uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática

2020-2021

Trabajo Fin de Grado

"Control mediante la mirada de una cámara integrada en un dron"

Melina Abril Hernández Velásquez

Tutor/es

Daniel Amigo Herrero

Leganés, 2021

RESUMEN

Los drones son un campo de investigación al alza ya que son económicos y se utilizan en una gran cantidad de campos desde la agricultura hasta la minería y el entretenimiento. Es por esto por lo que se buscan formas más intuitivas e inmersivas de controlarlos. El seguimiento de la mirada, por otro lado, tiene una extensa bibliografía ya que tiene igualmente múltiples aplicaciones como, por ejemplo, los sistemas de seguridad de los coches que detectan cuando el conductor está cansado. Con este proyecto se han querido unir estos dos sectores en la investigación del desarrollo de un sistema que permita controlar la cámara integrada de un dron, mediante el seguimiento de la mirada del operario que lo controla, utilizando técnicas de Visión Artificial. Se ha realizado un estudio de la viabilidad del sistema, así como de sus posibles aplicaciones, y se ha diseñado e implementando un prototipo. Los resultados del proyecto muestran que el desarrollo es funcional y que los objetivos de la investigación se han cumplido.

Palabras clave

Dron, Visión Artificial, seguimiento de la mirada, Dlib, OpenCV, Python, landmarks, Detección Facial, puntos de referencia faciales, gaze tracking.

ABSTRACT

Drones are a growing research field; they are inexpensive, and they are used in a myriad of fields from agriculture to mining to entertainment. Therefore, more intuitive and immersive ways to control them are sought. Gaze tracking, on the other hand, has an extensive bibliography as it also has multiple applications such as car safety systems that detect when the driver is tired based on his/her gaze. The intention of this project is to unite these two sectors in the research of the development of a system that allows to control the integrated camera of a drone by following the gaze of the operator who controls it, using Artificial Vision techniques. A feasibility study of the system has been carried out, and a prototype has been designed and implemented successfully. The results of the project show that the development is functional and that the research objectives have been met.

DEDICATORIA

Para empezar, quiero agradecer a mi familia que me hayan dado todos los recursos necesarios para poder formarme y llegar hasta aquí, así como todo su cariño y apoyo en estos años que no he podido estar tanto con ellos como me habría gustado.

A mi novio que siempre me ha ofrecido su ayuda cuando me veía agobiada con la universidad incluso cuando él estaba igual que yo. Y también quiero agradecer a sus padres que nos han cuidado y ayudado en todo lo que han podido en esta etapa que hoy termina.

A los amigos que he hecho en la universidad con los que he compartido tantas horas en la biblioteca.

Por último, quiero agradecerle a mi tutor, sus palabras de ánimo cuando me costaba seguir y sus sugerencias para mejorar el proyecto, me han ayudado a completar este trabajo.

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN 1
1.1 Motivación del trabajo 1
1.2 Objetivos 3
1.3 Estructura del proyecto 5
1.4 Marco regulador 6
2. FUNDAMENTOS TEÓRICOS Y ESTADO DEL ARTE 8
2.1 Drones introducción y situación actual 8
2.2 Inteligencia Artificial 9
2.2.1 Introducción 9
2.2.2 Visión Artificial 10
2.2.3 Aplicaciones en los drones de la Visión Artificial 10
2.2.4 Detección facial 11
2.2.4.1 Histograma de Gradientes Orientados 13
2.2.4.2 Machine Learning y Redes Neuronales 14
2.2.4.3 Puntos de referencia faciales 16
2.2.5 Comparación de distintas implementaciones de la Detección Facial 17
2.2.5.1 Tensorflow 18
2.2.5.2 OpenCV y Dlib 19
2.2.5.3 Conclusiones 20
2.3 Análisis de distintas herramientas para la implementación del sistema 21
2.3.1 Aplicación Web vs Aplicación Móvil 22
2.3.1.1 Aplicación Móvil Nativa 22
2.3.1.2 Aplicación Web 22
2.3.1.3 Híbridos entre Aplicación Web y Móvil Nativa 23
2.3.1.4 Conclusiones 23

2.3.2 REST-API 24
2.3.2.1 Flask Vs Django 24
2.3.3 Sockets 25
2.3.4 Hardware 28
2.4 Conclusiones sobre el estado del arte 29
3. DISEÑO Y DESARROLLO 31
3.1 Requisitos del sistema 31
3.2 Resumen del diseño 32
3.3 Arquitectura de software 32
3.3.1 Interfaz Web y adquisición de datos de la Webcam 33
3.3.2 Servidor principal API-REST 34
3.3.3 Streaming de video de la cámara del dron 35
3.3.4 Software de detección de mirada 35
3.3.5 Servidor de la Raspberry Pi 38
3.4 Implementación 40
4. RESULTADOS 43
4.1 Definición de las pruebas 43
4.1.1 Prueba de latencia 43
4.1.2 Prueba de precisión de la transformación 44
4.1.3 Prueba de precisión del modelo de detección 46
4.2 Resultados de las pruebas 47
4.2.1 Resultados de la prueba de latencia 47
4.2.2 Resultados de la prueba de precisión en la transformación 48
4.2.2.1 Resultados de la ruta 1 48
4.2.2.2 Resultados de la ruta 2 con paros y acelerones 52
4.2.3 Resultados de la prueba de precisión del modelo 53
4.3 Discusión de los resultados 54

5. IMPACTO SOCIOECONÓMICO 57
6. PRESUPUESTO 59
6.1 Introducción 59
6.2 Desarrollo del presupuesto 59
7. CONCLUSIÓN Y FUTUROS TRABAJOS 63
8. BIBLIOGRAFÍA 66
ANEXO A. REPOSITORIO DE GITHUB E INSTRUCCIONES DE USO 71
Instalación en Windows 71
ANEXO B. RESULTADOS DE LAS PRUEBAS 74
Ruta 1 74
Prueba 2 74
Prueba 3 75
Ruta 2 76
Prueba 2 76
Prueba 3 77

ÍNDICE DE FIGURAS

Figura 1. Clasificación de los métodos más populares de detección facial 12
Figura 2. Ejemplo descriptor HOG. Cada celda indica la orientación de los gradientes [19] 13
Figura 3. Una red neuronal artificial con tres inputs y un output [21] 15
Figura 4. Ejemplo de detección de puntos de referencia faciales [23] 16
Figura 5. Diagrama comunicación entre Cliente y Servidor mediante una REST-API . 24
Figura 6. Comunicación mediante WebSockets Vs. HTTP 27
Figura 7. Ejemplo de estructura en Pan and Tilt para el movimiento horizontal y vertical de la cámara 28
Figura 8. Diseño completo del sistema 33
Figura 9. Interfaz de la aplicación Web. Recoge las imágenes de la webcam y devuelve las imágenes de la cámara del dron 34
Figura 10. Clase DronVideoStreaming 35
Figura 11. Diagrama de clases del software de detección de la mirada 38
Figura 12 Clase DronMoveCamera 39
Figura 13. Relación entre los grados de rotación de un servo y el ancho de pulso 39
Figura 14. Representación de la arquitectura del diseño propuesto 40
Figura 15. Servos conectados en estructura pan and tilt 41
Figura 16. Diagrama de cómo medir la latencia entre Cliente y Servidor 44
Figura 17. Diagrama explicativo de la prueba 2 45
Figura 18. Diagrama de la segunda ruta con paros y acelerones 46
Figura 19. Transformación de la ratio vertical de la mirada al duty cycle vertical 48
Figura 20. Transformación de la ratio vertical de la mirada al duty cycle vertical con tramos 49
Figura 21. Transformación de la ratio horizontal de la mirada al duty cycle horizontal 50
Figura 22. Transformación de la ratio horizontal de la mirada al duty cycle horizontal con tramos 51
Figura 23. Transformación de la ratio vertical de la mirada al duty cycle vertical 52
Figura 24. Transformación de la ratio horizontal de la mirada al duty cycle horizontal 53
Figura 26. Resultados del software de detección de la mirada 54

ÍNDICE DE TABLAS

TABLA 1. RESULTADOS DE LA PRUEBA DE LATENCIA DEL SISTEMA 47
TABLA 2. PRESUPUESTO DESGLOSADO..... 59
TABLA 3. RESUMEN DE PRESUPUESTO..... 61

1. INTRODUCCIÓN

1.1 Motivación del trabajo

El siguiente trabajo consiste en el desarrollo de un sistema para el control de la cámara integrada en un dron mediante la detección de la mirada. Los drones tienen cada vez más variedad de usos y aplicaciones, se utilizan por ejemplo para captar datos [1], realizar fotografías, entregar paquetes [2] e incluso en agricultura [3]. Suelen estar controlados por un operario con un mando a distancia y muchos de ellos tienen cámaras incorporadas para realizar este tipo de tareas. La motivación de este trabajo surge cuando se plantea la necesidad de mover la cámara a la vez que se desplaza la máquina. Actualmente esto se consigue con mandos que permiten mover con unos controles al dron y con otros la cámara, sin embargo, esto no termina de ser cómodo para el usuario ya que tiene que estar pendiente de ambos controles. Además, el usuario siempre está buscando la forma de hacer más inmersiva la experiencia de pilotar un dron, es por ello por lo que se han popularizado las gafas de Realidad Virtual [4] pues permiten tener una experiencia en primera persona de lo que está viendo el dron . Este método sin embargo sigue utilizando los controles convencionales para mover la cámara por lo que, aunque gires la cabeza con las gafas puestas, seguirás viendo la misma imagen.

Con el siguiente trabajo se pretende solucionar esta problemática mediante el uso de técnicas de visión por ordenador para poder controlar el movimiento de la cámara con la dirección de la mirada. De este modo el control de drones usados para tomar imágenes desde el aire será mucho más inmersivo y sencillo para el usuario ya que solo tendrá que dirigir la mirada hacia donde quiere que se desplace la cámara. A parte, la investigación en el campo del seguimiento de la mirada tiene un gran interés en otras aplicaciones, no solo en los drones. Los SAAC o Sistemas Avanzados de Asistencia al Conductor por ejemplo, están estudiando el empleo de técnicas del seguimiento de la mirada para detectar cuando el conductor está cansado o incluso ha cerrado los ojos y se está quedando dormido, para así poder asistir al conductor en estas situaciones y preservar su seguridad [5]. Otro ejemplo del empleo de esta técnica es la corrección de la dirección de la mirada durante las videoconferencias [6], se detecta la mirada del usuario y se manipula de tal forma que parezca que la persona está mirando directamente a la Webcam en lugar de a su pantalla, lo que mejora la comunicación en este formato. La

investigación en esta área comenzó con técnicas que requerían de hardware unido a la piel para detectar la mirada, como electrodos en los métodos de Electrooculografía, pero actualmente la Inteligencia Artificial permite desarrollar aplicaciones de este tipo sin necesidad de hardware costoso, únicamente con un cámara se consigue detectar la mirada de una persona [7]. Con este trabajo se quiere investigar las mejores técnicas de seguimiento de la mirada para aplicarlas al control de la cámara de un dron.

La importancia de este proyecto también reside en la reducción de costes, ya que con este desarrollo se elimina el error humano a la hora de controlar la cámara, el usuario no tendrá que ser consciente de hacia dónde desplaza la cámara sino simplemente mirar hacia donde quiere que apunte la cámara. Con los controles manuales se pueden cometer errores, consumiendo recursos como las horas de grabación en el campo necesarias para completar un proyecto, así como las horas en postproducción eliminando tomas fallidas. Además, permite tener una experiencia más inmersiva sin la necesidad de comprar unas gafas de Realidad Virtual que tienen normalmente un coste bastante elevado [8].

El hecho de poder mover la cámara según hacia donde mira el operario también crea una situación más inmersiva para el control del dron. Lo que hace que esta tarea se disfrute mucho más, ya que al girar la vista puedes ver lo que hay en esa dirección como si el operario estuviera también en el aire. Esto es muy valorado por las personas que pilotan drones como hobby o entretenimiento, o incluso en las carreras competitivas de estos vehículos.

Como se puede ver no faltan razones para la implementación y la realización de este proyecto ya que soluciona una problemática real en este sector. Además, como se ha explicado antes, a pesar de que este trabajo se centra en una aplicación concreta, el control mediante la mirada desarrollado aquí puede tener aplicaciones en otros proyectos ya que no deja de ser una forma de controlar dispositivos, un mando a distancia que utiliza la mirada en vez de unos controles de mano.

Por otro lado, también existe una motivación didáctica. Con este proyecto se pretende profundizar en los conocimientos adquiridos en la carrera como la programación, el diseño de sistemas electrónicos y la automatización de tareas. Conseguir dirigir el

movimiento de una cámara con los ojos parecía una tarea imposible antes de ingresar en la carrera, pero ahora se trata de saber desplegar un servidor, programar un software que utilice visión por computadora y controlar el movimiento de un servomotor con una Raspberry Pi. La Inteligencia Artificial tiene cada vez más aplicaciones y es más accesible, por lo que aprender a utilizar herramientas de visión por ordenador permite adquirir experiencia en este tipo de desarrollos cada vez más demandados [9]. OpenCV es una herramienta muy utilizada por las empresas que se dedican a la visión artificial, y además Python es el lenguaje de programación más demandado actualmente. Este trabajo permitirá al alumno utilizar herramientas que podría incluir en su currículum más adelante.

1.2 Objetivos

Con este trabajo se persigue una primera aproximación al desarrollo de un sistema que sea capaz de controlar remotamente la cámara integrada en un dron mediante la dirección de la mirada, es decir que el operario que pilota el dron pueda desplazar la cámara de éste únicamente con el movimiento de sus ojos. Tras un estudio previo de la cuestión se tratará de determinar tanto si es viable un desarrollo de estas características como si es útil esta tecnología en el campo de los drones. De cumplirse estas dos premisas se estudiará la posibilidad de desarrollar un prototipo inicial, tanto del software de visión artificial, como del sistema completo de comunicación, que permita implementar el proyecto con el mínimo coste y tiempo de realización, es decir un MVP (*Minimum Viable Product*) en español un Producto Mínimo Viable. A esto se sumaría un estudio sobre el impacto social y económico de esta implementación que también serviría para determinar la viabilidad del diseño.

El objetivo del MVP será el de cumplir unas características mínimas establecidas por el cliente, las cuales serán:

- Que el software sea accesible desde un smartphone.
- Conseguir una transformación de las coordenadas de los ojos a coordenadas de los servos que mueven la cámara.
- Que la comunicación se realice mediante Wifi.

- Que la latencia sea mínima de tal forma que el usuario no la note, y pueda obtener una respuesta casi instantánea al movimiento de sus ojos. Lo que se resume en que el diseño sea funcional.
- Que se utilice como hardware para el prototipo una Raspberry Pi y unos servos sg90 en una estructura de *pan and tilt* (horizontal y vertical) para mover la cámara en dirección vertical y horizontal.

Para el diseño de este MVP se realizará una investigación detallada para determinar el mejor software de visión por computadora y las mejores herramientas y tecnologías para la conexión entre el smartphone y la Raspberry Pi. Además, se realizará el análisis del hardware necesario. Para facilitar el desarrollo, este prototipo estará desacoplado del dron y se realizarán las pruebas de la misma manera, es decir que las Raspberry Pi y los servos con la cámara no estarán montados en un dron. Igualmente, los servidores estarán desplegados en una red local, por lo que la aplicación solo estará disponible para aquellos smartphones o dispositivos conectados a la misma red local.

Este prototipo sentaría las bases para un diseño más avanzado o comercial para el cual se utilizarían mejores componentes, como unos servos mejores y más robustos con mayor rango de movimiento, ya que los sg90 solo pueden girar 180 grados. Además, se podría hacer en futuros trabajos todo el montaje en el dron y realizar pruebas en el aire con un dron real. Los servidores se podrían desplegar en algún servicio en la nube que tenga CPU más potentes o puedan aguantar más llamadas simultaneas al servidor, esto haría que fuesen por fin accesibles desde cualquier lugar del mundo con una conexión a Internet.

Para determinar si se podría llegar a este diseño más avanzado, primero se desarrollaría el prototipo con limitaciones en el coste y tiempo de desarrollo. En este producto mínimo el objetivo sería realizar una serie de pruebas para determinar el rendimiento del software de visión artificial, así como la latencia del sistema completo, para poder analizar si es un diseño funcional o no. A parte estas pruebas servirían para sacar conclusiones y encontrar mejoras de cara a una implementación más avanzada y de mayor coste.

1.3 Estructura del proyecto

El proyecto se estructura de la siguiente manera:

Capítulo 1: Introducción.
En esta parte del proyecto se contextualiza el contenido de la obra. Se presentan brevemente la motivación del trabajo, los objetivos que se quieren alcanzar y la estructura general del proyecto. También se presenta el marco regulador que se tendrá en cuenta para la realización del trabajo.

Capítulo 2: Fundamentos Teóricos y Estado del Arte.
Con este capítulo se presenta el estado actual de la tecnología Se trata de una revisión de las referencias bibliográficas que se han encontrado y de las que se han partido para la realización del proyecto. Además de una valoración del estado del tema. Se ha querido dividir en una introducción a los drones, la Inteligencia Artificial y distintas herramientas que podrían utilizarse para la implementación, pasando por el estado de la detección facial y los puntos de referencia faciales.

Capítulo 3: Diseño y desarrollo.
En este apartado se describe la solución final al problema, las tecnologías aplicadas y la arquitectura de software, la selección de los lenguajes de programación y los elementos de hardware empleados para diseñar la solución. Está dividido en cuatro partes: los requisitos del sistema, el resumen del diseño, la arquitectura del software y finalmente la implementación del diseño.

Capítulo 4: Resultados.
En el capítulo 4 se presentan los resultados obtenidos con la implementación del diseño. El rendimiento real de la aplicación desarrollada y los datos numéricos que determinan si la solución aportada es válida y funcional.

Capítulo 5: Impacto socioeconómico.
Aquí se describe el impacto social y económico del trabajo realizado. Los recursos necesarios para la implementación del proyecto y el impacto en las personas y el medioambiente.

Capítulo 6: Presupuesto.
En este apartado se hace un desglose del presupuesto total del desarrollo. Teniendo en cuenta todas las piezas de hardware necesarias, así como las horas de trabajo del personal las cuales incluirán la investigación previa para ver la viabilidad del producto, así como el diseño e implementación de este.

Capítulo 7: Conclusión y futuros trabajos.
En la conclusión se valora el trabajo realizado en función de los objetivos alcanzados con la solución propuesta. También se incluyen soluciones y mejoras para las partes del proyecto que no han alcanzado los objetivos deseados, así como los trabajos futuros.

1.4 Marco regulador

Hasta hace poco no existía una regulación o legislación oficial sobre la Visión Artificial en España. Recientemente, el 21 de abril de 2021 la Comisión Europea ha presentado la legislación para el uso de la Inteligencia Artificial [10]. Es la primera regulación oficial sobre el tema y se tendrá en cuenta para la realización del presente trabajo.

Esta propuesta establece niveles de riesgo de la Inteligencia Artificial en función de su amenaza para la seguridad y los derechos de las personas. Aquellas aplicaciones que se encuentren dentro del "riesgo inaceptable" estarán prohibidas en la Unión Europea. Los distintos niveles serían los siguientes:

Riesgo inadmisible: Se trata de usos nocivos de la IA que violen los derechos fundamentales de las personas, todo lo que se encuentre dentro de este nivel de riesgo estará prohibido.

Alto riesgo: Se considera de alto riesgo las aplicaciones de la IA que tienen un impacto negativo en la seguridad de las personas o en sus derechos fundamentales.

Riesgo limitado: Se incluyen aquí los sistemas que tienen obligaciones de transparencia, por ejemplo, cuando existe un riesgo de manipulación, el usuario debe estar informado de este riesgo.

Riesgo mínimo: Todos los demás sistemas de IA podrán utilizarse con normalidad como son las aplicaciones en videojuegos o aplicaciones de imagen. Tendrán la opción de cumplir unos códigos de conducta de una IA digna de confianza.

Es dentro de este último nivel de riesgo donde se encuentra la aplicación web planteada en este proyecto por lo que la normativa no especifica ninguna medida específica a seguir. La aplicación no violará los derechos humanos ni pondrá en riesgo la seguridad del usuario, además no tiene ninguna aplicación que pueda ser nociva para las personas. Esta Ley prohíbe el uso del reconocimiento facial en zonas públicas, lo cual podría pensarse que afecta a este proyecto. Pero no se debe olvidar la diferencia entre Reconocimiento y Detección faciales, el primero se trata de reconocer a personas específicas a partir de imágenes y diferenciarlas del resto de personas, mientras que el segundo se trata de detectar a personas en imágenes, igual que se pueden detectar otros objetos en las imágenes sin determinar de quién se trata, únicamente reconocer la etiqueta “persona”. Por lo que la detección facial utilizada en el desarrollo de este proyecto está permitida sin ninguna regulación especial.

Aparte, también se aplica la Ley de Protección de Datos [11], ya que se estarán recogiendo imágenes del usuario que pueden comprometer su privacidad, así como la cámara del dron podría estar grabando a personas. Todo esto implica el tratamiento de unos datos personales. Por tanto, toda empresa que utilice esta tecnología y esté tratando las imágenes de una persona deberá cumplir con la normativa vigente sobre tratamiento y protección de datos que se incluye en el RGPD (Reglamento General de Protección de Datos).

La empresa responsable del tratamiento de los datos deberá informar a los usuarios conforme a los arts. 13 y 14 del RGPD sobre todos los apartados que se mencionan en los artículos, como son por ejemplo los datos de contacto del responsable o el plazo durante el cual se guardarán los datos.

2. FUNDAMENTOS TEÓRICOS Y ESTADO DEL ARTE

2.1 Drones introducción y situación actual

Los drones son vehículos aéreos no tripulados controlados remotamente, pueden llevar a cabo tareas tan simples como hacer unas fotografías o grabar unas imágenes [3], sin embargo, su gran utilidad reside en que quizá estas fotografías son en zonas de desastres naturales ultrapeligrosos como un incendio o una erupción volcánica. Pueden ser controlados remotamente, pudiendo volar a distintas alturas y por bastante distancia, lo que los hace perfectos candidatos a algunas de las tareas más duras y extremas para los humanos. Este tipo de robots los puedes encontrar en todas partes, desde una zona de rescate de víctimas hasta en la puerta de tu casa dejándote un paquete que has pedido por Internet [2].

Los drones llevan años siendo usados con fines militares pues es para lo que originalmente fueron diseñados [12], realizan tareas de gran relevancia en el ejército siendo los ojos en las operaciones militares o sustituyendo a los vehículos aéreos militares para no poner en riesgo vidas humanas. Sin embargo, su eficiencia y seguridad los ha llevado a instaurarse en nuestro día a día desarrollando todo tipo de tareas cotidianas, como transportar pequeños paquetes o realizar fotografías y vídeos aéreos. Pero también realizan muchas otras tareas como monitorización de ganado, búsqueda y salvamento, transporte de mercancías, inspección de construcciones, mapeado de zonas inaccesibles, sembrado de campos, fumigación, reparto de medicamentos, seguimiento de tormentas o recogida de información de sensores.

Los UAV (*Unmanned Aerial Vehicles*) como su nombre en inglés indica operan sin un piloto a bordo, pero esto no quiere decir que no haya humanos implicados en su control. La autonomía de estos vehículos puede variar enormemente, variando desde el control remoto hasta pudiendo encontrar drones que no necesitan de un humano para pilotarlos

ya que se orientan gracias a una serie de sensores que llevan integrados. No obstante, la gran mayoría de ellos necesitan al menos a un operario para controlar su movimiento, lo que significa que debe haber una persona en todo momento manejando unos mandos.

Aunque aún están lejos de no necesitar la ayuda humana, el futuro de estos dispositivos pasa por realizar tareas cada vez más sofisticadas con la mayor autonomía posible. Es por ello por lo que los investigadores se centran en programar los drones para tareas complejas que requieran de la mínima intervención humana. Es aquí donde entra otra tecnología con la que los drones se complementan perfectamente para ganar la máxima autonomía, la visión por ordenador. En los siguientes apartados se realizará una breve introducción al mundo de la Inteligencia Artificial y a su aplicación en los vehículos no tripulados.

2.2 Inteligencia Artificial

2.2.1 Introducción

Según la definición de Bruno López Takeyas del Instituto Tecnológico de Nuevo Laredo “La IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: el razonamiento y la conducta.” [13]. La Inteligencia Artificial trata de resolver problemas mediante algoritmos que imitan el razonamiento humano utilizando algoritmos computacionales. Por ejemplo, permite que un ordenador o una máquina procese imágenes y las “vea”, es decir que procese y clasifique los distintos objetos que se encuentran en la imagen, tal y como lo haría una persona. Esta es solo una de las tantas ramas de la Inteligencia Artificial, la Visión por Computadora. Pero, así como la Inteligencia Artificial permite imitar en este caso la visión, también se encuentran algoritmos que pueden resolver problemas matemáticos que no han visto nunca, gracias a que son capaces de aprender a partir de otros datos y extrapolarlos a situaciones y problemas nuevos. Esta es la mayor ventaja de la Inteligencia Artificial, que no es necesario programar literalmente la tarea que se quiere realizar con ella, sino que basta con enseñarle a tomar sus propias decisiones, es decir programar el método de razonamiento que le permitirá después de forma autónoma llegar a una solución [13].

Como se ha mencionado antes una rama de la Inteligencia Artificial es la Visión Artificial, la cual nos puede permitir encontrar una solución para uno de los objetivos de este proyecto: la detección del movimiento de la mirada.

2.2.2 Visión Artificial

La visión artificial consiste en la extracción automatizada de información de las imágenes [14]. La visión para los humanos es la forma principal de obtener información sobre el entorno, permite reconocer formas y colores y clasificarlos y compararlos con otros. Programar un dispositivo para que tenga esta misma capacidad de recoger y clasificar información es lo que sería la Visión por ordenador, cuando se generan algoritmos para imitar este sentido humano a partir de imágenes y vídeos.

La visión por ordenador tiene ya 40 años de investigación a sus espaldas [15] , principalmente en torno a imitar la visión humana, como por ejemplo siendo capaz de reconocer objetos y personas y clasificarlos. El reconocimiento facial en concreto tiene una extensa bibliografía. Esta disciplina consiste en reconocer mediante algoritmos los rasgos faciales específicos de una persona y ser capaz de determinar con una alta precisión cuando aparece esta persona en una imagen. No debe confundirse con la detección de personas cuyo objetivo es reconocer personas en imágenes, es decir, diferenciar a un humano de todo lo demás sin caer en diferenciar a una persona de otra. La gran utilidad que tiene esta rama de investigación la podemos observar simplemente tomando nuestro smartphone y desbloqueándolo con nuestra cara. Al mismo tiempo podemos comprobar cómo de avanzada está esta tecnología, ya que, si otra persona intenta desbloquear nuestro móvil de esta forma, el dispositivo lo rechazará.

2.2.3 Aplicaciones en los drones de la Visión Artificial

Si mezclamos esta capacidad para analizar el entorno a través de código junto con un vehículo que es capaz de acceder casi a cualquier sitio tenemos una herramienta increíblemente potente que puede realizar diversas tareas de muchísima utilidad sin consumir excesivos recursos. La visión por computadora hace que los drones den un salto en autonomía y usos.

En concreto algunas de las aplicaciones de este campo de la inteligencia artificial en los drones serían el rastreo de objetos [1], la navegación autónoma y la detección de obstáculos para evitar colisiones [16]. Sin embargo, como dijimos anteriormente, todavía están lejos estas tecnologías de la autonomía total en el vuelo y lo normal sigue siendo tener un mando entre las manos para dirigir el vuelo. Es por ello por lo que la visión artificial tiene también aplicaciones no solo en el vuelo sino también a la hora de controlar el dron de forma remota.

En el estudio realizado por Y. Zhou, J. Hou y. Gong se consigue implementar mediante Inteligencia Artificial un control remoto del dron mediante la voz [17]. Utilizando algoritmos de reconocimiento del lenguaje consiguen dar instrucciones mediante la voz al dron para que se mueva en el aire. Esto permite que el operario del dron tenga las manos libres para realizar otras tareas, lo que tiene gran relevancia en el área de las operaciones militares ya que el operario tiene que estar atento a todos los cambios y los imprevistos de la operación y una interacción más sencilla como puede ser la comunicación por voz con el dron es más eficiente y no requiere una demanda tan grande física y psicológica para el operario.

Del mismo modo la visión por ordenador nos puede permitir una interacción con la máquina más eficiente y natural para el usuario, esto es lo que se pretende con la aplicación de la detección de objetos al control del vuelo de los drones.

2.2.4 Detección facial

Todos hemos probado los filtros de Realidad Aumentada que ofrecen ahora las Redes Sociales más famosas. Esta es solo una de las muchas aplicaciones de la detección y el reconocimiento facial.

Para empezar, la detección y el reconocimiento son dos cosas distintas y no deberían confundirse. La primera consiste en determinar si hay una o más caras en una imagen o video, y distinguirla del resto de objetos que compongan la imagen ya sean objetos y otros seres vivos. Por otro lado, el reconocimiento facial sería la disciplina que identifica los rasgos faciales característicos de una persona o personas en una imagen y la diferencia del resto de humanos. Estas personas deben haber sido previamente

introducidas en el sistema del algoritmo para poder identificarlas. Por lo tanto, lo que hace es comparar lo que está reconociendo con la información que ya tiene.

En esta memoria nos centraremos en la detección facial ya que el objetivo del proyecto es detectar los ojos de una persona sin concretar quién es esta persona para poder controlar el movimiento de los servos que lleva el dron.

Para que una computadora pueda distinguir a las personas de los distintos objetos en una imagen e incluso del fondo de la imagen, Yang, Kriegman, y Ahuja presentaron diferentes métodos de clasificación en su publicación *Face Detection: A Survey* [18] . Los métodos basados en las características y los métodos basados en imágenes.

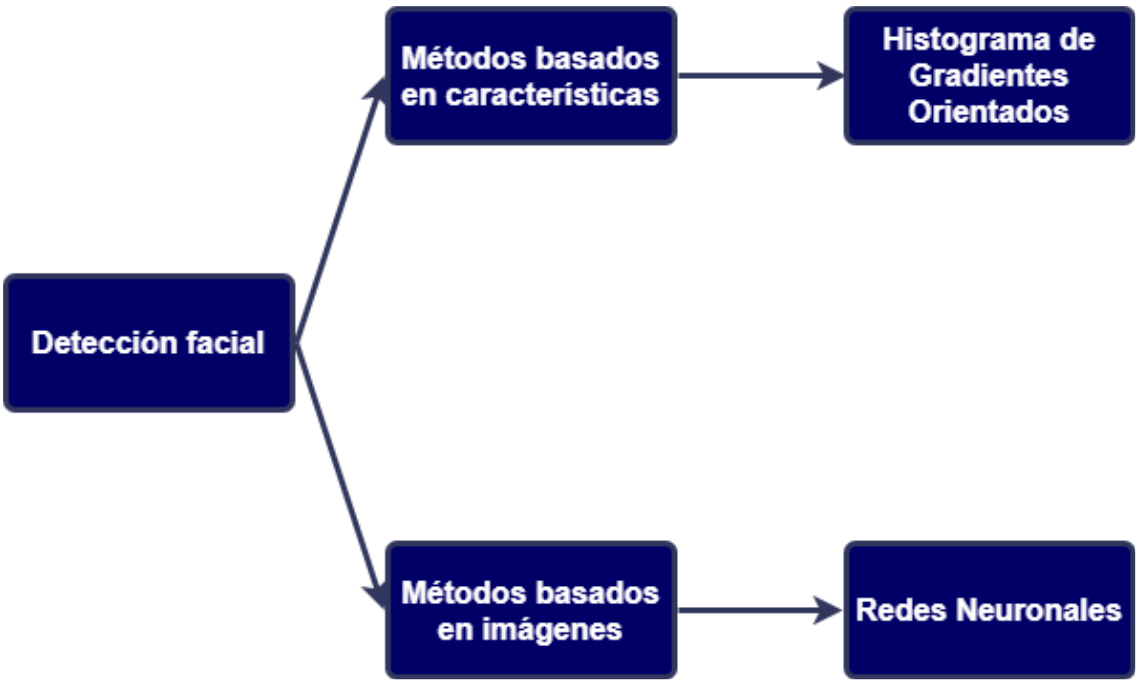

Figura 1. Clasificación de los métodos más populares de detección facial

El método basado en las características o *feature-based* trata de extraer de las imágenes distintas características como la distancia entre los ojos, el tamaño de estos, donde está la boca, la nariz o los colores y sombras. Este método permite ir haciendo descartes de aquellas zonas de la imagen donde claramente no se encuentran estas características de la cara humana. Se basa en entrenar clasificadores para detectar estas estructuras que determinan las zonas donde existen caras.

El problema con el empleo de la detección de características es que la apariencia de la cara puede tomar distintos ángulos y apariencias en incluso el fondo puede confundirse con rasgos de la cara. Esto limita este método a imágenes frontales con unas buenas condiciones de luz y donde se vean la cara y los hombros. Algunas técnicas más avanzadas de este método permiten solventar este problema cuando el fondo no está claro o las condiciones de luz no son las ideales.

2.2.4.1 Histograma de Gradientes Orientados

Una de estas técnicas es el Histograma de Gradientes Orientados (HOG), un extractor de características usado para problemas de optimización, detección de patrones y visión por ordenador. Se trata de crear un histograma a partir de la dirección de los gradientes que componen una imagen. De esta forma la imagen se divide en celdas y se calcula la dirección dominante de los gradientes en cada zona de la imagen.

Figura 2. Ejemplo descriptor HOG. Cada celda indica la orientación de los gradientes **[19]**

Los gradientes representan la estructura de bordes y contornos, permitiendo así detectar las distintas regiones, las sombras y luces y los segmentos de la imagen. En vez de analizar la intensidad de cada píxel se calcula el gradiente, esto es así para solucionar el

problema que explicamos antes sobre detectar características faciales en fondos con malas condiciones de luz.

Al analizar los píxeles, la misma cara puede tener valores muy distintos según el brillo de la imagen, si es más oscura o clara el resultado puede ser completamente distinto aun siendo la misma imagen, pero calculando el gradiente, es decir la dirección en la que cambia el brillo, las imágenes más claras y oscuras tendrán los mismos valores en el histograma y se podrá determinar más fácilmente las características de la cara sin el problema de las distintas condiciones de luz.

Aparte, con un Histograma de Gradientes Orientados no es necesario guardar la información píxel por píxel, sino que se pueden coger secciones de la imagen, es decir, cuadrados de píxeles que nos den suficiente información sobre cada zona. Esto simplifica las imágenes y el problema, además aumenta la velocidad de detección. A partir de esta segmentación simplemente habría que comparar el patrón encontrado con los gradientes con la nueva imagen en la que queremos detectar la cara de una persona. Los patrones serán similares sin importar las distintas condiciones de luz. El Histograma de Gradientes Orientados tienen un alto rendimiento en tareas de detección como se presenta en el estudio realizado por L. R. Cerna, G. Cámara-Chávez y D. Menott [20].

2.2.4.2 Machine Learning y Redes Neuronales

Aunque los métodos basados en características sirven en la tarea de detección facial, a nivel de rendimiento los métodos basados en imagen sobrepasan a cualquier otro ya que utiliza *machine learning* y estadística para detectar patrones y reconocer caras.

Cuando se enfrenta el problema de la detección facial mediante imágenes deja de ser necesario el conocimiento sobre las características concretas de la cara ya que este método se centra en reconocer patrones. No hay error de modelaje por no haber determinado correctamente las características determinantes de una cara, sino que se entrenan los modelos a partir de un montón de ejemplos y contraejemplos. La detección se realiza comparando contra los patrones que se encontraron en el conjunto de datos de entrenamiento.

Es aquí donde entran las Redes Neuronales, la herramienta más potente actualmente para los problemas de detección facial. Estas fueron propuestas inicialmente en 1944 por Warren McCullough y Walter Pitts. Desde ese entonces las redes neuronales han tenido sus altos y bajos en popularidad y actualmente gracias a la gran potencia de computación están superando a cualquier otra técnica de reconocimiento facial, de hecho, es la tecnología que podemos encontrar en por ejemplo el FaceID de Apple. Las redes neuronales son una técnica de Machine Learning la cual está compuesta por multitud de nodos organizados por capas, cada uno de los cuales tiene asignado un peso. Estos nodos reciben datos de otros nodos y los multiplican por su peso, el producto final se compara entonces con un valor umbral que determina si este resultado debe pasar a la siguiente capa de nodos.

Cuando se entrena una red neuronal todos los pesos se establecen con valores aleatorios, los datos para el entrenamiento son introducidos en la capa más baja y de ahí va pasando a las demás capas donde se van realizando las operaciones matemáticas hasta llegar a un resultado en la última capa. Estos pesos son ajustados durante el entrenamiento para después ser utilizados en la fase de inferencia donde se aplica lo aprendido a un nuevo conjunto de datos.

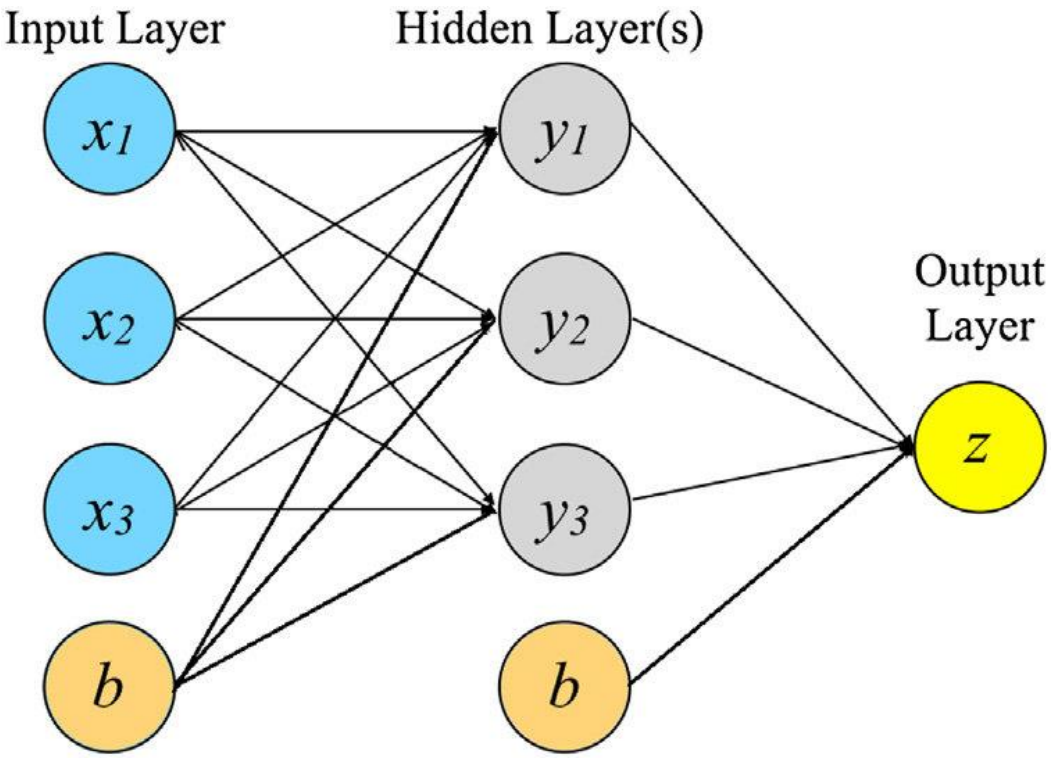

Figura 3. Una red neuronal artificial con tres inputs y un output **[21]**

2.2.4.3 Puntos de referencia faciales

Ya hemos visto distintos métodos de detección facial, pero queda profundizar en la tarea propuesta y las soluciones aplicables a ella. Uno de los objetivos de este proyecto es detectar la dirección de la mirada, por lo que la detección facial será importante, pero aparte hay que determinar la posición de los ojos dentro de la imagen y ser capaces de seguirla. Es aquí donde entran los puntos de referencia faciales o *landmarks* [22], en la detección facial por características ya explicamos cómo se puede afrontar el problema de detectar el rostro en imágenes buscando características concretas que lo definen, una forma de determinarlas es mediante referencias o etiquetas que las identifiquen. Estos puntos de interés pueden ser los ojos, la nariz, las cejas, es decir, estructuras faciales relevantes. Los puntos de referencia faciales son la determinación, localización y seguimiento de puntos en la cara que definen alguna de estas estructuras faciales. Este proyecto de reconocimiento facial trata de determinar unos puntos de referencia que serán en este caso los ojos, para proyectar la dirección de la mirada al movimiento de los servos del dron.

Podemos encontrar 2 clasificaciones principales en los puntos de referencia faciales. Los puntos de referencia primarios y secundarios, los primarios tratarán de determinar las referencias más importantes en el rostro como pueden ser las esquinas de la boca y los ojos, es decir, los puntos más fácilmente localizables con cualquier técnica de detección facial. Los secundarios, sin embargo, no son puntos extremos, sino que son más sutiles o están en mitad de puntos primarios, como pueden ser los puntos medios de los labios, la barbilla, las mejillas. Estos últimos puntos son más difíciles de localizar, pero son los más relevantes a la hora de determinar expresiones faciales y emociones.

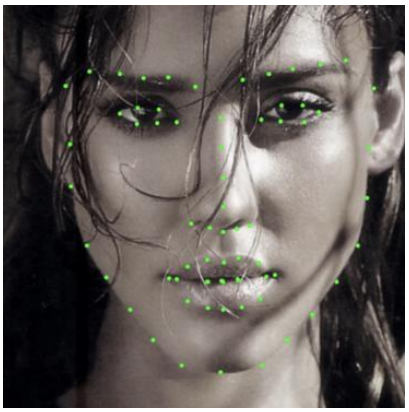

Figura 4. Ejemplo de detección de puntos de referencia faciales **[23]**

Los puntos de referencia faciales tienen muchas aplicaciones actualmente, de hecho, están detrás de los famosos filtros faciales de las redes sociales, en los cuales podemos apreciar cómo la inteligencia artificial determina los límites de nuestros rasgos faciales para adaptar el filtro a distintas formas de cara y seguir los movimientos de los ojos, la cabeza y los labios.

Esta técnica es exactamente lo que necesitamos en este proyecto para seguir el movimiento de los ojos, sin embargo, tienen algunas desventajas. Una de ellas es el tiempo, ya que la máxima utilidad de esta técnica reside en que se pueda utilizar en tiempo real para seguir el movimiento de los puntos. Esto quiere decir que se necesita una gran potencia de computación que pueda realizar este procesamiento sin demasiada latencia, esto limita este método a dispositivos potentes. Por otro lado, las condiciones del entorno también pueden condicionar los resultados ya que si son muy diferentes a las de las imágenes de entrenamiento puede verse afectado el rendimiento. Las poses extrañas o las malas condiciones de luz pueden reducir la precisión de los puntos, como se puede apreciar en los filtros de Realidad Aumentada cuando giras la cara o la pones en poses extrañas.

Estas desventajas de los puntos de referencia no serán un problema en este desarrollo ya que la cámara estará siempre posicionada de forma frontal a la cara de la persona, además como los drones se utilizan al aire libre y de día, siempre habrá buenas condiciones de luz. El único inconveniente que se podría apreciar es la potencia de computación, sin embargo, al ser una aplicación web, el algoritmo de detección estará situado en la parte del servidor, por lo que se puede correr el modelo en un servidor potente o en alguna máquina virtual en la nube sin que se requiera un gran ordenador en la parte del cliente. Una Raspberry Pi con cámara, un móvil o una Tablet será suficiente para acceder a la aplicación de detección facial.

2.2.5 Comparación de distintas implementaciones de la Detección Facial

Tras haber estudiado los distintos acercamientos a la detección facial y haber encontrado una técnica aplicable al problema que se trata en este proyecto, el siguiente paso lógico es analizar las distintas formas en que se pueden aplicar los puntos de

referencia faciales. Existen cantidad de lenguajes de programación, librerías y algoritmos para realizar el desarrollo de los puntos de referencia faciales.

En cuanto a los lenguajes de programación, Python es el lenguaje de programación más utilizado en proyectos de Inteligencia Artificial y Machine Learning ya que tiene una gran cantidad de librerías dedicadas a este fin como Pandas, Numpy. Keras, Scikit, entre otras, y además es sencillo y tiene una gran comunidad de ingenieros desarrollando con este lenguaje lo que hace que haya gran cantidad de bibliografía, cursos y ayudas para utilizar este lenguaje [24]. Además, tiene muchas oportunidades laborales ya que es uno de los principales lenguajes utilizados en grandes compañías como Google, Facebook o Instagram [25].

La precisión del modelo será otro requerimiento importante ya que será determinante para que la funcionalidad deseada se cumpla, que la cámara del dron se mueva en la dirección correcta. Sin embargo, una mayor precisión puede llevar a una mayor potencia de computación de la aplicación o un tiempo de procesamiento mucho mayor por lo que también se deberá encontrar un balance entre precisión, velocidad y potencia.

2.2.5.1 Tensorflow

La búsqueda para el análisis comienza mediante la búsqueda de herramientas que permitan entrenar modelos de Aprendizaje Profundo que tuvieran en cuenta la posición de la cabeza y estuvieran entrenadas para detectar la dirección de la mirada, no solo la circunferencia del ojo. Como ya se explicó en el punto 2.2.4.2 los métodos de machine learning y redes neuronales superan a cualquier otro, por lo que primera se apuntó en esta dirección.

De acuerdo con la página oficial “TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que permite que los investigadores innoven con el aprendizaje automático y los desarrolladores creen e implementen aplicaciones con tecnología de AA fácilmente.” [26]. Esta herramienta es ampliamente conocida y utilizada en el mundo de la Inteligencia Artificial y la Visión

Artificial. Tiene cantidad de herramientas que permiten la fácil y rápida implementación de soluciones mediante algoritmos de machine learning.

Además existe Tensorflow.js que permite utilizar el framework de aprendizaje profundo Tensorflow en una página web mediante JavaScript, además existe un convertidor para pasar un modelo ya entrenado con el módulo de Tensorflow en Python a Tensorflow.js [27].

Estas ventajas en cuanto a su potencia en tareas de Visión Artificial y su framework para Web hace que sea una opción muy a tener en cuenta para la implementación de la solución del software de detección de la mirada.

2.2.5.2 OpenCV y Dlib

Debido al gran soporte que tiene Python en el área de la Inteligencia Artificial, no es complicado encontrar librerías especializadas para la tarea que se quiere llevar a cabo. OpenCV es una biblioteca libre de visión artificial originalmente desarrollada por Intel [28]. Es muy utilizada para proyectos de visión por ordenador ya que es potente y libre, está disponible para muchos lenguajes de programación y sistemas operativos, es gratuita y tiene una comunidad de más de 47 mil personas, una gran cantidad de repositorios en GitHub y de tutoriales. Al ser libre, es utilizada en grandes compañías para tareas de visión por ordenador, además entre todos los algoritmos de machine learning y visión por computadora que incluye se encuentran tanto clásicos como estados del arte, por lo que no se queda atrás. No obstante, no permite entrenar modelos de aprendizaje profundo por lo que para aplicaciones de aprendizaje profundo y redes neuronales se usa para el procesamiento previo de las imágenes. En este proyecto OpenCV se va a utilizar para el procesamiento de las imágenes ya que la librería Dlib ofrece un detector de puntos de referencia faciales preentrenado capaz de localizar 68 puntos de referencia. Esta librería es la más popular para la detección de puntos de referencia faciales debido a su velocidad y fiabilidad, a pesar de que en tareas de reconocimiento OpenCV lo sobrepasa en rendimiento la tarea que nos ocupa es de detección [29].

Entre los algoritmos de detección de puntos de referencia faciales los dos más conocidos y utilizados son STASM y Dlib. Stasm es un software en C++ que permite localizar estos puntos de referencia en caras. Está diseñado para trabajar con vistas frontales y expresiones neutras lo cual no sería un problema para este proyecto ya que todas las imágenes que se pretenden tomar será de una vista frontal de la cara [30].

Por otro lado, Dlib es un framework escrito en C + + que contiene algoritmos de machine learning preentrenados y herramientas que permiten desarrollar softwares complejos de forma sencilla. Entre ellos se encuentra el detector de 68 puntos de referencia faciales. Un algoritmo preentrenado que permite detectar hasta 68 puntos de referencia, aunque en esta aplicación solo serían necesarios los puntos de los ojos para determinar la dirección de la mirada. Este algoritmo es una implementación de la publicación *One millisecond face alignment with an ensemble of regression trees* [31] para la estimación de la pose, más el uso de un Histograma de Gradientes Orientados clásico.

A pesar de que ambos algoritmos podrían servir para la implementación de este proyecto, de acuerdo al estudio realizado por Wouter Pool [32] Dlib muestra mejores resultados que STASM, ya que esta última a veces daba falsos positivos y encontraba caras donde no las había mientras que Dlib tuvo buenos resultados en todas las pruebas. Es por eso por lo que para la implementación de este proyecto se usaría Dlib antes que STAM para el desarrollo del software de detección.

La combinación de OpenCV y Dlib permitiría hacer un desarrollo fiable y rápido para determinar la dirección de la mirada para más tarde enviar las coordenadas a los servos que mueven la cámara.

2.2.5.3 Conclusiones

Las redes neuronales y los algoritmos de machine learning como los que implementa Tensorflow pese a ser muy potentes a nivel computacional no son siempre la solución más adecuada, ya que tienen algunas limitaciones, siendo la principal que requieren de mucha potencia de computación y, por lo tanto, de recursos económicos. La falta de esta potencia incrementa el tiempo tanto de desarrollo como de latencia a la hora de

utilizarlo y hace que esta tecnología pierda puntos frente a otras que, aunque no sean tan precisas y efectivas, son más rápidas cuando se tienen recursos limitados y además son más fáciles de desarrollar e implementar.

Es decir, una gran compañía y puede permitirse la gran infraestructura que se necesita para implementar las redes neuronales, sin embargo, la mayoría de las veces este no será el caso. Es por ello por lo que en este trabajo no se han utilizado redes neuronales con Tensorfllow sino técnicas más asequibles que se emplean en librerías como OpenCV y Dlib.

OpenCV es una librería de código abierto que proporciona un entorno de desarrollo sencillo, entendible y accesible para desarrollar aplicaciones de visión artificial [28]. Gracias a OpenCV se pueden desarrollar prototipos de aplicaciones de visión por ordenador con un bajo presupuesto. Al ser este desarrollo un producto mínimo para ver la viabilidad del proyecto, esta librería es la mejor opción y la que finalmente se ha determinado usar junto con Dlib aplicando el método de los puntos de referencia faciales.

Para detectar la cara se usará la función *get_frontal_face_detector* incluida en la librería Dlib. Esta función implementa un Histograma de Gradientes Orientados para determinar dónde se encuentra la cara de la persona en la imagen. Una vez detectada la cara con la función *shape_predictor* se determinan los 68 puntos de referencia faciales, de los cuales solo los puntos de los ojos y la pupila se tendrán en cuenta para determinar la dirección de la mirada.

2.3 Análisis de distintas herramientas para la implementación del sistema

En este apartado se estudiarán distintas herramientas aplicables a la implementación del trabajo. En cada parte se explicarán los distintos enfoques posibles para cada parte del desarrollo y la solución escogida finalmente para su implementación, teniendo en cuenta coste, dificultad y accesibilidad.

2.3.1 Aplicación Web vs Aplicación Móvil

El objetivo principal de este trabajo es crear una aplicación a la que el usuario del dron pueda acceder a través de un dispositivo con cámara y conexión a internet que ejecute el software especificado. Dentro de este planteamiento existen dos opciones principales que serían desarrollar una aplicación móvil nativa y desarrollar una aplicación web.

2.3.1.1 Aplicación Móvil Nativa

Una aplicación nativa es la aplicación que podemos instalar en nuestros dispositivos a través de una tienda de aplicaciones como App Store o Google Play. La podemos reconocer en nuestro smartphone gracias a un icono que aparece en el escritorio. Estas apps se desarrollan específicamente para un sistema operativo, siendo los más utilizados Android e IOS. Al ser nativa, permite utilizar al máximo las funcionalidades del dispositivo, tanto el software como el hardware. No obstante, implica que hay que hacer un desarrollo específico para cada sistema operativo por lo que se puede multiplicar el tiempo de desarrollo, además implica tener conocimientos sobre los lenguajes de programación concretos de cada sistema operativo. El rendimiento se verá optimizado gracias a esta especialización, pero no siempre es conveniente esta optimización frente al tiempo de desarrollo y la accesibilidad.

2.3.1.2 Aplicación Web

Por otro lado, existen las aplicaciones web que serían el mismo concepto que conocemos de aplicación móvil o de escritorio, pero alojada en un servidor web y accesibles a través del navegador. No están optimizadas para cada dispositivo ya que existe una única aplicación a la que se puede acceder a través de una dirección web. Sin embargo, tienen la ventaja de que implican un único desarrollo siendo funcional en cualquier dispositivo con acceso a Internet. No es necesario tener conocimientos sobre un lenguaje de programación específico para un sistema operativo y a parte el procesamiento no recae en el cliente sino en el servidor, por lo que no consume tantos recursos cuando es utilizado en un teléfono móvil, por ejemplo. Otra ventaja importante de las aplicaciones web es que no ocupan espacio en el dispositivo y no hay que descargar nada, lo que hace que los usuarios estén más dispuestos a utilizarlas.

2.3.1.3 Híbridos entre Aplicación Web y Móvil Nativa

También existen los híbridos entre aplicaciones web y móvil, y pueden ser de dos tipos principalmente, el primero serían las aplicaciones web que se pueden descargar como app de tal forma que la aplicación móvil sería simplemente un visor de la web en un formato más adaptado a móvil. Y la segunda sería una aplicación móvil cuya interfaz está escrita en algún lenguaje de programación de aplicaciones web que funciona tanto en Android como IOS como es React.js, y cuyo back-end es una REST-API que se aloja en otro servidor al que se accede mediante peticiones POST o GET, de tal forma que la parte importante del procesamiento se puede realizar en un servidor más potente y no en el dispositivo. Este último diseño es el que se utiliza en aplicaciones como Instagram, ya que permite tener una app nativa potente sin consumir recursos del dispositivo.

2.3.1.4 Conclusiones

Teniendo en cuenta estas tres posibles soluciones se ha elegido la aplicación web ya que es más escalable que una aplicación móvil, no se requieren distintos desarrollos para cada sistema operativo. Haciendo el diseño una sola vez funciona en portátil, móvil y Tablet en cualquier sistema operativo. Además, la potencia no depende del dispositivo en el que se use y no hay limitaciones por el tamaño de la descarga. Teniendo en cuenta el coste de implementación, una aplicación móvil implica dos desarrollos por separado uno para Android y otro para IOS, los principales sistemas operativos de los smartphones, mientras que la aplicación web solo necesita uno. De igual forma una aplicación no es solo el desarrollo sino también el mantenimiento, en una app móvil el coste de mantenimiento se multiplica por dos. Se podría desarrollar la aplicación solo para un sistema operativo, pero entonces no podríamos alcanzar a todo el público. También podría haberse desarrollado un sistema híbrido, pero como explicaremos más adelante, hacer el front-end en React.js para desarrollar una aplicación móvil compatible con los dos sistemas operativos habría supuesto un tiempo mayor de desarrollo que la solución en Flask que finalmente se ha desarrollado. Además, la solución en Flask no limita un posterior desarrollo del front-end de la aplicación en React.js.

2.3.2 REST-API

Hemos mencionado en la explicación sobre las aplicaciones web las REST-APIs, un estilo importante de arquitectura de software que permite separar el cliente y el servidor de tal forma que la comunicación se realiza mediante peticiones en las que el cliente envía unos datos y obtiene una respuesta, que puede ser una operación de cierta complejidad procesada en el servidor. Es un diseño muy importante ya que permite separar front-end y back-end de tal forma que el back-end es independiente de la interfaz por lo que se puede cambiar la el front-end sin rehacer el back-end.

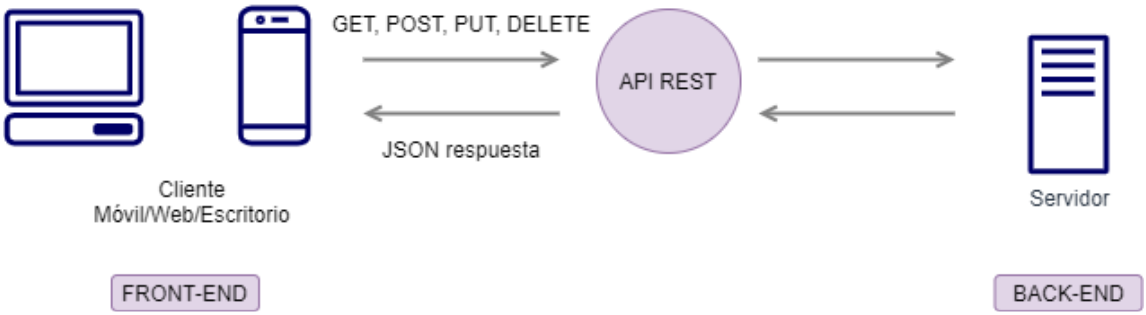

Figura 5. Diagrama comunicación entre Cliente y Servidor mediante una REST-API

2.3.2.1 Flask Vs Django

Para el desarrollo de la aplicación web se han estudiado dos las dos opciones más utilizadas en el mercado para el desarrollo de REST-APIs las cuales son Flask, un framework escrito en Python con el que crear fácilmente APIs y Django, otro framework de Python para el desarrollo de aplicaciones. Apps tan famosas como Instagram utilizan Django ya que al igual que Flask, es sencillo, ágil y potente, perfecto para desarrollar el back-end de cualquier aplicación [33].

Django permite el desarrollo de aplicaciones escalables y de buena calidad, además tiene cantidad de plugin y permite administrar bases de datos a través de su panel de administración. Es perfecto para aplicaciones grandes y robustas con un gran equipo implicado, sin embargo, tiene mayor complejidad frente a Flask a la hora de su implementación.

Flask, por el otro lado, es igual de bueno que Django, es sencillo y potente. Las principales razones para elegirlo frente a Django es que desarrollar una API desde el principio es mucho más ágil, además es más rápido que Django ya que es un framework más sencillo y para el objetivo de este proyecto la velocidad de las llamadas al API eran claves. Además, un punto muy importante y decisivo es que la utilización de sockets con Django es mucho más compleja que con Flask y este era un punto muy relevante ya que los sockets son esenciales para poder hacer el *streaming* del dron y el seguimiento de la mirada en tiempo real. Más adelante se explicarán los Sockets en profundidad, pero en resumen se trata de un protocolo de comunicación que permite mantener la conexión entre cliente y servidor abierta para que las peticiones entre uno y otro fluyan mucho más rápido que con HTTP, donde se tiene que abrir una nueva conexión cada vez que se quiere hacer una petición.

En conclusión, Django es un gran framework para el desarrollo del back-end de aplicaciones, pero Flask es más sencillo de implementar y es la mejor opción para el desarrollo de un prototipo ya que entre otras cosas incluye un servidor para el desarrollo, por lo que la parte de la instalación del servidor queda solventada desde el principio. Además, existen muchas librerías para la implementación sencilla de Sockets con Flask. Igualmente, no podemos pasar por alto que este framework está escrito en Python por lo que será mucho más sencillo integrar el software de detección escrito también en Python.

2.3.3 Sockets

Las aplicaciones web son posibles gracias a la comunicación mediante peticiones asíncronas dentro de la página web, estas peticiones permiten acceder a otras funciones o vistas sin la necesidad de estar recargando la página constantemente. Antes de estas peticiones las páginas web eran simplemente código HTML en el que para acceder a otras funciones o hacer distintas operaciones tenías que acceder a links con el correspondiente tiempo de recarga de la página. Gracias a la comunicación asíncrona o la implementación de AJAX, es posible comunicarse con el servidor sin la necesidad de recargar la página, así como comunicarse con API externas para obtener información o enviarla. Es decir, las peticiones HTTP se realizan en segundo plano, esto ha hecho que

las aplicaciones web ganen mucha popularidad pues ya no es necesario de una app nativa para poder tener la fluidez de una aplicación.

HTTP es el método de comunicación con el servidor más estandarizado, permite hacer peticiones al servidor para solicitar, por ejemplo, un archivo HTML, es decir, la vista que vemos de una página web. Se basa en peticiones GET y POST, cuyos nombres en inglés resultan bastante auto explicativos, GET sirve para solicitar un recurso como puede ser una página y POST sirve para enviar datos al servidor, como por ejemplo cuando subimos una imagen a una página web esta hace un POST para enviarla al servidor que la guardará en la base de datos.

AJAX ha permitido la comunicación asíncrona en las páginas web gracias a peticiones HTTP, sin embargo, existe un método de comunicación mejor si el objetivo es intercambiar información con el servidor de forma más rápida, los WebSockets. WebSockets es un protocolo de red basado en TCP que se diferencia en la comunicación HTTP en que mantiene la conexión abierta y en esta se pueden realizar tantas peticiones y respuestas como se requieran. La ventaja de este protocolo es que al no tener que estar abriendo y cerrando la conexión con cada intercambio, es mucho más rápida. En la publicación *Real-Time Monitoring using AJAX and WebSockets* [34] se comparan estas dos tecnologías y se demuestra que los WebSockets consumen un 50% menos de ancho de banda que AJAX, pudiendo enviar hasta un 200% más de datos en el mismo consumo de ancho de banda que AJAX. Es por esto por lo que es el protocolo elegido a la hora de hacer streaming de video o chats en tiempo real como WhatsApp Web.

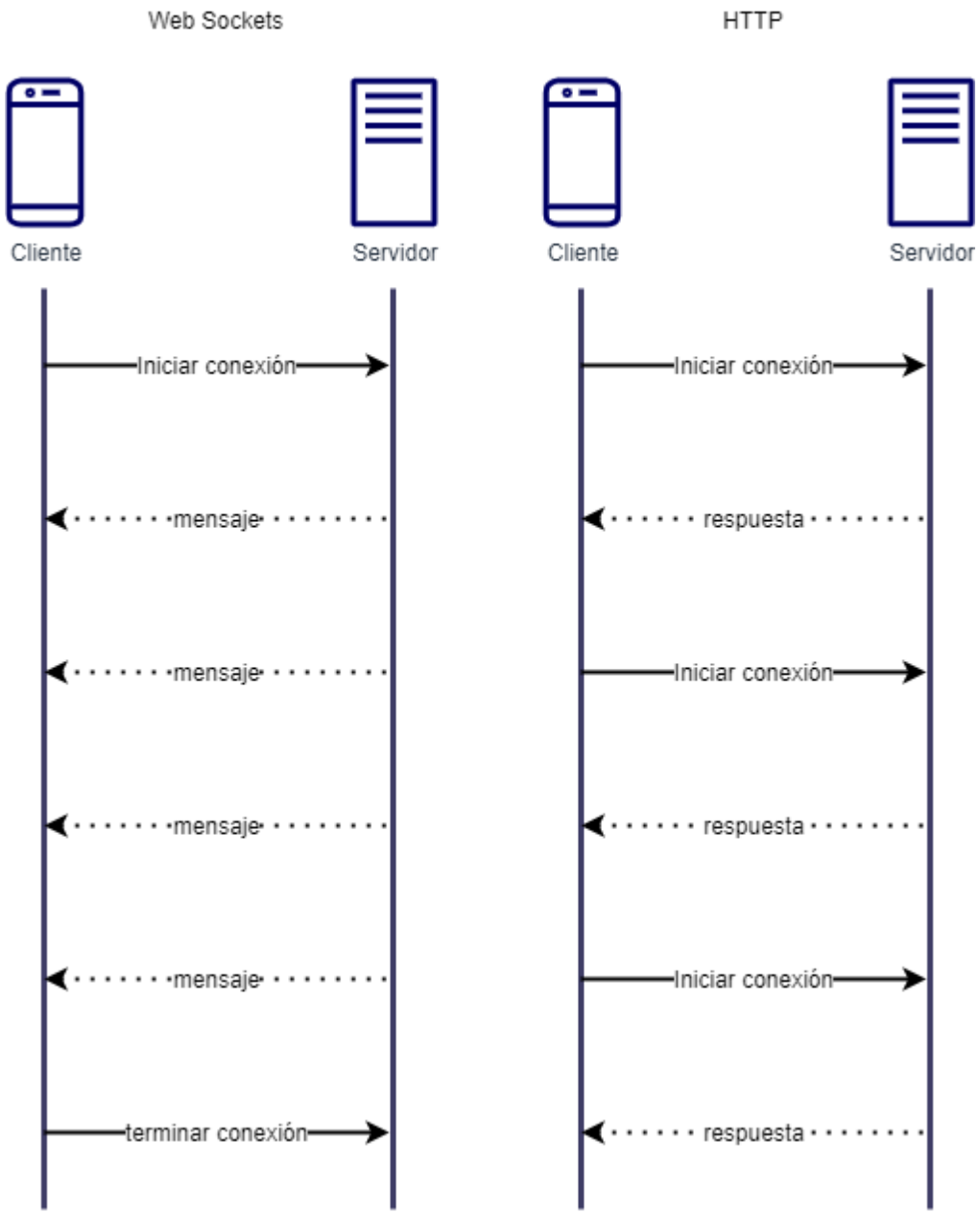

Figura 6. Comunicación mediante WebSockets Vs. HTTP

Como ya se ha mencionado varias veces a lo largo de la memoria, la latencia será clave para el funcionamiento del prototipo que se pretende implementar. Se tendrán hasta tres streamings de datos en tiempo real, el streaming de las imágenes captadas por el dron, el streaming de imágenes captadas por la webcam que se enviarán al servidor para ser procesadas por el software de detección y el streaming de datos al servidor de la Raspberry con las coordenadas para los servos. Todas estas comunicaciones tienen que realizarse lo más cercanas posibles al tiempo real para que el usuario pueda mover la cámara a voluntad sin que resulte incómodo o no merezca la pena frente a la alternativa de usar otro mando. Es por ello por lo que se ha decidido que el servidor utilizará

conexiones por WebSockets para comunicarse con los dos clientes, la web y la Raspberry.

2.3.4 Hardware

La estructura de hardware se hará a partir de una Raspberry Pi ya que es lo que especificó el cliente para el prototipo, este ordenador en miniatura junto con unos servos sg90 y una cámara recogerá las imágenes desde el aire. Este pequeño ordenador es ampliamente utilizado para prototipar ya que es barato y fácil de programar. El sistema operativo utilizado será Raspbian, una distribución de Linux basada en Debian. Será el servidor al que llegarán las coordenadas de los ojos para mover la cámara. Los servos estarán directamente conectados a la Raspberry por lo que será fácil mover los servos gracias a la librería de Python RPi.GPIO para controlar los GPIO.

Los servos serán unos sg90 atornillados a una estructura de tal forma que uno se mueva en horizontal y otro en vertical. Esto permitirá los movimientos derecha, izquierda, arriba y abajo. La cámara estará también acoplada a esta estructura para poder moverse y conectada a la Raspberry Pi para poder enviar las imágenes captadas. Es una disposición sencilla para el prototipado pudiendo mejorarse la precisión de los servos con unos mejores o pudiendo aumentar la calidad de las imágenes con una cámara mejor.

Figura 7. Ejemplo de estructura en Pan and Tilt para el movimiento horizontal y vertical de la cámara

2.4 Conclusiones sobre el estado del arte

En este estudio previo se ha realizado un análisis para determinar la viabilidad del sistema y encontrar el mejor software y hardware para su implementación. En el apartado 2.1 estudiamos la situación actual de los drones o UAV y se vio cómo el futuro de éstos pasa por mejoras en la forma de relacionarse el humano con la máquina, como vimos en el estudio realizado por Y. Zhou, J. Hou, y Y. Gong [17], en el que se hacía un desarrollo similar al propuesto en este trabajo pero mediante un control por voz.

Las técnicas de Visión Artificial actuales nos permiten determinar la dirección de la mirada en imágenes mediante software de una forma accesible, gracias a las potentes librerías de código abierto para Python OpenCV y Dlib. El desarrollo de software de Inteligencia Artificial no requiere de una gran inversión inicial lo que permite desarrollar un prototipo de bajo coste para probar el sistema.

Por otro lado, se han estudiado las distintas implementaciones posibles para la aplicación que usará el operario en su smartphone, una aplicación móvil o una aplicación web. Con este análisis se ha decidido que la mejor opción para un prototipo era una aplicación web ya que es compatible con todos los sistemas operativos de los smartphones más utilizados y requiere un menor tiempo de desarrollo, además, es escalable pudiendo, de ser necesario, acercarla más a la aplicación móvil nativa mediante librerías especializadas en crear estos híbridos entre aplicación web y móvil.

Para la comunicación de todo el sistema en el apartado 2.3 se recoge el estudio sobre distintas herramientas de comunicación y se llega a la conclusión de que la mejor es una REST API en Flask con comunicación mediante Sockets ya que es sencilla y escalable, Flask tiene un buen rendimiento y además tiene la ventaja de que tiene un servidor incorporado para las pruebas de desarrollo por lo que no es necesario estar instalando un servidor. Los Sockets permiten una comunicación asíncrona más rápida que mediante peticiones AJAX, al mantener la comunicación abierta se reducirá la latencia de estar abriendo y cerrando la comunicación cada vez que se envía o se recibe un mensaje del servidor.

Con todo esto la implementación resulta viable por lo que se diseñará un prototipo que utilice todas estas herramientas para llegar a un Producto Mínimo Viable que nos permita medir resultados y sacar conclusiones de cara a una versión comercial o más avanzada del sistema.

3. DISEÑO Y DESARROLLO

3.1 Requisitos del sistema

A continuación, se enuncia los requisitos del sistema que se han tenido en cuenta para el desarrollo de la solución.

- Acceso a través del navegador de un smartphone.
- Una aplicación web que pueda acceder a la cámara frontal del teléfono.
- El envío de las imágenes recogidas a un servidor.
- El tratamiento de las imágenes en el servidor utilizando Visión Artificial de tal forma que se determine la dirección de la mirada del usuario a partir de las imágenes de la cámara frontal del móvil.
- Convertir los puntos de referencia faciales que devuelve el modelo de visión por ordenador a unas coordenadas que entienda el controlador del motor de la cámara del dron.
- Enviar esta información del servidor a la Raspberry Pi integrada en el dron.
- Que la Raspberry controle la dirección del motor de la cámara del dron.
- Las imágenes de la cámara del dron deben enviarse a la aplicación web en vivo.
- Estas imágenes enviadas por el dron deben mostrarse en el teléfono del usuario a través de la aplicación web.

Con este diseño se persigue una primera aproximación al desarrollo de un sistema que sea capaz, para empezar, de recoger las imágenes de la mirada del operario que controla el dron mediante la cámara frontal de un smartphone. Una vez se hayan recogido estas imágenes se procesarán mediante un modelo de visión por ordenador que traducirá la dirección de la mirada a coordenadas. Esta tarea se realizará en un servidor online en Python que recibirá las imágenes y enviará la información sobre la dirección de la mirada a una Raspberry Pi mediante una conexión Wifi. Este último dispositivo se encuentra integrado en el dron y controla el motor de la cámara del dron. Con la información que vaya recibiendo del servidor actualizará la posición de la cámara para que se ajuste a lo que el operario quiera ver en cada momento.

El objetivo del diseño será desarrollar una aplicación web que permita al usuario de un dron controlar la cámara de este mediante la mirada, se tratará de reducir el tiempo de

respuesta lo suficiente para que sea cómodo para el usuario y cuando mire hacia un cierto punto de la pantalla la cámara le siga sin demasiada latencia. Esta será la mayor dificultad junto con el desarrollo del software que traduce imágenes a coordenadas, ya que habrá muchas comunicaciones distintas y cada una de ellas hará que aumente el tiempo entre el movimiento de los ojos y el de la cámara.

3.2 Resumen del diseño

El diseño se puede dividir en tres partes, la aplicación web a la que se accede para utilizar el software y ver las imágenes de la cámara del dron en streaming, el software de visión por ordenador y el servidor del dron que recibirá las coordenadas para mover los servos. El servidor donde se alojará la aplicación web será la interfaz para el usuario donde podrá acceder a través de su dispositivo, como por ejemplo su smartphone, para ver en tiempo real las imágenes captadas por la cámara del dron, y al mismo tiempo a través de la webcam se recogerán las imágenes de la dirección de la mirada para poder procesarlas y sacar las coordenadas. En este servidor también se ejecutará el software de visión por ordenador, un detector de puntos de referencia faciales desarrollado con las librerías de código abierto Dlib y OpenCV. Este servidor se ejecutará en un ordenador portátil para este trabajo, pero podría alojarse en algún servicio en la nube como Amazon Web Services y usar la potencia del servicio para aumentar la velocidad del software de detección. Por otro lado, tendremos la Raspberry Pi con el servidor que hará de cliente para recibir las coordenadas de los ojos y mover los servos en función de estas, tendremos en esta parte también un pequeño software para mover los servos. En la figura de la siguiente página se puede ver un esquema del diseño completo.

3.3 Arquitectura de software

El software que se quiere desarrollar en este trabajo tiene como objetivo determinar la dirección a la que se quiere mover la cámara integrada en un dron según la mirada de un usuario operario. La arquitectura de software se ha dividido en la interfaz y adquisición de datos, el servidor donde se corre el software de detección de la mirada y el servidor en la Raspberry Pi que mueve los servos.

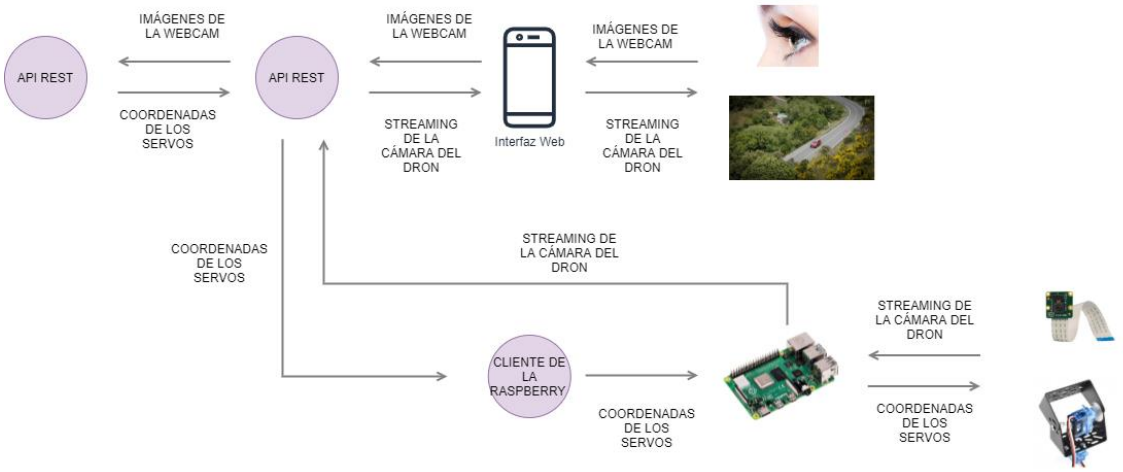

Figura 8. Diseño completo del sistema

3.3.1 Interfaz Web y adquisición de datos de la Webcam

La interfaz será una aplicación web a la que se podrá acceder desde cualquier navegador. En este caso tiene que ser desde un dispositivo con cámara que pueda recoger las imágenes de la cara del usuario. La web será una única vista donde se podrán ver las imágenes que se retransmiten en streaming de la cámara integrada en el dron. Además, la web deberá acceder a la cámara del dispositivo que apunte a la cara del usuario, estas imágenes serán las que se envíen al servidor para poder detectar el movimiento de la mirada.

La web estará escrita en HTML5, CSS y JavaScript. Constará únicamente de un contenedor para el streaming de la cámara del dron. Se comunicará con el servidor mediante Sockets y tendrá 3 funciones para esta comunicación, la función para conectarse al servidor e iniciar la conexión mediante sockets, la función para desconectarse del servidor y la función principal que será una emisión de fotogramas al servidor cada segundo. Esta última función será la que envíe al servidor los datos necesarios para analizar la dirección de la mirada.

Figura 9. Interfaz de la aplicación Web. Recoge las imágenes de la webcam y devuelve las imágenes de la cámara del dron

3.3.2 Servidor principal API-REST

El servidor principal será el que procese las imágenes de la cara y realice las comunicaciones entre todas las demás partes. Para este servidor se usará Flask-SocketIO, una librería de Python que permite usar Sockets en Flask. Estará corriendo en un portátil potente para poder usar el software de detección de mirada, pero podría alojarse en cualquier servicio en la nube para mayor velocidad en las llamadas.

La aplicación de Flask se llamará Web Server en el directorio del proyecto. El *endpoint* principal o índex será la interfaz de la aplicación donde accede el usuario, de esta forma no hará falta un servidor a parte para alojar la web, aunque podría hacerse en caso de tener alguna ventaja. Para este proyecto no será necesario.

A parte del endpoint principal se encuentran las comunicaciones principales con la web, es decir las funciones de conectar y desconectar la conexión con esta. El *namespace* para esta comunicación será "/web" para diferenciar la conexión con la interfaz de la de la Raspberry. Para conectarse con el cliente de la Raspberry Pi se usará otro namespace "/dron" con las funciones conectar y desconectar correspondientes.

Por otro lado, se encuentra la función que recibirá los fotogramas de la webcam. En esta función se procesará la imagen y se pasará al software de detección facial. El resultado se transformará en coordenadas para los servos y se guardará en una variable global para enviarlo al servidor de la Raspberry.

3.3.3 Streaming de video de la cámara del dron

Para terminar, la última función del servidor será el envío del streaming de la cámara del dron a la página web donde se podrán ver las imágenes en directo. Para poder proyectarlas en la web, el streaming primero se procesa en el servidor para adaptarlo a un formato compatible con la web, esta es la función de la clase DronVideoStreaming.

La clase DronVideoStreaming instancia un objeto VideoCapture de OpenCV a partir de la IP del streaming de video de la cámara de la Raspberry Pi. Además, comienza el hilo de comunicación del streaming. Esta clase tendrá tres métodos:

update(): Lee el siguiente fotograma del stream en un hilo distinto.
get_frame(): Coge un fotograma del streaming y cambia su tamaño al deseado.
maintain_aspect_ratio_resize(): Cambia el tamaño del fotograma a la resolución elegida.

De esta manera se adapta el tamaño de cada fotograma al deseado para después enviarlo a la web que muestra las imágenes.

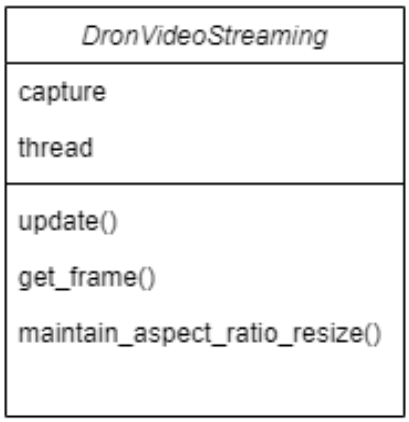

Figura 10. Clase DronVideoStreaming

3.3.4 Software de detección de mirada

Para el software de detección de mirada se creará la clase GazeTracking para detectar el movimiento de los ojos en los distintos fotogramas enviados por la aplicación web. Al inicializar esta clase se crean distintos atributos: el fotograma, el ojo derecho, el ojo izquierdo y el objeto de detección facial. Como se explicó anteriormente la detección facial se hará utilizando la librería Dlib, en concreto se usará la función

get_frontal_face_detector() que determina la cara en una imagen utilizando un Histograma de Gradientes Orientados. Para nuestra clase se creará una instancia de este objeto llamada *face_detector*. Una vez detectada la cara los puntos de referencia facial se extraerán con la función shape_predictor() de Dlib, a este atributo se le llamará *predictor*. La clase GazeTracking también crea una instancia del objeto *Calibration*() para la calibración del algoritmo detector de la pupila.

Los métodos de esta clase se explicarán en profundidad a continuación.

pupils_located(): este método comprueba que las pupilas han podido ser localizadas en el fotograma, retorna un booleano.

analyze(): la función analice transforma el fotograma de BGR a gris para poder detectar correctamente la cara. Una vez detectada la cámara inicializa los objetos de los ojos con los landmarks detectados.

refresh(): esta función renueva el fotograma y lo analiza con el método analyze().

pupil_left_coords(): retorna las coordenadas de la pupila izquierda en el fotograma. pupil_right_coords() sería lo mismo para el ojo derecho.

horizontal_ratio(): retorna un ratio que indica la proyección horizontal de la mirada, es decir hacia donde está mirando la persona en el fotograma en la dirección horizontal. vertical_ratio() es la misma función para la dirección vertical.

tilt_duty_cycle(): este método retorna el *duty cycle* que se debe aplicar al servo que se mueve de arriba abajo para que coincida con la posición de la mirada. pan_duty_cycle() sería lo mismo para el servo que va de derecha a izquierda.

Aunque la clase GazeTracking es la principal, se han creado otras para apoyar el proceso de detectar la mirada como son Eye, Pupil y Calibration.

La clase Calibration como su nombre indica se encarga de calibrar el algoritmo de detección encontrando el mejor *threshold* para el fotograma que le llega. Es decir, se

adapta a las condiciones de luz y la posición de la cara de la persona en la imagen. Entre sus atributos se encuentran el threshold_left y el threshold_rigth ya que habrá distintos thresholds para cada ojo. Pasaremos ahora a explicar los principales métodos de esta clase.

is_complete(): una vez se haya completado la calibración este método retornará True, en caso contrario retorna False.

threshold(): es el método más importante ya que retorna el valor de threshold para el ojo que se pasa como argumento.

iris_size(): retorna el tamaño del iris necesario para determinar el valor del threshold.

find_best_threshold(): retorna el mejor threshold para la situación concreta del fotograma.

evaluate(): evalúa el threshold seleccionado en el fotograma.

La clase Pupil detecta el iris en la cara y determina la posición de la pupila en el ojo. Consta de dos métodos, el primero image_processing() procesa el fotograma para que sea más sencillo determinar la posición de la pupila. El segundo detect_iris() determina las coordenadas de la pupila en el ojo con los valores x e y.

La clase Eye representará un ojo conteniendo todos los elementos relevantes para este objeto, como pueden ser el centro del ojo o la pupila. La pupila será una instancia de la clase Pupil que vimos anteriormente. Para inicializar este objeto se le pasa el fotograma y los puntos de referencia detectados para que aísle los landmarks de los ojos y determine la pupila, el método isolate() se encargará de esta función de aislar el ojo del resto de la imagen. Con la función analyze() se analiza el ojo, es decir se aísla el ojo en la imagen, se hace la calibración para determinar el threshold y se inicializa el objeto Pupil que determina las coordenadas de la pupila.

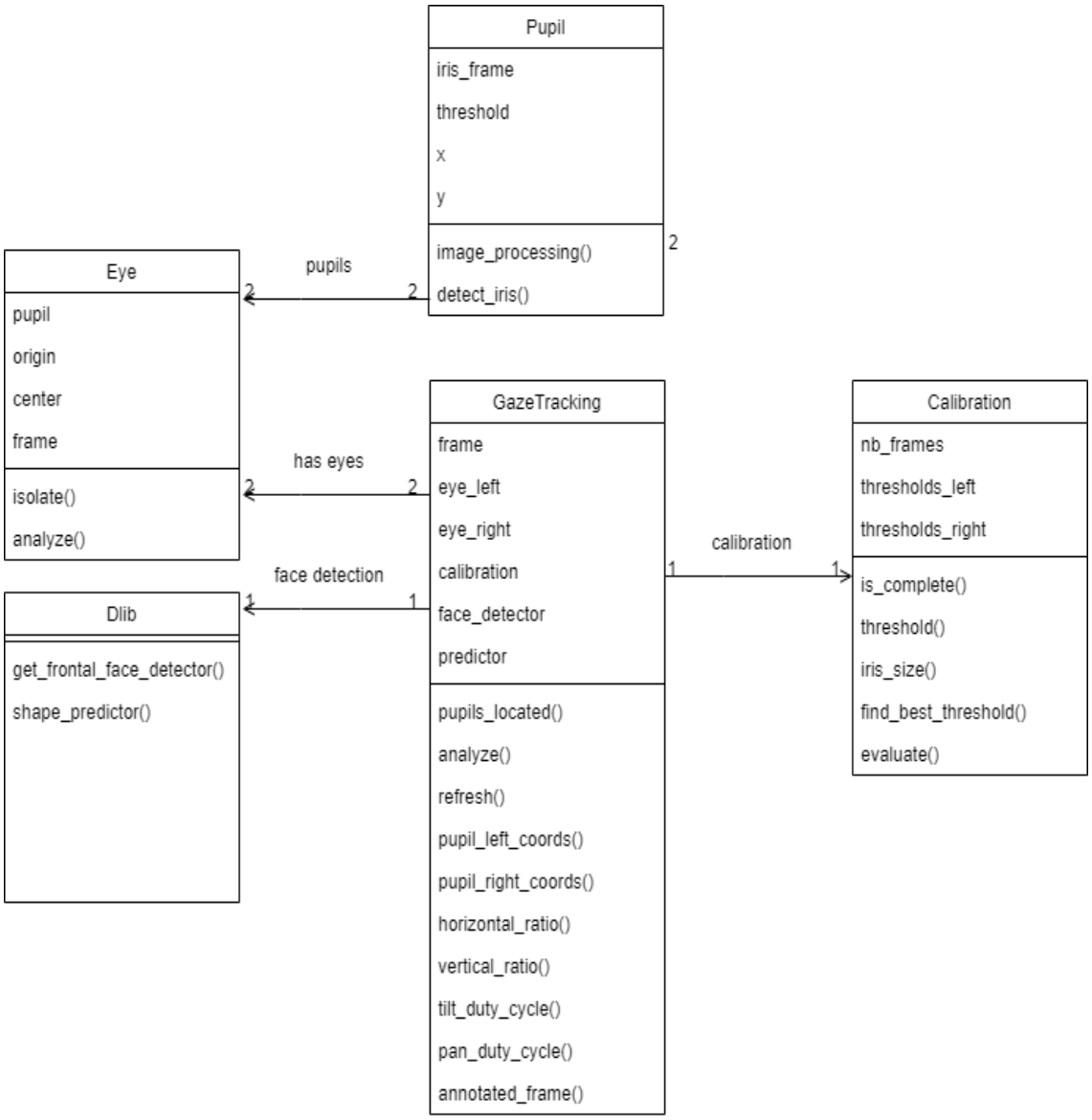

Figura 11. Diagrama de clases del software de detección de la mirada

3.3.5 Servidor de la Raspberry Pi

El servidor de la Raspberry Pi será el cliente que recibirá las coordenadas de la mirada para mover los servos en la dirección correcta. Se trata de una aplicación de Python desarrollada con la librería SocketIO. Aparte de las conexiones conectar y desconectar del servidor principal, tendremos la función gaze_position_to_dron() donde llegarán las coordenadas para mover los servos. El cliente establecerá una conexión con el servidor principal de tal forma que cada segundo le enviará un mensaje para preguntar las coordenadas a las que tiene que mover los servos.

Para controlar el movimiento de los servos se creará la clase DronMoveCamera. Al crear una instancia de esta clase se crea un objeto que inicializa los servos y los lleva a la posición inicial. A partir de esta posición hay cuatro funciones que representan los cuatro movimientos que puede hacer la cámara, arriba, abajo, derecha e izquierda. Este movimiento se consigue cambiando el duty cycle de los servos con la función set_duty_cycles(). Si los servos alcanzan su límite de movimiento se pinta un mensaje en la consola informando de que los servos han alcanzado este límite y no se van a desplazar más en esa dirección. Para terminar la función stop_camera() para los servos.

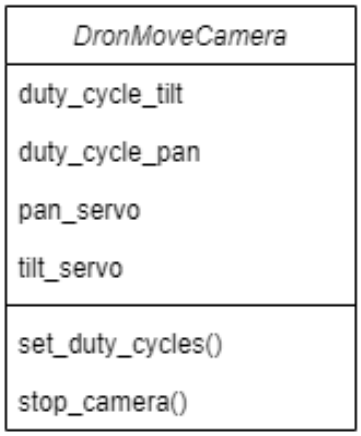

Figura 12 Clase DronMoveCamera

Los servos se mueven gracias a una señal modulada, en función del ancho de pulso será la rotación del servo. De esta forma modificando el duty cycle se puede cambiar la dirección del servomotor.

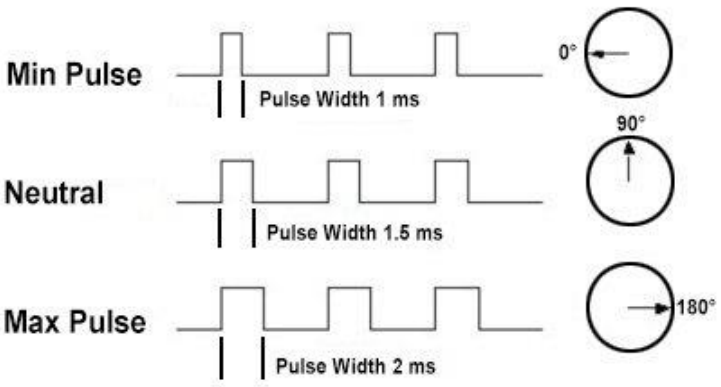

Figura 13. Relación entre los grados de rotación de un servo y el ancho de pulso

Para resumir la arquitectura se adjunta una imagen del diseño con las tres capas existentes.

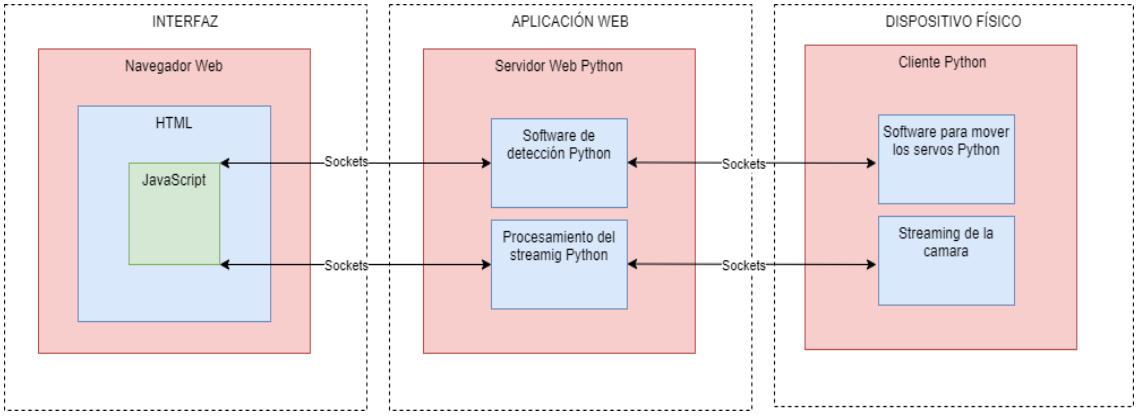

Figura 14. Representación de la arquitectura del diseño propuesto

3.4 Implementación

Durante la implementación del diseño propuesto surgieron algunos problemas propios de la aplicación a la realidad. Para empezar a pesar de que los sockets permiten un gran ancho de banda y una gran cantidad de intercambio de mensajes, siempre surgía un error "*Too many packets in payload*" ya que la librería Flask-SocketIO tiene una capa de seguridad que limita la cantidad de paquetes en el *payload* o carga útil, ya que por lo general una gran cantidad de pequeños paquetes implica que alguien está intentando romper el servidor.

Esta limitación generaba una caída de la comunicación constante, ya que se enviaban al servidor los fotogramas a una velocidad de 20FPS. Pero además de estos fotogramas se tenía que enviar la respuesta de vuelta, más el streaming de la cámara del dron más enviar las coordenadas al servidor de la Raspberry. Es decir, se hacían una gran cantidad de envíos por segundo lo que generaba este error por la carga útil. Para intentar solucionar este error se aumentó el *MAX_DECODE_PACKETS* a 500, pero no fue suficiente, por lo que se tuvo que reducir los fotogramas por segundo a 1 FPS. A pesar de que ya no se desconectaba la comunicación constantemente, enviar 1 FPS por segundo hacía que la diferencia entre la dirección de la mirada y el movimiento de los servos estuviera desfasada 1 segundo, lo que se sumaba al resto de latencias.

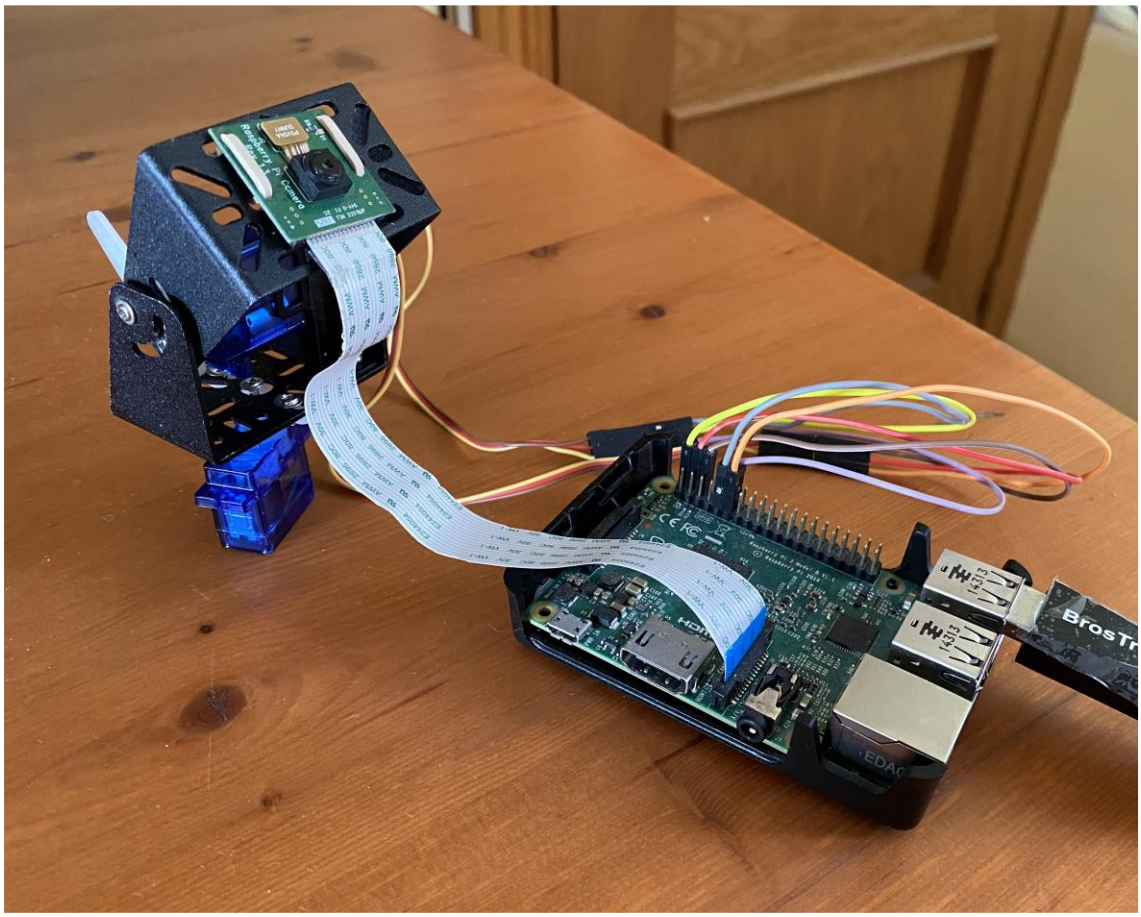

Figura 15. Servos conectados en estructura pan and tilt

Siguiendo con los problemas con la latencia, al hacer la comunicación entre las distintas partes mediante Wifi, la velocidad estaba limitada a la velocidad de la fibra y la velocidad que soportaban los dispositivos, sumada a las interferencias y la distancia al rúter, las paredes de por medio y la saturación de los canales. Una conexión por Ethernet reduciría esta latencia, pero no es posible utilizarla para el cliente de la Raspberry ya que se supone que estará integrado en un dron que estará volando al aire libre, por lo que Ethernet no es una opción para esta parte del diseño. Sin embargo, el servidor principal o servidor web, sí que podría estar conectado a Ethernet para tener una mayor velocidad a la hora de enviar las respuestas a los clientes y recibirlas, ya que se evitarían las interferencias y la saturación de las redes Wifi. Este servidor puede estar en cualquier sitio por lo que se podría mejorar la conectividad y la latencia.

El dispositivo utilizado para abrir la aplicación web y utilizar la Webcam también se supone que está al aire libre por lo que igualmente tiene la limitación de la conexión. Todas estas pequeñas latencias suman y la aplicación deja de rendir como se esperaba

en la teoría. El usuario mira hacia un lado y hay una diferencia considerable en la respuesta por lo que la cámara se mueve un poco después de que el usuario haya cambiado de dirección. Esto empeora la experiencia de usuario.

La gran cantidad de peticiones también presenta un problema a nivel de carga de la CPU, como se explicó anteriormente se utiliza un gran ancho de banda, el servidor se instaló en un portátil que no podía aguantar esta gran carga durante mucho tiempo y pasado un tiempo empezaba a sobrecalentarse y a fallar. Además, se sumaba que tenía que correr el software de detección que también requiere mucha potencia de procesamiento.

Sumado a esto surgió un problema que no se había previsto con el diseño y es que la aplicación se debe calibrar cada vez que se quiera usar ya que la transformación de la mirada a las coordenadas para los servos o el duty cycle que se debe aplicar al servo cambia en función de la resolución de la pantalla y la distancia de la persona a la pantalla. La transformación de la dirección de la mirada al movimiento de los servos cambia en cada uso, por lo que se tiene que calibrar continuamente. Esto se habría solucionado incorporando a la aplicación una calibración automática cada vez que se accede nuevamente a la aplicación, pidiéndole al usuario que mire a una serie de puntos en su pantalla se podría haber ajustado la transformación para mover los servos en función del tamaño de la pantalla.

4. RESULTADOS

En el capítulo 1 se describen los objetivos del proyecto, en resumen, se trata de desarrollar una aplicación web que permita al usuario de un dron controlar la dirección de la cámara de éste mediante la dirección de su mirada. Con la cámara frontal del smartphone se recogerán las imágenes de la mirada que serán enviadas al servidor, donde el software de Visión por Ordenador traducirá los fotogramas a coordenadas que serán enviadas a la Raspberry Pi integrada en el dron. Las coordenadas servirán para mover los servos que mueven la cámara. Al mismo tiempo en la aplicación web se reproducirá en streaming las imágenes del dron ya que tiene que poder visualizarlas el usuario para saber hacia dónde mirar.

El procedimiento para medir los resultados tratará de buscar principalmente la relación entre la entrada y la salida del sistema, siendo estas la dirección de la mirada y el movimiento de los servos respectivamente. Además, se buscará reflejar el tiempo en el que se realiza la acción ya que el sistema no será eficiente si no consigue realizar la transformación en un tiempo que resulte cómodo y útil para el usuario. Éste debe dirigir la mirada a un punto de la pantalla y que el servo se mueva hasta este punto pasando la información a través de todo el sistema, de la Webcam a la web, de la web al servidor principal, donde el software de Inteligencia Artificial determinará el resultado y este se enviará al servidor de la Raspberry que moverá los servos.

4.1 Definición de las pruebas

4.1.1 Prueba de latencia

Una parte importante del sistema es la latencia de éste, ya que solo será efectivo si tiene una latencia baja que permita a los servos seguir la mirada del usuario. Por ello la primera prueba tratará de medir la latencia del sistema completo, desde que la web envía las imágenes de los ojos al API hasta que recibe de vuelta las imágenes de la cámara desplazadas del punto inicial.

Para esto se medirá el ping o latencia que es el tiempo que tarda en transmitirse un paquete dentro de la red, se mide en milisegundos y cuanto más pequeño mejor. En el sistema hay distintas latencias, la latencia entre la web y el API, es decir el tiempo que

tarda la web en enviar las imágenes de la webcam al API donde se procesarán, la latencia del envío de las coordenadas de los servos desde el API a la Raspberry Pi y, por último, la latencia del streaming de la cámara del dron. Se tratarán de medir todas estas latencias para tener una idea del tiempo de respuesta del sistema.

El proceso para medir la latencia es simple, por ejemplo, para medir la latencia entre la Web y el API desde el lado del cliente, en este caso la Web se emite un mensaje mediante Sockets, los cuáles se explicaron en el apartado 2.3.3. Es mensaje será el "ping", y en el momento que lo recibe el servidor del API emite un "pong" como respuesta que es simplemente enviarle un mensaje de vuelta a la web. La web lo recibe y mide desde su lado el tiempo transcurrido desde que envió el "ping" hasta que recibió el "pong". Para que la medida sea más precisa, hace la media entre las 30 últimas medidas y el resultado sería la latencia entre cliente y servidor de esta parte del sistema. El resto de las partes se medirá de igual manera.

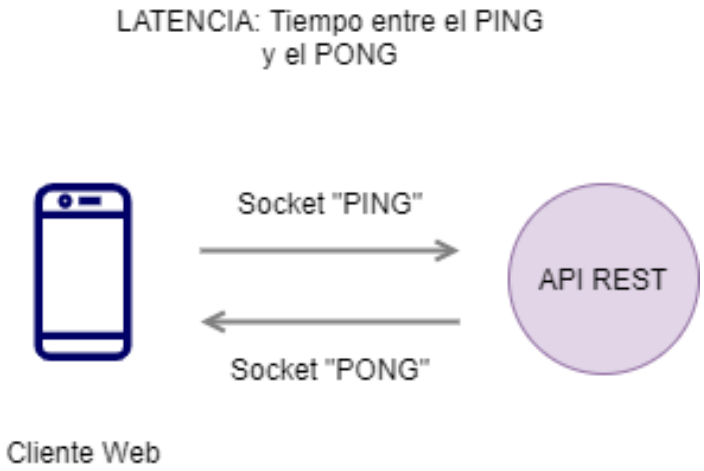

Figura 16. Diagrama de cómo medir la latencia entre Cliente y Servidor

4.1.2 Prueba de precisión de la transformación

Para medir la precisión en la transformación de una mirada al movimiento de los servos se ha desarrollado una prueba que tratará de medir y representar la relación entre los siguientes datos: la dirección real de las pupilas, las coordenadas de las pupilas detectadas por el software y las coordenadas recibidas por los servos. Para ello se han determinado dos rutas distintas a realizar por el usuario, éste deberá realizar el recorrido con la mirada por la pantalla siguiendo cada ruta tres veces seguidas. Con esto se pretende demostrar que los servos son capaces de reproducir la ruta que realizan las

pupilas y que la transformación es efectiva, sin tener en cuenta el tiempo que requiere esta operación. Serán dos casos distintos con distintos movimientos que se repetirán tres veces para conseguir una media de los resultados en cada ruta.

Para realizar estas pruebas se asumirá que las coordenadas de las pupilas son cien por cien fiables y precisas, para más adelante medir la precisión del modelo de detección con una prueba aparte, la cual se explica en el apartado 4.1.2.

La primera ruta pasará por los extremos de la pantalla donde tiene que mirar el usuario, las cuatro esquinas de la pantalla y al final el centro, tratando de mantener un ritmo constante de movimiento con la mirada. Antes de realizar la prueba ya se conocen las coordenadas de estos puntos tanto en el sistema del software, cuyo resultado se mide en dos ratios, ratio vertical y ratio horizontal, como después de la transformación al duty cycle de los servos, teniendo igualmente dos medidas, una para el duty cycle del servo que se mueve en vertical y otra para el servo que se mueve en horizontal. Estos puntos teóricos serán la referencia que se usará para medir si el software de detección consigue detectar estos puntos en la dirección de la mirada, así como si la transformación llega correctamente a los servos. Gracias a esta medida más la latencia del sistema podremos tener una idea de si el sistema está funcionando correctamente.

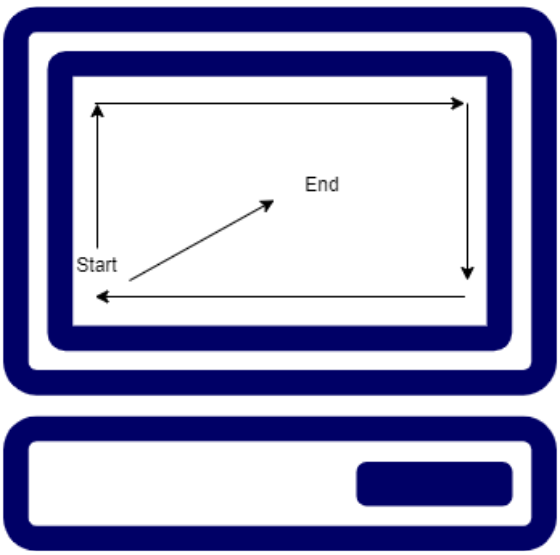

Figura 17. Diagrama explicativo de la prueba 2

La segunda ruta tratará de medir los paros y acelerones en el movimiento de la mirada, es decir esta vez el movimiento de los ojos no será constante, sino que realizará la

primera ruta con paros y acelerones en distintos puntos para ver cómo se adapta el sistema. La transformación de la ratio de la mirada al duty cycle de los servos es una transformación lineal con un alisado ya que seguir exactamente la mirada detectada por el software hacía que los servos se movieran bruscamente de un lado para otro. El software de detección da algunos valores atípicos o *outliers* por lo que al alisar la curva se consigue eliminar estos puntos erróneos que hacían que de repente el servo tuviese que gira al extremo completamente opuesto a la mirada para luego volver a donde estaba. Con esta prueba se pretende medir la influencia de este alisado en el movimiento de los servos.

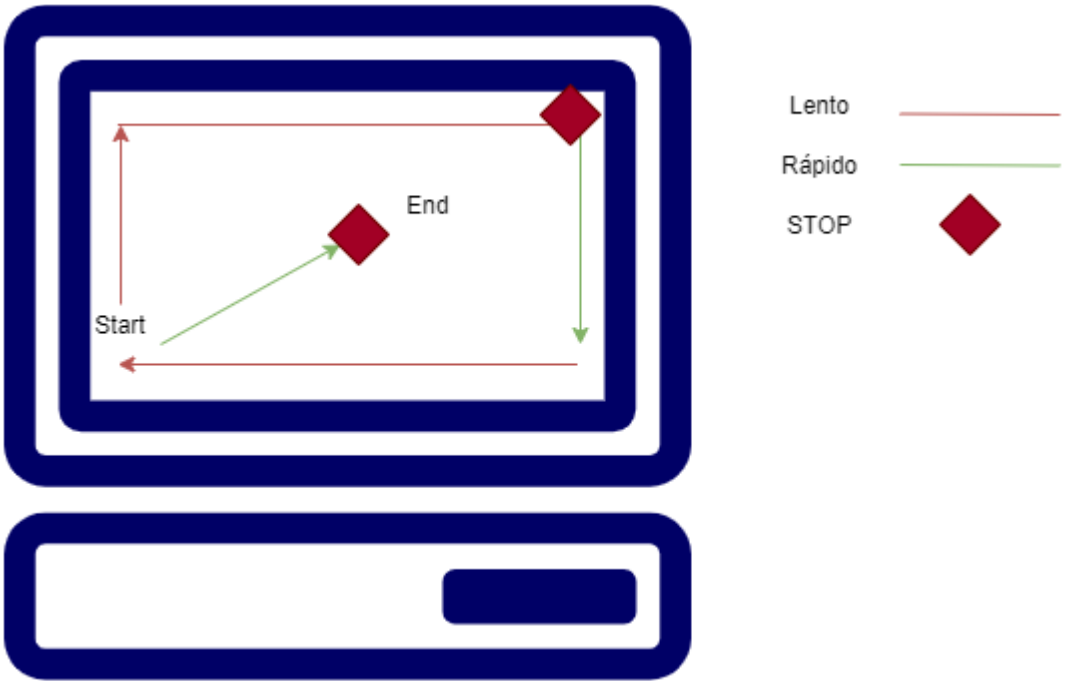

Figura 18. Diagrama de la segunda ruta con paros y acelerones

4.1.3 Prueba de precisión del modelo de detección

La última prueba se realizará para medir la precisión del modelo de detección facial, como ya se explicó en el capítulo 2, se ha utilizado un algoritmo de la librería de Python Dlib que utiliza puntos de referencia faciales para encontrar los rasgos de la cara en la imagen.

Este modelo incluye una función propia para medir el Error Absoluto Medio, una medida común en Inteligencia Artificial para medir la precisión de los modelos. dlib.test_shape_predictor() que sería esta función lo que hace es medir la distancia

media entre los puntos de referencia faciales encontrados con el modelo utilizado en una imagen con los puntos de referencia faciales reales, los cuales han sido previamente etiquetados en la imagen. Cuanto menor sea este valor mejor será la precisión del modelo.

4.2 Resultados de las pruebas

A continuación, se presentan los resultados obtenidos con el sistema propuesto.

4.2.1 Resultados de la prueba de latencia

TABLA 1. RESULTADOS DE LA PRUEBA DE LATENCIA DEL SISTEMA

	Raspberry Pi	Streaming	Web
	Ping (ms)	**Ping 2 (ms)**	**Ping 3 (ms)**
	216,8	760	7,5
	176,5	800	7,7
	150,5	870	8,3
	127,7	900	8,4
	118,4	780	8,2
	123,9	910	8,3
	112	840	8,2
	107,1	920	8,1
	104,2	800	8,1
	100,1	870	8
Media	**133,72**	**845**	**8,08**

TOTAL (ms)	**986,8**
TOTAL (s)	**0,9868**

En total pasan 0.99 segundos, casi 1 segundo entre que la Web envía las imágenes de la Webcam hasta que el usuario puede ver en la web el movimiento de la cámara del dron. Tal y como se sospechaba el mayor tiempo se consume en el streaming de video. Las mejoras en este aspecto de discutirán en el apartado 4.3.

4.2.2 Resultados de la prueba de precisión en la transformación

4.2.2.1 Resultados de la ruta 1

En la primera ruta se trataba de recorrer todo el contorno de la pantalla con la mirada para terminar en el centro empezando por la esquina inferior izquierda en sentido horario. El primer resultado se muestra en la siguiente tabla donde se compara la ratio vertical de la mirada con el duty cycle que se envía al servo después de la transformación. Con estos datos se mide la transformación del movimiento vertical de los ojos al duty cycle del servo que hace el movimiento vertical.

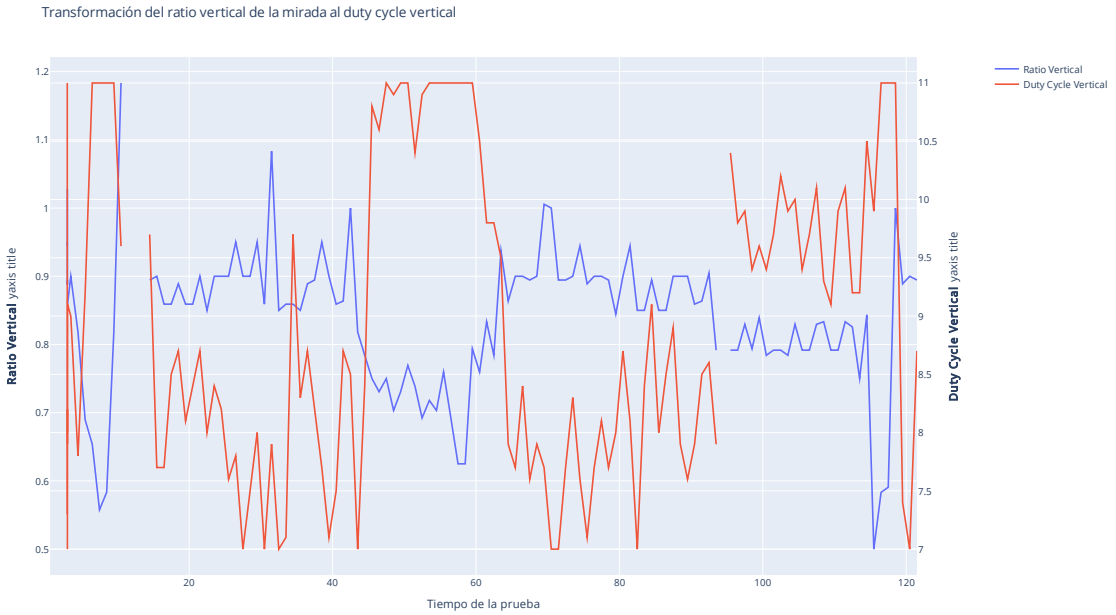

Figura 19. Transformación de la ratio vertical de la mirada al duty cycle vertical

Para entender este gráfico se van a superponer unas líneas para explicarlo. Solo se mirará la línea naranja ya que es la que nos indica el movimiento vertical del servo.

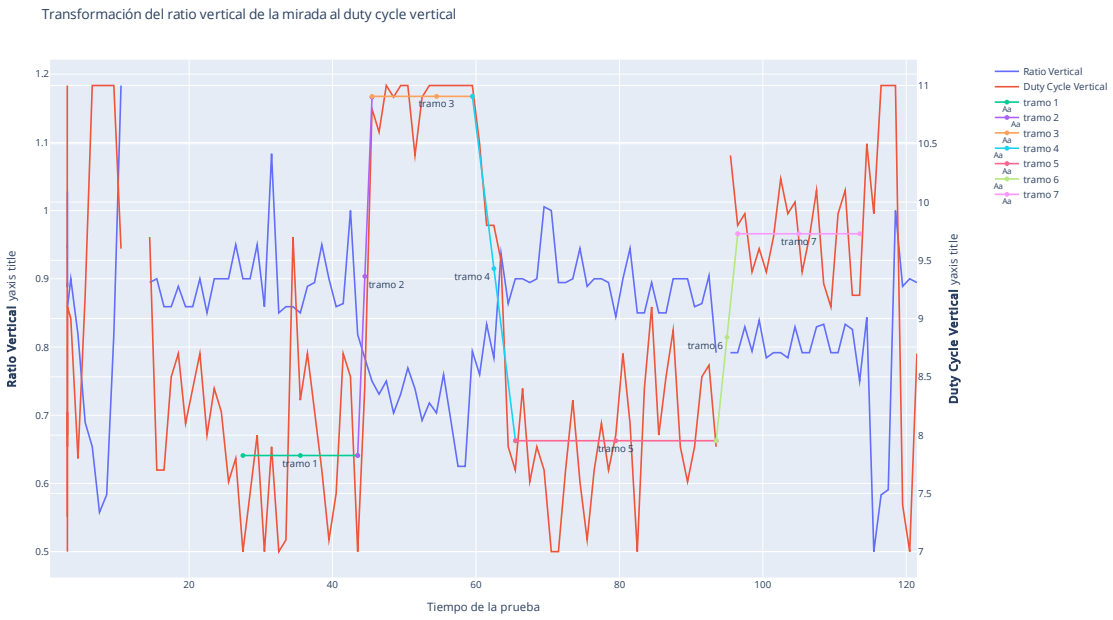

Figura 20. Transformación de la ratio vertical de la mirada al duty cycle vertical con tramos

Los distintos tramos del 1 al 7 indican la media calculada para ese tramo y en su conjunto representan la tendencia del movimiento del servo, hasta el inicio del primer tramo no se tienen en cuenta los datos obtenidos ya que son al encender los distintos servidores y la cámara y ponerse en posición de inicio para comenzar la prueba. Como se explicó en el apartado 4.1.2 la prueba comienza por el extremo inferior izquierdo de la pantalla por lo que en este gráfico debería empezar por el límite inferior del servo, el cual es un duty cycle de 7. Como se puede ver en la imagen el primer tramo oscila entre el 7 y el 8 [tramo 1], lo que indica que empieza abajo. Sube abruptamente [tramo 2] hasta colocarse en el límite superior que es 11 [tramo 3] y luego baja algo más lentamente [tramo 4], aquí se mantiene abajo porque tiene que recorrer el contorno inferior de la pantalla [tramo 5] y finalmente sube de nuevo [tramo 6]. Esta última subida se queda hasta más o menos la mitad entre 7 y 11 por lo que sabemos que ha subido hasta el medio de la pantalla [tramo 7]. Esto último tiene coherencia con la ruta establecida ya que la prueba debía terminar en el centro de la pantalla. Los últimos puntos son al retirar los ojos del software para apagar la prueba por eso se ve una bajada al final.

En esta prueba se debía recorrer toda la pantalla y terminar en el centro moviendo los ojos a un ritmo constante, sin embargo, se puede ver como los distintos tramos no duran lo mismo. Esto se debe a que la pantalla no es un cuadrado perfecto, sino que es un

rectángulo con una ratio de 16:9 por lo que subir la mirada por los laterales a un ritmo constante supone que el tiempo en las subidas y bajadas es menor que al recorrer el marco superior e inferior, pero también puede ser por error humano ya que es complicado mover los ojos a un ritmo constante.

Con esta gráfica podemos ver que realmente la mirada es detectada y el servo se mueve en función de esta tanto en la dirección de la mirada como a la velocidad de la mirada. Se puede apreciar la relación de la transformación y como se suavizan los valores detectados de la mirada para enviar el duty cycle al servo. Estos valores podrían suavizarse mucho más con distintas técnicas y de esto se hablará en las conclusiones.

Por otro lado, la ratio horizontal se ha medido de la misma manera y el resultado se muestra en la Figura 21 a continuación.

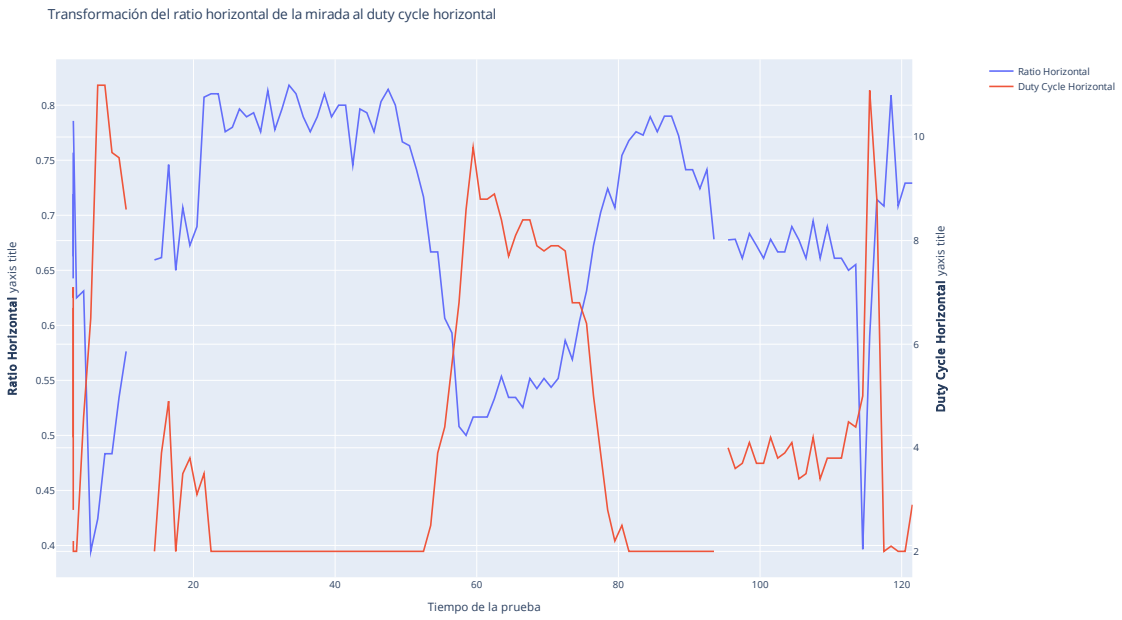

Figura 21. Transformación de la ratio horizontal de la mirada al duty cycle horizontal

Otra vez, la línea naranja indica el movimiento del servo, esta vez en horizontal. La línea azul recoge el movimiento de los ojos en horizontal.

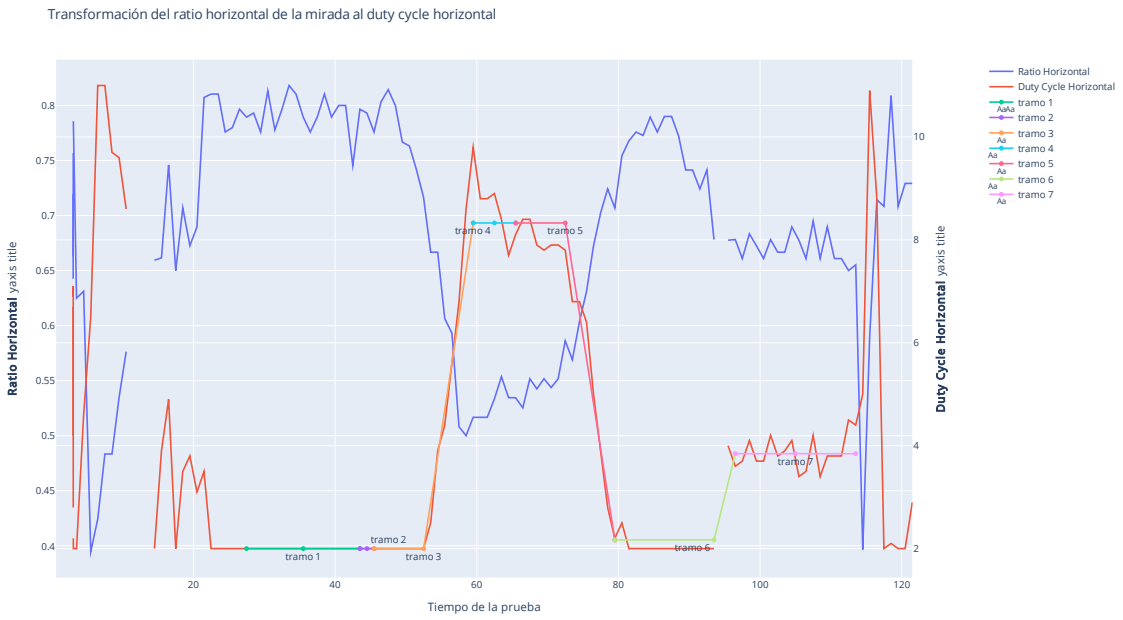

Figura 22. Transformación de la ratio horizontal de la mirada al duty cycle horizontal con tramos

Es una gráfica muy similar a la de la ratio vertical, pero esta vez observando la línea naranja 2 es izquierda y 11 derecha, es decir un punto bajo en la gráfica representa la izquierda y un punto arriba representa la derecha. En el primer tramo comenzamos en el punto más bajo porque estamos en la esquina inferior izquierda de la pantalla, al subir se mantiene la mirada a la izquierda por lo que el tramo 2 sigue estando abajo. En el tramo 3 que sería recorrer la pantalla por la parte superior, comienza a la izquierda y termina en la derecha. Durante el tramo 4 y 5 sucede lo mismo, pero al revés, empieza en la derecha y termina a la izquierda. El final del tramo 5 sería la esquina inferior izquierda desde la que se desplaza la mirada al centro por lo que en el tramo 6 el duty cycle se encuentra a mitad entre 2 y el 11.

Se puede observar que la transformación de la ratio horizontal de la mirada es mucho mejor que el de la vertical, ya que no hay tantos picos en las curvas, sino que se mantiene más o menos constante lo que hará que el servo tenga un movimiento más fluido. Sin embargo, en ningún momento se llega al límite superior por lo que la transformación ha fallado al leer los valores de la mirada a la derecha de la pantalla, lo que implicaría que el servo nunca llegaría a su límite de movimiento horizontal. Esto puede ser debido a un fallo en la calibración del extremo derecho, que se ha proyectado mucho más a la derecha de lo que realmente está. La calibración del software es otro punto que mejorar del sistema tal y como se explicará en el apartado 4.3.

El resto de los resultados de la prueba se adjuntan en el Anexo I.

4.2.2.2 Resultados de la ruta 2 con paros y acelerones

Con la segunda prueba se quería probar cómo respondía el sistema a paros y acelerones.

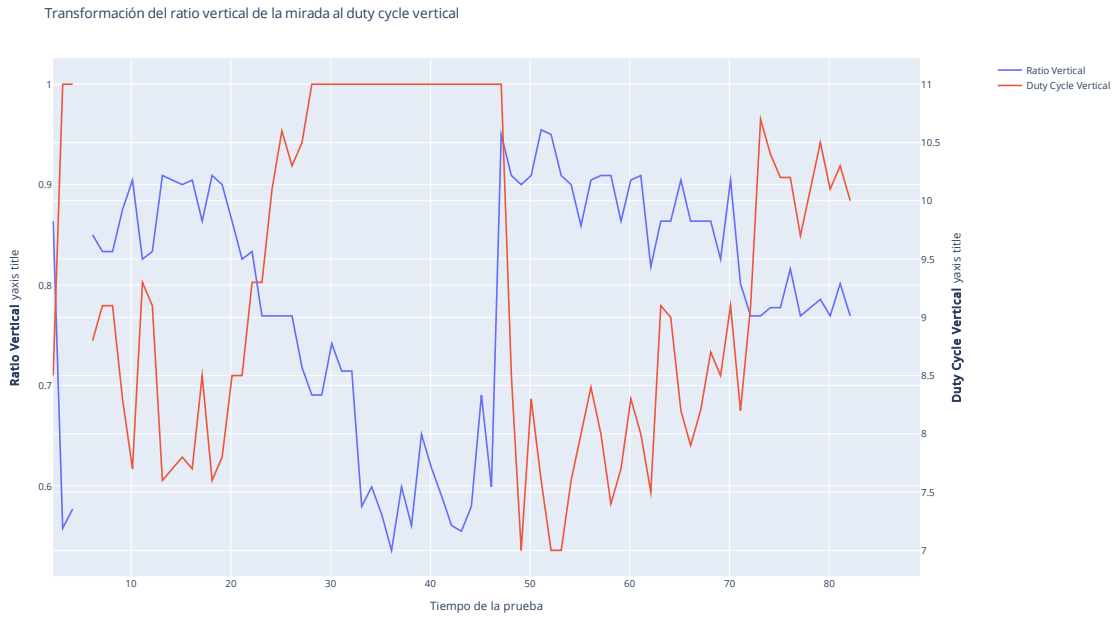

Figura 23. Transformación de la ratio vertical de la mirada al duty cycle vertical

En la gráfica se puede observar cómo se comienza desde el punto más bajo pues la prueba comienza en la esquina inferior izquierda y poco a poco va subiendo hasta el límite superior. Como se explicó en el punto 4.1.2 se debía hacer un paro en la esquina superior derecha, esto se refleja claramente en la gráfica donde se ve el punto más alto de la línea naranja mantenido durante más tiempo que en la prueba anterior. Seguidamente se debía hacer una bajada rápidamente hasta la esquina inferior derecha como se ve en la gráfica en el segundo 47. De ahí vuelve lentamente al principio hasta el segundo 73 donde hace una subida abrupta hasta más o menos la mitad de la pantalla, donde se mantiene un tiempo antes de terminar la prueba.

Si ahora observamos la tabla del movimiento horizontal de los ojos y el servo se pueden observar curvas similares. Al principio está a la izquierda y va cambiando hacia la derecha donde se mantiene pues hay un paro en la ruta en la esquina superior derecha.

Después va volviendo hacia el punto inicial a la izquierda, se mantiene a la izquierda y rápidamente se mueve al medio, pues en el recorrido de la ruta se especifica un movimiento rápido de la esquina izquierda al centro. En el centro se mantiene un tiempo hasta que termina la prueba como se puede observar claramente en el último tramo de la gráfica.

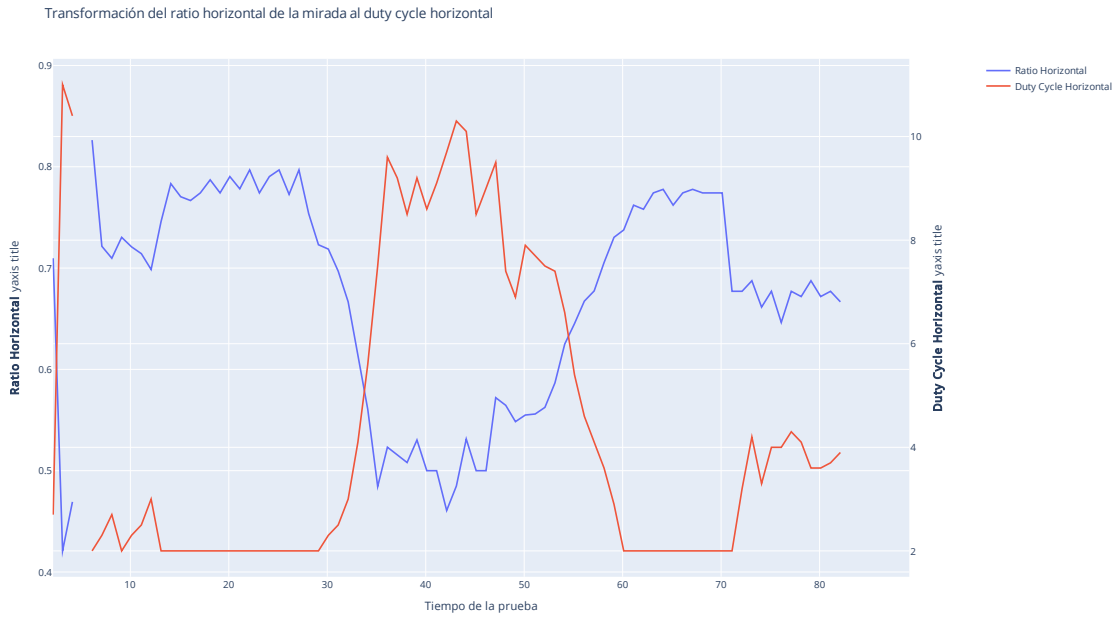

Figura 24. Transformación de la ratio horizontal de la mirada al duty cycle horizontal

Con esta prueba se revela que el servo puede mantenerse quieto en un punto y acelerar y desplazarse rápidamente a otro punto. A la hora de mantenerse quieto en un punto influye en gran medida que los valores de la detección de la mirada van variando ligeramente, se podría suavizar aún más la curva para evitar este ruido o mejor el software de detección. También la precisión de la transformación dependerá del tamaño de la pantalla ya que cuanto más pequeña sea menor será el rango de movimiento de los ojos y más difícil medir el movimiento de estos sin cometer errores.

4.2.3 Resultados de la prueba de precisión del modelo

Para realizar la prueba de precisión del modelo se ha utilizado la función test_shape_predictor() incluida en la librería de Dlib, con la cual se realiza el cálculo del Error Absoluto Medio utilizando para ello unas imágenes con las que no se haya entrenado el modelo.

Esta función recibe el PATH del modelo y el PATH de los puntos de referencia faciales reales de la imagen que se encuentran en un archivo XML y calcula el error absoluto medio. Para las imágenes utilizadas el error ha sido de 0.061272969444244145, aproximadamente 0.06. Es un error muy bajo para el modelo de puntos de referencia faciales.

Se puede apreciar en la Figura 25 cómo el software detecta correctamente la posición de las pupilas en cada imagen, indicadas con cruces verdes y además gracias a las ratios vertical y horizontal calculadas indica la dirección de la mirada en la imagen.

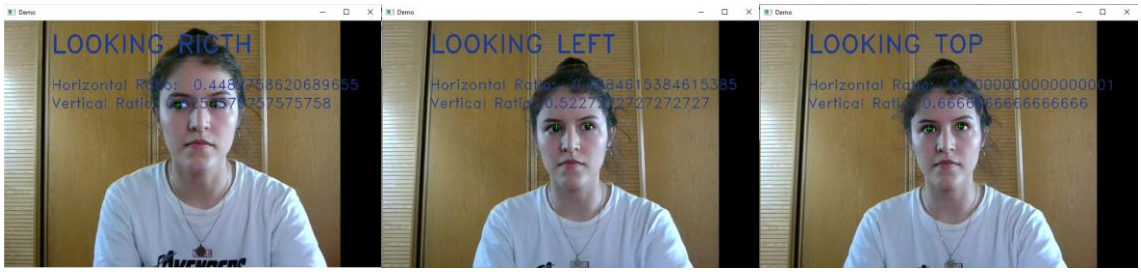

Figura 25. Resultados del software de detección de la mirada

4.3 Discusión de los resultados

Analizando los resultados obtenidos se demuestra que el diseño es funcional, los motores cambian su duty cycle en función de la dirección de la mirada con un tiempo de latencia razonable. Sin embargo, también muestran las debilidades del sistema, aquellos aspectos a mejorar con más tiempo y recursos económicos.

El sistema responde a la dirección de la mirada, no obstante, es muy sensible a la calibración que referencia los límites de la pantalla para determinar el punto exacto al que está mirando el usuario. Esto se podría solucionar integrando por ejemplo la calibración automática al principio de cada uso, de tal forma que se pida al usuario que mire a distintos puntos en su pantalla determinados previamente para que sirvan de referencia a la hora de determinar qué coordenadas de los ratios vertical y horizontal implican estos puntos. Dependiendo del tamaño de la pantalla, la disposición de esta o la distancia del usuario a la Webcam, las coordenadas de la mirada extraídas de la imagen van a indicar distintos puntos en la pantalla. En la Figura 22 por ejemplo, se

puede observar como el punto más a la derecha de la pantalla no está bien calibrado, por lo que, aunque el usuario mire al borde derecho de la pantalla el servo nunca llega a moverse hasta su límite de rango horizontal, por lo que se pierde información y el hardware no está bien aprovechado. Como se ha explicado existe una solución sencilla para este problema que podría implementarse en futuras mejoras del software.

Otro aspecto del software que se podría optimizar es el alisado de la curva, aunque la mirada pueda temblar, los servos no deberían percibir variaciones pequeñas en la ruta ya que hace que éstos se desplacen temblando y la experiencia de usuario no sea la mejor. Existen muchas técnicas avanzadas de estadística para alisar las curvas realizadas por una serie de puntos. Por supuesto con unos servos mejores el movimiento también sería más fluido, los servos empleados para la implementación del diseño tienen muchas limitaciones para desplazarse de forma fluida ya que se mueven mediante una señal PWM generada por un pin de la Raspberry Pi, además solo pueden girar 180 grados. La precisión del sistema la han limitado en gran parte los servos ya que la resolución de estos es de solo un decimal en el duty cycle, e incluso esta afirmación no es del todo correcta ya que no se pueden desplazar del 7.1 al 7.2 sino que depende del ancho de pulso, además los servos no pueden mantener una posición, si se cambia el duty cycle, se desplazan unos grados y se quedan oscilando en el punto, no se detienen, para detenerlos en este proyecto se han tenido que parar cada vez que se cambiaba el duty cycle, lo que hace que se incremente el tiempo de latencia y además no se paran en el punto exacto.

Unos servos mejores solucionarían muchos de estos problemas y además podrían permitir un mayor grado de movimiento de la cámara llegando incluso a los 360 grados. Esto combinando con un mejor alisado de la curva daría mayor sensación de fluidez a la aplicación. Sin embargo, estos aspectos no influyen en la funcionalidad del sistema que sigue desplazándose, siguiendo la mirada, únicamente mejorarían el aspecto de UX (User eXperience) que es importante, pero en un Producto Mínimo Viable puede quedar relegado a un segundo plano.

La latencia del sistema como se explicó en los objetivos es un punto muy importante para que el sistema resulte funcional, ya que si tardase mucho en desplazarse se perdería toda la sensación de inmersión y control. Los resultados obtenidos muestran que desde

que el usuario mueve la mirada hasta que ve su movimiento reflejado en las imágenes de la pantalla pasa tan solo 1 un tiempo razonable de latencia que permite la sensación de control de la cámara y no resulta tedioso para su uso. El 80% del tiempo se va en el streaming de video de la cámara de la Raspberry, esto puede ser debido a que la resolución aplicada al video es demasiado grande o a que no se está optimizando correctamente la codificación, por ejemplo, es posible que se esté enviando un archivo de audio del video cuando no es necesario, o que el códec utilizado no sea el mejor. Del mismo modo se podrían codificar mejor las imágenes recogidas por la webcam para conseguir mejor resolución con menor peso en el envío.

Respecto a la precisión en la detección de la mirada, Dlib muestra resultados excelentes detectando los puntos de referencia faciales, sin embargo, el algoritmo aplicado para determinar el ratio vertical y horizontal que se traduce a los duty cycle de los servos compara las coordenadas de las pupilas con el total del blanco en los ojos para sacar estas ratios. Esto significa que la precisión depende de lo grande que se vea el ojo en la imagen y de la proporción que ocupe en ésta, es decir que depende de lo cerca que esté la persona de la pantalla. Además, el software es muy sensible a los parpadeos que reducen el blanco de los ojos e incluso hacen que se pierda la posición de la mirada. Se podrían encontrar mejoras para el cálculo de la proyección de la mirada y la transformación al duty cycle de los servos sin necesidad de cambiar el modelo de Inteligencia Artificial, por ejemplo, detectando los parpadeos y eliminando estos puntos que no tiene ninguna utilidad de la curva.

En conclusión, los resultados muestran un sistema funcional abierto a futuras mejoras y que no tiene problemas o fallas que sean imposibles de arreglar o mejorar.

5. IMPACTO SOCIOECONÓMICO

En este proyecto se desarrolla una forma de controlar la cámara integrada en un dron mediante la dirección de la mirada de tal manera que pilotar un dron se vuelve más inmersivo. Los drones tienen gran cantidad de aplicaciones en distintos campos y su uso va en aumento cada año. Los usuarios de estos dispositivos demandan un control más inmersivo, en concreto en las aplicaciones que tienen estos vehículos para tomar imágenes y videos el piloto necesita estar atento a las imágenes que está tomando y no tanto al control del dron.

Las gafas de Realidad Virtual son muy utilizadas para el vuelo de drones con cámara ya que permiten una experiencia en primera persona o FPV (*First Person View*). El éxito de estas gafas recae precisamente en la inmersión que producen. El problema con estas gafas es que son muy caras y no todos los aficionados a pilotar drones pueden permitírselas [8]. Es aquí donde entra el desarrollo realizado en este trabajo, se propone una forma mucho más barata de acceder a esta sensación de inmersión en el vuelo, ya que el usuario no necesita de hardware adicional para el uso de la aplicación web, únicamente su propio smartphone o Tablet.

Es una aplicación web sencilla que no necesita de mucho mantenimiento ni de un coste inicial elevado. Se puede costear con anuncios en la página web o podría ser adquirida mediante una suscripción. La primera opción no tendría ningún coste para el usuario y la segunda opción seguiría siendo significativamente más económica que adquirir unas gafas de realidad virtual.

Como se explicó en la introducción los drones son muy utilizados en catástrofes naturales ya que pueden acceder a zonas peligrosas sin poner en riesgo la vida de una persona. Para estas operaciones humanitarias el servicio podría ser gratuito, además como está alojado en una web es accesible desde cualquier parte del mundo simplemente con una conexión a internet en el smartphone.

Aparte de la aplicación a la que está destinada, esta tecnología puede tener un impacto en áreas completamente ajenas a los drones, ya que no deja de ser un software que permite controlar unos motores a través de la mirada completamente en remoto. Esto

deja las manos libres para realizar otras operaciones mientras que el usuario mueve algo simplemente con la dirección de su mirada.

6. PRESUPUESTO

6.1 Introducción

En este apartado se especifica el presupuesto asociado al presente trabajo, el cual se dividirá en tres secciones: hardware y personal. Se incluirán tanto los costes de implementación del software, así como los de la investigación necesaria para el desarrollo de este trabajo.

6.2 Desarrollo del presupuesto

El capítulo 1 de la tabla de presupuesto recogerá todas las estimaciones de coste asociadas al hardware indicando la cantidad, el precio unitario y el importe total del artículo. Todas las referencias de coste estarán indicadas en la bibliografía.

El capítulo 2 del presupuesto recogerá los importes del personal. Estos se indicarán mediante el cálculo precio/hora indicando la cantidad de horas en CANTIDAD y el precio por hora en PRECIO, el total se verá reflejado en la columna IMPORTE. La estimación de horas del personal incluye la realización del trabajo, es decir la investigación previa para ver la viabilidad del proyecto, así como el desarrollo e implementación del software. El precio/hora del personal se ha extraído de las tablas sobre Retribuciones proporcionadas por la UC3M [35].

TABLA 2. PRESUPUESTO DESGLOSADO

CÓDIGO	Uds	DESCRIPCIÓN	CANTIDAD	PRECIO	IMPORTE
CAPÍTULO C1: HARDWARE					
C1.1	**ud**	**Raspberry Pi 2 Model B** Quad-Core Cortex A7 a 900 MHz con 1 GB de RAM. Puertos: 4 x USB 2.0, 1 x 40 GPIO pin, 1 X HDMI, 1 x Ethernet, 1 x Combo audio/MIC, 1 x Interfaz de cámara (CSI), 1 X Interfaz de Pantalla (DSI), 1 x MicroSD, 1 x Núcleo Gráfico 3D	1,00	39,87 [36]	39,87

C1.2 ud Ordenador Portátil MSI GF65 Thin 9SD-072XES

Ordenador Portátil MSI con procesador Intel Core i7-9750H RAM de 16GB, memoria SSD de 512GB, tarjeta gráfica GTX 1660 Ti y 15.6.". Se usará como servidor para la aplicación.

1,00	1048,99 [37]	1048,99

C1.2 ud Adaptador Wifi USB BrosTrend

Adaptador wifi de doble banda para Linux. 5GHz/2,4GHz, Antena 5dBi

1,00	19,99 [38]	19,99

C1.3 ud Kit Pan-Tilt - Desplazamiento Horizontal-Vertical

Pequeño kit especialmente diseñado para hacer el movimiento horizontal (pan) y vertical (tilt) de una cámara o accesorio en tu robot. Utiliza dos piezas de aluminio de color negro para sostener ambos servos

1,00	7,75 [39]	7,75

C1.4 ud Fuente de alimentación

Fuente de alimentación 5V, 2 A, 2000 mAh, Micro USB.

1,00	8,99 [40]	8,99

C1.5 ud Cable de puente macho a hembra

Cable de puente macho a hembra 20 cm, 1 pin a 1 pin.

1,00	0,09	3,60 [41]

C1.6 ud Micro servo miniatura SG90

Peso: 9 gramos
Dimensiones: 22.2 x 11.8 x 31 mm
Torque: 1.8 kg/cm
Velocidad: 0.1 s/60 grados
Alimentación: 4.8 V (~5V)
Temperatura de funcionamiento: 0 º C – 55 º C

	2,00	2,45 [42]	4,90

TOTAL, CAPÍTULO C1 HARDWARE 1134,09

CAPÍTULO C2 PERSONAL

CÓDIGO	Uds	DESCRIPCIÓN	CANTIDAD	PRECIO	IMPORTE
C2.1	**h**	**Alumno UC3M**			
		Alumno que desarrollará el TFG.			
			324,00	9,64 [35]	3123,36
C2.2	**h**	**Tutor**			
		Tutor del alumno para el desarrollo del TFG.			
			20,00	11,24 [35]	224,8

TOTAL, CAPÍTULO C2 PERSONAL 3348,16

TABLA 3. RESUMEN DE PRESUPUESTO

CAPÍTULO	RESUMEN	IMPORTE	%
C1	HARDWARE.....	1134,09	25,30
C2	PERSONAL.....	3348,16	74,70
	PRESUPUESTO DE EJECUCIÓN	**4482,25**	

Asciende el presupuesto a la expresada cantidad de CUATRO MIL CUATROCIENTOS OCHENTA Y DOS euros y VEINTICINCO céntimos.

Del total del presupuesto es personal es casi el 75% de éste ya que se han requerido de muchas horas para investigar la cuestión y desarrollarla. El resto de los componentes solo representan un 25% del presupuesto.

7. CONCLUSIÓN Y FUTUROS TRABAJOS

El objetivo de este proyecto era estudiar la viabilidad de desarrollar un sistema que permita controlar la cámara integrada en un dron mediante la mirada, a través de algoritmos de Visión Artificial. La motivación para este trabajo parte de que los drones y las tecnologías aplicadas al control de éstos son un tema de investigación que en los últimos años está teniendo su mayor crecimiento. Esto se debe a que los drones tienen aplicaciones en gran cantidad de campos distintos, se emplean en el transporte de productos, en las operaciones militares e incluso en las operaciones de rescate durante catástrofes naturales.

Se buscan nuevas formas de interactuar y controlar estos vehículos, en el capítulo 2 tras realizar una investigación sobre la situación actual de la cuestión, tanto de los drones como de la Visión Artificial aplicada a la detección de la mirada, se determina que el objetivo propuesto tiene utilidad en el mercado. Durante las operaciones militares a parte de control del UAV se deben procesar muchas tareas al mismo tiempo, es por esto por lo que se están investigando formas más eficaces de interactuar con el dron como se ve en las distintas publicaciones estudiadas. Con la propuesta de este trabajo la cámara del dron podría moverse únicamente con la mirada, dejando libres las manos para ocupar otras tareas y además haciendo más natural e inmersivo el control de la cámara.

En este proyecto también se investiga la viabilidad técnica de un sistema de estas características, así como las herramientas más adecuadas para implementarlo. De esta forma se determina que la mejor solución es una aplicación Web, ya que tiene la ventaja de ser accesible desde cualquier dispositivo a través del navegador. Además, todo el motor de la aplicación se encontrará en una API-REST de tal forma que quede desligado de la interfaz, lo que permitiría utilizar el software de detección facial en distintas interfaces las cuales se pueden desarrollar a posteriori, como por ejemplo una APP nativa para un sistema operativo concreto.

Por otro lado, para la conexión de todas las partes del sistema se comparó el método de peticiones mediante HTTP con la comunicación mediante WebSockets, esta última permite una mayor velocidad en el envío de mensajes, ya que mantiene la comunicación abierta todo el tiempo al contrario que las peticiones HTTP que deben abrir una nueva

conexión cada vez que envían un mensaje. La velocidad en la comunicación entre las distintas partes del sistema es un factor clave en el desarrollo del proyecto ya que la latencia determinará si el sistema es funcional.

Con todo esto se realiza el diseño la aplicación Web para controlar la cámara del dron. Este diseño es el de un Producto Mínimo Viable por lo que se ha implementado mediante una Raspberry Pi y unos servos conectados en pan and tilt que mueven la cámara conectada a la Raspberry, todo completamente desligado del dron. En función de los resultados obtenidos con esta implementación se podrían buscar mejoras de software y mejor hardware.

Los resultados obtenidos muestran que el diseño es funcional, los motores cambian su duty cycle en función de la dirección de la mirada con un tiempo de latencia razonable. El usuario solo necesita entrar en la dirección web y encender la Webcam, a partir de ahí puede acceder al streaming de la cámara conectada a la Raspberry Pi. Si mueve los ojos por la pantalla, los servos le seguirán. Los objetivos del proyecto se han alcanzado.

Los resultados muestran claramente los próximos pasos para el diseño. Por ejemplo, se podría implementar una calibración automática del sistema antes de cada uso para que el sistema responda correctamente al movimiento de la mirada en todo momento. En cuanto a la transformación del ratio horizontal y vertical de los ojos al duty cycle de los servos, se podría mejorar el alisado de la curva y eliminar mediciones erróneas, como cuando se parpadea, que se miden puntos extremos cuando el usuario no estaba cambiando la dirección de la mirada.

Otro punto importante que hay que tener en cuenta en futuros trabajos es que el hardware también tenía bastantes limitaciones, los servomotores no son adecuados para el control de la precisión ya que intentan girar según los pulsos, pero siempre se quedan por detrás por lo que nunca llegan al punto exacto, además cuando se paran siguen oscilando con el pulso por lo que no dejan de vibrar. Con unos motores mejores el movimiento de la cámara podría ser más fluido y preciso.

Por último, la latencia del sistema podría mejorarse bastante únicamente optimizando la codificación de las imágenes de los videos tanto de la webcam como de la cámara del dron, esto es posible mediante códecs mejores.

En resumen, se han conseguido los objetivos y se ha diseñado e implementado un sistema que permite mover la cámara integrada en un dron mediante la mirada. Existen varios aspectos que se podrían mejorar con mayores recursos, pero el Producto Mínimo Viable se considera alcanzado.

8. BIBLIOGRAFÍA

[1] «Detección de Objetos en Movimiento utilizando Drones», *Embention*, may 15, 2020. https://www.embention.com/es/news/deteccion-de-objetos-en-movimiento-con-drones/ (accedido ago. 21, 2021).

[2] D. Wang, P. Hu, J. Du, P. Zhou, T. Deng, y M. Hu, «Routing and Scheduling for Hybrid Truck-Drone Collaborative Parcel Delivery With Independent and Truck-Carried Drones», *IEEE Internet Things J.*, vol. 6, n.º 6, pp. 10483-10495, dic. 2019, doi: 10.1109/JIOT.2019.2939397.

[3] M. R. Dileep, A. V. Navaneeth, S. Ullagaddi, y A. Danti, «A Study and Analysis on Various Types of Agricultural Drones and its Applications», en *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, nov. 2020, pp. 181-185. doi: 10.1109/ICRCICN50933.2020.9296195.

[4] C. Hutton, «Augmented Reality Interfaces for Semi-Autonomous Drones», en *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, mar. 2019, pp. 1361-1362. doi: 10.1109/VR.2019.8797893.

[5] Khan y Lee, «Gaze and Eye Tracking: Techniques and Applications in ADAS», *Sensors*, vol. 19, p. 5540, dic. 2019, doi: 10.3390/s19245540.

[6] «Eye Gaze Correction | Eye Gaze Correction for Vid...» https://www.leadingedgeonly.com/innovation/view/eye-gaze-correction- (accedido sep. 02, 2021).

[7] H. Chennamma y X. Yuan, «A Survey on Eye-Gaze Tracking Techniques», *Indian J. Comput. Sci. Eng.*, vol. 4, dic. 2013.

[8] «The Best VR Headsets for 2021», *PCMAG*. https://www.pcmag.com/picks/the-best-vr-headsets (accedido sep. 02, 2021).

[9] C. Connley, «These are the 10 most in-demand A.I. jobs according to Indeed—and they all pay at least $95,000», *CNBC*, jun. 01, 2021. https://www.cnbc.com/2021/06/01/10-of-the-most-in-demand-ai-jobs-that-pay-at-least-95000.html (accedido sep. 02, 2021).

[10] «Coordinated Plan on Artificial Intelligence 2021 Review | Shaping Europe's digital future». https://digital-strategy.ec.europa.eu/en/library/coordinated-plan-artificial-intelligence-2021-review (accedido ago. 17, 2021).

[11] Jefatura del Estado, *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*, vol. BOE-A-2018-16673. 2018, pp. 119788-119857. Accedido: ago. 17, 2021. [En línea]. Disponible en: https://www.boe.es/eli/es/lo/2018/12/05/3

[12] J. Buckley, *Air Power in the Age of Total War*. Routledge, 2006.

[13] B. L. Takeyas, «INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL», p. 3.

[14] «Visión por computadora - Libro online de IAAR». https://iaarbook.github.io/vision-por-computadora/ (accedido ago. 17, 2021).

[15] «La Visión Artificial y el reconocimiento de imágenes: procesamiento automatizado», *Santander Global Tech*, dic. 14, 2020. https://santanderglobaltech.com/vision-artificial-reconocimiento-imagenes-procesamiento-automatizado/ (accedido ago. 21, 2021).

[16] J. Martínez-Carranza, L. Valentín, F. Márquez-Aquino, J. C. González-Islas, y N. Loewen, «Detección de obstáculos durante vuelo autónomo de drones utilizando SLAM monocular», *Res. Comput. Sci.*, vol. 114, n.º 1, pp. 111-124, dic. 2016, doi: 10.13053/rcs-114-1-9.

[17] Y. Zhou, J. Hou, y Y. Gong, «Research and Application of Human-computer Interaction Technology based on Voice Control in Ground Control Station of UAV», en *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, dic. 2020, pp. 1257-1262. doi: 10.1109/ICCC51575.2020.9344892.

[18] E. Hjelmås y B. K. Low, «Face Detection: A Survey», *Comput. Vis. Image Underst.*, vol. 83, n.º 3, pp. 236-274, sep. 2001, doi: 10.1006/cviu.2001.0921.

[19] «Image Recognition using Histogram of Oriented Gradients (HOG) Descriptor», *DebuggerCafe*, jun. 08, 2020. https://debuggercafe.com/image-recognition-using-histogram-of-oriented-gradients-hog-descriptor/ (accedido ago. 21, 2021).

[20] L. R. Cerna, G. Cámara-Chávez, y D. Menotti, «Face Detection: Histogram of Oriented Gradients and Bag of Feature Method», p. 5.

[21] «Fig. 1 -An artificial neural network with three inputs and one output.», *ResearchGate*. https://www.researchgate.net/figure/An-artificial-neural-network-with-three-inputs-and-one-output_fig1_327440209 (accedido ago. 21, 2021).

[22] H. Ouanan, M. Ouanan, y B. Aksasse, «Facial landmark localization: Past, present and future», oct. 2016, pp. 487-493. doi: 10.1109/CIST.2016.7805097.

[23] X. Dong, Y. Yan, W. Ouyang, y Y. Yang, «Style Aggregated Network for Facial Landmark Detection», *ArXiv180304108 Cs*, mar. 2018, Accedido: ago. 21, 2021. [En línea]. Disponible en: http://arxiv.org/abs/1803.04108

[24] T. Taulli, «Python Language: What You Need To Know», *Forbes*. https://www.forbes.com/sites/tomtaulli/2020/07/24/python-language-what-you-need-to-know/ (accedido ago. 21, 2021).

[25] R. Python, «8 World-Class Software Companies That Use Python – Real Python». https://realpython.com/world-class-companies-using-python/ (accedido ago. 21, 2021).

[26] «TensorFlow», *TensorFlow*. https://www.tensorflow.org/?hl=es-419 (accedido ago. 21, 2021).

[27] «TensorFlow.js | Aprendizaje automático para desarrolladores de JavaScript», *TensorFlow*. https://www.tensorflow.org/js?hl=es-419 (accedido ago. 21, 2021).

[28] «OpenCV», *Wikipedia, la enciclopedia libre*. jun. 04, 2021. Accedido: ago. 17, 2021. [En línea]. Disponible en: https://es.wikipedia.org/w/index.php?title=OpenCV&oldid=136091757

[29] N. Boyko, O. Basystiuk, y N. Shakhovska, «Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library», ago. 2018, pp. 478-482. doi: 10.1109/DSMP.2018.8478556.

[30] S. Milborrow, «Stasm 4 User Manual», p. 20.

[31] V. Kazemi y J. Sullivan, «One millisecond face alignment with an ensemble of regression trees», en *2014 IEEE Conference on Computer Vision and Pattern Recognition*, jun. 2014, pp. 1867-1874. doi: 10.1109/CVPR.2014.241.

[32] «Bachelorpaper_Wouter_Pool.pdf». Accedido: ago. 21, 2021. [En línea]. Disponible en: http://essay.utwente.nl/77533/1/Bachelorpaper_Wouter_Pool.pdf

[33] I. Engineering, «Web Service Efficiency at Instagram with Python», *Medium*, jun. 26, 2016. https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366 (accedido ago. 21, 2021).

[34] D. G. Puranik, D. C. Feiock, y J. H. Hill, «Real-Time Monitoring using AJAX and WebSockets», en *2013 20th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS)*, abr. 2013, pp. 110-118. doi: 10.1109/ECBS.2013.10.

[35] «Retribuciones | UC3M». https://www.uc3m.es/conocenos/recursos-humanos/retribuciones (accedido ago. 18, 2021).

[36] «Raspberry Pi 2 Model B - Placa Base (Arm Quad-Core 900 MHz, 1 GB RAM, 4 x USB, HDMI, RJ-45) : Amazon.es: Informática». https://www.amazon.es/Raspberry-Pi-Model-Placa-Quad-Core/dp/B00T2U7R7I (accedido ago. 18, 2021).

[37] «MSI GF65 Thin 9SD-072XES Intel Core i7-9750H/16GB/512GB SSD/GTX 1660Ti/15.6" | PcComponentes.com». https://www.pccomponentes.com/msi-gf65-thin-9sd-072xes-intel-core-i7-9750h-16gb-512gb-ssd-gtx-1660ti-156 (accedido ago. 18, 2021).

[38] «BrosTrend Adaptador WiFi de Largo Alcance USB 1200Mbps, Velocidad de Red inalámbrica de Banda Dual de 5 GHz a 867 Mbps, 2,4 GHz a 300 Mbps, 2 Antenas WiFi de 5dBi, USB 3.0, Soporta Windows 10/8/7/XP : Amazon.es: Informática». https://www.amazon.es/Adaptador-BrosTrend-1200Mbps-Velocidad-inal%C3%A1mbrica/dp/B01IEU7UZ0/ref=asc_df_B01IEU7UZ0/?tag=googshopes-21&linkCode=df0&hvadid=301444355645&hvpos=&hvnetw=g&hvrand=4976857416359230712&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9061034&hvtargid=pla-377904052743&psc=1 (accedido ago. 18, 2021).

[39] E. Embajadores, «Kit Pan-Tilt - Desplazamiento Horizontal-Vertical - ROB-10335», *Electrónica Embajadores*. https://www.electronicaembajadores.com/es/Productos/Detalle/ROBMB04/mecanica-y-robotica/movimiento-robotica/kit-pan-tilt-desplazamiento-horizontal-vertical-rob-10335 (accedido ago. 18, 2021).

[40] «Tree-on-Life 5V 3A Fuente de alimentación Cargador Adaptador de CA Cable Micro USB con Interruptor de Encendido/Apagado para Raspberry Pi 3 pi Pro Modelo B B + Plus : Amazon.es: Informática». https://www.amazon.es/Tree-Life-alimentaci%C3%B3n-Adaptador-Interruptor/dp/B07R7LK1YS/ref=sr_1_12?adgrpid=62319073379&dchild=1&gclid=CjwKCAjw3_KIBhA2EiwAaAAlikAci6YuQY8zVMcg9cY4gcxznB-7wTglDlSMx6R5NRsKw9bJMldmTBoC8YwQAvD_BwE&hvadid=311318742431&hvdev=c&hvlocphy=9061034&hvnetw=g&hvqmt=b&hvrand=9326949847435919800&hvtargid=kwd-296183610226&hydadcr=5126_1831071&keywords=fuente+de+alimentaci%C3%B3n+de+5v&qid=1629316024&sr=8-12 (accedido ago. 18, 2021).

[41] «Dupont - Cable de puente macho a hembra (40 hilos, 20 cm, 1 pin a 1 pin) : Amazon.es: Informática». https://www.amazon.es/Dupont-Cable-puente-macho-hembra/dp/B00D7SDDLU (accedido ago. 18, 2021).

[42] «Micro servo miniatura SG90 BricoGeek | BricoGeek.com». https://tienda.bricogeek.com/servomotores/968-micro-servo-miniatura-sg90.html (accedido ago. 18, 2021).

ANEXO A. REPOSITORIO DE GITHUB E INSTRUCCIONES DE USO

En este Anexo se incluye el enlace al repositorio de GitHub, así como las instrucciones para instalar y correr el programa en Windows.

Se puede acceder al repositorio a través del siguiente link:
https://github.com/MelinaHernandezUC3M/control-drone-camera-by-gaze-tracking

Instalación en Windows

Para empezar, se debe instalar en un ordenador Python 3.6 con los módulos que se encuentran en requirements.txt.

Para descargarse el modelo de detección de landmarks de Dlib solo hay que ir a la siguiente dirección y la descarga debería comenzar automáticamente:
http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

Este modelo se debe incluir en una carpeta llamada trained_models de tal forma que la ruta quede como:
control-drone-camera-by-gaze-tracking/WebServer/GazeTracking/gaze_tracking/trained_models/shape_predictor_68_face_landmarks.dat

Para crear el streaming de la cámara conectada a la Raspberry se debe abrir la consola y escribir los siguientes comandos:

```
sudo modprobe bcm2835-v4l2

sudo rmmod bcm2835-v4l2

sudo modprobe bcm2835-v4l2

cvlc v4l2:///dev/video0 --v4l2-width 1920 --v4l2-height 1080 --v4l2-chroma h264 --sout '#standard{access=http,mux=ts,dst=0.0.0.0:12345}'
```

Seguidamente hay que acceder al código en la carpeta WebServer el archivo app.py y cambiar la dirección del streaming de video de la Raspberry a la IP que corresponda en la función encode_video_streaming().

Ahora ya se puede encender el servidor cuya aplicación se encuentra en la carpeta WebServer desde la consola del ordenador con el comando python app.py.

Una vez que está encendido el servidor se puede acceder a él desde el navegador con la dirección http://127.0.0.1:5000. Se deben aceptar los permisos para permitir al sitio acceder a la Webcam. Ya debería verse en la página las imágenes de la Webcam y al lado las de la cámara de la Raspberry Pi. Como es un prototipo se ven en todo momento las imágenes de las dos cámaras, aunque en el producto final debería aparecer a pantalla completa las imágenes de la cámara de la Raspberry Pi de tal forma que ocupe toda la pantalla.

Ahora hay que correr el cliente en la Raspberry Pi para que reciba los duty cycles que muevan los servos. El esquema para conectar los servos a la Raspberry Pi está representado en la siguiente Figura.

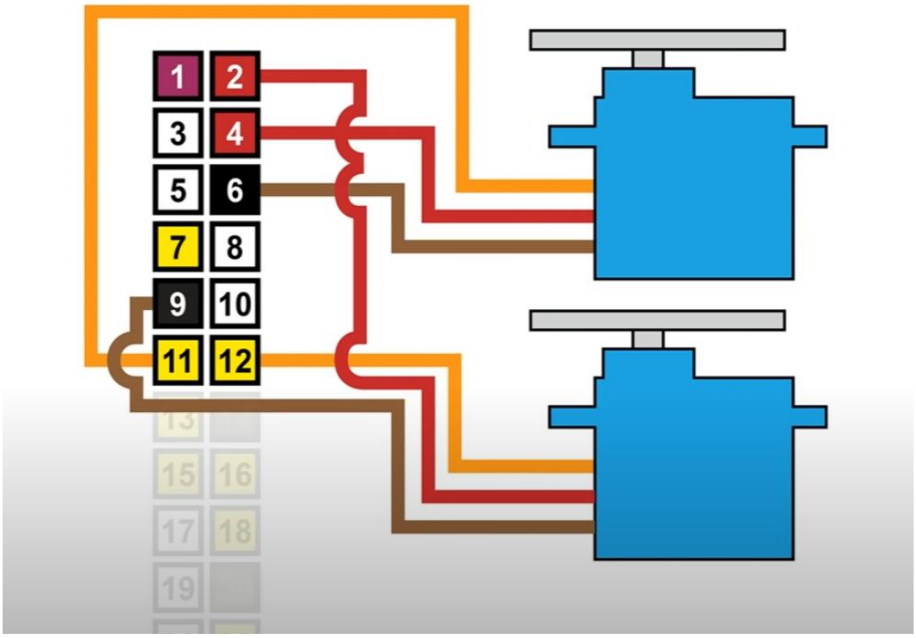

Figura A. 1. Conexión de los servos en pan and tilt

Desde la Raspberry Pi hay que acceder a la aplicación dentro de la carpeta DronMoveCamera y escribir en la consola el comando python3 app.py para encender el cliente.

Ahora deberían empezar a moverse los servos según lo que reciben del servidor principal, es decir según la dirección de la mirada de la persona que está mirando a la Webcam. Hay que tener en cuenta que en este prototipo los límites del movimiento de los servos serían los bordes de la pantalla del ordenador.

ANEXO B. RESULTADOS DE LAS PRUEBAS

En este Anexo se recogen el resto de los resultados de las pruebas realizadas.

Ruta 1

Prueba 2

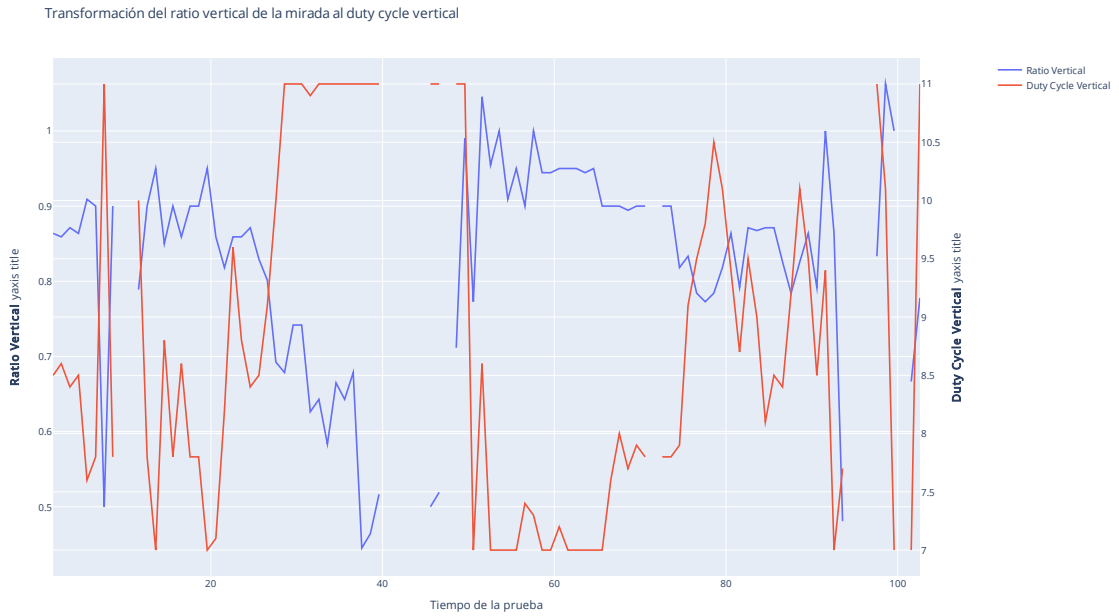

Figura B. 1. Resultados movimiento vertical ruta 1 prueba 2

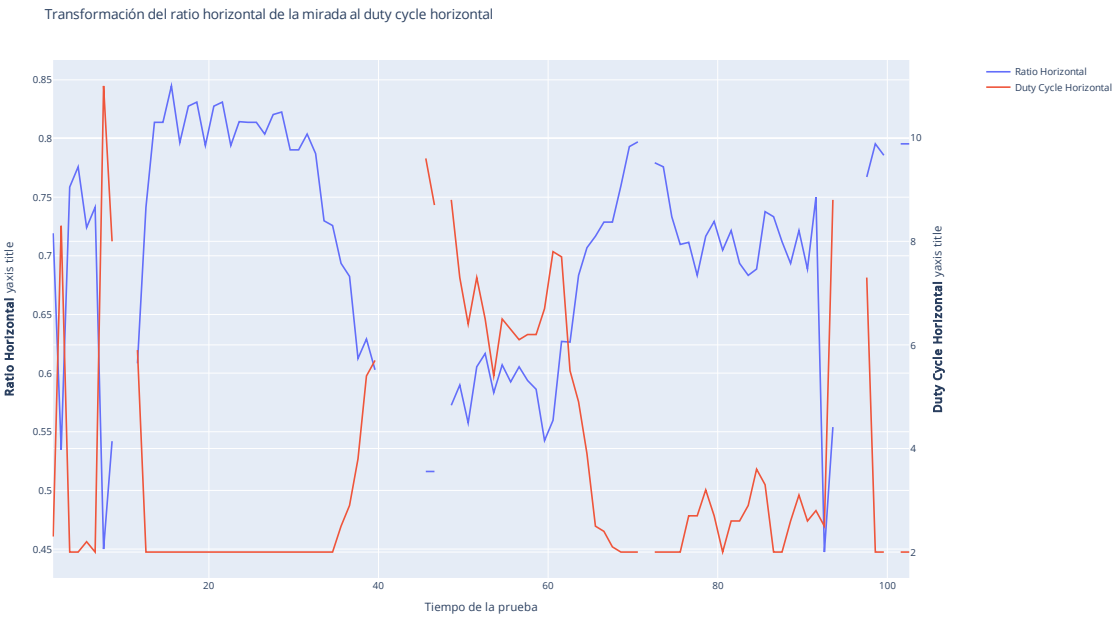

Figura B. 2. Resultados movimiento horizontal ruta 1 prueba 2

Prueba 3

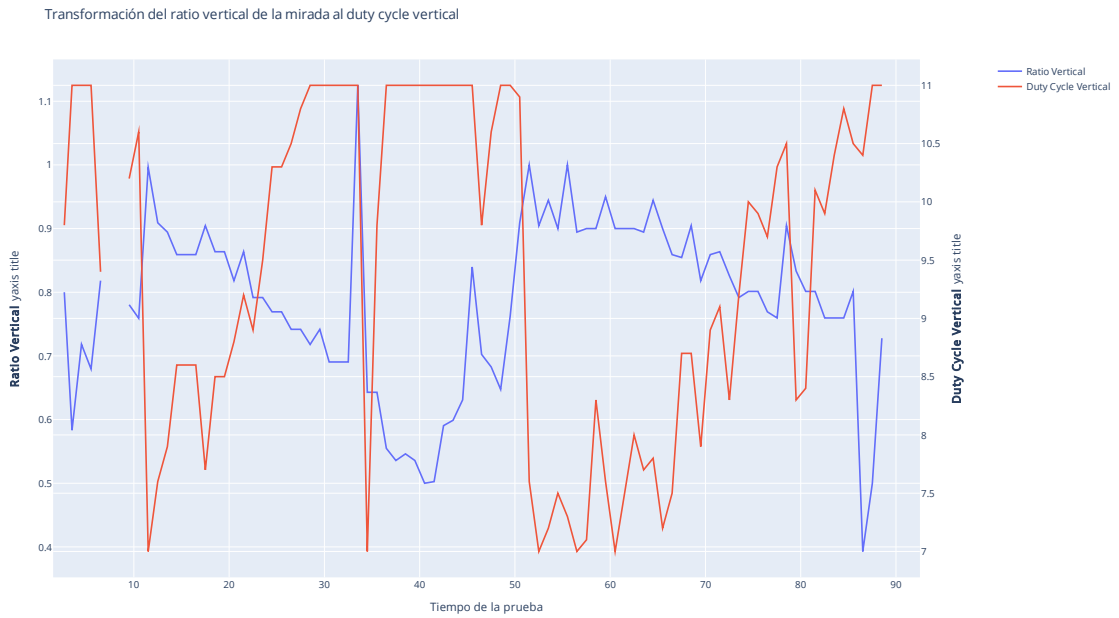

Figura B. 3. Resultados movimiento vertical ruta 1 prueba 3

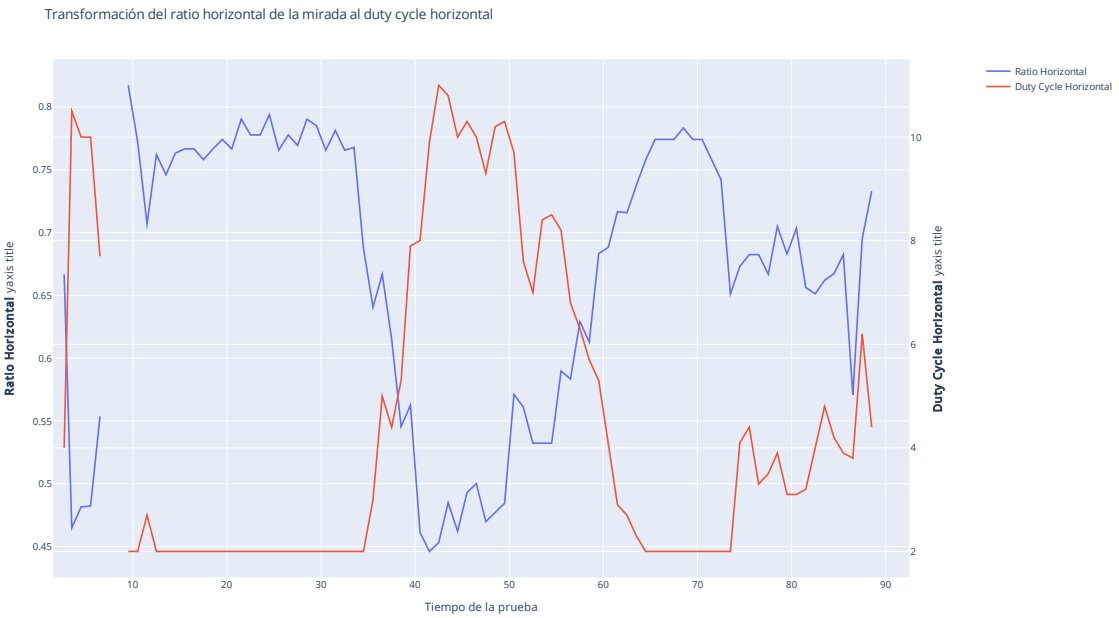

Figura B. 4. Resultados movimiento horizontal ruta 1 prueba 3

Ruta 2

Prueba 2

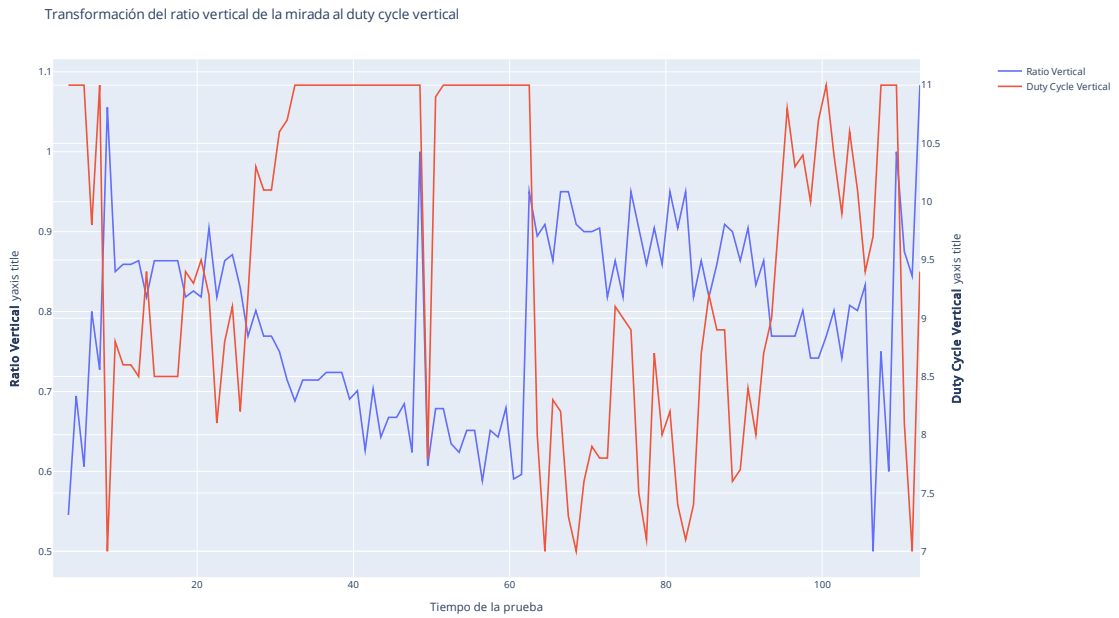

Figura B. 5. Resultados movimiento vertical ruta 2 prueba 2

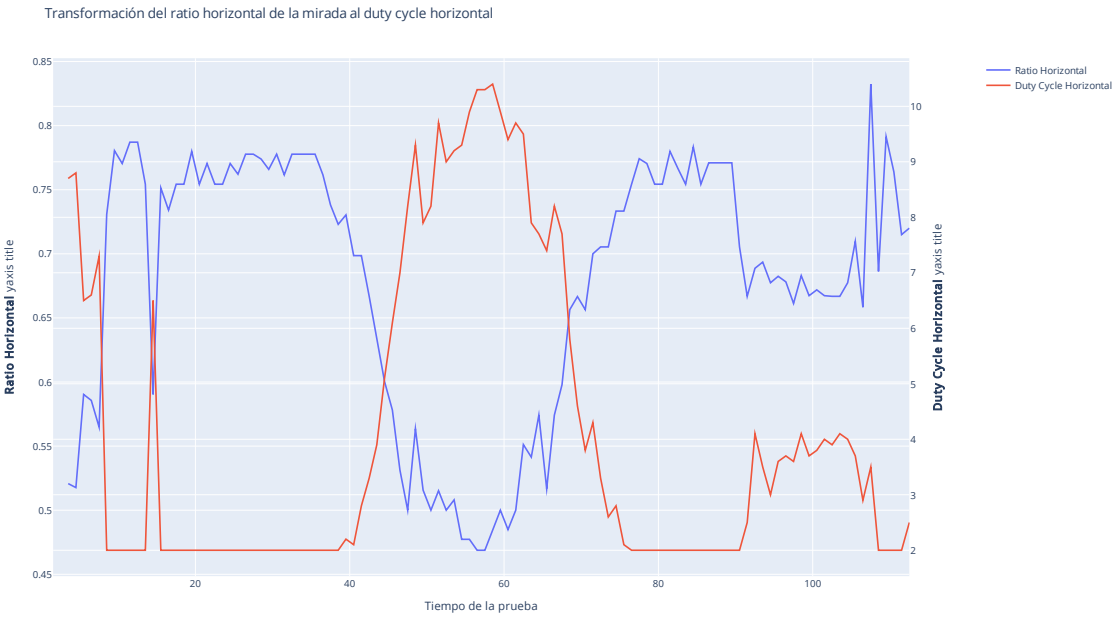

Figura B. 6. Resultados movimiento horizontal ruta 2 prueba 2

Prueba 3

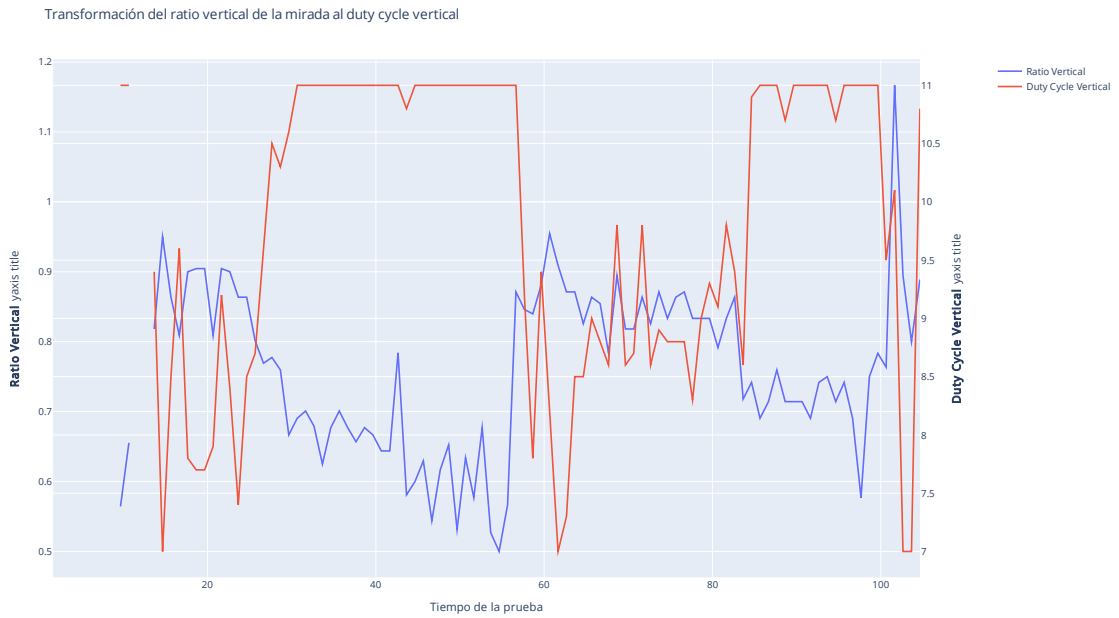

Figura B. 7. Resultados movimiento vertical ruta 2 prueba 3

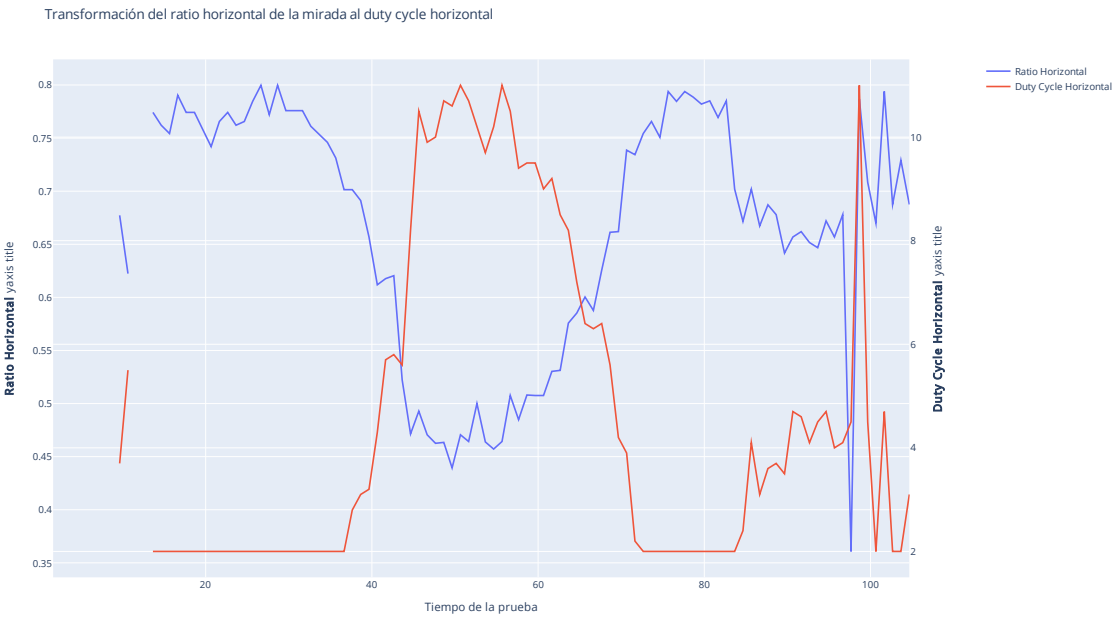

Figura B. 8. Resultados movimiento horizontal ruta 2 prueba 3