Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

MANEJO DE UN DRON MEDIANTE POSES

TÍTULO DEL TFG: Manejo de un dron mediante poses

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: David Sánchez Cozar

DIRECTOR: Miguel Valero García

FECHA: 6 de septiembre de 2023

RESUMEN

Este proyecto constituye el **Trabajo de Fin de Grado (TFG)** en Ingeniería en Sistemas de Telecomunicación. El objetivo principal ha sido desarrollar un sistema de control para un dron mediante el **reconocimiento de poses** captadas a través de una cámara, ya sea mediante gestos de las manos, posiciones corporales o movimientos de las extremidades.

El proyecto se ha fundamentado en la detección de puntos clave del cuerpo humano. Utilizando algoritmos y scripts en **Big Data**, se logró identificar con precisión las manos y el cuerpo de cualquier persona en diversas situaciones. Esto permitió programar un sistema que compara la posición de los puntos clave y determina la pose específica, lo que desencadena diferentes movimientos del dron en función de los gestos detectados.

Además, en el proyecto han incluido elementos interactivos, como un sistema de puntuaciones visibles en pantalla que se incrementa en **tiempo real** según las interacciones con el dron. Para enriquecer la experiencia del usuario, también se han incorporado efectos sonoros y música, aportando un componente lúdico e inmersivo al manejo del dron.

OVERVIEW

This project constitutes the **Final Degree Project (TFG)** in Telecommunication Systems Engineering. The main objective was to develop a control system for a drone using **pose recognition** captured through a camera, whether through hand gestures, body positions, or limb movements.

The project was based on detecting key points of the human body. By using algorithms and **Big Data** scripts, it was possible to accurately identify the hands and body of any person in various situations. This enabled the programming of a system that compares the position of the key points and determines the specific pose, triggering different drone movements based on the detected gestures.

Additionally, the project included interactive elements, such as an on-screen scoring system that updated in **real-time** based on interactions with the drone. To enrich the user experience, sound effects and music were also incorporated, adding a playful and immersive component to controlling the drone.

ÍNDICE

Capítulo 1. Introducción

En este apartado se pretende poner en situación general al lector del trabajo.

1.1 Qué he hecho

He contribuido al avance y a la mejora de un espectáculo realizado con drones que pretende ser una herramienta de ocio. Toda mejora ha sido realizada llevando a cabo ideas mediante programación en el ordenador.

Esto ha consistido en una aplicación que permite controlar un dron Tello mediante posturas con las manos y con el cuerpo. Más adelante se explica con detalle.

1.2 Por qué es importante

Hoy en día la imagen que nos viene a la cabeza cuando oímos la palabra dron es un tipo de dron utilizado como medio en una guerra o conflicto.

Es por eso por lo que normalmente tenemos una concepción negativa de este tipo de vehículos aéreos no tripulados.

Este trabajo pretende ser una herramienta para hacer ver que los drones tienen aplicaciones que aportan muchas cosas y muy buenas a la sociedad y no solo son un arma usada en las guerras.

La percepción negativa hacia los drones puede ser atribuida a varios factores, entre ellos:

- La privacidad: Existe preocupación en relación con la invasión de la privacidad debido a la capacidad de los drones para capturar imágenes y videos desde el aire. Esto ha generado debates en torno al uso responsable de los drones y la necesidad de establecer regulaciones adecuadas para proteger la privacidad de las personas

- La seguridad: Ha habido informes de incidentes en los que los drones han representado un riesgo para la seguridad, como vuelos no autorizados cerca de aeropuertos o zonas restringidas. Estos incidentes han aumentado las preocupaciones sobre la seguridad aérea y han llevado a una mayor regulación en muchos países.

A pesar de estas preocupaciones, los drones también tienen numerosas utilidades y beneficios. Algunas de las aplicaciones más comunes son:

- Fotografía y videografía aérea: Los drones permiten capturar imágenes y videos desde perspectivas aéreas únicas, lo que ha revolucionado la industria de la fotografía y el cine. Se utilizan en la producción audiovisual, la filmación de eventos, la promoción turística y mucho más.

- Entrega de paquetes: Empresas como Amazon están experimentando con el uso de drones para entregar paquetes de manera más eficiente. Esto podría reducir los tiempos de entrega y mejorar la logística en general.

- Monitoreo y mapeo: Los drones equipados con cámaras y sensores pueden utilizarse para monitorear áreas remotas, vigilar la vida silvestre, realizar inspecciones de infraestructuras y generar mapas detallados del terreno.

- Agricultura de precisión: Los drones pueden proporcionar información valiosa sobre los cultivos, permitiendo a los agricultores optimizar la siembra, la irrigación y el uso de fertilizantes. Esto puede conducir a una mayor eficiencia y una reducción en el consumo de recursos

- Búsqueda y rescate: Los drones pueden utilizarse en operaciones de búsqueda y rescate, ya que pueden cubrir grandes áreas rápidamente y proporcionar información en tiempo real sobre la ubicación de personas en peligro.

- Combate: Como se menciona anteriormente los drones pueden ser utilizados en la guerra como un arma poderosa. Esto se debe a que no llevan tripulación y en caso de ser destruidos no causan daño humano como podría pasar con un avión de combate.

Observando que los drones pueden ofrecer muchas cosas y muy buenas, en este trabajo se pretende añadir una más a esta larga lista: el ocio. Es por eso que este trabajo se hace tan interesante, intenta cambiar la perspectiva respecto a estos vehículos aéreos no tripulados.

1.3 Esto se inscribe en el Circo de drones

Así se llama: Circo de Drones. Y no porqué se haya decidido ahora, ya hay otros compañeros de carrera que han estado trabajando en el proyecto que he pretendido evolucionar.

Este circo consiste en varios números de interacción con los usuarios del público que intentarán resolver los diferentes retos planteados en distintos modos de juego.

El “Dron Lab”, que con este nombre nos referimos al circo de drones que se sitúa en la EETAC de Castelldefels, atrae a un número elevado de gente que se interesa en visitar las instalaciones y en comprobar cómo funciona este laboratorio experimental de drones.

La idea es que haya juegos de diferentes dificultades y niveles, aunque todos muy asequibles. Que sean todos para pasar un buen rato en grupo. En todos los modos de juego se utiliza el dron

1.4 Trabajo en equipo

Hasta ahora solo hablaba en singular en referencia al autor de este trabajo. Pues bien, cabe comentar que esta evolución en el circo de drones la he llevado a cabo con mi compañero y amigo Víctor González a punto de finalizar como yo el grado en ingeniería de Sistemas de Telecomunicaciones.

Al ser un trabajo en pareja, aunque él redactará también su memoria habrá partes que serán idénticas, algunas muy parecidas y otras en las que nos diferenciaremos, aunque estas últimas serán pocas. Pues hemos trabajado bastante unidos y en mini proyectos dentro de este trabajo muy similares.

En una parte más avanzada de la memoria se explica las tareas a las que se ha dedicado más mi compañero y se diferencian de las que me he dedicado yo más. En cualquier caso, siempre que uno necesitaba algo de ayuda en su tarea nos hemos podido ayudar.

1.5 Estructura de la memoria

Este trabajo empieza tratando el protagonista del circo: el dron que usamos. El siguiente apartado habla sobre la base que partimos. Este trabajo como antes hemos mencionado no se ha iniciado de cero, tiene otro trabajo de base a partir del cual se progresa y evoluciona. Parte del trabajo es hacer ver la base de partida de este.

Seguidamente se trata la planificación respecto al tiempo que se ha planteado para llevar a cabo este trabajo (GANTT) y se explican las tecnologías y herramientas necesarias para llevar a cabo todos los progresos realizados.
A continuación, se muestra cómo queda la versión final de la aplicación, es decir todas las posibilidades que se le ofrecen al usuario en la pantalla del ordenador y qué significan las diferentes opciones.

Finalmente se muestran las pruebas realizadas para garantizar un mínimo de calidad en todo lo realizado, algunas evaluaciones externas y las conclusiones finales extraídas.

Capítulo 2. Cómo es el Tello

En este capítulo se describen las características del hardware del dron Tello así cómo la librería característica que permite realizar acciones básicas con este tipo de dron.

Para poder entender que tipo de espectáculo se puede ofrecer al cliente es necesario saber qué tipo de dron se utiliza y cómo se puede usar.

Esta es una breve descripción del dron que se usa en este trabajo. El dron Tello EDU es un dron que puede programarse a gusto del programador, en este caso a nuestro gusto y con el lenguaje Python. Es considerado ultraligero por su reducido peso y puede alcanzar hasta los 30 metros de altura volando. Es capaz de realizar todo tipo de acrobacias.

El mismo dron presenta una aplicación que hace que se pueda controlar de manera sencilla de manera que el control del dron sea total e incluso semi automático.

A continuación, en la **Fig 2.1**, se muestran las especificaciones técnicas del dron Tello EDU, seguido de una breve descripción de su kit de desarrollo de software (SDK) (Panel de leds). Por último, se menciona el Dron Robomaster TT el cual es una versión mejorada del Tello EDU como se verá en su descripción.

Aeronave	Peso	80g con hélices y batería
	Dimensiones	98x92.2x41 mm
	Hélice	3 pulgadas
	Funciones integradas	Sensor telemétrico
		Barómetro led
		Sistema de visión
		Wi-Fi 2.4 GHz 802.11n
		Vista en directo a 720p
Rendimiento de vuelo	Distancia máx. de vuelo	100 m
	Velocidad máx.	10 m/s
	Tiempo máx. de vuelo	13 min
	Altitud máx. de vuelo	30 m
Batería	Batería desmontable	1.1 Ah / 2.8 V
Cámara	Foto 5 MP	(2592x1936)
	Campo de visión	82.6 °
	Vídeo	HD 720p 30 fps
	Formato	JPG (foto); MP4 (vídeo)
	Estabilización electrónica	Sí

Fig. 2.1 Características del dron

El Tello EDU viene con una serie de sensores integrados, incluyendo un sensor de vuelo y un sensor de visión que le permite volar de manera estable y evitar obstáculos. Además, cuenta con una cámara HD que captura imágenes y vídeos de alta calidad, lo que permite realizar proyectos de fotografía y videografía aérea.

Fig. 2.2 Dron Tello

En esta imagen, la **Fig 2.2**, apreciamos el dron que se utiliza en este proyecto. Es un dron de 4 hélices, diseñado para interior. Es decir, si el dron te atropella o atropella un obstáculo no hay de qué preocuparse.

Este dron ofrece la posibilidad de incorporar un adaptador de sensores, para conectarse a módulos externos como por ejemplo una matriz 8x8 de leds que se puede acoplar sobre el mismo dron el cual proporciona una posibilidad de experimentos mucho más amplia al incluir variantes con leds.

Esta posibilidad obtiene el nombre de Robomaster TT.

El dron Robomaster TT tiene el aspecto mostrado en la **Fig 2.3.**

Fig. 2.3 Dron Tello Robomaster TT

El paso cero para iniciar este proyecto es conectar el dron con el ordenador correspondiente y así hacer posible la interacción.

Para conectar el dron Tello EDU a un ordenador, necesitarás seguir los siguientes pasos:

Asegúrate de que tanto el dron Tello EDU como el ordenador están encendidos y con suficiente batería (la batería de este dron dura poco, es recomendable tenerlas todas las que no se utilicen cargando)

En tu ordenador, asegúrate de tener una conexión Wi-Fi disponible.

Busca la red Wi-Fi emitida por el dron Tello EDU. Por defecto, el nombre de la red debería ser algo similar a "TELLO-XXXXXX" como se muestra en la **Fig 2.4**. Conéctate a esta red Wi-Fi desde tu ordenador.

Fig. 2.4 Conexión Wi-Fi Dron

Este tipo de conexión no tiene interacción posible con el panel de leds, para hacerlo posible hay que conectarse a otra red wifi. Nos situamos en la lista de conexiones Wi-Fi disponibles. Se selecciona la red con las letras “RMTT” al principio del nombre como se muestra en la **Fig 2.5**. Esto establece la conexión con el panel también y se podrá programar lo que aparezca en este.

Fig 2.5 Conexión WiFi dron con Leds

Una vez mostradas las características hardware del dron y cómo conectarse al mismo nos centramos en la librería que nos permite realizar operaciones básicas. La librería “TelloPy” para Python.

TelloPy es una librería de Python desarrollada específicamente para interactuar con el dron DJI Tello. El DJI Tello es un dron pequeño y asequible diseñado para propósitos educativos y recreativos. TelloPy proporciona una interfaz de programación que permite controlar el dron y recibir información de vuelo en tiempo real.

Con TelloPy, puedes escribir scripts en Python para comunicarte con el dron y enviarle comandos. Algunas de las funciones principales de la librería incluyen:

Control de vuelo: TelloPy te permite enviar comandos al dron para controlar su vuelo. Puedes programar el despegue, aterrizaje, movimientos en diferentes direcciones (adelante, atrás, izquierda, derecha), giros y ajustes de altitud.

Captura de imágenes y videos: Puedes utilizar TelloPy para capturar imágenes y grabar videos desde el dron. Esto te permite realizar proyectos de fotografía y videografía aérea utilizando Python.

Recepción de datos de vuelo: TelloPy proporciona información en tiempo real sobre el estado del dron. Puedes recibir datos como la altura actual, la velocidad, el nivel de batería y la velocidad de vuelo.

Manejo de eventos: La librería permite manejar eventos generados por el dron, como notificaciones de estado o respuestas a comandos enviados. Esto te permite programar acciones específicas basadas en los eventos del dron.

TelloPy es una herramienta útil para desarrolladores y entusiastas que desean explorar la programación de drones y realizar proyectos con el DJI Tello utilizando Python. La librería simplifica la comunicación con el dron y proporciona una interfaz fácil de usar para controlarlo y recibir datos de vuelo en tiempo real.

Un ejemplo de código en el que se muestran las operaciones básicas con el dron mediante la librería TelloPy es el siguiente:

```
“import tellopy
import time

def main():
    drone = tellopy.Tello()     # Crea una instancia de la clase Tello
    try:
        drone.connect()          # Conecta al dron Tello
        drone.start_video()    # Inicia la transmisión de video
        drone.takeoff()          # Despegue
        time.sleep(5)          #  El dron despegará y se  mantendrá en el aire
durante 5 segundos


        drone.land()           # Aterriza
    except Exception as e:
        print(f"Error: {e}")
    finally:
        drone.quit()  # desconexión del dron “
```

En este trozo de código se muestran algunas de las operaciones básicas que se pueden realizar con esta librería como por ejemplo las de despegar o aterrizar el dron.

Capítulo 3. Base del circo de drones y herramientas usadas

Este trabajo tiene la base en el trabajo de Jonathan Palacios titulado “Guía para programar un circo de drones”. En este capítulo se muestra la base de la que parte el circo de drones y las herramientas que se usan para implementar

El trabajo de base muestra el aspecto mostrado en la **Fig 3.1**

Fig. 3.1 Portada circo de drones

Una vez el usuario entra en el menú tiene varios modos de juego como se muestra en la **Fig 3.2**

Fig. 3.2 Modos de juego

Todos los modos de juego tienen una base común. El usuario se sienta delante de una cámara, puede ser la del dron (en el modo “Follow Me”) o la del ordenador (resto de modos) y con diferentes posturas o movimientos previamente indicados en pantalla va controlando el dron.

En este trabajo se tratan por encima dos modos de juego en los que no nos hemos focalizado para mejorar.

Estos dos modos son el “Follow me” y el “control caras”.

El modo “Follow me” consiste en que el dron con la cámara que tiene incorporada detecta a la persona y la sigue.

La descripción genérica de este modo de juego es la siguiente;

El modo "Follow Me" utiliza tecnologías como el GPS y la detección de objetos para rastrear y seguir al objetivo designado. Aquí se muestra una descripción general de cómo funciona:

Preparación: Antes de utilizar el modo "Follow Me", debes asegurarte de que el dron tenga una señal GPS sólida y esté conectado al control remoto o a la aplicación móvil. También debes activar el modo "Follow Me" en la aplicación o en la configuración del dron.

Selección del objetivo: En la mayoría de los casos, puedes seleccionar el objetivo que deseas seguir directamente en la pantalla de tu dispositivo móvil o en el control remoto del dron. Puedes elegir seguir una persona o un objeto, y el dron ajustará automáticamente su velocidad y dirección para mantenerse cerca del objetivo.

Inicio del seguimiento: Una vez que hayas seleccionado el objetivo, puedes iniciar el modo "Follow Me". El dron despegará automáticamente y se posicionará en el aire a una altura predeterminada.

Seguimiento: A medida que te muevas o el objeto se desplace, el dron utilizará los datos del GPS y la información de los sensores para ajustar su posición y orientación. Mantendrá una distancia y altura constante con respecto al objetivo, siguiéndolo de manera suave y precisa.

Funciones adicionales: Algunos drones ofrecen características adicionales en el modo "Follow Me". Por ejemplo, pueden permitirte establecer un punto de interés para que el dron orbite alrededor de ti mientras te sigue. También pueden incluir modos de vuelo predefinidos, como el seguimiento desde atrás o el seguimiento en espiral, que agregan variedad y creatividad a tus tomas.

Esto ha sido adaptado a este dron que no tiene funciones GPS pero que como se ha comentado si que es capaz de seguir a una persona con su cámara.

En la **Fig 3.3** se muestra alguna imagen tomada del trabajo anterior que muestra el funcionamiento de este modo de juego:

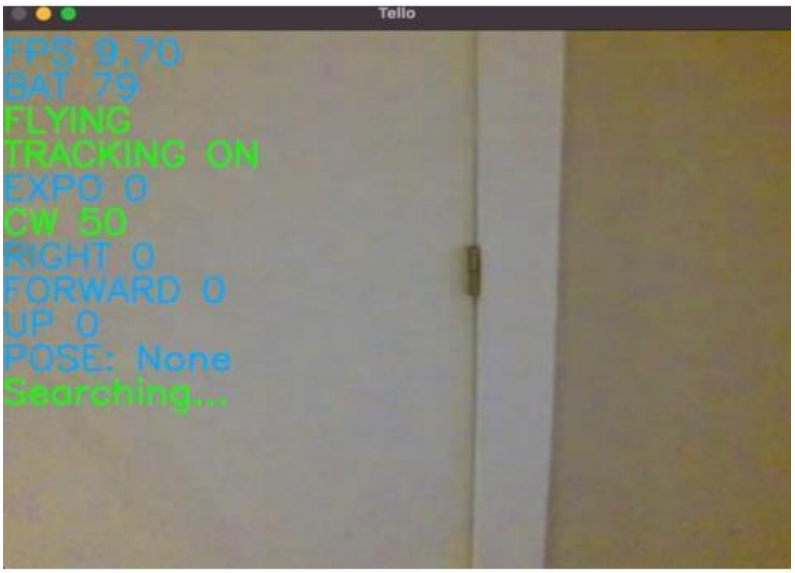

Fig. 3.3 Cámara Dron

En esta imagen extraída del trabajo mencionado del que partimos se muestra la cámara del dron en busca de una cara para que, haciendo poses, pueda dirigirlo.

El segundo modo de juego en el que tampoco se avanza en este proyecto pero que cabe comentar es el “control caras”.

Este método consiste en guiar al dron mediante diferentes posturas con la cara. La diferencia con el “Follow me” es que en este modo de juego se utiliza la cámara del ordenador.

Existen unas poses ya establecidas con la cara como por ejemplo las que se muestran en la **Fig. 3.4**

Fig. 3.4 Poses Control cara

El modo de juego consiste en copiar las poses deseadas para mover el dron. La cámara del ordenador detecta que la pose que hace el usuario es parecida o casi igual a la que tiene registrada (con herramientas como Mediapipe), esa pose significa un tipo de movimiento, por ejemplo, despegar, y el ordenador envía la orden al dron. El dron despega.

Este modo de juego es importante ya que los modos de juego en los que este trabajo evoluciona son muy similares.

Para empezar el modo “Control dedos”. Este modo de juego del que partimos consiste en controlar el dron mediante unas poses con las manos ya preestablecidas. El usuario se coloca delante de la cámara del ordenador que está conectado al dron como anteriormente hemos mencionado. Ahí, en el mismo ordenador puede consultar las poses que hay ya establecidas, en este caso son las que se pueden observar en la **Fig 3.5**

Fig. 3.5 Poses Control dedos

Iniciando el modo de juego y por tanto la cámara del ordenador que gracias a opencv y mediapipe como antes hemos mencionado es capaz de identificar poses (también comparando las poses de la cámara con las preestablecidas, esto se consigue programando la comparación), el usuario puede guiar al dron a su gusto.

Por último, mencionar el modo de juego “control poses”. Es completamente igual al modo de juego anterior, pero se controla el dron con poses utilizando los puntos de todo el cuerpo. Las poses que se observan en la **Fig 3.6** son aquellas ya establecidas que el usuario intentará imitar:

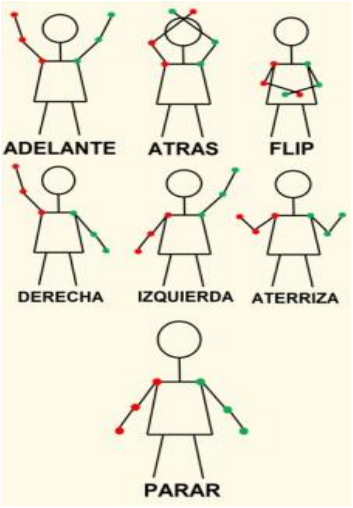

Fig. 3.6 Control pose

Este trabajo tiene las bases ya explicadas. El área de mejora que se explota consiste en evolucionar los modos de juego “control dedos” y “control pose

Capítulo 4. Objetivos y plan de trabajo

En este capítulo se exponen los objetivos que se pretenden alcanzar al final del trabajo, así como las diferentes tareas que se han ido llevando a cabo para alcanzarlos.

4.1 Objetivos

Nuestros objetivos comunes eran evolucionar en el circo de drones. Es decir, a partir de lo que ya se conocía como el circo de drones, mejorarlo todo lo que pudiéramos.

Este objetivo es muy amplio y es por eso que nos hemos ido marcando objetivos a más corto plazo. Estos objetivos han sido:

- Crear modos de juego con diferentes dificultades a partir de los ya creados
- Permitir al usuario introducir las poses con las que quiere jugar y no establecerlas como predeterminadas
- Mejorar el método de detección de poses para alcanzar mayor robustez
- Crear nuevos modos de juego
- Crear interfaces de las aplicaciones de modo de juego libre y modo de juego guiado
- Finalizar el trabajo: memoria, documentación de código, demostraciones y presentación

4.2 Plan de trabajo

Estos objetivos los hemos ido llevando a cabo realizando tareas prácticamente semanales.
Estas tareas las mostramos reflejadas en el diagrama de GANTT de la **Fig 4.1.**

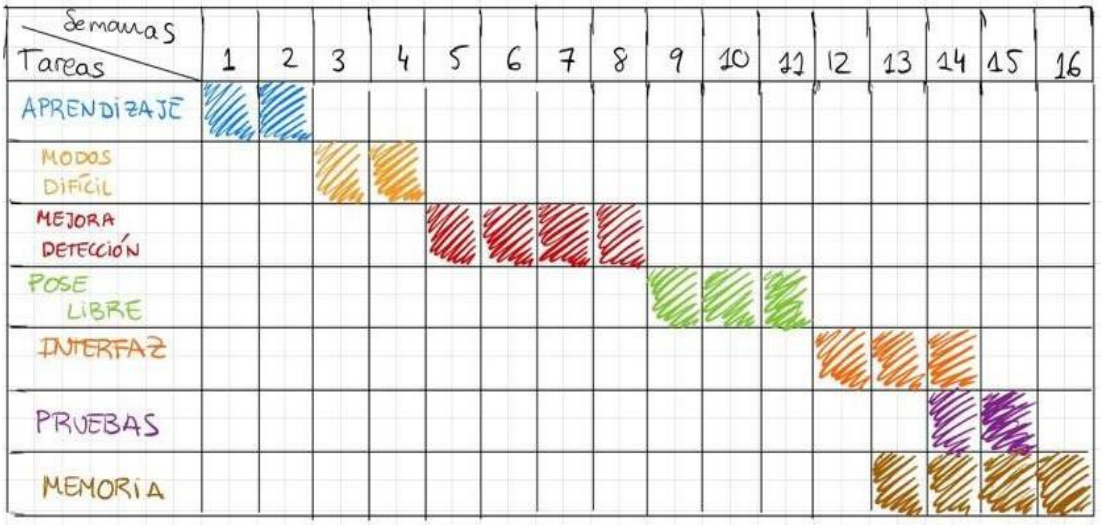

Fig. 4.1 Diagrama de Gantt

Como observamos las primeras dos semanas consistieron en una fase de aprendizaje.

El objetivo de esta primera fase del trabajo es aprender a usar las herramientas básicas y familiarizarse con el trabajo que se toma como base, el mencionado anteriormente de Jonathan Palacios.

Para ello fue necesario instalar Python versión 3.7 y Pycharm (community edition) que tiene un aspecto de aplicación como el mostrado en la **Fig 4.2**

Fig. 4.2 PyCharm Community Edition

Una vez realizado esto el objetivo era, mediante tutoriales de YouTube (algunos proporcionados por nuestro tutor) crear una aplicación muy básica de Python con Pycharm.

Para la interfaz gráfica de esta aplicación teníamos dos opciones: Tkinter y CustomTkinter que más tarde se explican con detalle. Yo me centré en Tkinter mientras mi compañero Víctor realizó el estudio de CustomTkinter.

Las dos siguientes semanas, como observamos en el diagrama tenemos una fase en la que nos dedicamos a generar un modo difícil en dos de los modos de juego que ya estaban realizados: “Control dedos” y “Control poses”. El método que utilizaba Jonnathan Palacios era muy sencillo ya que las poses solo se podían programar antes de estar en la interfaz, por tanto, solo detectaba las poses que estuvieran programadas, no se podían añadir nuevas. Apenas utilizamos el código de Jonnathan ya que, con un tutorial de YouTube, aprendí a sacar los puntos de las manos de otra manera y a partir de ahí ya los saqué así en todo el proyecto.

Las siguientes semanas entramos en una fase de mejora en la detección de poses ya que para hacer un modo lo más complicado posible, es decir, que el usuario pueda hacer la pose que quiera y sea detectada, se necesita un mecanismo de detección de poses robusto.

Para empezar, necesitamos mejorar el problema de que la distancia de la pose con la cámara del ordenador nos dé fallos. Es decir, interesa que

independientemente de la distancia con la cámara del usuario la posea sea detectada.

En la mejora del modo de detección de poses hay dos caminos. Al principio, trabajamos ambos en un algoritmo basado en la distancia de los diferentes puntos de la mano. Avanzamos un poco con esta idea hasta que tuvimos ya un pequeño programa que funcionaba, entonces se lo mostramos a nuestro tutor y él nos propuso otro camino, otra idea: un algoritmo basado en rectángulos. En cuanto Miguel nos propuso esta idea, decidimos dividirnos. Víctor continuaría mejorando el algoritmo de las distancias y yo empezaría a trabajar en el algoritmo de los rectángulos. Finalmente, nos decantamos con trabajar con el algoritmo de los rectángulos ya que daba mejores resultados, sobre todo con poses en que los puntos estuvieran muy cerca o incluso se solaparan.

La siguiente fase consiste en poder hacer que el usuario pueda poner en el juego la pose que quiera. Es decir que pueda ser capaz de ajustar que tal pose significa que el dron se desplace hacia la derecha, por ejemplo. En esta parte se centró más Víctor y consiste en hacer un menú donde el usuario vaya registrando las poses y las pueda cambiar si necesita. Además de que estas poses funcionen en el juego es decir sean detectadas.

Todas estas fases de mejora y experimentales se llevan a cabo con el modo de juego “Control dedos” y con una sola mano. Es por eso que una vez tenemos ya el método de detección afinado, entramos en una fase de expansión al juego con dos manos y con poses de todo el cuerpo. Esta parte la realicé más yo.

A continuación, entramos en una fase de juntar todo lo que teníamos de la versión anterior y lo que hemos mejorado nosotros en una interfaz gráfica y poder darle forma al juego que hemos realizado, con diferentes pantallas, botones, etc.

Esta fase la realizamos entre los dos ya que hay que coordinarlo todo de manera correcta para que funcione la versión previa a realizar la interfaz gráfica y por tanto la versión final.

Finalmente, la última fase del trabajo consiste en redactar la memoria y hacer pruebas o evaluaciones de nuestro trabajo.

En estas dos últimas etapas, el enfoque de Víctor se centra en la redacción, mientras que yo me encargo de mejorar la estética de la interfaz del juego, la cual finalmente decidimos implementar utilizando CustomTkinter.

Capítulo 5. Tecnologías y herramientas usadas

En este capítulo se exponen las herramientas y las tecnologías que permiten realizar la mayoría de las tareas básicas que se llevan a cabo en este proyecto.

5.1 Mediapipe

Todos los modos de juego tienen una base común. El usuario se sienta delante de una cámara, puede ser la del dron (en el modo Follow Me) o la del ordenador (resto de modos) y con diferentes posturas o movimientos previamente indicados en pantalla va controlando el dron.

La base principal para el reconocimiento y procesado de movimientos del cuerpo es la herramienta de MediaPipe. La librería de MediaPipe disponible para Python hace que esta herramienta sea perfecta para hacer este proyecto ya que se puede utilizar tanto la transmisión por parte de la cámara del dron en directo, o usar la transmisión desde una estación base como puede ser un ordenador o un portátil que procese el video y transmita las órdenes al dron. Es conveniente por eso realizar un estudio profundo sobre esta herramienta.

Empezamos por Mediapipe Hands. Esta herramienta solo detecta y por tanto solo permite controlar el dron con las posturas de la mano. Es la que utilizamos en el modo de juego “Control Dedos”.

MediaPipe Hands es una biblioteca de código abierto desarrollada por Google que utiliza el aprendizaje automático para detectar y seguir las manos en tiempo real a través de una cámara. Está diseñada para ser una herramienta fácil de usar y flexible para desarrolladores que deseen incorporar la detección de manos en sus aplicaciones de Python.

La biblioteca MediaPipe Hands utiliza una red neuronal convolucional (CNN) para detectar y localizar las manos en una imagen o secuencia de video. Utiliza un modelo pre entrenado y optimizado que ha sido entrenado en un gran conjunto de datos de imágenes de manos. El modelo está entrenado para ser rápido y eficiente, lo que permite una detección en tiempo real.

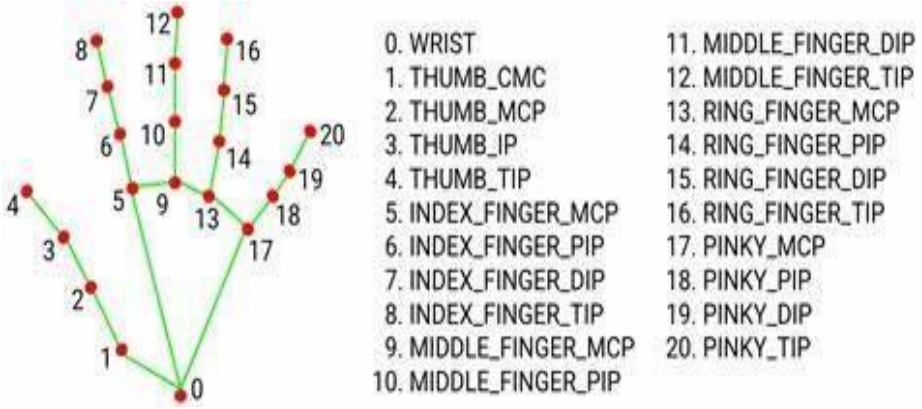

Fig. 5.1 Mediapipe Hands

En la **Fig 5.1** observamos lo comentado anteriormente, esta herramienta nos permite coger los puntos de la mano y utilizarlos para lo que nos convenga.

En una parte más avanzada del trabajo veremos cómo se utilizan para hacer que el dron haga lo que quiere el usuario.

A continuación, cabe estudiar también Mediapipe poses. Es una herramienta muy similar a Mediapipe hands, pero esta se utilizará en los modos de juego “Follow me”, “control poses” y “control cara”. Mediapipe poses coge puntos de todo el cuerpo en vez de coger sólo los de las manos. Más avanzado el trabajo podremos observar cómo, aunque coge todos los del cuerpo, podemos hacer que coja solo los que nos interesa y, como en Mediapipe hands, utilizar esos puntos para mover el dron.

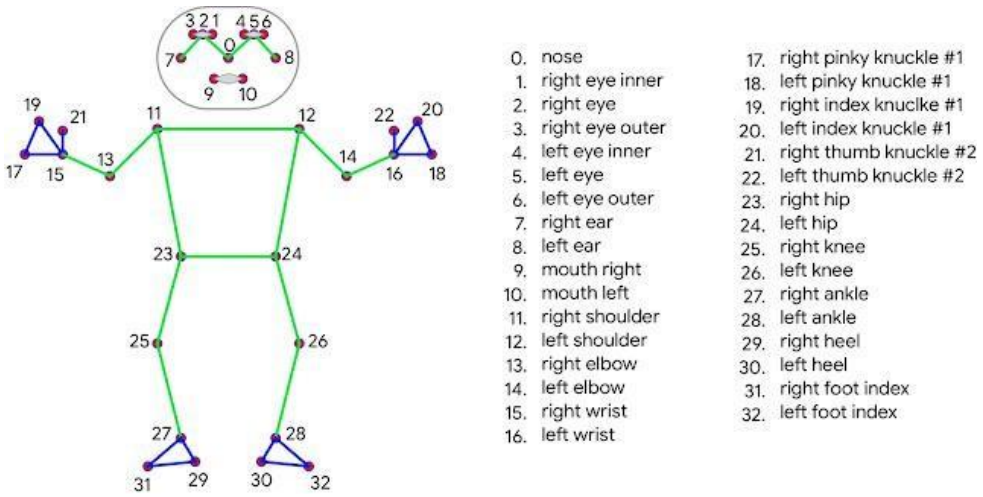

Fig. 5.2 Mediapipe poses

Observamos en la **Fig 5.2** los treinta y dos puntos que gracias a esta librería podemos extraer de la imagen de la cámara y trabajar con ellos.

Estos puntos se utilizan como si estuvieran en un sistema de coordenadas. Es decir, a cada punto le corresponde una coordenada X y una coordenada Y. Mediante este sistema de coordenadas se pueden realizar comparaciones para saber por ejemplo si una pose registrada en la base de datos se parece a la pose que está realizando el usuario.

5.2 CustomTkinter

La segunda herramienta importante que cabe comentar es la que nos permite configurar el aspecto gráfico al juego.

Para la interfaz gráfica de esta aplicación teníamos dos opciones: Tkinter y CustomTkinter.

La librería Tkinter es una interfaz de programación de aplicaciones (API) de Python que permite crear interfaces gráficas de usuario (GUI) de forma sencilla. Tkinter se basa en Tk, una biblioteca gráfica desarrollada originalmente para el lenguaje de programación Tcl. Al estar integrada en Python, Tkinter se encuentra disponible en la mayoría de las distribuciones de Python sin necesidad de instalación adicional.

Tkinter proporciona una amplia gama de widgets (elementos de la interfaz gráfica) que se pueden utilizar para construir ventanas, botones, campos de texto, cuadros de diálogo y muchos otros componentes de la GUI. Estos widgets son altamente personalizables y pueden ser configurados con diferentes propiedades como el tamaño, la posición, el color y el estilo.

La programación con Tkinter sigue un enfoque basado en eventos, donde se definen funciones o métodos que se ejecutan en respuesta a acciones del usuario, como hacer clic en un botón o escribir en un campo de texto. Tkinter también proporciona un sistema de manejo de geometría que permite organizar y colocar los widgets en la ventana de forma flexible.

Aunque Tkinter es una biblioteca muy útil para crear interfaces gráficas básicas, algunos desarrolladores pueden encontrar que su aspecto visual predeterminado no es lo suficientemente moderno o personalizable para sus necesidades. En este sentido, han surgido bibliotecas y frameworks que extienden Tkinter y ofrecen estilos y personalizaciones adicionales.

CustomTkinter es una de esas bibliotecas que se construye sobre Tkinter y proporciona un conjunto de estilos y temas personalizables. Permite a los desarrolladores crear interfaces gráficas más atractivas y modernas utilizando elementos gráficos como botones estilizados, barras de progreso animadas y temas visuales predefinidos. CustomTkinter amplía la funcionalidad de Tkinter y facilita la creación de interfaces de usuario más visualmente atractivas sin necesidad de aprender frameworks más complejo

En resumen, Tkinter es una biblioteca estándar de Python que permite la creación de interfaces gráficas de usuario. Es una herramienta útil y fácil de usar para desarrolladores que desean crear aplicaciones con una interfaz visual. CustomTkinter es una biblioteca que se basa en Tkinter y proporciona estilos y temas personalizables para crear interfaces gráficas más atractivas y modernas.

Para iniciarnos con esta herramienta realizamos una pequeña aplicación.

Yo la hice con Tkinter a diferencia de mi amigo Víctor. Tenía un aspecto como el que podemos observar en la **Fig 5.3**

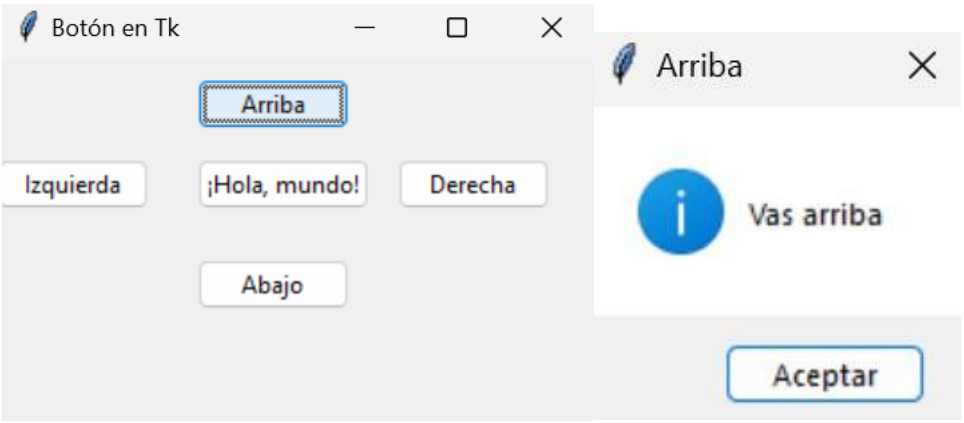

Fig. 5.3 Primera App con Tkinter

Esta aplicación consistía en cuatro botones que, al pulsar en cada uno de ellos saltaba un MessageBox conforme habías pulsado.

Para poder demostrar la diferencia entre Tkinter y CustomTkinter y observar cómo esta segunda ofrece un aspecto más profesional podemos comparar el formulario de la **Fig 5.4** con los formularios que se muestran en la **Fig 5.3** realizados con Tkinter.

Observamos cómo con CustomTkinter podemos editar los bordes de los botones, el fondo del formulario y nos ofrece más formas de botones, así como más posibilidades dentro del formulario como por ejemplo el brillo con el que queremos observarlo.

En cambio, con Tkinter el formulario es mucho más básico sin tantas oportunidades de cambio respecto a colores, formas o aspectos visuales.

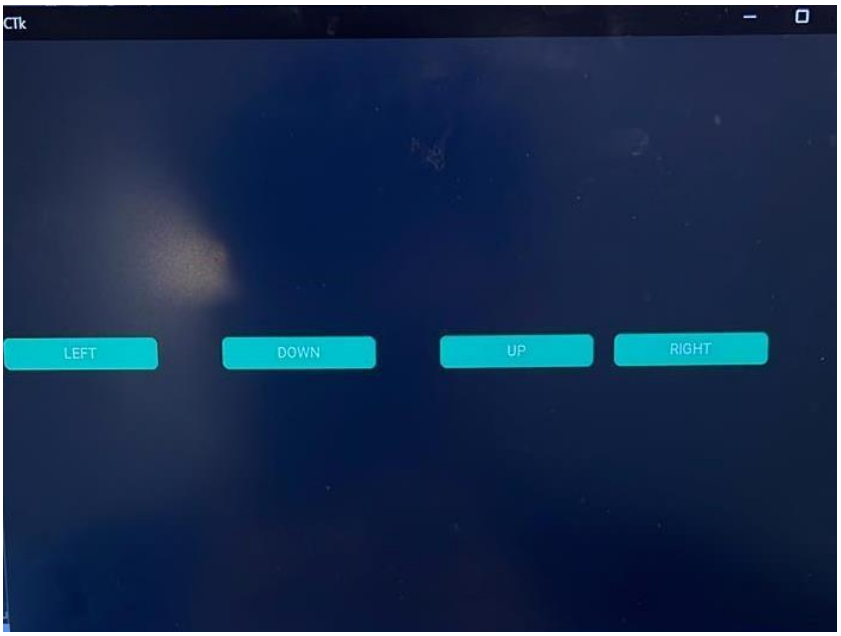

Fig. 5.4 Aspecto app con CustomTkinter

Capítulo 6. Aspecto de la aplicación

En este capítulo mostramos el aspecto de todas las pantallas que se puede encontrar el usuario en las dos aplicaciones realizadas centrándonos siempre en la más elaborada, la app del juego de imitación.

6.1 App modo libre

En este trabajo nos hemos dedicado a dos aplicaciones. Una tiene el modo de juego libre y otra el modo de juego imitación. Esta segunda es la que más tiempo hemos dedicado y así se ve en el aspecto.

Empezamos por comentar la aplicación sencilla del modo de juego libre con sus pantallas:

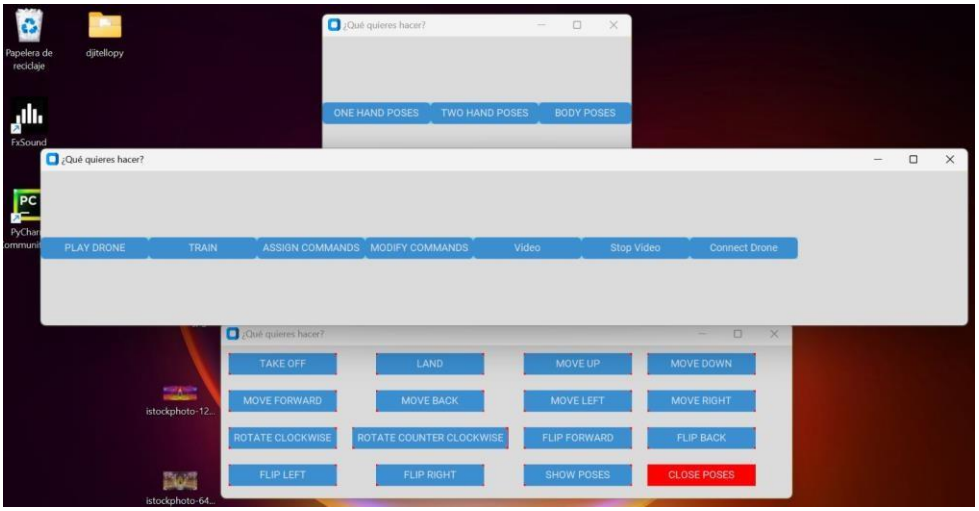

Fig. 6.1 App modo libre

Como observamos al abrir la aplicación nos aparece la ventana principal que es la de la parte superior de la **Fig. 6.1**. Ahí podemos escoger el modo libre con una mano, con dos manos o con poses de todo el cuerpo. Una vez escogemos y apretamos el botón, nos aparece la siguiente ventana que sería la situada en el medio de la figura anterior. En esta ventana podemos conectar con el dron, iniciar la cámara del dron, parar la cámara del dron, asignar poses a comandos del dron y modificar algunas poses ya asignadas. Vemos que tenemos dos modos de juego: el entreno y el juego del dron.

En la captura anterior finalmente se muestra todos los comandos del dron a los que les podemos asignar poses diferentes.

6.2 App modo imitación

A continuación, vamos a explicar las diferentes pantallas y opciones del modo de juego imitación. Al abrir el juego nos encontramos con la pantalla mostrada en la **Fig 6.2**

Fig. 6.2 Pantalla principal juego imitación

En esta pantalla observamos tres botones: el botón “start”, el botón “how to play” y el botón “Music on”.

Si pulsamos el botón “how to play” nos aparece una breve explicación del funcionamiento del juego como observamos en la **Fig 6.3**

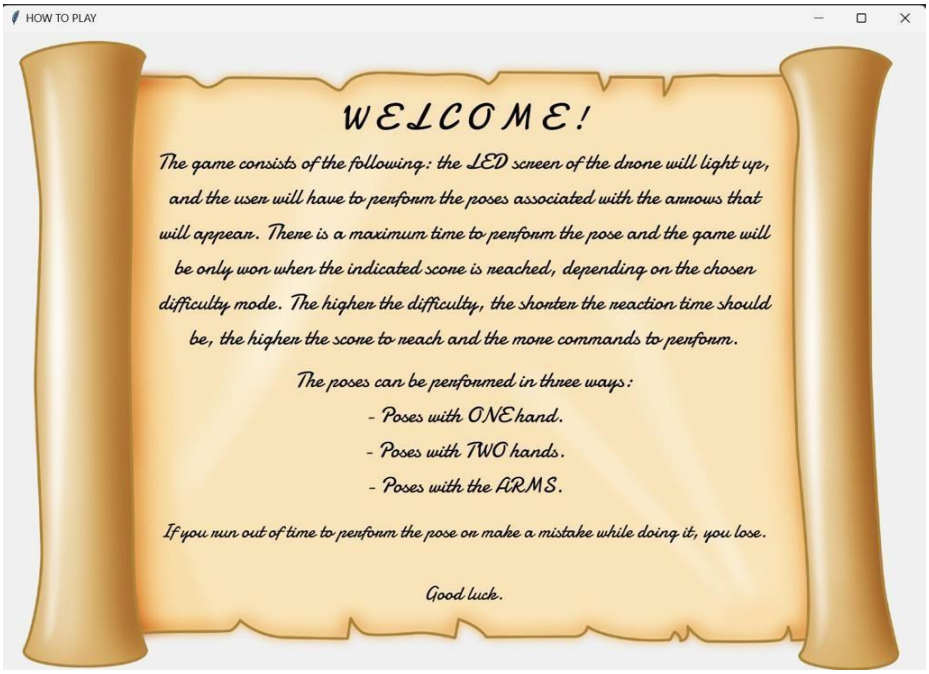

Fig 6.3 Descripción de cómo se juega

Si en cambio pulsamos el botón “music on” podremos activar o desactivar la música del juego.

Finalmente, si pulsamos el botón “start” pasaremos a la siguiente pantalla que será como observamos en la **Fig 6.4**

Fig. 6.4 Pantalla modos de juego

En esta pantalla se nos muestran los diferentes modos de juego. Hay tres: modo de juego con una mano, modo de juego con dos manos y modo de juego con las poses de todo el cuerpo. Cabe comentar que también tenemos un botón en el que pone “back” que pulsándolo nos desplazamos hasta la ventana anterior. El botón para activar y desactivar la música permanece también en esta ventana.

Una vez hemos decidido el modo de juego al que queremos jugar, pulsamos el botón y nos aparecerá la siguiente ventana mostrada en la **Fig 6.5**

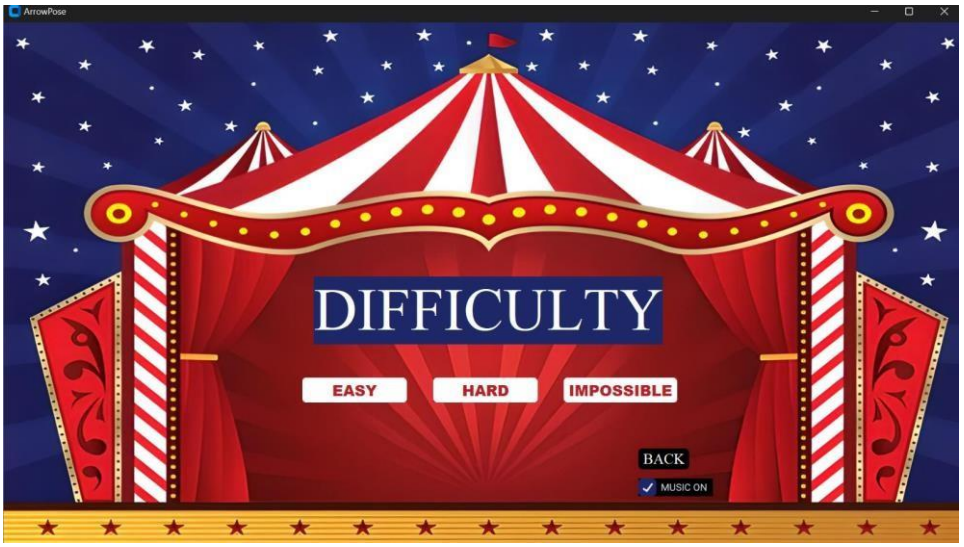

Fig. 6.5 Pantalla niveles de dificultad

Este menú nos da la opción de escoger en que dificultad queremos jugar. La dificultad consiste en tener menos tiempo para realizar la pose que pide el dron y en tener más posibles poses que hacer. Siguen en esta ventana los botones para ir una ventana atrás y para activar o desactivar la música.

Una vez escogemos dificultad pasamos al último menú que tiene el aspecto de la **Fig 6.6**

Fig. 6.6 Pantalla previa inicio vuelo

Este último menú es el más completo. Tiene también los botones para tirar hacia atrás y para desactivar la música. Tiene el botón de asignar poses a los diferentes movimientos y el botón de modificarlas en caso de habernos equivocado. También tiene el botón para mostrar las poses guardadas en la base de datos y para cerrarlas. Aparece el botón que aparecía en la primera ventana en el que se explican las normas del juego. Observamos también el botón para conectar con el dron y finalmente el botón para empezar a jugar

Capítulo 7. Proceso para capturar las poses

En este capítulo se explica la manera que se usa en nuestra app para poder capturar poses de un “frame” de video.

Este apartado es delicado ya que no es fácil de explicar. Para empezar cabe comprender que la captura de las poses se obtiene gracias a los hilos. Los hilos son diferentes caminos del programa.
Es decir, tenemos un hilo del programa principal que se dedica a hacer funciones como por ejemplo abrir la cámara de video. El problema está en que cuando queremos capturar una pose, si lo intentamos hacer en el hilo principal, la cámara se queda congelada y por tanto no es lo que queremos. Para realizarlo tenemos que crear un hilo con un temporizador que se encargue de cuando el hilo principal le de la orden capturar la pose que se muestre en pantalla.

Aquí, en la **Fig. 7.1** mostramos un trozo de código con el hilo secundario para capturar la pose:

```
“if cv2.waitKey(1) == ord(" "):
  temporizador_thread = threading.Thread(target=temporizador)
  temporizador_thread.start()”
```

Fig. 7.1 Captura hilo principal

```
"def temporizador():
 print("TOMANDO FOTO EN 3 SEGUNDOS...")
time.sleep(3)
global take_photo
global frame
global frame_captured
global results
global results_captured

take_photo = True
# cv2.imshow("Foto_3_segundos", frame)
results_captured = results
frame_captured = frame"
```

Fig. 7.2 captura hilo secundario

En la imagen 7.1 observamos como el hilo principal del programa llama al hilo secundario mostrado en la **Fig. 7.2** en el que se inicia un temporizador de tres segundos y se toma la foto.

Otro apartado que hay que tratar es el problema de las distancias al captar una pose. Es decir, si la base de datos tiene una pose guardada, se necesita que el programa detecte esa pose, aunque el usuario realice la pose más cerca o más lejos de la cámara respecto a la que hay guardada.

Este problema lo solucionamos con el rectángulo de adaptación. Consiste en adaptar la pose captada por la cámara a las medidas de la pose que hay guardada en la base de datos. Son matemáticas.
Se toman las medidas de la pose de la base de datos, se toman las medidas de la pose capturada y se adapta dividiendo o multiplicando la pose capturada por el factor de adaptación.

Esta fue la primera medida tomada y la que antes se nos ocurrió. Cabe comentar que una vez llegamos al método de detección de poses mediante rectángulos visto más adelante ya no hizo falta el rectángulo de adaptación.

Capítulo 8. Proceso para detectar las poses

Este proceso es al que mayoritariamente nos hemos dedicado a lo largo del trabajo. En este capítulo se muestran las diferentes etapas que hemos seguido para llegar al resultado final.

8.1 Coordenadas

En el trabajo anterior, la detección de poses se realizaba mediante una comparación de coordenadas. Es decir, sin él la pose de la base de datos la coordenada 12 del hands Mediapipe está por encima que la 10, en la imagen captada tiene que ser igual.

Esto implica que, al tener que realizar comparaciones punto a punto, para cada pose hay que intentar detectar las coordenadas clave y programar cada comparación que se quiere realizar. Esto hace que sea muy poco ágil al tener que cambiar una pose de la base de datos.

Para explicar de mejor manera como se realizaban estas comparaciones nos ayudamos de una captura de código en la **Fig. 8.1.**

```
"if ((leftHandLandmarks[18][1] > leftHandLandmarks[19][1])
        and (leftHandLandmarks[19][1] > leftHandLandmarks[20][1])
        and (leftHandLandmarks[14][1] > leftHandLandmarks[15][1])
        and (leftHandLandmarks[15][1] > leftHandLandmarks[16][1])
        and (leftHandLandmarks[10][1] > leftHandLandmarks[11][1]

      and (leftHandLandmarks[8][0] < leftHandLandmarks[5][0])
        and (leftHandLandmarks[6][1] < leftHandLandmarks[5][1])
        and (leftHandLandmarks[6][1] < leftHandLandmarks[7][1])
        and (leftHandLandmarks[7][1] < leftHandLandmarks[8][1])
        and (leftHandLandmarks[4][0] < leftHandLandmarks[3][0])
        and (leftHandLandmarks[4][1] < leftHandLandmarks[3][1])
        and (leftHandLandmarks[3][1] < leftHandLandmarks[2][1])
        and abs(leftHandLandmarks[8][1] - leftHandLandmarks[4][1])
< 0.05
```

```
                and abs(leftHandLandmarks[8][0] - leftHandLandmarks[4][0])
< 0.05):

    return True
 else:
    return False"
```

Fig. 8.1 Comparaciones mediante coordenadas

Las posturas estaban ya establecidas en una base de datos. De estas se podían extraer las coordenadas de los treinta y dos puntos que nos permitía detectar Mediapipe.

Entonces la idea es ver si en esta postura era clave ver por ejemplo que la yema del dedo índice estaba por encima de la yema del dedo corazón ya que este estaba doblado hacia abajo. Es decir, la idea era coger las características claves de la postura preestablecida en relación con los puntos detectados. Estas características clave también incluían comparar algunas distancias. Si por ejemplo en esa pose había dos dedos muy juntos, una condición clave era que la distancia entre esos dedos no fuera superior a un cierto valor umbral (que se podía ir ajustando a medida que se probaba).

Una vez teníamos estas condiciones claves de puntos para que se diera esta postura determinada, la idea era coger los puntos de la imagen del usuario y comparar para ver si se cumplían estas condiciones

Es por eso por lo que en la **Fig 8.1** observamos tantos “AND” que implican que solo si se cumplen todas las condiciones claves para que sea la figura que buscamos se detecte como tal.

Es por esto por lo que nos llevó bastante tiempo este apartado. Teníamos claras las poses que queríamos hacer tanto en el modo de juego de “Control Dedos” como en el modo de juego “Control Poses” pero a la hora de buscar los puntos clave y programar esas condiciones se complicaba la cosa.

Es aquí donde nos dimos cuenta tanto nosotros como nuestro tutor que esto era poco práctico. ¿Cada vez que quisiéramos añadir una nueva pose tendríamos que hacer todo este proceso? Registrar la pose en la memoria, coger los puntos clave y programar las condiciones para comparar. Este método no era muy práctico, aunque en todo caso sí efectivo.

No llegamos a programar todas las poses del modo difícil en estos dos modos de juego sobre los que estábamos trabajando. Vimos que había un problema y, con la ayuda de nuestro tutor, empezamos a trabajar en la siguiente idea.

8.2 Distancias

Coma previamente hemos mencionado este segundo método de detección de poses consiste en la comparación de distancias. Esto implica que las distancias entre los puntos de la imagen de la base de datos sean similares, y sí similares porque establecemos un margen de error, a las distancias entre los puntos de la imagen capturada.

Aquí, en la **Fig. 8.2** se muestra una captura del código en el que se realizan las comparaciones por distancias:

```
" NewPose1_YdiferenciaTHUMB_INDEX = y1 - y2
        DistanciaY_NewPose1_THUMB_INDEX_Modified =
NewPose1_YdiferenciaTHUMB_INDEX

    if (NewPose1_YdiferenciaTHUMB_INDEX < 0):
            DistanciaY_NewPose1_THUMB_INDEX_Modified
DistanciaY_NewPose1_THUMB_INDEX_Modified / (-1)
            NewPose1_YdiferenciaTHUMB_INDEX =
NewPose1_YdiferenciaTHUMB_INDEX / (-1)
    if (20 > NewPose1_YdiferenciaTHUMB_INDEX > -20):
            DistanciaY_NewPose1_THUMB_INDEX_Modified =
DistanciaY_NewPose1_THUMB_INDEX_Modified * 10.00"
```

Fig. 8.2 Comparación mediante distancias

Como observamos hay un margen de error que fuimos ajustando para hacer lo más fiable posible la detección.

La principal ventaja de este método respecto al anterior es que no hay que programar las condiciones clave de cada pose e ir comparando coordenada a coordenada. En este caso se buscan siempre que las distancias sean similares y es por eso que no hace falta en cada pose nueva se programen nuevas condiciones. Un gran avance.

La principal desventaja y por la que descartamos el método e intentamos encontrar otro o mejorar este es la fiabilidad. Es muy poco robusto ya que si hay puntos que no detecta la cámara, al pasar por la comparación mediante distancias peta. Por ejemplo, si el usuario quiere introducir la pose en “control dedos” de los tres dedos índice, corazón y pulgar extendidos mientras anular y meñique están plegados con la parte trasera de la mano mirando a la cámara, los puntos de los dedos plegados cuestan de detectar o no los detecta y eso hace que colapse este método.

8.3 Rectángulos

Este último método es una mejora del anterior que permite garantizar fiabilidad casi del cien por cien en la detección de poses. Anteriormente había unos márgenes de distancias que tenías que ir ajustando según la pose y, aunque era bastante menos laborioso que en el primer método, daba faena. Otro problema era que dependiendo lo que escondieras algunos dedos tras la mano, no los detectaba.

En este método lo que se establecen son unos rectángulos de margen en los 5 puntos de la mano más utilizados o en el caso de las poses de todo el cuerpo en los 6 puntos que tomamos como datos, como se muestra en la imagen de la **Fig. 8.3**

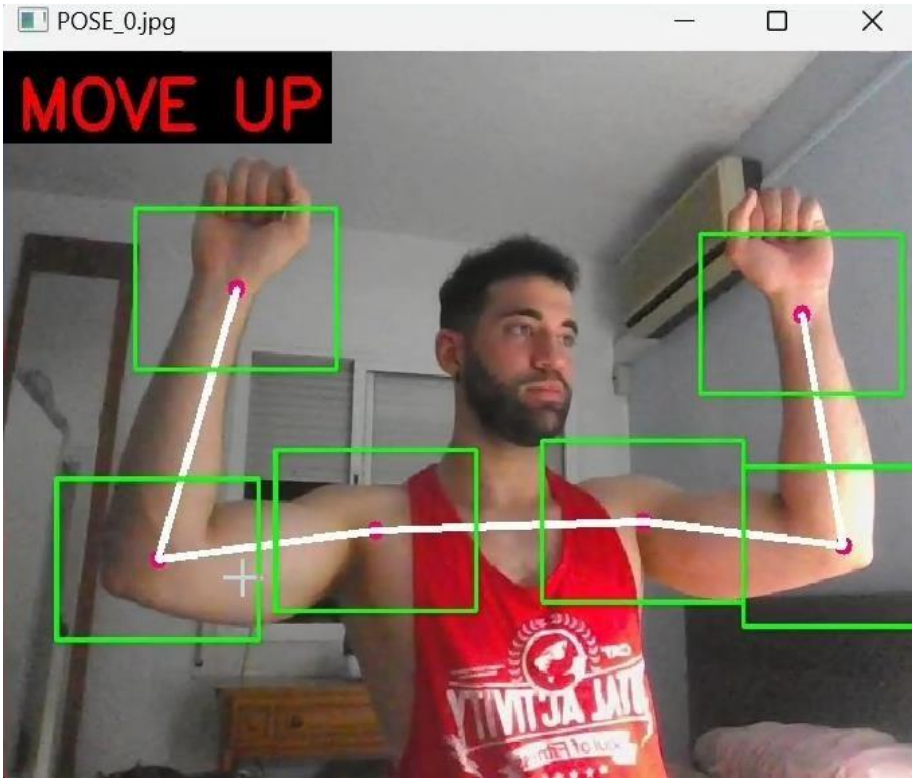

Fig. 8.3 Método de los rectángulos

Estos rectángulos permiten dar siempre ese margen de error que ya no será necesario ajustar cada vez. Los rectángulos permiten también detectar más fáciles puntos que queden ocultos tras la mano por ejemplo ya que si entran en el mismo margen del rectángulo podrán ser detectados.

Aquí, en la **Fig 8.4** ,mostramos la programación del rectángulo con los márgenes establecidos

```
" for i in range(5):
        x1_rectangle_normalized, y1_rectangle_normalized = lista_x[i] + 0.05, lista_y[i] + 0.05
        x2_rectangle_normalized, y2_rectangle_normalized = lista_x[i] - 0.05, lista_y[i] - 0.05

        x1_rectangle = int(x1_rectangle_normalized * width)
        x2_rectangle = int(x2_rectangle_normalized * width)

        y1_rectangle = int(y1_rectangle_normalized * height)
        y2_rectangle = int(y2_rectangle_normalized * height)

    # Dibujamos el cuadrado en la imagen
        #cv2.rectangle(frame, (x1_rectangle, y1_rectangle), (x2_rectangle, y2_rectangle), (0, 255, 0), 2)

        lista_marginsRectangle_finger = "lista_marginsRectangle_finger_" + str(i)

        sub_diccionario[lista_marginsRectangle_finger] = [x1_rectangle_normalized,
                                                          x2_rectangle_normalized,
                                                          y1_rectangle_normalized,
                                                          y2_rectangle_normalized]"
```

Fig. 8.4 Programación rectángulos

La medida del rectángulo ha sido optimizada probando muchas poses diferentes y viendo que vaya bien para todas.

Todo el proceso, como hemos comentado se ha llevado a cabo con poses en una sola mano, pero cuando se ha llegado a un resultado que hemos considerado válido y robusto se ha extendido a dos manos y a poses con todo el cuerpo.

Cabe comentar que todo el código está explicado en un video. En él se aprecian mejor todos los detalles específicos de cada apartado.

El link se encuentra al final del trabajo[1].

Capítulo 9. Mecánica del juego

En este capítulo se explica la mecánica de las dos aplicaciones con sus respectivos modos de juego.

Una vez mostradas las diferentes pantallas que tendrá el juego, sabemos que hay dos modos: el modo libre y el modo imitación.

El modo libre consiste en que el usuario entra a la interfaz gráfica, una vez conectado el dron, registra las poses que quiere que signifique cada movimiento concreto de este y ya puede llevar al dron con sus poses por donde quiera.

El modo imitación en cambio es un juego un poco más elaborado que permite al usuario ganar, es decir superar el juego o perder, es decir no superarlo.

Este modo de juego tiene diferentes dificultades ya que recordemos que la idea es que pueda ir dedicado a público de todas las edades.

Consiste en, como siempre el usuario entra a la interfaz grafica y registra las poses para los movimientos arriba, abajo, izquierda y derecha en el caso del modo fácil. Una vez lo ha realizado, le aparecen las poses que ha decidido guardar en la base de datos en la pantalla. Por ejemplo, las siguientes mostradas en la **Fig 9.1**

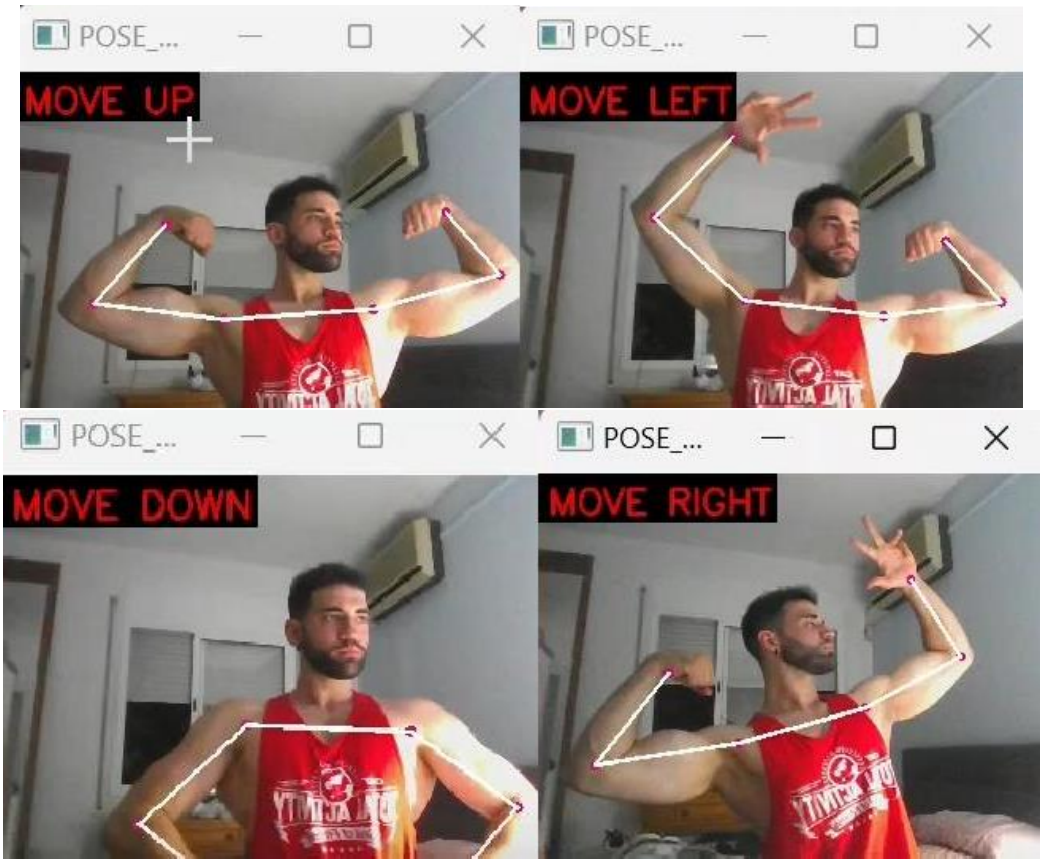

Fig. 9.1 Poses establecidas

Entonces iniciamos el juego. El dron mediante su panel de leds indicará al usuario qué pose tiene que realizar delante de la cámara. Un ejemplo lo vemos en la **Fig 9.2**

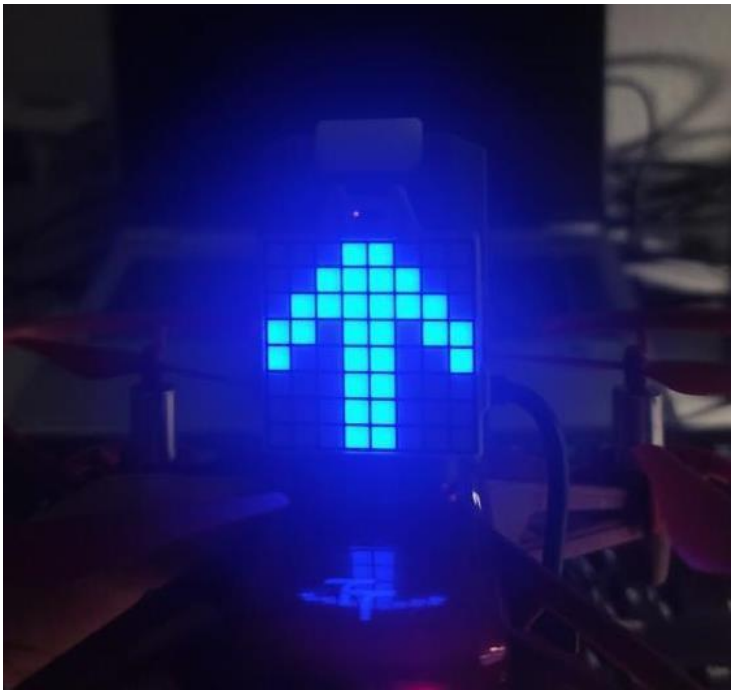

Fig 9.2 Panel de leds con la pose a ejecutar

Si el usuario realiza la pose correcta sumará un punto, si realiza una pose incorrecta o tarda más de diez segundos (en el modo fácil) en realizarla, el usuario habrá perdido.

Se pasa el juego cuando se realizan tres poses seguidas de manera correcta como indica el dron. Cabe comentar que obviamente después de realizar cada pose el dron se desplaza hacia donde le toca. En otros modos más complicados se reduce el tiempo que se le ofrece al usuario para realizar la pose y se aumentan el numero de poses que hay que realizar de manera consecutiva para ganar.

Para mostrar mejor cómo funcionan hemos realizado un video que lo demuestra con claridad. Aquí se encuentra el link del video [1]

Capítulo 10. Pruebas y evaluación

En este capítulo se establecen un conjunto de pruebas básicas para comprobar que funciona bien el programa y se buscan personas aleatorias para que den su opinión respecto al trabajo realizado, así como áreas de mejora.

10.1 Pruebas

Durante todo el trabajo se habla de los términos fiabilidad o robustez y esto se ha ido valorando mediante diferentes pruebas que hemos realizado en el modo imitación que es donde se ha centrado la mayoría de nuestro trabajo.

Hemos intentado ir a pillar: haciendo que las poses capturadas por el usuario no le gustan y las quiere cambiar diez veces, por ejemplo. poniéndonos siempre en los peores casos para ver si en estos el programa responde y si no mejorarlo para que responda.

La primera prueba que hemos realizado es probar todos los modos de juego con todas las dificultades posibles de manera correcta: es decir pruebas un modo, ganas, sales del modo de juego y pruebas el siguiente. Así sucesivamente. La aplicación responde correctamente.

La segunda prueba que realizamos fue una prueba sorpresa. El martes día 27 de junio asistimos a nuestra reunión rutinaria con nuestro tutor Miguel Valero al cual le íbamos a enseñar ya finalizado nuestro modo de juego imitación.

Nuestra sorpresa fue que nos trajo público para tal demostración y enseñamos delante de unas quince personas este modo de juego en el que los usuarios fuimos nosotros: capturamos las poses y empezamos a jugar delante de ellos. Salió todo bien y al parecer les gustó.

Por último, hemos realizado la prueba de repetir poses que no gustan al usuario y volver a guardarlas. Resultó una prueba satisfactoria finalmente.

Así hemos ido realizando pruebas como una de las últimas realizadas en este trabajo que es la realización del video intentando mostrar todos los modos de juego sin ningún fallo.

Obviamente siempre puede haber un fallo, de hecho, a lo largo del trabajo hemos experimentado muchos fallos que hemos ido solucionando: desde la distancia cámara-usuario al detectar una pose hasta el errático método de detección de poses mediante distancias.

10.2 Evaluaciones

Respecto a las evaluaciones, hemos dejado nuestro programa a familiares cercanos tanto Víctor como yo y hemos extraído algunas cosas positivas y otras que se podrían mejorar.

En mi caso, mi hermano Emilio probó el juego y le gustó. Me comentó que era un buen proyecto, aunque el juego final lo veía sencillo. Él comprendió que lo importante era sobre todo la detección de las poses y dijo que eso sí estaba muy bien porque no fallaba ninguna.

En segundo lugar, lo probaron mis padres y al estar en inglés la interfaz pues no sabían que tenían que hacer, así que les fui guiando. Me dieron una buena opinión y disfrutaron con el modo de dificultad difícil, ya que en la imposible no consiguieron vencer.

Por tanto, las conclusiones de las opiniones de las personas que hemos preguntado son que una de las cosas positivas es que hemos creado un juego amigable con diferentes niveles para todas las edades que crea buenas sensaciones en los usuarios: diversión y entretenimiento.

Además, nos han comentado que, al ser un juego rápido, es decir que no dura mucho tiempo, no se hace pesado hacer varias partidas seguidas intentando superarte.

Finalmente nos han comentado que es un juego fácil de usar, es decir es difícil perderse entre las diferentes pantallas o equivocarse de botones. Está todo bastante bien indicado.

Algunas de las cosas a mejorar son principalmente la robustez del juego. Hay veces que dependiendo de la pose rebuscada que haga el usuario, le cuesta detectarla. De todas las evaluaciones que hemos hecho, ha habido veces que el dron o el programa ha fallado.
Otra de las cosas a mejorar es el dron. Tiene baterías muy débiles que se agotan a los minutos de estar jugando y eso hace que haya que estar cambiándolas continuamente.

A parte de las opiniones ajenas, nosotros también hemos detectado cosas que se podrían mejorar o incluso algunas ideas que se podrían llevar a cabo.

Empezamos por comentar una propuesta que nos hizo nuestro tutor que consiste en jugar con la cámara del dron. Una posibilidad sería hacer un juego tipo tenis. Dos usuarios se sitúan uno frente al otro cada uno con una raqueta. La raqueta tiene una cara del color azul por ejemplo y la otra de color amarillo. El dron vuela entre ellos y cuando detecta el color azul en un mediante su cámara da la vuelta y se dirige hacia el otro usuario en cambio si detecta el amarillo sigue avanzando hacia la raqueta. Este juego se podría añadir con aleatoriedad del trayecto y velocidades del dron para que así no sea un juego monótono y en línea recta.

Obviamente la robustez o las baterías del dron que son problemas que ha detectado nuestro público nosotros ya lo habíamos detectado. En el caso de la robustez hemos hecho lo posible por arreglarlo o al menos mejorarlo y en el caso de las baterías no hemos podido hacer nad

Capítulo 11. Conclusiones

En este capítulo se exponen las conclusiones extraídas al finalizar el trabajo.

Personalmente, este trabajo me ha ayudado mucho a ver cómo es trabajar en conjunto en un proyecto de programación. Me ha servido mucho porque he aprendido a programar en Python desde cero. Sobre todo, en lo que más me ha aportado este proyecto es en el ámbito de actuar como programador, es decir, a pensar como un programador. A día de hoy, cuando tengo que programar en un lenguaje que desconozco, soy capaz de aprender gracias a la base que me ha dado este trabajo.

Ha habido partes muy buenas durante el proyecto, sobre todo cuando después de semanas intentando que funcionase una función, al final funcionaba.

También han habido partes malas y tediosas. Las baterías del dron son de muy poca duración, por tanto, cuando querías probar las nuevas funciones, tal vez tenías problemas para utilizarlo. Además, no sabes cuándo están cargadas al 100 %. Me acuerdo que hubieron días que las dejé cargando durante horas y cuando fui a utilizarlas, apenas tenían batería o me duraban unos minutos.

Finalmente concluimos que hemos realizado un buen trabajo.

Creo que hemos avanzado de manera notable en la manera de detección de poses, así como en el reconocimiento de estas: hemos cambiado los métodos de reconocimiento, hemos hecho un reconocimiento más robusto y eficaz y hemos hecho que el usuario pueda registrar sus propias poses.

A parte hemos realizados avances que proponen más atracción de cara al usuario como por ejemplo diferentes dificultades de juego o el hecho de ir sumando puntos a medida que avanza el juego para llegar a un objetivo concreto.

Cabe comentar también que hemos dedicado un tiempo a la interfaz gráfica que ha quedado en muy buen estado y amigable para cualquier edad.

Hay muchas áreas de mejora en este proyecto interesante ya que se pueden realizar miles de juegos diferentes como entretenimiento.

Esto nos lleva a concluir que hemos sido capaces de demostrar que los drones son una gran herramienta de ocio y esperemos que trabajos como este sirvan para cambiar la imagen que, en general, el mundo tiene de ellos.

Capítulo 12. Referencias

En este capítulo se citan las referencias de las ayudas que hemos precisado en el trabajo para documentarnos y poder llevarlo a cabo.

[1] David y Victor- Video de explicación TFG (2023) https://youtu.be/YkIxwt4Gaac?si=0EzKFxkkCyIZwvwE

[2] Jonathan Palacios – Guía para programar un circo de drones (2023) [Documento]

[3] Custom Tkinter – información general https://customtkinter.tomschimansky.com/

[4] Tutorial Mediapipe – información general https://youtu.be/ipHKQVtwRas

[5] Inteligencia artificial- información general de todos los temas tratados https://chat.openai.com/

[6] Dron Tello – Especificaciones técnicas – [Online] https://www.ryzerobotics.com/es/tello/specs

[7] Jonatan Palacios – Repositorio de GitHub drone-circus (2022) – [Online] https://github.com/jonatanpalaciosacevedo/drone-circus

[8] Nicholas Renotte – Video de YouTube: detección de gestos (2021) – [Online] https://youtu.be/doDUihpj6ro

[9] MediaPipe – Herramienta de detección del cuerpo y cara – [Online] https://mediapipe.dev/