gesto de paz en imagen estática*

Fuente: [Elaboración propia]

Como hemos podido observar no todas las librerías consiguen detectar todos los gestos cuando les pasamos imágenes estáticas, la librería DLIB no la hemos podido probar. Las demás en imágenes estáticas GestIA se queda un poco por detrás a la hora de detectar de OpenCV y MediaPipe. Mientras que si tenemos en cuenta el tiempo que tardan en detectar el que se quedaría por detrás sería OpenCV.

La última prueba a la que sometemos estas librerías en este segundo bloque es pasarle un vídeo con todos los gestos que hemos utilizado en las anteriores pruebas de las imágenes estáticas. Como vemos en la tabla 1, Mediapipe se comporta perfectamente, reconociendo todos los gestos en el vídeo, mientras que OpenCV y GestIA fallan en un gesto cada uno. Además OpenCV para poder pasar el vídeo y que detectará los gestos le ha costado más de 20 minutos mientras que Mediapipe y GestIA no han tardado nada.

	INDICE	PALMA	OK	PUÑO	PAZ
MEDIAPIPE	Sí	Sí	Sí	Sí	Sí
OPENCV	Sí	Sí	Sí	Sí	No
GESTIA	Sí	Sí	No	Sí	Sí

Tabla 1 *Resultados detección de gestos en el vídeo*

Fuente: [Elaboración propia]

Finalmente termina este segundo bloques de pruebas a los que hemos enfrentado las APIs, para poder hacerlo en la gran mayoría hemos tenido que cambiar el código que teníamos de un principio para conseguir que no solo detectará las manos sino también los gestos. Por lo que para realizar las pruebas de este bloque y los siguientes hemos tenido que investigar acerca de cada librería y realizar los cambios necesarios en el código.

Seguramente de este segundo bloque de pruebas las librerías que más destacan son OpenCV y Mediapipe, esta última destaca un poco más por la rapidez de detección.

BLOQUE 3

En este bloque se continúa con pruebas similares al anterior pero en este caso se realiza con un vídeo en tiempo real, que es como queremos que funcione nuestra aplicación final. Estas pruebas se realizarán en la misma ubicación y con una iluminación lo más parecida posible a las imágenes utilizadas en el bloque de pruebas anterior. La diferencia más clara que tendremos respecto a las imágenes anteriores es que esta vez aparecerá una cara en la imagen, pues estás pruebas serán como si se tratase de un usuario que lo utiliza con el mejor entorno posible. En todas las APIs que se sometan a estas pruebas se intentará realizar de la forma más similar posible.

Respecto a la medición de los recursos utilizados por el computador hay que tener claro que una vez se vayan a ejecutar estas pruebas el único programa que se estará ejecutando será el de la API, teniendo todos los recursos disponibles para el consumo que se necesite por cada solución. Se monitorizarán los recursos de CPU y memoria.

En este apartado de pruebas sí que será posible realizar las pruebas a la librería DLIB, pues parece que sí que es capaz de detectar los gestos cuando se trata de un vídeo a tiempo real.

Del mismo modo que hicimos con las imágenes estáticas realizaremos el mismo gesto en tiempo real diez veces en cada librería, así tendremos en cuenta el porcentaje de aciertos y el tiempo que tarda en vídeo a hacer el reconocimiento.

El primer lugar hay que recalcar que todas las librerías se han podido utilizar en un vídeo a tiempo real, entendiéndose por tiempo real a que es capaz de coger al momento las imágenes que obtienen de la cámara del computador. Bien es cierto que aunque coja las imágenes al momento puede que tarde varios segundos en ejecutarse, es por esto que compraremos los tiempos de las distintas opciones.

Al evaluar el gesto de la palma abierta obtenemos los gráficos de la figura 24, dónde encontramos a Mediapipe con un cien por cien de acierto y además la más rápida. Destaca que tanto Mediapipe como GestIA superan el porcentaje del gesto reconocido a tiempo real que pasándole imágenes estáticas. OpenCV sigue en el mismo rango tanto de aciertos como los segundos que tarda en detectar el gesto, que al tratarse de un vídeo a tiempo real se nota mucho más y hace que no sea óptimo. Por último, DLIB obtiene unos buenos resultados siendo de los más rápido en detectar y con un 70% de detecciones correctas.

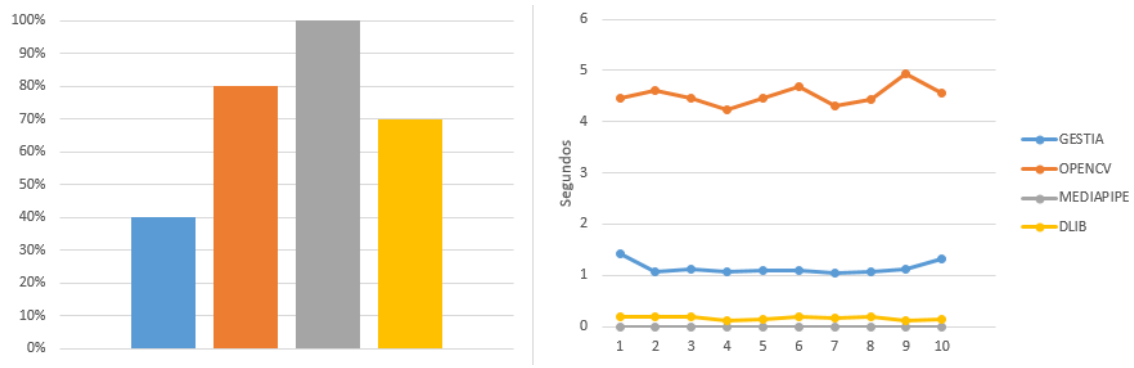


Figura 24 *Resultados detección palma en vídeo a tiempo real*

Fuente: [Elaboración propia]

Respecto a detectar el gesto del dedo índice obtenemos los resultados de la figura 25, dónde también tenemos la librería Mediapipe como la que más destaca, pero es cierto que tanto esta como OpenCV como GestIA han empeorado sus registros de aciertos en vídeo a tiempo real respecto a las imágenes estáticas. En el caso de DLIB no consigue detectar ni la mitad de los gestos de dedo índice pero vuelve a conseguir hacerlo en pocos segundo, igualando en algunos casos a Mediapipe.

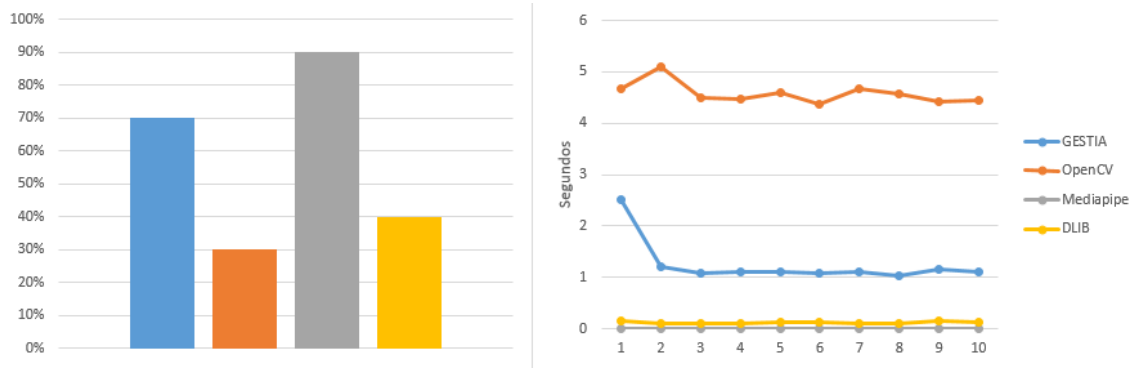


Figura 25 *Resultados detección índice en vídeo a tiempo real*

Fuente: [Elaboración propia]

El siguiente gesto que utilizamos para comparar es el gesto del pulgar hacia arriba, el conocido como “Ok!”. En la figura 26 observamos que todas nuestras librerías a tiempo real tienen menos acierto para detectar el gesto que cuando se hacía mediante imágenes estáticas. Pese a tener menos acierto sigue estando Mediapipe muy por encima del resto. Además DLIB no ha obtenido ningún acierto en estas pruebas.

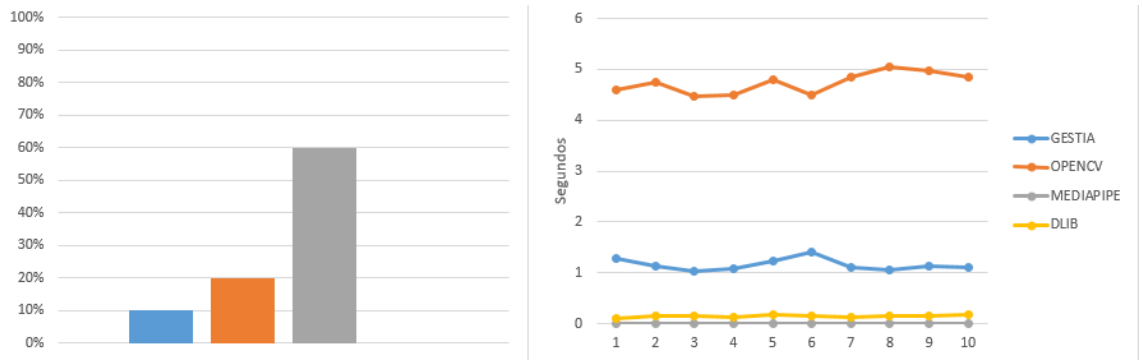


Figura 26 *Resultados detección ok en vídeo a tiempo real*

Fuente: [Elaboración propia]

En el caso de la detección del gesto del puño los resultados se pueden observar en la figura 27. En este gesto es la única vez en que Mediapipe es superado por alguna otra librería, en este caso obtiene un cien por cien de acierto GestIA. Esto se debe principalmente a que GestIA en la mayoría de gestos que no acierta es porque detecta un puño, por lo que en el caso de detectar el puño siempre acierta pues casi siempre resulta ser la primera predicción que hace. Le sigue Mediapipe, el cual vuelve a destacar en los segundos que tarda en detectar el gesto. Con DLIB pasa exactamente igual que en el caso anterior, no es capaz de detectar ningún puño que se le muestra en la imagen.

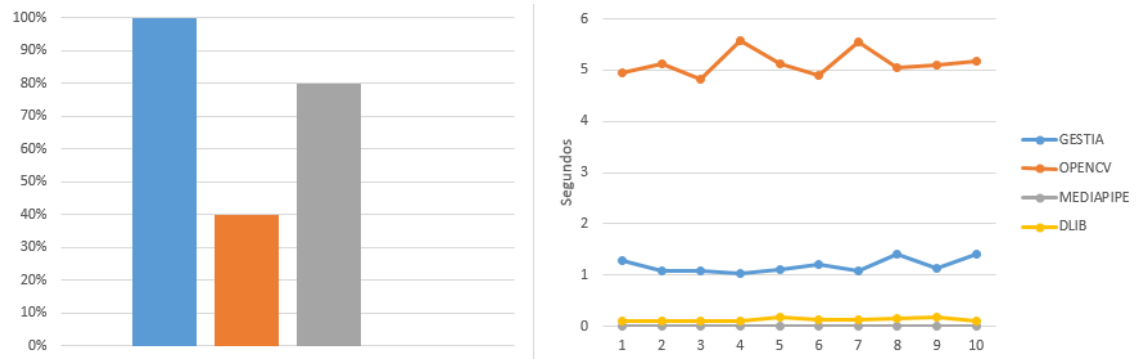


Figura 27 *Resultados detección del puño en vídeo a tiempo real*

Fuente: [Elaboración propia]

Por último, comparamos la detección del gesto que nombramos de paz, con el dedo índice y corazón levantados. En la figura 28 observamos que GestIA mejora respecto a la imágenes estáticas hasta un 70% de acierto, igualando a OpenCV. Mediapipe empeora un poco, pese a ello obtiene un 80% de acierto y se queda igualado con DLIB, donde ambas librerías tardan muy pocos segundos en ser capaces de detectar el gesto.

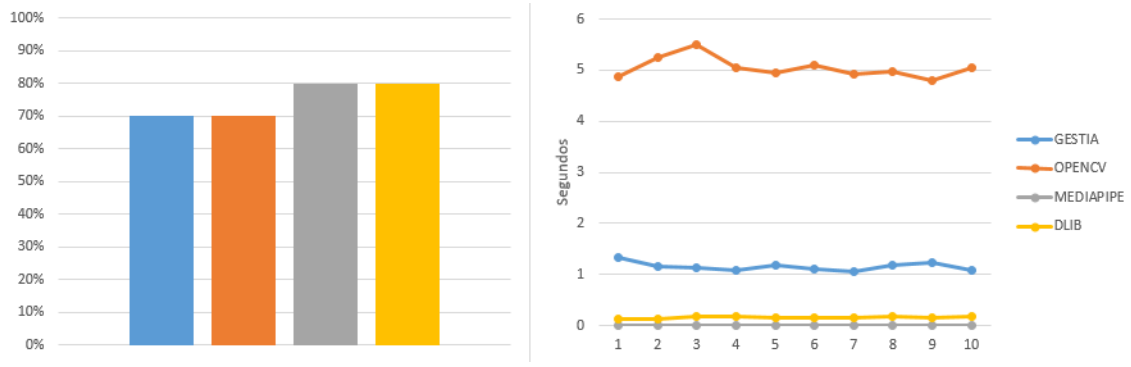


Figura 28 *Resultados detección gesto de paz en vídeo a tiempo real*

Fuente: [Elaboración propia]

Llegados a este punto tendríamos la comparación de todos los gentos en todas las APIS tanto en vídeo a tiempo real como mediante imágenes estáticas, comparado en el bloque anterior. En este momento podemos comenzar a sacar algunas conclusiones pues parece que con las imágenes estáticas tanto Mediapipe como OpenCV nos daban resultados de ciertos muy similares pero respecto al tiempo siempre era más rápido Mediapipe. Pero una vez realizadas las pruebas de vídeo a tiempo real se hace más evidente la rapidez y el acierto que obtiene Mediapipe, mejorando a todas las librerías en estas pruebas, además tardando milisegundos en ser capaz de detectar cualquiera de los gestos.

Por último en este bloque nos queda evaluar los recursos que utilizan cada una de las librerías empleadas, respecto al dato de recursos consumidos en la CPU se refiere a la media cada 60 segundos, mientras que la memoria se trata de la memoria virtual reservada por el sistema operativo para el proceso. En la tabla 2 vemos como el que menor uso hace de la CPU es la librería Mediapipe, mientras que la que más uso hace es OpenCV gastando demasiado para lo que tarda en detectar los gestos. Respecto a la memoria la que menos ocupa es DLIB muy cercano a Mediapipe, mientras que GestIA necesita casi el doble.

	GESTIA	OPENCV	DLIB	MEDIAPIPE
CPU	43%	63%	26%	22%
MEMORIA	525,256KB	405,344KB	179,500KB	190,568KB

Tabla 2 *Recursos utilizados por cada librería*

Fuente: [Elaboración propia]

Como conclusión a este bloque tenemos bastante claras las diferencias de las librerías, posiblemente este bloque era más necesario para acabar de destacar a una librería sobre las otras. Tanto por velocidad de detección como por acierto la librería más adecuada con la que podemos seguir nuestro proyecto debería ser Mediapipe. Pero antes de tomar esta decisión realizaremos el último bloque de pruebas.

BLOQUE 4

Hasta este último bloque todas las pruebas han sido realizadas en las mejores condiciones posibles, es decir con una imagen o vídeo a tiempo real con una buena

iluminación y una distancia adecuada de la cámara. Sin embargo no todos los usuarios podrán o querrán utilizar la aplicación en las mejores condiciones, puede que alguno quiera utilizarlo a oscuras o desde lejos del computador.

En este bloque se realizarán varios tipos de pruebas, una donde no tengamos mayor iluminación en la habitación que la que tenemos de la pantalla del computador. Otra de las pruebas consistirá a realizar los gestos en un sitio con mucha luz, incluso pudiendo hacer a contraluz. También se realizará una prueba donde los gestos se realicen muy próximos a la cámara y otra donde haya mayor distancia que la habitual. Otra de las pruebas será el realizar el gesto muy rápido, así comprobaremos si todos son capaces de detectarlo. Por último, realizaremos una prueba donde además del usuario aparezca en cámara algún otro sujeto, debiendo la librería detectar únicamente el gesto que realiza el usuario, pues será el más cercano.

Todas estas pruebas serán realizadas con todos los gestos que hemos utilizado en los anteriores bloques de pruebas y en cada una de las APIs en vídeo a tiempo real.

Los resultados que se muestran en la tabla 3, se observa como la librería Mediapipe es la más completa, donde en todas las pruebas que se han realizado con dificultades no le ha costado detectar el gesto que estábamos realizando. El resto de librerías en cada una de las pruebas al menos algún gesto fallan, excepto OpenCV que incluso habiendo otras personas en la imagen era capaz de detectar el gesto necesario.

		Indice	Palma	Ok	Puño	Paz
Baja luminosidad	GestIA	No	Sí	No	Sí	No
	DLIB	Sí	Sí	No	No	No
	OpenCV	No	Sí	No	No	No
	Mediapipe	Sí	Sí	Sí	Sí	Sí
Alta luminosidad	GestIA	Sí	No	No	Sí	Sí
	DLIB	No	No	No	No	No
	OpenCV	No	Sí	No	No	No
	Mediapipe	Sí	Sí	Sí	Sí	Sí
Próximo	GestIA	Sí	Sí	No	Sí	Sí
	DLIB	Sí	No	No	Sí	No
	OpenCV	No	No	No	No	No
	Mediapipe	Sí	Sí	Sí	Sí	Sí
Distante	GestIA	No	No	No	No	No
	DLIB	Sí	Sí	No	No	Sí
	OpenCV	No	No	No	No	No
	Mediapipe	Sí	Sí	Sí	Sí	Sí
Alta velocidad	GestIA	No	No	No	No	No
	DLIB	Sí	No	No	No	Sí
	OpenCV	No	No	No	No	No
	Mediapipe	Sí	Sí	Sí	Sí	Sí
Varios sujetos	GestIA	No	No	No	No	No
	DLIB	Sí	Sí	No	No	Sí
	OpenCV	Sí	Sí	Sí	Sí	Sí
	Mediapipe	Sí	Sí	Sí	Sí	Sí

Tabla 3 *Pruebas con imágenes no óptimas en cada librería*

Fuente: [Elaboración propia]

Después de haber realizado las pruebas de todos los bloques y haber analizado los resultados obtenidos en cada uno de ellos vemos como sobre las demás librerías destaca Mediapipe. Sobre todo uno de los bloques más importantes sería ese tercer bloque donde se analiza la detección de gestos en cada librería en vídeo a tiempo real, es de los más importantes ya que nuestra aplicación final será utilizada en vídeo a tiempo real.

4. Solución propuesta

Como hemos visto en las distintas pruebas realizadas a todas las APIs, Mediapipe es la que mejor comportamiento nos ha dado al realizar las pruebas tanto estáticas como de vídeo a tiempo real e incluso es en la cual hemos obtenido mejores resultados respecto a tiempo y recursos del computador empleados. Además Mediapipe es una solución universal, que simplemente se trata de un esqueleto por lo que no se aprenden gestos como tal de un entrenamiento de una persona concreta, sino que mediante los puntos clave obtienes donde está la mano y como se realizan ciertos gestos. Es por todo esto que será Mediapipe la base de nuestro proyecto.

Es importante saber de qué código base partimos, pues el código que encontramos en un primer momento únicamente detectaba la mano y es en el apartado anterior de pruebas de APIs que se pudo obtener la detección de diferentes gestos. Así pues nuestro código base de desarrollo sería el utilizado finalmente en las pruebas, es decir, una vez ya detecta tanto la mano como los gestos.

Nuestro objetivo principal es realizar un prototipo de aplicación que funcione a modo de script, donde mediante gestos en tiempo real interactúe con el computador. Para poder lograrlo lo primero que debemos de tener es un código que detecte gestos, esta primera fase ya la tenemos, pero es importante entender cómo funciona.

Además de la detección de gestos la otra parte del código más esencial es la capacidad de interactuar con el computador, así que esta será la segunda fase del desarrollo de la solución. En esta fase desarrollaremos la interacción con el computador pero de una forma sencilla, simplemente al hacer un gesto que suceda un evento como puede ser una simulación de presionar la flecha hacia abajo del teclado. Una vez obtenido al menos la simulación de presionar una tecla, todos los eventos de teclados serán sencillos de obtener.

Así pues la siguiente fase seguirá por el camino de la interacción con el computador, pero esta vez a través del ratón, consiguiendo que mediante gestos podamos mover el cursor y realizar un clic.

La última fase consistirá en hacer un gesto dinámico, es decir, en lugar de hacer un solo gesto de un segundo que se detecte hacer un gesto que suponga un movimiento, como puede ser juntar el dedo pulgar e índice e ir separándose. Este gesto dinámico se asociará a la interacción con el computador como la subida o bajada del volumen.

Por cada fase que se concluya habrá que realizar la prueba correspondiente para comprobar que la implementación es correcta y funciona. Todas estas pruebas se pueden realizar en una aplicación de pasar diapositivas o un visualizador de fotos.

Finalmente, una vez se pasen las diferentes fases se puede decir que habremos llegado a nuestro objetivo principal, crear un prototipo de aplicación que interactúa con el computador a tiempo real.

Cuando ya tenemos nuestro script listo para ejecutarlo bastará con llamarlo desde la línea de comandos, funcionará correctamente en todos los ordenadores que

dispongan del lenguaje Python, las librerías necesarias instaladas y una cámara conectada al computador.

Por último se realizarán pruebas en otros ordenadores para comprobar que se puede usar, además de ser probados por usuarios diferentes que mediante una guía de usuario sepan utilizarlo para así saber todas las ventajas e inconvenientes que tiene nuestro proyecto.

5. Desarrollo de la solución

Antes de comenzar a desarrollar la solución es importante tener en cuenta la diferencia del código que obtuvimos por primera vez en la web y el código que tenemos actualmente como base. Como se ha comentado en el anterior apartado, el primer código que teníamos simplemente era capaz de detectar una mano y pintar los 21 puntos clave de esta y las conexiones que lo unen.

Así pues este primer código que teníamos no era capaz de detectar qué gesto se estaba realizando, es en este apartado de pruebas donde ya investigamos acerca del script que teníamos de software de código abierto.

De esta investigación observamos tres variables claves a la hora de cómo se comporta el detector, una de ella es “static_image_mode” que tiene que ser verdadero cuando se trate de pasarle una imagen estática, mientras que si le pasamos un vídeo debería de encontrarse a falso. La otra es el “max_num_hands” que tiene en cuenta el máximo número de manos que puede detectar, generalmente dos o una. La última de las variables es “min_detection_confidence” que ajusta la confianza del modelo a un mínimo, es decir, en un rango de 0 a 1, 1 sería que está totalmente seguro de los puntos clave que señala y 0 nada, por defecto esta variable se encuentra a 0.5. En el caso de que “static_image_mode” se encuentre en verdadero el valor que le pasemos a “min_detection_confidence” no se tendrá en cuenta.

Además en este primer código vemos como mediante las funciones de la librería de mediapipe es capaz de obtener las coordendas de los diferentes puntos y cómo las dibuja y los une.

Pero vayamos paso por paso, a la hora de investigar el código hay que destacar que no se trata de un modelo que tengas que entrenar con tus propias imágenes para que te detecte sino que al ejecutar el script ya está listo para usar. De este modo vemos que no es posible entrenar un modelo con varios gestos, pero al darnos los puntos clave que se encuentran en la mano y saber sus coordenadas para poder detectar un gesto solo nos hace falta basarnos en estas coordenadas. Por ejemplo a la hora de hacer el gesto del dedo índice la coordenada en la que se sitúa la punta del dedo debe de ser la única que esté posicionada en la imagen como la más arriba respecto al eje Y.

Así pues lo único necesario para poder detectar gestos sería saber las coordenadas de los distintos puntos clave, que variarán según el gesto que se realice.

Para poder reconocer los gestos hemos implementado el código que se ve en la figura 29.

```
            lms = [4,8,12,16,20]
            lmList = []
            if results.multi_hand_landmarks:
                for handlandmark in results.multi_hand_landmarks:
                    for lm in handlandmark.landmark:
                        h,w,_ = frame.shape
                        lmList.append([int(lm.x*w),int(lm.y*h)])
                        mp_drawing.draw_landmarks(frame, handlandmark,mp_hands.HAND_CONNECTIONS)

            if lmList!=[]:
                fingers = []
                if lmList[0][1] < lmList[8][1]:
                    cv2.putText(frame,'Hand Gesture Not Recognisable!!',(20,50),cv2.FONT_HERSHEY_COMPLEX,0.9,(0,255,0),2)
                else:
                    if lmList[5][0] < lmList[17][0]:
                        fingers.append(True) if lmList[lms[0]][0] < lmList[lms[0]-2][0] else fingers.append(False)
                    else:
                        fingers.append(True) if lmList[lms[0]][0] > lmList[lms[0]-2][0] else fingers.append(False)

                    for lm in range(1,len(lms)):
                        fingers.append(True) if lmList[lms[lm]][1] < lmList[lms[lm]-2][1] else fingers.append(False)

                    cv2.putText(frame,simpleGesture(fingers,lmList[4],lmList[8]),(20,50),cv2.FONT_HERSHEY_COMPLEX,0.9,(0,255,0),2)
```

Figura 29 *Parte del código donde se dibuja y se detecta los gestos*

Fuente: [Elaboración propia]

Para explicarlo de forma sencilla, lms es una lista que guarda los puntos de cada uno de los dedos de la mano comenzando por el pulgar que corresponde al número 4 y siguiendo hasta el meñique que será el 20. El caso de la variable lmList la inicializaremos como una lista vacía, que posteriormente con esos primeros bucles se irá rellenando de las coordenadas del eje X y del eje Y de cada uno de los puntos, hasta obtener 21 puntos claves. Esto último será posible porque las implementaciones de las librerías de mediapipe es capaz de extraer las coordenadas. Es importante destacar que las coordenadas que obtenemos de estas librerías es necesario multiplicarlo por el ancho y el alto de la imagen para obtener los puntos reales en la imagen, además los hemos convertido a int pues un punto X e Y siempre deben de ser enteros. Por último en este bucle se dibujan ya los puntos y sus conexiones.

Es importante antes de adentrarnos en el código tener en cuenta cómo guardas las coordenadas de los puntos y las posiciones de ambos ejes, así nos será más sencillo entender cómo se detectan los diferentes gestos. A la hora de recorrer la lista de coordenadas tendrá varias listas anidadas, el primer índice señalará que punto queremos y el segundo si es 0 será sobre el eje X y si es 1 sobre el eje Y. Sin embargo no tendremos un plano como al que estamos habituados de que en la esquina inferior izquierda se encuentren los puntos (0,0), en este caso este punto se encontrará en la esquina superior izquierda y tenderán ambos a infinito en la esquina inferior derecha.

Una vez hecho este apunte, para detectar los gestos primero inicializamos una lista “fingers”, que se referirá a cuantos dedos hay levantados. La primera comprobación que realizaremos será que el punto 0, el cual marca la muñeca se encuentre más abajo en la imagen que el dedo índice, en caso de que no se cumpla y la muñeca se encuentre por encima de la punta del dedo índice entonces el gesto no será reconocible.

Bien es cierto que este hecho nos puede llevar a que ciertos gestos no puedan ser reconocidos, lo cual podría ser un problema en un futuro depende del gesto que se quiera reconocer. La decisión de tomar las distintas posiciones entre la muñeca y el dedo índice viene de pensar que a la hora de ejecutar la aplicación e interactuar con el

computador normalmente las personas lo llevarían a cabo sentadas, por lo que girar la mano en tal posición es más incómodo y poco útil para el usuario.

Una vez tenemos descartado que se trate de un gesto no reconocible pasamos a observar que dedos se encuentran levantados. Para ello hay que tener en cuenta que la mayoría de dedos están juntos y seguirán las mismas normas, pero sin embargo el pulgar no se encuentra en la misma posición que el resto por lo que sus normas son diferentes. A la hora de decidir si el dedo pulgar esta levantado o no la primera comprobación que realizaremos será la posición de la mano, no será igual si tenemos el pulgar más en la parte derecha o izquierda, para saber esto nos basamos en los puntos de la base del dedo índice y meñique. Si la base del dedo meñique se encuentra más a la derecha que la base del dedo índice entonces tendremos el pulgar mirando hacia la parte izquierda, lo que supone que si la punta del pulgar se encuentra más hacia la izquierda en la imagen que el punto de flexión de este dedo podemos concretar que el dedo pulgar se encuentra levantado o estirado. Sin embargo, si la punta del dedo pulgar se encuentra más hacia la derecha que el punto de flexión de este dedo diremos que tenemos el dedo encogido. Por otro lado en el caso de que el pulgar se encuentre en la parte derecha, es decir, que la base de dedo meñique se encuentre en una posición más cerca de la parte izquierda que la base del dedo índice, para saber si tenemos el pulgar levantado comprobaremos si la punta de este se encuentra más a la derecha que el punto de flexión de este. En caso de que el punto de flexión del dedo pulgar se encuentre más a la derecha que la punta concretaremos que tenemos el dedo encogido. Para poder tener en cuenta si los dedos están levantados simplemente pondremos en la lista “fingers” verdadero en caso de que lo estén y falso si no lo están.

En el caso del resto de dedos será más sencillo pues comprobaremos mediante un bucle que empiece a partir del dedo índice hasta el meñique si el punto que se encuentra en la punta de cada dedo está más arriba en la imagen que el punto de flexión de ese mismo dedo, cuando esto ocurra diremos que el dedo se encuentra levantado. En caso contrario donde el punto del medio del dedo esté más arriba que la punta se determinará que esta bajado.

Una vez obtenido los dedos que se encuentran levantados y los que no, llamamos a la función “simpleGesture” del fichero “gesture.py” el cual encontramos en el Anexo I (Apartado 12.1), desde donde detectaremos realmente el gesto que se realiza gracias a la lista “fingers”. Así dependiendo de que dedo se encuentre levantado sabremos que realizamos un gesto u otro, por ejemplo si solo tenemos levantados el dedo índice y el dedo corazón entonces se mostrará un texto en la imagen que dirá que realizas el gesto de la paz.

Una vez conseguido que se detecten los gestos que son necesarios para el uso de nuestra aplicación podemos concluir que la primera fase del proyecto, aquella de detectar gestos se ha conseguido.

Avanzamos a la segunda fase, aquella en la que buscamos que mediante un gesto se interactúe con el computador a través del teclado. Como ya habíamos dicho anteriormente esta segunda fase buscaremos una interacción sencilla como puede ser la simulación de presionar una tecla. En un primer momento hemos decidido simular que pulsamos la flecha de hacia abajo del teclado, esto ha sido decidido porque si

queremos utilizar un visualizador de fotos o de diapositivas una forma sencilla de ir pasando hacia delante podría ser mediante las flechas del teclado. Para poder conseguir esta interacción hay algunas librerías gratuitas disponibles como keyboard o pyautogui que ya hemos comentado en el apartado de librerías. Es importante destacar algunas diferencias que existen entre ellas, con la librería keyboard puedes seleccionar cualquier tecla que se encuentra en el teclado e incluso permite combinar algunas teclas como “shift +Ctrl”, además permite la internacionalización del teclado. Sin embargo en el caso de utilizar la librería pyautogui las únicas teclas que permite seleccionar son los números o las letras, se puede llegar a combinar algunas teclas pero ya se necesitaría una mayor cantidad de líneas de código. Es por la facilidad que nos da la librería keyboard respecto a la combinación de teclas que nos hemos inclinado hacia el uso de esta. Además en nuestra aplicación tendrá gran importancia la tecla “f5”, la cual en algunos casos se activa mediante la combinación de teclas, siendo esta necesaria en algunos editores de presentaciones para comenzar a presentar las diapositivas.

Esta librería tiene un método que con solo llamarlo se presiona la tecla que le digas, así pues cuando el método del detector de gestos devuelva el nombre del gesto que queremos le diremos mediante esta librería que se presione la flecha hacia abajo.

De esta manera conseguiríamos abordar esta segunda fase y pasar a la tercera fase donde se sigue requiriendo la interacción con el computador, pero esta vez del ratón. Nuestra intención únicamente es conseguir a través de los movimientos de la mano que se pueda mover el ratón alrededor de la pantalla y poder hacer clic.

En este punto debíamos pensar sobre varias cuestiones, necesitamos un punto clave de la mano en el cual situar el cursor y que gesto pueda significar hacer clic. A la hora de ver el punto clave que fuera el cursor puede que el primer punto que todo el mundo se le venga a la cabeza sea la punta del dedo índice, puede que te haga tener una mayor precisión a la hora de moverte pero también es cierto que no es un punto muy e estable cuando se calcula la coordenada. Además ambas cuestiones planteadas deben medirse conjuntamente, en caso de que el cursor fuera la punta del dedo índice puede que el gesto más sencillo de realizar para hacer clic sea juntar el dedo índice y corazón para no moverlo del lugar donde queremos hacer clic. Esta podía ser la solución más lógica ya que es lo más similar a hacer clic con un ratón físico. Pero obtuvimos algunos problemas, la precisión que esperábamos no fue la que obtuvimos pues para poder hacer clic el movimiento natural era mover el dedo índice hacia el corazón con lo que el cursor se movía del lugar de la pantalla y no se hacía clic al elemento que se quería. Además pensamos que uno de los propósitos de nuestra aplicación era ser accesible a todo tipo de personas, incluso aquellas que tengan algún tipo de dificultad con la movilidad, por lo que el gesto para hacer clic podía ser muy dificultoso y además la punta del dedo índice como cursor les podía suponer algún problema.

Una vez descartada esta primera prueba del punto de referencia del cursor y el gesto de la interacción, investigamos acerca del punto que es más estable y pensamos en dos posibilidades, podíamos escoger el punto de la muñeca o la base del dedo corazón. Estos dos puntos podían ser el punto clave que interactuará con el cursor, finalmente nos decantamos por elegir la base del dedo corazón sobre la muñeca y que para llevar el cursor a las esquinas de la pantalla habría que realizar un mayor recorrido,

además una vez no se encuentran los dedos en la imagen es difícil que se detecta únicamente el punto de la muñeca.

Respecto al hacer clic, una vez establecido el punto de la base del dedo corazón como cursor, pensamos que una de las maneras era bajar la punta del dedo índice por debajo de su base, así manteníamos el punto del cursor en el mismo lugar. Sin embargo se convertía en un gesto difícil de realizar para algunas personas, pero a pesar de dicha dificultad se ha mantenido en el código, ya que el gesto de bajar todos los dedos incluye bajar el dedo índice por lo que también daría resultado a la interacción con el computador, realizándose un clic.

Para determinar si se baja el dedo índice para poder hacer clic podríamos haber pensado en los gestos directamente, pero en este caso como estábamos haciendo por scripts separados cada una de las fases, lo calculamos como que la distancia entre la muñeca y la base del dedo corazón debe de ser mayor que la distancia entre la muñeca y el dedo índice, al suceder esto se realizaría el clic.

Ahora que ya tenemos claros los puntos y el gesto que necesitamos para la interacción, hemos investigado acerca de que librería utilizar y como se comentó en el apartado librerías, consideramos que la librería pyautogui para este caso sí que es válida. Así con los métodos de esta librería podemos decir los puntos en los que se debe de mover y el gesto a detectar para hacer clic.

De esta fase es importante destacar que hay que tener en cuenta la pantalla del computador que se utiliza, pues es necesario saber la relación aspecto para luego interpolar las coordenadas obtenidas de los puntos clave para poder mover el cursor y clicar. Este aspecto es algo que se comprobará si funciona correctamente cuando se pruebe en otros computadores con un tamaño de pantalla distinto al que estamos realizando el desarrollo, pues puede que sea un valor que haya que modificar según el ordenador del usuario.

Finalmente llegamos a la última fase del desarrollo que nos habíamos marcado en la solución propuesta, donde el computador debe de interactuar al detectar un gesto dinámico.

Hasta ahora todos los gestos que habíamos realizado se trataban de gestos estáticos, es decir, haces un gesto en un momento determinado con la mano y ese es el que se captura y el que hace que se envíe un evento al computador. Ahora mismo lo que buscamos es un gesto dinámico, es decir, un gesto que se realiza en movimiento, según el movimiento a lo largo del tiempo que se realice el evento enviado será uno u otro.

Como gesto dinámico que podríamos añadir a nuestra aplicación hemos pensado en uno sencillo, uno que juntando la punta del dedo pulgar y la del índice mida la distancia que se encuentre entre ellos, así según la distancia que haya el volumen variará. Cuando la distancia sea mínima el volumen del equipo se pondrá a cero, conforme vaya aumentando la distancia de ambos dedos el volumen aumentará, en el momento en que la distancia sea máxima se llegará al cien por cien del volumen.

Al implementar este gesto dinámico nos hemos encontrado con una dificultad, dependiendo de la distancia que se encuentre entre la cámara y la mano, la distancia que captará entre la punta del dedo índice y del dedo pulgar será una u otra. Realmente esto podría suponer un problema si la posición de la mano se encuentra muy cerca de la cámara porque puede que nunca llegue a poder subir el volumen al 100%, mientras que si está muy lejos es posible que el volumen suba al 100% demasiado rápido cuando aún la distancia real entre ambos puntos no sea la máxima.

Para poder solucionar este problema deberíamos de tener un sensor de profundidad para así calcular la distancia real que pueda existir. Al no disponer de este dispositivo, hemos optado por implementarlo pensando en la distancia más correcta que debe de haber entre un ordenador y su usuario, donde lo recomendable es al menos 40cm de separación.

Para poder implementar esta funcionalidad hemos necesitado importar nuevas librerías de código abierto que interactúen con el volumen del computador, para poder variarlo. En nuestro caso hemos encontrado la librería pycaw de Python, la cual sirve para modificar el audio de un computador, aunque para poder usarla ha sido necesario importar las librerías ctypes y comtypes, por los requisitos de los métodos que tiene pycaw.

Una vez instalamos e importamos las tres librerías anteriores obtenemos el rango de volumen de nuestro equipo, para así decirle que cuanta más distancia entre la punta del dedo pulgar e índice aumente hacia el máximo y cuanta menor distancia baje al mínimo. Igual que en la fase anterior vuelve a ser importante interpolar las distancias obtenidas con el rango de volumen de nuestro computador.

En el momento en que ya está implementada la última funcionalidad del volumen y probada podríamos concluir que hemos llegado al final de las fases de la solución propuesta.

Hemos llegado al punto en el que comprobamos que las funcionalidades de todas las fases implementadas en un mismo script se ejecutan correctamente, pero pensamos en una pequeña mejora que ayudaría al usuario centrándonos en el uso para presentaciones. En un primer momento, en las fases de interacción con el teclado nos basamos en un gesto estático que al realizarlo se envíe un evento de teclado con el que podamos avanzar a la siguiente diapositiva o retroceder a la anterior, también tenemos la interacción con el ratón y con el volumen del dispositivo. Pensando en la interacción con una diapositiva que avances a otra o retrocedas en este momento estaba implementado con un gesto como podía ser levantar solo el dedo pulgar o levantar tres dedos, esto funciona correctamente pero no es algo útil para el usuario. En caso de querer utilizar nuestra aplicación el usuario debía de leerse antes una guía pero no era fácil de recordar el gesto para cuando los usuarios volvieran a utilizar la aplicación por lo que no acababa siendo un gesto útil.

En este momento pensamos posibles mejoras y se nos ocurrió que podríamos dividir la imagen que obtiene la cámara en tres, una parte izquierda y derecha que medirían exactamente lo mismo y una central un poco más amplia. Al ponerse en el lugar del usuario y su experiencia parece más lógico que para avanzar de diapositiva si colocas la mano en la parte derecha, la diapositiva cambie a la siguiente. Igual que si

colocas la mano a la parte de la izquierda vuelva a la diapositiva anterior. Si nos paramos a pensarlo, estos movimientos podrían ser muy parecido a la hora de tener una pantalla táctil y deslizas desde la izquierda hacia la derecha y viceversa, por lo que al usuario le será un gesto más fácil de recordar para poder utilizar la aplicación.

Así pues en las divisiones de los laterales con simplemente que se detecte la palma de una mano, es decir, como un gesto mostrando los 5 dedos, la diapositiva ya cambiaría. La parte central de las divisiones la hemos reservado para únicamente la interacción con el ratón, así que a la hora de poner la mano en algún lateral el ratón no se moverá ni hará clic, para ello se realizarán los gestos descritos en la parte central. Por último respecto a la interacción del volumen podría haberse puesto en esta parte central, pero consideramos más lógico pasarla al lateral derecho, ya que en todos los computadores encontramos el volumen en la esquina inferior derecha. Así que en caso de querer variar el volumen será necesario realizarlo en el lateral derecho separando o juntando las puntas de los dedos pulgar e índice.

Finalmente, ya tendremos nuestra aplicación lista y en condiciones de funcionar, como se ha nombrado en el apartado 1.3 todo el código desarrollado de la aplicación se encuentra en el enlace de github.

6. Presupuesto

En este apartado del trabajo se quiere hacer una estimación del dinero que nos costaría realizar la aplicación. Hay que destacar que solamente se tendrá en cuenta en este presupuesto los materiales que necesitaríamos así como los recursos humanos para la realización de la aplicación, es decir, no se tendrá en cuenta lo que posteriormente costaría su comercialización en términos de gestión de empresa ni marketing.

Antes de empezar a presupuestar es necesario tener claro los supuestos que queremos abarcar con la aplicación, saber que usabilidad queremos que tenga. Hasta este punto en nuestro proyecto hemos buscado conseguir la interacción con el computador mediante gestos, es decir, poder hacer con gestos lo que actualmente haríamos con un teclado o un ratón. En nuestro caso nos hemos centrado en la utilización de un editor de presentaciones, pero hay muchos programas que pueden ser interesantes para utilizarlos mediante gestos. Por lo que el presupuesto dependerá de cuantos programas se quieran abarcar para poder ser utilizado por gestos.

Para la creación de nuestro proyecto hemos pasado varias fases, empezando por la investigación de las librerías disponibles, el desarrollo y la puesta en marcha del prototipo. Para finalmente decir que se trata de una aplicación faltaría realizar la parte del diseño de esta, donde debemos tener una interfaz cómoda para el usuario final.

Respecto a los recursos humanos que se necesitarían a la hora de acabar de realizar esta aplicación, pensando en una aplicación bastante completa que abarque varios programas, deberíamos de contar con tres desarrolladores que realicen investigación de las librerías disponibles y la posterior implementación del código necesario. Además necesitaríamos un diseñador para realizar la interfaz, así la utilización de la aplicación será sencilla.

Teniendo en cuenta estas cuatro personas, habrá que darles el material hardware necesario, por lo que dispondremos de cuatro computadores. Tres computadores serán iguales, los tres que irán destinados para los desarrolladores, necesitaremos que se tengan una gran capacidad de computación y una buena cámara para poder hacer pruebas. El cuarto computador será para el diseñador, este tendrá que contar con una buena tarjeta gráfica. Además cada computador tendrá el material software necesario para el trabajo realizado.

Un aspecto importante a tener en cuenta en nuestro presupuesto será en cuanto tiempo pensamos que se podría obtener esta aplicación lista, pues como no hablamos del mantenimiento sino simplemente de realizarla podemos pensar que con los recursos humanos que contamos en cuestión de cuatro meses estaría lista si trabajan a tiempo completo. Dentro de estos cuatro meses se contaría con la investigación, el desarrollo, las pruebas y la implementación final.

Así pues el presupuesto final calculado con los precios que existen en este momento en el mercado será el propuesto en la tabla 4.

DESCRIPCIÓN	Nº UNIDADES	COSTE UNITARIO	COSTE TOTAL
Recursos humanos	4	2.000€/mes	32.000€
Computador para el desarrollo	3	1.400€	4.200€
Computador para el diseño	1	1.000€	1.000€
Recursos software	4	150€	600€
			TOTAL
			37.800€

Tabla 4 *Presupuesto aplicación completa*

Fuente: [Elaboración propia]

El presupuesto que hemos extraído es en vistas al futuro, es decir, pensando en realizar la aplicación completa y no solo el prototipo como se ha hecho en este trabajo de final de grado. Así que en este presupuesto se tiene en cuenta muchas mejoras que podrían suponer las líneas de futuro que podemos plantearnos.

Para obtener el presupuesto del trabajo que hemos realizado en el proyecto no obtendríamos un producto final como tal, solo el prototipo de la aplicación. Para poder realizar este proyecto se han invertido 430 horas, donde la mitad han ido destinadas a la investigación de las librerías y el mercado actual mientras el resto hacia la implementación. Teniendo en cuenta las horas que se han invertido y que en una jornada de trabajo normal como mucho se podrían hacer 8 horas al día y en nuestro sector normalmente es de lunes a viernes, estaríamos hablando de más o menos dos meses.

Por lo que si obtenemos el presupuesto estimado de lo realizado en nuestro trabajo de final de grado sería los datos que tenemos en la tabla 5.

DESCRIPCIÓN	Nº UNIDADES	COSTE UNITARIO	COSTE TOTAL
Recursos humanos	1	1.500€/mes	3.000€
Recursos hardware	1	1.400€	1.400€
Recursos software	1	0€	0€
			TOTAL
			4.400€

Tabla 5 *Presupuesto del trabajo realizado*

Fuente: [Elaboración propia]

Este último presupuesto sería el presupuesto estimado de lo que ha costado realizar el actual proyecto final de grado. Es importante destacar de este que los recursos software son gratuitos ya que todo lo empleado ha sido software de código abierto. Los recursos hardware se ha tenido en cuenta el precio del ordenador que pese a no ser comprado especialmente para este uso es un gasto que se podría computar al presupuesto. Por último, comentar acerca de los recursos humanos, nos hemos puesto el salario medio de una persona junior, es decir, de alguien que está a punto o que acaba de terminar sus estudios. Este salario es un poco menor que el puesto en el anterior presupuesto, pues pensamos que necesitaremos a alguien con un poco más de experiencia.

7. Implantación

Para que un usuario pueda utilizar el prototipo de aplicación que hemos creado durante este trabajo, primero deberemos de implantarlo en el computador que quiera ser usado mediante gestos. Además para su utilización será necesario que el usuario disponga de una guía que le indique que gestos debe realizar para interactuar con el computador.

7.1 Requisitos para implantación

Para poder implantar este trabajo en cualquier otro computador necesitaremos que el computador donde lo vayamos a utilizar disponga de algunos requisitos. Como elemento hardware necesitaremos que tenga una cámara, no es necesaria que sea una cámara que ya esté integrada en el portátil puede ser una cámara auxiliar.

Para poder lanzar la aplicación únicamente es necesario tener los scripts que se encuentran en el enlace de github mencionado en el punto 1.3 de este documento.

Se lanzará en la terminal del computador mediante la línea de comandos, donde únicamente tendremos que ir a la carpeta donde se encuentra el script y llamarlo, escribiendo únicamente el nombre del fichero “hand_control.py”.

Antes de poder ejecutar la aplicación será necesario que el equipo tenga instalado el lenguaje de programación Python, la misma versión que hemos utilizado en el proyecto (v. 3.6.5) o superior.

Además también será necesario que se disponga de las librerías que importamos en el código. Estas son:

- Opencv-python
- Mediapipe
- Comtypes
- Pycaw
- Keyboard
- Pyautogui

En caso de no disponder de ellas previamente en el computador, se pondrán instalar de manera sencilla lanzando el comando “pip install” seguido de los nombres de las librerías anteriormente nombrados. Por ejemplo, para instalar la librería de mediapipe escribiremos “pip install mediapipe” y al cabo de unos segundos ya estará instalada y lista para ser utilizada.

7.2 Guía de usuario

En este apartado pretendemos realizar un escrito para que el usuario pueda utilizar la aplicación sin ningún tipo de dificultad. Será útil tanto para usuarios nuevos que quiera saber que gestos debe de realizar para llevar a cabo cierta interacción, como para antiguos usuarios que quieran recordarlo.

Al ejecutar el script veremos nuestra imagen en la pantalla, una imagen dividida en tres partes por dos líneas rojas.

En la parte central tendremos la interacción con el ratón, podremos mover el cursor al mover nuestra mano por toda esta parte y se realizará clic al bajar todos dedos, dejando la mano en forma de puño. Tal y como se muestra en la figura 30, observaremos en la imagen que el punto que será el cursor se dibujará de un color rosa y para hacer clic la línea azul clara deberá ser más corta que la línea azul oscura.

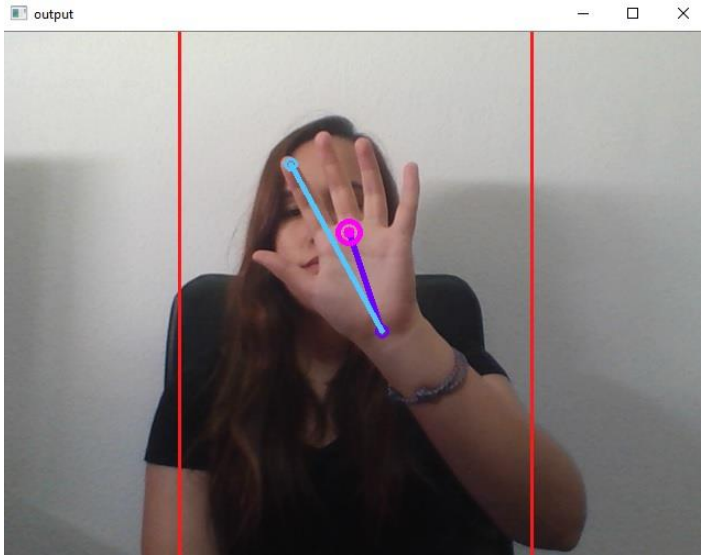


Figura 30 *Gesto para mover el cursor*

Fuente: [Elaboración propia]

En la parte central si únicamente bajamos el dedo anular y el dedo corazón, como vemos en la figura 31, haremos un gesto que significará que se presiona la tecla “f5”, que en el caso de nuestro programa de presentaciones se utiliza para comenzar a presentar las diapositivas. El gesto será el que se observa en la siguiente imagen:

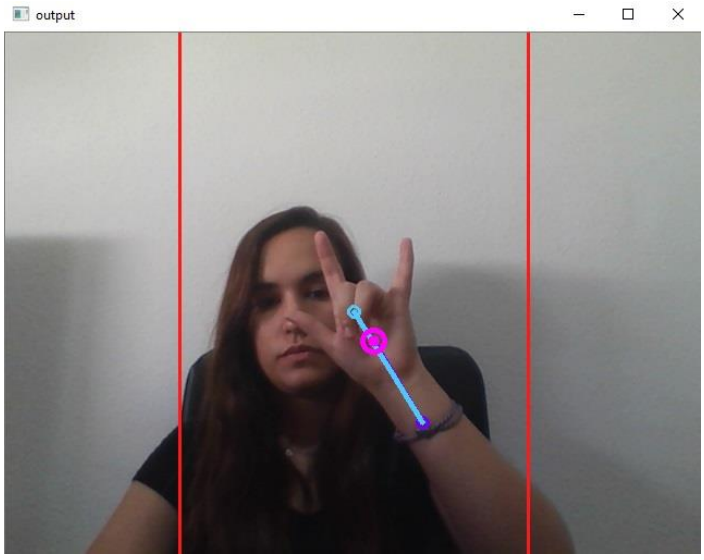


Figura 31 *Gesto para iniciar presentación*

Fuente: [Elaboración propia]

En la parte izquierda al pasar nuestra mano con la palma abierta se presionará la tecla de la fecha izquierda, así conseguiremos volver a la diapositiva anterior. En la figura 32 tenemos un ejemplo.

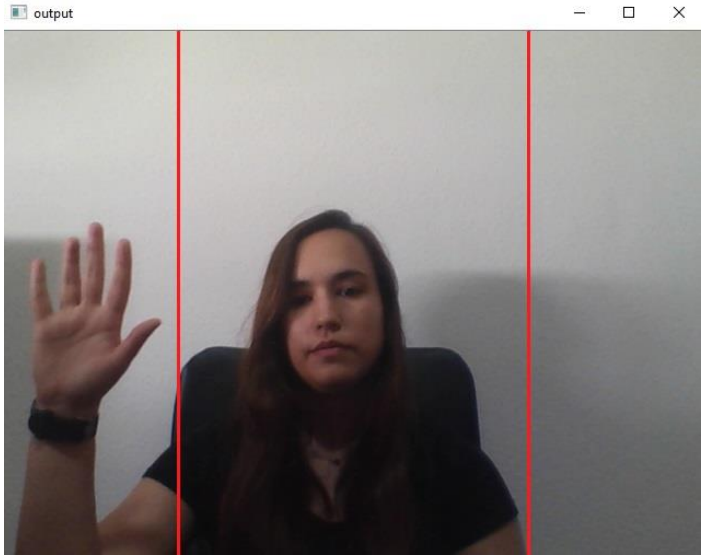


Figura 32 *Gestos para volver a la anterior diapositiva*

Fuente: [Elaboración propia]

En la parte derecha funcionará de la misma forma que la anterior, pero al pasar la mano con la palma abierta se presionará la flecha derecha, así se pasará a la siguiente diapositiva. La figura 33 muestra la forma de realizarlo.

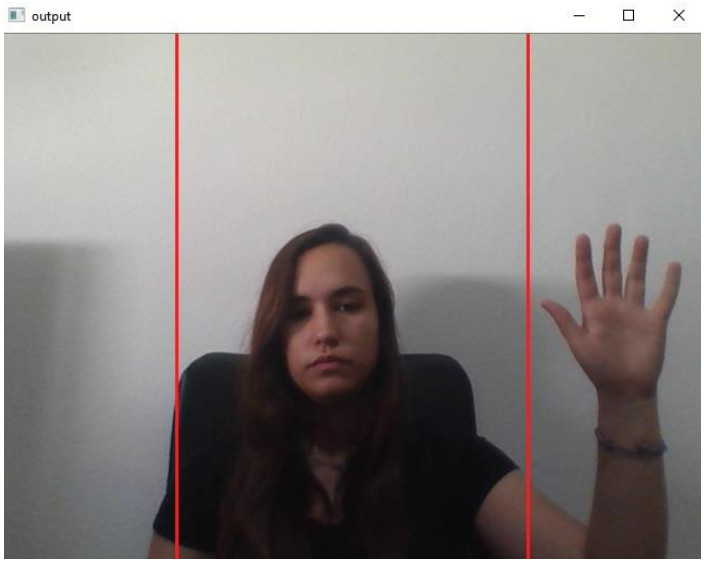


Figura 33 *Gesto para pasar a la siguiente diapositiva*

Fuente: [Elaboración propia]

Por último en esta parte derecha también podremos modificar el volumen del equipo, separando o juntando el dedo pulgar y el dedo índice. Para poder tener el volumen al mínimo posible realizaremos el gesto que vemos en la figura 34 mientras

que para subirlos iremos separando ambos dedos, cuando se llegue a una imagen parecía a la figura 35 se obtendrá el volumen máximo.

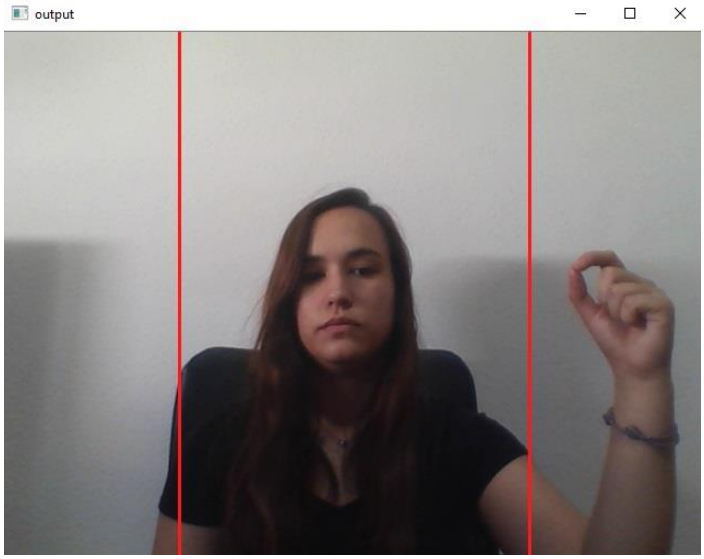


Figura 34 *Gesto para el volumen mínimo*

Fuente: [Elaboración propia]

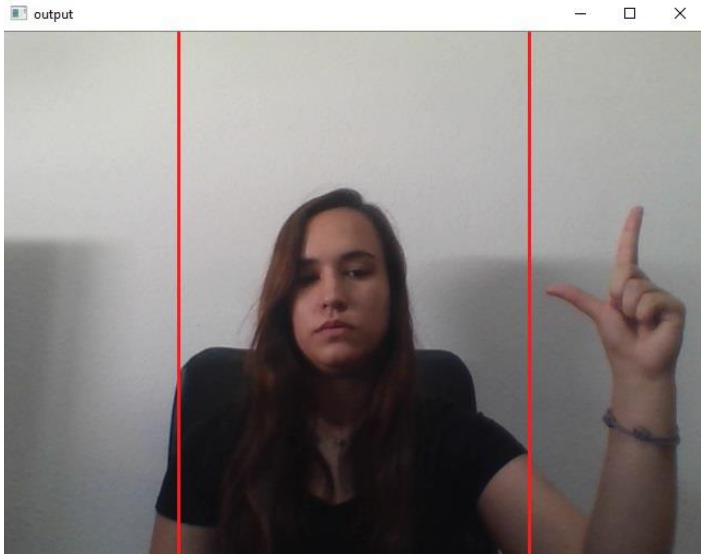


Figura 35 *Gesto para volumen máximo*

Fuente: [Elaboración propia]

Para poder finalizar la aplicación volver a la imagen de la cámara y pulsar la tecla escape.

8. Pruebas

En este apartado de pruebas se pretende que se utilice el prototipo de aplicación por distintos usuarios a los que los hemos desarrollado, así obtendremos una visión distinta. Para ello hemos elegido a seis personas, algunas más jóvenes que durante toda su vida han vivido rodeados de tecnología y otros más mayores, los cuales no siempre han estado en contacto con ordenadores.

Cada persona que realice las pruebas se leerá la guía de usuario de este documento y posteriormente utilizará la aplicación, por último completará un cuestionario en el que obtengamos respuesta a las siguientes preguntas:

- ¿Suele utilizar en su día a día un computador?
- ¿Alguna vez ha tenido que realizar una presentación?
- Ha sido fácil utilizar la aplicación.
- ¿Le ha sido sencillo mover el cursor y realizar un clic?
- ¿Le ha resultado sencillo pasar las diapositivas?
- ¿Ha sido fácil bajar y subir el volumen?
- ¿Cómo de útil le parece la aplicación?

Para utilizar la aplicación se les mandó una serie de pruebas, debían abrir PowerPoint, nuestro programa de presentaciones, que se encontraba previamente minimizado en la barra de tareas. Una vez abierto debían inicializar la presentación, pasar al menos seis diapositivas para adelante y volver a la diapositiva número cinco. En la diapositiva cinco encontrarían un vídeo que debía de ejecutar con el uso del clic del ratón y variar el volumen. Por último debía finalizar la aplicación pulsando la tecla “esc”.

La presentación que se utilizó durante las pruebas se encuentra disponible en el mismo lugar que los códigos de la aplicación.

Además no todas las personas que realizaron las pruebas lo hicieron desde el mismo ordenador, la mitad de ellas lo hicieron desde un ordenador Lenovo, con el que se desarrolla la aplicación. Mientras que la otra mitad lo hicieron desde un computador Huawei Matebook 14, el cual tiene una diferencia muy importante con el anterior, en lugar de tener la cámara en la parte superior de la pantalla esta se encuentra en el teclado, por lo que el ángulo no es el mismo.

A partir del Anexo II y hasta el Anexo VII (Apartados desde 12.2 hasta 12.7) se encuentras las respuestas individuales de cada sujeto. Hay que tener en cuenta que este cuestionario ha sido realizado a muy pocas personas, por lo que para obtener unas conclusiones más exactas será necesario tener una mayor muestra de la población.

Pese a que se realizó la encuesta a una muestra demasiado pequeña, vamos a extraer las respuestas a modo de resumen. Ya que pueden resultar muy interesante las conclusiones que extraigamos de cara al futuro de nuestra aplicación.

A la primera pregunta de si el usuario suele utilizar en su día a día un computador obtenemos un 83,3% de los votos afirmativos, como vemos en la figura 36. En este caso

se muestra bastante bien como es la realidad, pues hoy por hoy casi todo el mundo utiliza un ordenador en su día a día, ya sea para trabajar o por diversión, pero es extraño estar en una vivienda donde no exista un computador.

Figura 36 *Respuestas a la pegunta: ¿Suele utilizar día a día un computador?*

Fuente: [Elaboración sforms]

Del mismo modo que pasaba en la pregunta anterior, a la pregunta de si alguna vez ha tenido que realizar alguna presentación la mayoría de votos han sido afirmativos. De hecho la única persona que no ha realizado una presentación es aquella que no utiliza un computador en su día a día. Podemos observar las respuestas en la figura 37.

Figura 37 *Respuestas a la pegunta: ¿Alguna vez ha tenido que realizar alguna presentación?*

Fuente: [Elaboración sforms]

La siguiente pregunta se trataba de una afirmación en las que los sujetos debían puntuar la frase, mediante una escala de Likert, si estaban totalmente desacuerdo con un uno y con un 5 si estaban totalmente de acuerdo. En la figura 38 observamos las respuestas obtenidas, dónde la mitad está totalmente de acuerdo y la otra mitad de acuerdo, por lo que podemos extraer que para algunas personas no fue tan sencillo como para otras pero finalmente para todas les fue fácil.

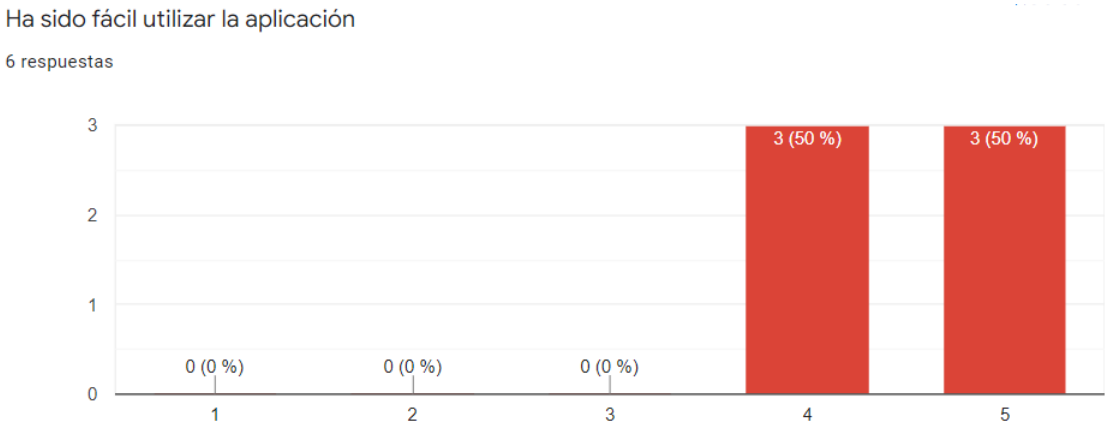


Figura 38 *Respuestas a la afirmación: Ha sido fácil utilizar la aplicación*

Fuente: [Elaboración sforms]

En la siguiente pregunta se cuestionaba si había sido sencillo utilizar la mano para controlar el ratón y hacer clic. En la figura 39 se muestra como a la gran mayoría les pareció sencillo, también es verdad que en la prueba no tuvieron que estar mucho rato utilizando y la gran mayoría conseguía mover el ratón y seleccionar lo que realmente se proponían.

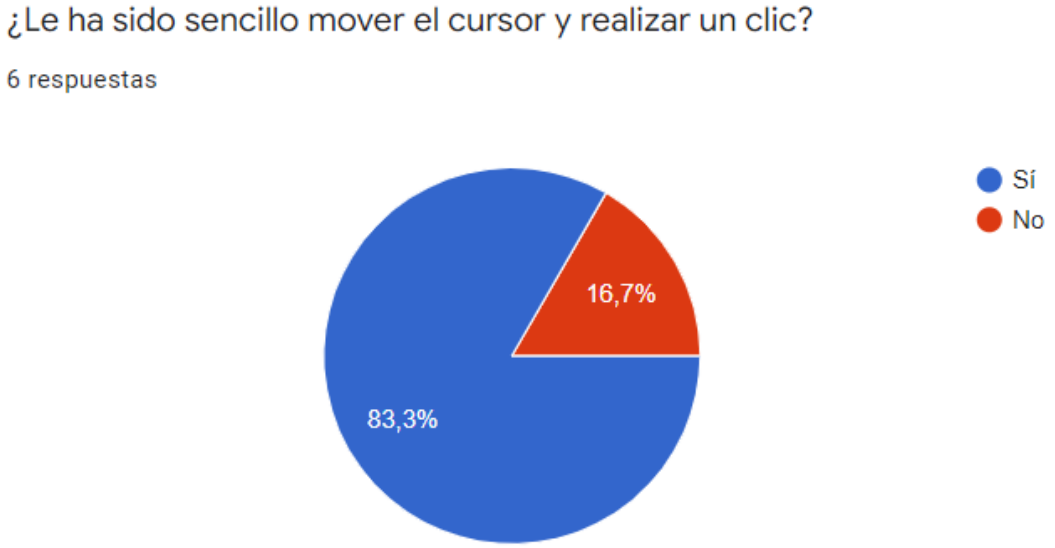


Figura 39 *Respuestas a la pegunta: ¿Le ha sido sencillo mover el cursor y realizar un clic?*

Fuente: [Elaboración sforms]

Á la pregunta de si les parecia sencillo pasar las diferentes diapositivas, volvemos a tener una gran mayoría a los que les resultó sencillo. Únicamente una persona voto que no le había parecido sencillo. Las respuestas las podemos observar en la figura 40.

Figura 40 *Respuestas a la pegunta: ¿Le ha resultado sencillo pasar las diapositivas?*

Fuente: [Elaboración sforms]

En la figura 41, tenemos las respuestas a la pregunta sobre si había sido fácil variar el volumen, dónde obtuvimos un 100% de respuestas afirmativas. Pudimos ver mediante las pruebas como los usuarios si no conseguían llegar al máximo acercaban el gesto hacia la pantalla para poder aumentar el volumen. También se observó durante las pruebas que resultaba ser el gesto más cómodo a realizar y que mejor se entendía.

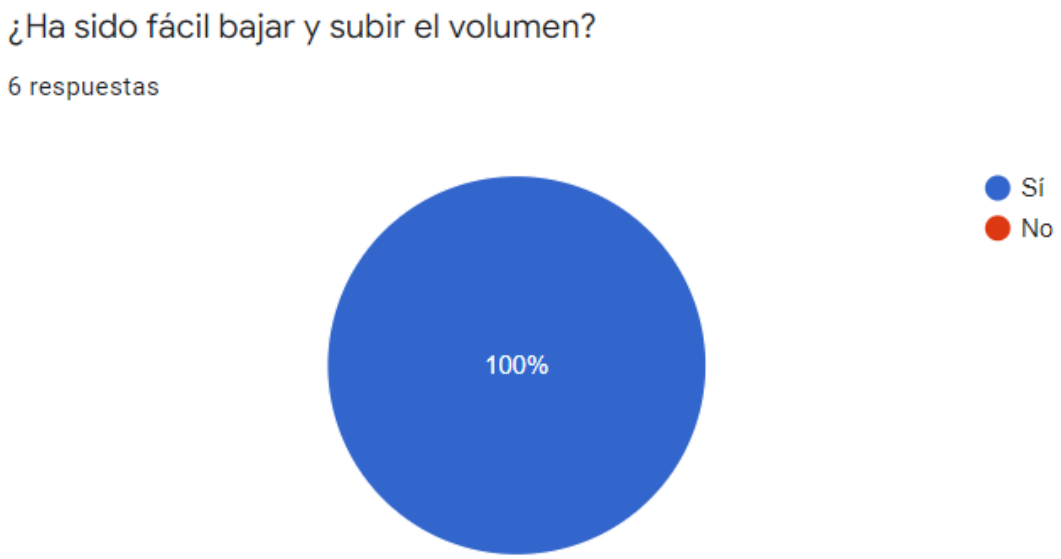


Figura 41 *Respuestas a la pegunta: ¿Ha sido fácil bajar y subir el volumen?*

Fuente: [Elaboración sforms]

Por último en la figura 42 observamos las respuestas a la pregunta que más nos podría marcar el futuro de nuestra aplicación, como de útil es. Igual que anteriormente se trata de una escala Likert, donde el uno significa totalmente inútil y el cinco totalmente útil. Como vemos todas las respuestas se encuentran divididas entre útil y totalmente útil. Como veíamos anteriormente solo dos personas han encontrado algún tipo de dificultad a la hora de interactuar con la presentación, un sujeto con las diapositivas y otro con el movimiento de ratón. Pese a encontrar alguna dificultad todos los votos han dado lugar a que la aplicación puede ser útil, para unas personas un poco más útil que para otras.

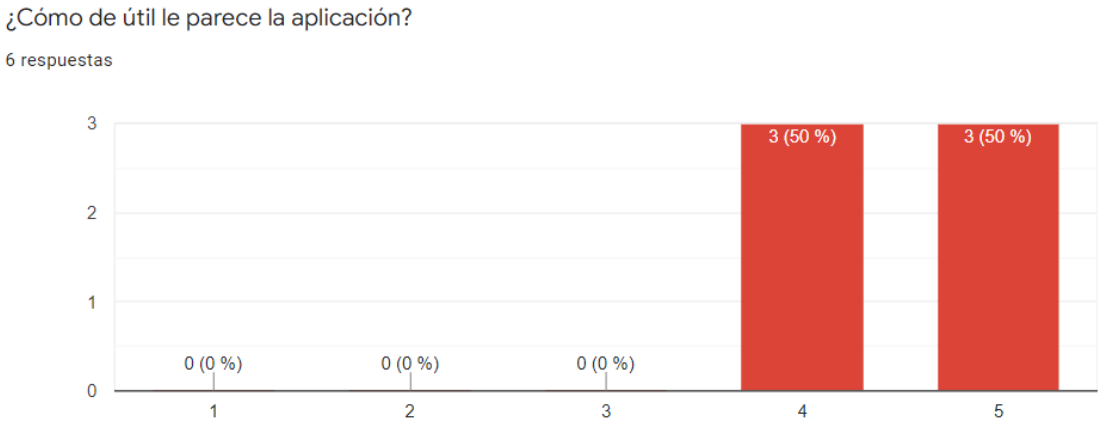


Figura 42 *Respuestas a la pegunta: ¿Cómo de útil le parece la aplicación?*

Fuente: [Elaboración sforms]

Mientras los usuarios realizaban las pruebas estuvimos pendientes de lo costoso que les parecía y si debían de volver a la guía de usuario por olvido de algún gesto. De esta observación extrajimos que las personas de más edad sí que tuvieron que volver a releer la guía de usuario porque el gesto de inicio de presentación se les olvidaba.

Como conclusiones a todas las respuestas que obtuvimos podemos ver como no todos los usuarios les parece algo sencillo el realizar los gestos en el aire, es por esto que no es una aplicación que pueda sustituir el uso del teclado y del ratón sino complementarlo. Pese a que no fuera del todo sencillo realizar algunos gestos comprobamos como a todos los sujetos en una mayor o menor medida les pareció una aplicación lo bastante útil.

Gracias a nuestra observación mientras los usuarios realizaban las pruebas nos dimos cuenta de que utilizarlo en un computador con la cámara en un ángulo más bajo, como sucede con el computador de Huawei, dificulta saber a partir de qué lugar ya es posible que te detecte la mano. Lo bueno es que pese al ángulo en el que se encontraba la cámara no hacía que el gesto fuese más difícil de reconocer.

Además en ambos computadores nos dimos cuenta como mejoraría que al ejecutar la aplicación se quedará la imagen que obtiene la cámara en pequeño en una esquina, algo similar a lo que sucede cuando haces una videoconferencia y queda minimizada tu cámara o el resto.

9. Conclusiones

Todos los objetivos propuestos al inicio de este documento se han ido consiguiendo de manera satisfactoria.

En primer lugar se tuvo que realizar una laboriosa búsqueda de distintas opciones de las que partir para realizar el proyecto. Gracias a esta búsqueda se pudo aprender sobre distintas librerías que utilizan machine learning para poder detectar las manos y sus gestos.

Después de someter las distintas librerías a unas minuciosas pruebas se pudo seleccionar Mediapipe como librería que utilizaríamos para nuestra aplicación a la hora de detectar las manos y sus gestos. Durante estas pruebas tuvimos que aprender acerca de los métodos de las otras librerías para poder ejecutar todas las pruebas.

Mediapipe es una librería muy actual, que fue sacada a la luz por Google hace poco tiempo, esto puede parecer una ventaja ya que es la más novedosa. Pero realmente trae inconvenientes, pues no hay mucha documentación que podamos encontrar en internet acerca de esta. Así que el conseguir los objetivos con esta librería nos podía dar más dificultades por no saber cómo funciona cada método con exactitud.

A la hora de implementar las distintas funcionalidades en el código, tuvimos algunas dificultades ya que era la primera vez que desarrollaba una aplicación. Además las funcionalidades se fueron implementando por separado en diferentes scripts. Esto fue un error pues si directamente lo hubiéramos implementado junto puede que hubiéramos utilizado menos librerías de las que tuvimos que importar finalmente. Este error se debió a la poca experiencia pero servirá para no volver a comértelo en un futuro.

Desde un primer momento nuestro objetivo principal era realizar un prototipo de aplicación con el que poder interactuar con el computador mediante gestos, este objetivo se ha cumplido correctamente. En nuestro caso lo cumplimos haciendo que se interactúe con un programa que sea editor de presentaciones, fue el elegido porque nos pareció muy útil, aunque bien se hubiera podido elegir un juego o cualquier otro programa.

La realización de pruebas de nuestra aplicación con distintos usuarios nos ha dado una visión más amplia de la que podemos tener nosotros mismos al ser los desarrolladores. Nos ha servido para verlo desde otro punto de vista y observar cómo se comportaría un usuario al utilizarlo en su día a día.

Una de las conclusiones claves que obtenemos al utilizar nosotros mismos la aplicación, es que no debe de sustituir esta aplicación al teclado y al ratón. La aplicación se debe de utilizar como complemento al teclado y al ratón. Como hemos podido comprobar, manejar el ratón con unos gestos manteniendo la mano en el aire no es sencillo ni obtenemos una gran precisión. Mientras que el teclado si necesitamos un gesto accesible por todas las personas para cada tecla sería casi imposible. Por ejemplo, si quisiéramos utilizar la aplicación para escribir un documento de texto sería bastante incómodo de realizar además de cansado.

Nuestra aplicación puede ser útil para ciertos usos como en un programa de presentaciones como hemos demostrado en este proyecto y como han ratificado en las pruebas de algunos usuarios. Incluso quizás para programas de visualizar vídeos o fotos, pero no para todos los programas será útil.

9.1 Relación del trabajo desarrollado con los estudios cursados

Pese a que en el grado de ingeniera informática no he estudiado el lenguaje de programación Python, sí que he aprendido Java y C++ que han resultado ser una buena base para el aprendizaje de Python. Tampoco había visto durante la carrera nada acerca de machine learning y mucho menos las librerías de reconocimiento de manos, pero las bases que he ido aprendiendo en los distintos cursos del grado me han hecho capaz de aprender nuevos conocimientos a medida que avanzaba en el trabajo y se iban consiguiendo con éxito cada uno de los hitos planteados en él.

10. Trabajos futuros

Aunque podamos afirmar el éxito del trabajo realizado, ya que se han conseguido todos los objetivos que nos hemos propuesto, siempre se puede ir más allá. En nuestro caso quizás es bastante lógico cómo podemos continuar desarrollando la aplicación, pues nosotros hemos realizado un simple prototipo de aplicación así que podríamos realizar directamente una aplicación.

En nuestro caso el prototipo está pensado directamente para ser utilizado en un editor de presentaciones, pero se podría implementar para varios programas. Sería interesante en un futuro realizar una aplicación que dejará seleccionar distintos programas y según el programa con el que se quiera interactuar mediante gestos, poner unos gestos u otros y adaptar la aplicación a cada programa. Sería interesante que al seleccionar un programa concreto la aplicación ofreciera una guía de usuario específica y después de leer esto el usuario pudiera interactuar con el computador mediante gestos.

Es importante también nombrar una mejora, muy necesaria, que hemos tenido en cuenta al ver a distintos usuarios hacer las pruebas. Para los usuarios era muy difícil saber en qué lugar debían colocar la mano en caso de querer pasar una diapositiva o utilizar el ratón. Esto sucede porque no ven la cámara entonces al no saber la imagen que coge no saben exactamente en qué punto están colocando la mano. Sin embargo si utilizábamos un monitor donde vieran la imagen de la cámara eran capaces de realizar todas las pruebas con éxito al momento. Por lo que una mejora necesaria sería que la imagen de la cámara se quedará en una esquina de la pantalla cuando pases a utilizar un programa, así todos los usuarios serán capaces de ver dónde deben hacer cada gesto.

De esta forma se conseguiría una aplicación muy completa que de acceso a un mayor número de programas con los que interactuar, por lo que su funcionalidad será mayor y sus usos también, además de contar con la facilidad de verse en la imagen.

Esta podría ser una línea futura para seguir con la intención que teníamos al crear esta aplicación, pero es cierto que se podrían seguir distintos caminos. No es necesario seguir con una aplicación que nos permita interactuar con el computador, la investigación que hemos realizado respecto a la detección de manos es muy importante y nos ha abierto nuevos caminos.

Otro posible trabajo futuro sería realizar un estudio de mercado del presente trabajo o de la posible aplicación más concreta. Como hemos investigado en un principio hay muchos dispositivos en el mercado que utilizan una tecnológica parecida, incluso existe en el mercado una aplicación que detecta gestos. Es por esto que sería interesante realizar un estudio sobre la comercialización de la aplicación.

También es posible que no queramos seguir con una aplicación que interactúe con el ordenador pero sí una que detecte los gestos, por ejemplo podría ser interesante realizar una aplicación que lea el lenguaje de signos y según los gestos que se realicen traduzca a un lenguaje normal lo que queremos decir. Esta posibilidad podría ayudar a

una gran parte de la población a aprender lenguaje de signos o que otras personas sean capaces de entenderles.

Existen numerosos caminos hacia los que dirigirse a través de este proyecto, elegir uno de ellos dependerá del fin que se busca conseguir.

11. Bibliografía

[1] *Machine Learning, Deep Learning e Inteligencia Artificial: ¿Qué es y por qué todo el mundo habla de ello?* (s. f.). devAcademy. Recuperado 29 de junio de 2022, de https://www.devacademy.es/machine-learning-deep-learning-e-inteligencia-artificial-mundo-habla-ello

[2] Ai, B. (2021, 15 diciembre). *Árboles de decisión (Práctica) - Bootcamp AI*. Medium. Recuperado 29 de junio de 2022, de https://bootcampai.medium.com/%C3%A1rboles-de-decisi%C3%B3n-pr%C3%A1ctica-62ee5c578b08

[3] *¿Qué es el Deep Learning?* (2019, 10 octubre). SmartPanel. Recuperado 29 de junio de 2022, de https://www.smartpanel.com/que-es-deep-learning/

[4] Heras, J. M. (2019, 28 mayo). *Máquinas de Vectores de Soporte (SVM)*. IArtificial.net. Recuperado 29 de junio de 2022, de https://www.iartificial.net/maquinas-de-vectores-de-soporte-svm/

[5] A, M. (2021, 13 diciembre). *Machine Learning: definición, funcionamiento, usos*. DataScientest.com. Recuperado 29 de junio de 2022, de https://datascientest.com/es/machine-learning-definicion-funcionamiento-usos

[6] *Inteligencia Artificial - ¿Qué es la Inteligencia Artificial?* (s. f.). SumUp. Recuperado 29 de junio de 2022, de https://sumup.es/facturas/glosario/inteligencia-artificial/

[7] P. (2010, 4 junio). *Hemos probado PlayStation Move y nos ha gustado*. Xataka. Recuperado 29 de junio de 2022, de https://www.xataka.com/analisis/hemos-probado-playstation-move-y-nos-ha-gustado

[8] *¿Como Funciona la Wii?* (2020, 25 enero). Electrónica Básica. Recuperado 29 de junio de 2022, de https://electronica-basica.com/wii/

[9] Puerto, K. (2014, 7 octubre). *A Kinect no se le escapa un dedo*. Xataka. Recuperado 29 de junio de 2022, de https://www.xataka.com/videojuegos/a-kinect-no-se-le-escapa-un-dedo

[10] *Prime X*. (s. f.). Manus. Recuperado 29 de junio de 2022, de https://www.manus-meta.com/products/prime-x

[11] *Tracking | Leap Motion Controller | Ultraleap*. (s. f.). Ultraleap. Recuperado 29 de junio de 2022, de https://www.ultraleap.com/product/leap-motion-controller/

[12] *[Mate 40 Series] - Controla tu teléfono con gestos, sin contacto o ¡sólo con la mirada!* (s. f.). consumer.huawei.com. Recuperado 29 de junio de 2022, de https://consumer.huawei.com/es/community/details/Mate-40-Series-Controla-tu-tel%C3%A9fono-con-gestos-sin-contacto-o-%C2%A1s%C3%B3lo-con-la-mirada/topicId_39245/

[13] *Gesture device control - Gestoos*. (s. f.). Gestoos. Recuperado 30 de junio de 2022, de https://www.gestoos.com/applications/gesture-device-control

[14] Sharmaa, A. (2020). *Hand Gesture Recognition using Image Processing and Feature Extraction Techniques*. ScienceDirect. Recuperado 29 de junio de 2022, de https://www.sciencedirect.com/science/article/pii/S187705092031526X

[15] Yee, R. (2019, julio). *Touch Type- Experiments with Google*. Experiments.withgoogle. Recuperado 29 de junio de 2022, de https://experiments.withgoogle.com/touch-type

[16] Campos, P. T. (2020, 24 septiembre). *GestIA: control your computer with your hands - Saturdays.AI*. Medium. Recuperado 29 de junio de 2022, de https://medium.com/saturdays-ai/gestia-control-your-computer-with-your-hands-6bd65dba09b6

[17] Gupta, V. (2018, 8 octubre). *Hand Keypoint Detection using Deep Learning and OpenCV | LearnOpenCV*. LearnOpenCV. Recuperado 29 de junio de 2022, de

https://learnopencv.com/hand-keypoint-detection-using-deep-learning-and-opencv/

[18] Mallick, S. (2016, 6 diciembre). *Histogram of Oriented Gradients explained using OpenCV*. | *LearnOpenCV*. LearnOpenCV. Recuperado 29 de junio de 2022, de https://learnopencv.com/histogram-of-oriented-gradients/

[19] Anwar, T. (2020, 7 septiembre). *Training a custom Object Detector with DLIB and Making Gesture Controlled Applications* | *LearnOpenCV*. LearnOpenCV. Recuperado 29 de junio de 2022, de https://learnopencv.com/training-a-custom-object-detector-with-dlib-making-gesture-controlled-applications/

[20] *Hands*. (s. f.). Mediapipe. Recuperado 29 de junio de 2022, de https://google.github.io/mediapipe/solutions/hands.html

[21] *On-Device, Real-Time Hand Tracking with MediaPipe*. (2019, 19 agosto). Google AI Blog. Recuperado 29 de junio de 2022, de https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html

[22] Campos, P. T. (2020b, septiembre 24). *GitHub - PaburoTC/GestIA*. GitHub. Recuperado 29 de junio de 2022, de https://github.com/PaburoTC/GestIA

[23] Mallick, S. (2016b, diciembre 6). *learnopencv/Training_a_custom_hand_detector_with_dlib/spmallick/learnopencv*. GitHub. Recuperado 29 de junio de 2022, de https://github.com/spmallick/learnopencv/tree/6c11af621e2ef358f6307ec001b3fab93baceba4/Training_a_custom_hand_detector_with_dlib

[24] Mallick, S. (2022, 29 junio). *learnopencv/HandPose at master · spmallick/learnopencv*. GitHub. Recuperado 29 de junio de 2022, de https://github.com/spmallick/learnopencv/tree/master/HandPose

[25] Python. (2020, 23 marzo). *keyboard*. PyPI.Org. Recuperado 30 de junio de 2022, de https://pypi.org/project/keyboard/

[26] Python. (s. f.). *json — Codificador y decodificador JSON — documentación de Python - 3.10.5*. Python.org. Recuperado 30 de junio de 2022, de https://docs.python.org/es/3/library/json.html

[27] NumPy. (s. f.). *NumPy*. Recuperado 30 de junio de 2022, de https://numpy.org/

[28] PyAutoGUI. (s. f.). *Welcome to PyAutoGUI's documentation! — PyAutoGUI documentation*. Recuperado 30 de junio de 2022, de https://pyautogui.readthedocs.io/en/latest/index.html

[29] DLIB. (s. f.). *dlib Library*. Recuperado 30 de junio de 2022, de http://dlib.net/

[30] OpenCV. (s. f.). *OpenCV*. Recuperado 30 de junio de 2022, de https://opencv.org/

12. Anexos

12.1 Anexo I: Archivo “gesture.py”

Imports

```
from math import hypot

import numpy as np

import cv2
```

Function

```
def simpleGesture(fingers,p1,p2):

    f_list = ['FIST!','ONE!','TWO!','THREE!','FOUR!','FIVE!']

    if fingers[0]==False and fingers[1]==True and fingers[2]==False and
fingers[3]==False and fingers[4]==True:

        return 'ROCK!'

    elif fingers[0]==True and fingers[1]==True and fingers[2]==False and
fingers[3]==False and fingers[4]==True:

        return 'SPIDERMAN!'

    elif fingers[0]==False and fingers[1]==True and fingers[2]==True and
fingers[3]==False and fingers[4]==False:

        return 'PEACE!'

    elif fingers[0]==True and fingers[1]==False and fingers[2]==False and
fingers[3]==False and fingers[4]==True:

        return 'TELEPHONE!'

    elif fingers[0]==True and fingers[1]==False and fingers[2]==False and
fingers[3]==False and fingers[4]==False:

        return 'Pulgar arriba -> OK'

    else:

        return f_list[fingers.count(True)]
```

12.2 Anexo II: Respuestas a la encuesta del sujeto 1

EDAD *

59

¿Suele utilizar en su día a día un computador? *

- ● Sí
- ○ No

¿Alguna vez ha tenido que realizar una presentación? *

- ● Sí
- ○ No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	●	○

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ● Sí
- ○ No

¿Le ha resultado sencillo pasar las diapositivas? *

- ● Sí
- ○ No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil					Totalmente útil
	○	○	○	●	○

12.3 Anexo III: Respuestas a la encuesta del sujeto 2

EDAD *

53

¿Suele utilizar en su día a día un computador? *

- ○ Sí
- ● No

¿Alguna vez ha tenido que realizar una presentación? *

- ○ Sí
- ● No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	○	●

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ○ Sí
- ● No

¿Le ha resultado sencillo pasar las diapositivas? *

- ● Sí
- ○ No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil					Totalmente útil
	○	○	○	●	○

12.4 Anexo IV: Respuestas a la encuesta del sujeto 3

EDAD *

33

¿Suele utilizar en su día a día un computador? *

- ● Sí
- ○ No

¿Alguna vez ha tenido que realizar una presentación? *

- ● Sí
- ○ No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	○	●

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ● Sí
- ○ No

¿Le ha resultado sencillo pasar las diapositivas? *

- ● Sí
- ○ No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil					Totalmente útil
	○	○	○	○	●

12.5 Anexo V: Respuestas a la encuesta del sujeto 4

EDAD *

28

¿Suele utilizar en su día a día un computador? *

- ● Sí
- ○ No

¿Alguna vez ha tenido que realizar una presentación? *

- ● Sí
- ○ No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	●	○

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ● Sí
- ○ No

¿Le ha resultado sencillo pasar las diapositivas? *

- ● Sí
- ○ No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil	○	○	○	○	●	Totalmente útil

12.6 Anexo VI: Respuestas a la encuesta del sujeto 5

EDAD *

35

¿Suele utilizar en su día a día un computador? *

- ● Sí
- ○ No

¿Alguna vez ha tenido que realizar una presentación? *

- ● Sí
- ○ No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	●	○

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ● Sí
- ○ No

¿Le ha resultado sencillo pasar las diapositivas? *

- ○ Sí
- ● No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil					Totalmente útil
	○	○	○	●	○

12.7 Anexo VII: Respuestas a la encuesta del sujeto 6

EDAD *

61

¿Suele utilizar en su día a día un computador? *

- ● Sí
- ○ No

¿Alguna vez ha tenido que realizar una presentación? *

- ● Sí
- ○ No

Ha sido fácil usar la aplicación *

Totalmente desacuerdo					Totalmente de acuerdo
	○	○	○	○	●

¿Le ha sido sencillo mover el cursor y realizar un clic? *

- ● Sí
- ○ No

¿Le ha resultado sencillo pasar las diapositivas? *

- ● Sí
- ○ No

¿Ha sido fácil bajar y subir el volumen? *

- ● Sí
- ○ No

¿Cómo de útil le parece la aplicación? *

Totalmente inútil					Totalmente útil
	○	○	○	○	●

12.8 Anexo VIII: Objetivos de Desarrollo Sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades**.			X	
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Los Objetivos de Desarrollo Sostenible que se pueden relacionar con el proyecto presentado son tres. En primer lugar el objetivo que más se vincula a nuestro proyecto se trata del objetivo número nueve, industria, innovación e infraestructuras. Durante todo nuestro proyecto hemos realizado una investigación acerca de la forma en que se puede interactuar con un computador hoy en día. De esta investigación hemos extraído que las formas más comunes son mediante dispositivos periféricos de entrada, como el ratón o el teclado. Nuestro proyecto trata de crear una aplicación que innove la forma de comunicarnos con el computador, en este caso se busca interactuar mediante gestos realizados en el aire. Se trata de una tecnología que puede no parecer muy innovadora pues algunas videoconsolas la implementan, pero para ello necesitan del uso de un dispositivo extra. En nuestro caso se implementa en un computador y únicamente utilizando la cámara, un hecho que lo hace un poco más innovador. Realmente lo más

novedoso sería el poder interactuar con el computador mediante gestos, ya que no se trataría de una aplicación única como puede ser solo en un juego, sino que se busca la interacción de una forma generalizada.

Por otro lado, otro de los objetivos que podemos relacionar con el proyecto, aunque en menor medida que el anterior, es respecto al objetivo de salud y bienestar. Hoy en día existen muchas personas con problemas dermatológicos, que les impiden tocar cosas o que al tocarlas les puede salir alguna herida. Además como hemos aprendido recientemente hay muchas enfermedades que se transmiten por contacto, como paso con la pandemia de 2019, el contacto de una persona infectada con un dispositivo podía hacer que una persona sana se contagiará al tocar el dispositivo. Nuestra aplicación puede ser útil para las personas con problemas dermatológicos, ya que no es necesario el contacto con ningún periférico. Además puede ser útil para computadores que se utilizan por muchas personas o cualquier dispositivo al que se le pueda agregar una cámara para que no sea necesario que todo el mundo tenga contacto con este.

Por último, otro de los objetivos con una pequeña relación con nuestro proyecto es el de la reducción de las igualdades. Debemos de tener en cuenta que nuestro proyecto se trata de una aplicación disponible para todas las personas. Hoy por hoy en la mayoría de oficios necesitan de un ordenador para realizar las tareas, para comunicarnos con los computadores siempre hemos utilizado el teclado y el ratón. Pero hay que tener en cuenta que hay personas que pueden tener algún problema de movilidad y puede que mover el ratón, hacer un clic o el poder presionar únicamente una tecla les sea muy difícil. La dificultad de estas personas al utilizar los periféricos a los que estamos habituados puede suponer una desigualdad, pues puede que necesiten más tiempo para realizar alguna tarea que conlleve un movimiento que les es difícil realizar y acaben prefiriendo contratar a otra persona que no tenga estas dificultades. Si personas que tuvieran ciertos problemas de movilidad pudieran utilizar nuestra aplicación para interactuar con el computador, posiblemente se reduciría una desigualdad, pues podrían hacer el mismo trabajo sin que les suponga una dificultad añadida.