INCIDENCIA DE LA SUPER-RESOLUCIÓN USANDO REDES NEURONALES CONVOLUCIONALES EN LA PRECISIÓN Y EXACTITUD DEL SEGUIMIENTO OCULAR BASADO EN PROCESAMIENTO DE VIDEO

Brayan Camilo Alarcón Hoyos

Arlinson Danilo Chimborazo Imbachí

Director:

Msc. Diego Enrique Guzmán Villamarín

Co-director:

Msc. Elena Muñoz España

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones

Departamento de Electrónica, Instrumentación y Control

Ingeniería en Automática Industrial

Popayán, 22 de julio de 2022

INCIDENCIA DE LA SUPER-RESOLUCIÓN USANDO REDES NEURONALES CONVOLUCIONALES EN LA PRECISIÓN Y EXACTITUD DEL SEGUIMIENTO OCULAR BASADO EN PROCESAMIENTO DE VIDEO

Brayan Camilo Alarcón Hoyos
Arlinson Danilo Chimborazo Imbachí

Trabajo de grado presentado a la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca para la obtención del título de:

Ingeniero en Automática Industrial

Popayán, 22 de julio de 2022

Dedicatoria

Esta tesis está dedicada a mis dos madres, Yolanda Alarcón y Marta Hoyos, quienes me han apoyado incondicionalmente a lo largo de toda mi vida y sobre todo en esta importante etapa. A una, por inculcarme desde mi niñez la necesidad de ser un profesional, de luchar contra cualquier adversidad, de no rendirme nunca y por ayudarme a construir una mentalidad positiva e inquebrantable. La otra, quien ha orado innumerables noches por mí, me enseñó de los valores de la vida, me hizo conocer de Dios y a apoyarme en él cuando sintiera que ya no podría seguir más. Mil gracias madres, esta tesis está dedicada a ustedes dos.

Brayan Camilo Alarcón Hoyos

Este trabajo de grado está dedicado a: mi madre, Gilma Cabezas, quien desde niño me educó y veló por cada necesidad, inculcándome siempre los valores como el respeto y la empatía hacia los demás; a mi tía Yudy Galindez, mi otra madre, a quien también le agradezco por siempre estar pendiente y nunca dejarme solo, por apoyarme y enseñarme todo lo que sé hasta el momento.

Arlinson Danilo Chimborazo Imbachí

Agradecimientos

Quiero agradecer enormemente a todas las personas que estuvieron a lo largo de esta grandiosa etapa. A mis maestros, quienes tuvieron la tolerancia y la fuerza para transmitirme sus conocimientos. A la institución por abrirme sus puertas y brindarme sus herramientas para mi crecer intelectual y personal. A los compañeros de clases, mis colegas, quienes estuvieron ahí conmigo para aprender juntos y apoyarnos en los momentos difíciles. A mis amigos que me impulsaron siempre a no rendirme a pesar de las adversidades. A mi compañero de tesis, Arlinson, quien ha sido un pilar fundamental a lo largo de la carrera y de esta etapa final. A los integrantes de mi familia que siempre tuvieron la intención de apoyarme e impulsarme a seguir. A los que^o estuvieron y fueron un apoyo fundamental y que hoy ya no están. Finalmente, deseo expresar mi más grande y sincero agradecimiento al Ingeniero Diego Guzmán, principal colaborador en esta etapa final, quien con su conocimiento, dirección, enseñanza y colaboración me permitió dar este gran paso. A todos infinitas gracias.

Brayan Camilo Alarcón Hoyos

Quiero expresar mi gratitud al Msc. Diego Guzmán, principal colaborador en esta etapa final, quien con su conocimiento, dirección, enseñanza y paciencia nos orientó para concebir este trabajo.

Arlinson Danilo Chimborazo Imbachí

Resumen

En la actualidad se encuentran bastantes seguidores oculares en el mercado, pero su costo es elevado. Dentro de este proyecto se busca establecer si existe un valor positivo o negativo en la precisión y exactitud de un algoritmo de seguimiento ocular al aplicar super-resolución en sus imágenes por medio de redes neuronales. Se presenta una descripción de diferentes métodos de seguimiento ocular y se plantean plataformas adecuadas para el desarrollo del proyecto. Se desarrolla una configuración de plataforma tipo estática para el proceso de seguimiento ocular con una metodología de video-oculografía basada en algoritmo 3D, donde se usa una cámara de tipo webcam para la obtención de datos. Dicha plataforma posee un soporte para la frente y una mentonera para garantizar la estabilidad del paciente. Se realizaron pruebas con cinco participantes y se obtuvieron registros del seguimiento ocular con y sin aplicación de super-resolución, mostrando que dicha aplicación genera diferencias en exactitud y precisión con valores pequeños, cercanos a cero. Se desarrollaron comparaciones estadísticas y se demostró que la incidencia es mínima.

Abstract–Nowadays there are many eye-trackers in the market, but their cost is expensive. This project aims to establish whether there is a positive or negative worth in the precision and accuracy of an eye-tracking algorithm applying super-resolution on the images captured by it using neural networks. A description of different eye-tracking methods is presented and suitable platforms are proposed for the development of the project. A static platform configuration is developed for the eye-tracking process, with a videoculography methodology based on 3D algorithm, where a webcam is used to obtain data. This platform has a forehead support and a chin rest to ensure patient stability. Tests were performed with five participants and eye-tracking records were obtained with and without the application of super-resolution. Statistical comparisons are developed and the incidence is show to be minimal.

Keywords: Eye tracking, Super-resolution, Neural networks, precision and accuracy.

Tabla de contenido

Índice de figuras

Índice de tablas

Capítulo 1

Generalidades

1.1. Introducción

El seguimiento ocular (SO) es el proceso que registra y estudia los movimientos de los ojos [1]. Dentro de la literatura se encuentran diferentes técnicas de seguimiento ocular [2], [3] y actualmente la video-oculografía (VOG) es la técnica mayormente usada. Su metodología se enfoca en la grabación del movimiento de los ojos. Se graba al participante y se fracciona el video en imágenes a las que se les aplica un tratamiento determinado para detectar automáticamente su rostro y ojos. Después de esto se visualiza su pupila y se detecta la dirección de esta. Cabe resaltar que este proceso se puede desarrollar en tiempo real.

La VOG generalmente comprende una, dos o más cámaras digitales, de acuerdo a la estructura y objetivo específico del rastreador en cuestión. Por lo general, la VOG usa LED de infrarrojo cercano (NIR) y una interfaz mostrada por computadora [4] en la cual se distingue el enfoque de la mirada de un observador en un momento dado, para luego adquirir datos de su sistema ocular. Con esta información se pueden generar registros y realizar análisis como entrada para aplicaciones en tiempo real, estudios cognitivos [5], psicológicos y de investigación médica [6], además de aplicaciones en realidad virtual y juegos serios [7], entre otros [8].

Los rastreadores oculares a través de sus algoritmos de seguimiento de la visión permiten medir las rotaciones físicas de los ojos para determinar la dirección de la mirada. Multiplicidad de algoritmos SO basados en video se encuentran en la literatura [9], [10], [11]. Estos son los encargados de realizar las tareas de detección y seguimiento de la visión. Muchos de ellos varían en su estructura hardware, su plataforma software, su calibración, pero principalmente en precisión y exactitud, siendo estos últimos dos puntos los más influyentes para garantizar el éxito de un rastreador ocular.

Es importante mencionar que una buena calibración también permite aumentar la precisión y exactitud de un algoritmo de visión. No obstante, existen algunos sistemas que no hacen uso de esta calibración porque usan otras metodologías. Sin embargo, la constitución de estos algoritmos suele ser a menudo complicada y requiere un tratamiento especial [12], [13].

Las condiciones del ambiente juegan un papel importante dentro del procedimiento de detección de la mirada [14], [15]. La falta o el exceso de iluminación, el ruido interno producido por la cámara, el desenfoque del lente o su falta de nitidez afectan

considerablemente el tratamiento de las imágenes para el procedimiento de detección y seguimiento de la pupila, incidiendo finalmente en la precisión y exactitud del algoritmo en cuestión.

El proceso de tratamiento de imágenes se desarrolla con el propósito de cambiar el impacto visual que la imagen genera en su intérprete. Este proceso permite extraer mayor información al realizarle procedimientos como resaltar sus bordes, cambiar su contraste, autorrellenar con píxeles y demás procedimientos existentes que son bastante útiles en el procesamiento de imágenes [16]. Dentro de la literatura [17] se encuentran aportes del uso de Redes Neuronales Artificiales (ANN, sigla en inglés) [18] al tratamiento de imágenes para el desarrollo en diferentes ámbitos. Las redes neuronales desarrollan técnicas que permiten identificar de forma menos compleja los componentes de una imagen. Sin embargo, en el dominio de SO se han encontrado pocos aportes dentro de la literatura moderna, además de tener poca consecución a través de los años.

Como consecuencia de esto, las imágenes usadas dentro del proceso de SO a menudo suelen tener variedad de inconvenientes que impiden generar un adecuado rastreo de sus componentes y consiguiente a esto una reducción en la efectividad del algoritmo de visión. El proceso de super-resolución se lleva a cabo empleando ANN, la cual se usa en tareas como aumento de resolución con mínima pérdida de información y predicción y relleno de información de imagen. Este proceso permite observar la constitución de una imagen distorsionada condicionando a la red neuronal entrenada previamente para que busque la mejor opción. Posteriormente, realiza un relleno de imagen. Finalmente, genera el relleno de imagen más probable. Este documento busca realizar un análisis sobre la incidencia de esta técnica en la precisión y exactitud aplicada en SO basado en procesamiento de video.

1.2. Planteamiento del problema

El SO es definido como el conjunto de herramientas empleadas para la extracción y análisis de información de los movimientos oculares de un usuario, realizando el monitoreo y registro de la forma en la que una persona mira determinada imagen, el tiempo y la secuencia de su investigación visual [19]. El SO ha desarrollado aportes sobre el comportamiento ocular para el diagnóstico de patologías cognitivas [20], detección de disfunción oculomotora [21], control de dispositivos [22], publicidad [23] y una amplia variedad de disciplinas y áreas de estudio [14].

Dicha tecnología ha demostrado ser útil tanto para el campo de la investigación como también para la ciencia, pues es en estos campos es donde más ha sido aplicado. Desde sus inicios, aproximadamente en el año 1879 con Louis Emile-Javal [24], hasta la actualidad ha tenido una evolución constante, desarrollando metodologías altamente invasivas [25], [26], [24] y poco amigables con el usuario, pero, también, otras en las cuales solo se necesita un video o múltiples fotos de su rostro [4]. Por tal motivo, esta tecnología actualmente es clasificada en el área de la visión por computadora o computer vision [27].

La configuración más básica de un sistema de SO debe hacer uso de dispositivos para la adquisición de imagen (cámaras o videocámaras) y dispositivos utilizados para el procesamiento de los datos (CPU/GPU) [28], [29]. Estos dos en conjunto son los encargados de dar vida al SO. Debido a la variedad de técnicas utilizadas, se encuentra de manera

común el uso de una cámara o varias cámaras, CPU de cuatro o más núcleos y el uso de luz infrarroja [4], [30], [31], [32], [33]. Cabe resaltar que dentro de un sistema de SO se encuentran dispositivos con diferentes características tales como los móviles, los estáticos, los que necesitan procesamiento en vivo, los que el procesamiento es posterior a la grabación del video. Estos dispositivos varían dependiendo de su aplicación [4]. El tiempo de procesamiento dependerá de la CPU si el proceso se realiza en línea o fuera de ella.

Actualmente, el SO se realiza utilizando técnicas basadas en métodos fundamentados en modelos del ojo [34], métodos basados en apariencia [35] y métodos combinados o híbridos [36]. En algunos casos se hace uso de redes neuronales para detección y segmentación [37], [38]. Pero la mayoría de rastreadores oculares comerciales hacen uso de los dos primeros métodos, obteniendo buenos valores de precisión y exactitud. Sin embargo, su elevado precio limita sus aplicaciones al no ser asequible a la mayoría de usuarios [39].

La resolución juega un papel importante en el proceso de SO. [40], [26], [41] comentan que la baja resolución también tiene un efecto negativo en la precisión y exactitud del SO. Además, el costo de los dispositivos para generar un aumento en precisión es alto, lo cual es negativo. Las redes neuronales desarrollan diferentes técnicas para el tratamiento de imágenes [42], [43], [44], [45], [46]. Sin embargo, en la literatura no se encuentra el uso de redes neuronales para mejorar la resolución de imágenes [47], [48] en el campo de SO. [48] muestra cómo la super-resolución busca expandir la imagen para posteriormente rellenar las partes menos distinguibles por medio de redes neuronales hasta obtener una imagen mayormente reconocible.

Dado que no existe mucha información sobre las ventajas de la super-resolución y su uso en SO, este trabajo de grado busca determinar cómo el uso de ANN puede realizar un aporte de calidad en resolución de imagen, favoreciendo la precisión y exactitud en el proceso de SO. De acuerdo a la problemática anteriormente expuesta, surge la siguiente pregunta de investigación: ¿hasta qué punto el uso de Redes Neuronales Convolucionales (CNN, sigla en inglés) usadas en el incremento de resolución de imágenes favorece la precisión y la exactitud del SO basado en procesamiento digital de video? Para el desarrollo del estado del arte se consultaron bases de datos, principalmente, IEEE Xplore, ScienceDirect, la Biblioteca Científica Electrónica en Línea (SciELO), Springer, Association for Computing Machinery (ACM), el motor de búsqueda Google Académico, entre otros. Como metodología de investigación se plantearon tres preguntas con el objetivo de particularizar el tema a tratar. Las preguntas fueron: ¿qué formas existen para medir posición de la pupila?, ¿qué aplicaciones en general tiene medir la posición de la pupila?, ¿cuáles algoritmos o filtros se utilizan para mejorar la precisión de la medida de la posición de la pupila?. Se obtuvo la revisión bibliográfica de 300 artículos, de los cuales se realizó una selección de acuerdo a criterios como año de publicación y estrecha relación con el tema. La selección resultante fue de 160 artículos pertinentes para el desarrollo de la investigación.

El eye-tracking se define como un conjunto de técnicas que buscan detectar y seguir los movimientos del ojo, registrando la forma en la que una persona mira una determinada imagen con el objetivo de determinar la duración y la secuencia de su investigación visual [19]. Dentro de la literatura se encuentran numerosas técnicas, las cuales pueden clasificarse en dos grandes grupos: las metodologías invasivas y las no invasivas [11].

En los últimos años, el uso de ANN ha aumentado en gran porcentaje debido a la integración de un marco de trabajo de aprendizaje que supera al marco tradicional. En este nuevo marco se encuentra el aprendizaje profundo [49]. Este ha demostrado ser el método más eficiente para procesamiento de imágenes, por lo que en el campo de la visión artificial se ha aplicado satisfactoriamente para múltiples propósitos [42], tales como detección de objetos [43], estimación de pose [44], clasificación [45], segmentación [50], seguimiento de objetos [51], eliminación de ruido [52] y super-resolución [53].

Actualmente, la super-resolución [47] busca ampliar el contenido espacial de sus fotogramas para conseguir resoluciones mayores en sus imágenes. Ayudada, también, por la interpolación de fotogramas genera imágenes intermedias entre los cuadros que ya se conocen para, así, incrementar el número de fotogramas por segundo, consiguiendo mayor fluidez en la reproducción de un video.

En el proceso de aumento de resolución se desarrollan varios procedimientos, un método importante es llamado reescalado bilineal, en el que al ampliar una imagen se separan sus píxeles y se rellenan con otros píxeles que tienen su color ya sea por un gradiente y una interpolación matemática se va combinando proporcionalmente estos píxeles con el color de los píxeles originales.

Otra estrategia de aumento de resolución llamada re-escalado por proximidad se puede realizar generando el relleno con otros píxeles del mismo color del píxel original más cercano. En la llamada super-resolución se realiza la percepción donde se predice qué hay en la imagen distorsionada. Así, la red neuronal queda condicionada para que busque la mejor opción, la más probable para esa imagen. La super-resolución genera aquellos pequeños detalles que definen a los objetos percibidos. Sobre estos se fundamentan los métodos de escalado basados en aprendizaje profundo. En los últimos años gracias al desarrollo del campo del aprendizaje profundo, se han posesionado como los métodos mayormente usados para este procedimiento, siendo más usados que los métodos de re-escalado mencionados anteriormente.

Es importante entender que no se trata de que el algoritmo adivine el objeto a partir de un "manchón" en la imagen sino en que la red pueda predecir cuál imagen es. Esto se logra por el arduo entrenamiento que la red neuronal haya recibido previamente (razón por la cual el adecuado entrenamiento es imprescindible) y los patrones que haya aprendido para predecir, aplicando aquellas texturas y patrones aprendidos tras observar muchas otras imágenes durante su entrenamiento. A esto se le llama alucinar.

Así como en super-resolución, en el campo del SO se encuentran variedad de casos en donde se aplica ANN. Algunos de ellos son [54], el cual hace uso de CNN. Para la detección de los movimientos de la cabeza [55], donde hace uso de CNN para el análisis de datos obtenidos por seguidores oculares, [56] realiza detección de estrabismo haciendo uso de CNN y datos obtenidos por seguidores oculares, [57] con DeepVOG, proyecto código abierto de segmentación de la pupila y estimación de la mirada en neurociencia mediante aprendizaje profundo, [58] implementación de ANN para el seguimiento de mirada en imágenes de baja calidad provistas por sistemas de consumo.

De acuerdo a la información anteriormente mostrada, se evidencian los usos de ANN en el campo de SO y las ventajas de la super-resolución, mostrando que esta podría

realizar aportes en precisión y exactitud en el campo de SO. Allí radica la importancia de este proyecto, ya que busca generar un aporte al adentrarse en esta temática, para ello se realizará un análisis comparativo en los resultados obtenidos al aplicar super-resolución a los algoritmos actuales utilizados para SO.

1.3. Objetivos

Los objetivos establecidos para el desarrollo de este proyecto son:

Objetivo general

Evaluar la incidencia de las CNN para el incremento de resolución de imagen en la precisión y la exactitud del SO basado en procesamiento digital de video.

Objetivos específicos

- Desarrollar una plataforma hardware de tipo estática mono-cámara con tecnología infrarroja para el uso del método de modelo geométrico tridimensional en el proceso de SO.
- Implementar un algoritmo de CNN para el incremento de la resolución de imagen.
- Comparar estadísticamente la incidencia con y sin el algoritmo de aprendizaje profundo en la precisión y la exactitud en el proceso de SO.

1.4. Estructura de la monografía

En el capítulo 1, llamado "Generalidades" se realiza una descripción de la temática y la problemática a abordar, se presenta el planteamiento del problema, estado del arte y los objetivos tanto general como específicos para el desarrollo de este trabajo de grado. En el capítulo 2, llamado "Plataforma física para seguimiento ocular", se realiza una descripción de plataformas de SO encontradas en trabajos relacionados a esta investigación, se describe y se presenta el tipo de plataforma a usar, el tipo de cámara y se muestra su estructura física. En el capítulo 3, llamado "Seguimiento ocular basado en modelo geométrico 3D", se presentan las técnicas de SO basadas en video-oculografía y se muestra el funcionamiento del algoritmo 3D implementado con las descripciones de los parámetros y partes que lo componen. En el capítulo 4, llamado "Super-resolución", se realiza una descripción de ANN y CNN, definiendo la super-resolución. Se muestra el método de super-resolución y se muestra una implementación sencilla de su aplicación en una imagen con código de ejemplo. En el capítulo 5, llamado "Pruebas, comparaciones y resultados", se presenta el desarrollo del proceso de seguimiento a realizar, interfaz, pruebas y resultados obtenidos, junto con las estadísticas de los resultados obtenidos al aplicar super-resolución a una imagen, para luego ser usada en SO. Finalmente, en el capítulo 6, se presenta la discusión y las conclusiones asociadas al presente trabajo.

Capítulo 2

Plataforma física para el Seguimiento Ocular

En este capítulo se realiza una descripción sobre las configuraciones de plataforma implementadas en trabajos relacionados con SO, donde se tiene en cuenta la cantidad y el tipo de cámara que mejor se adapta al proceso de seguimiento y cuáles son las ventajas y desventajas que posee cada configuración. Se elige una configuración y se muestran las partes que la componen.

Para elegir la configuración de plataforma se tomó como base el artículo [4], el cual brinda una amplia comparación de plataformas y sistemas de SO con mono-cámaras y multi-cámaras, de acuerdo a la necesidad requerida.
Desde el punto de vista de los métodos invasivos encontramos la electro-oculografía (EOG), en la cual se ubican pares de electrodos alrededor de los ojos [59], la bobina de búsqueda escleral (SSC), en la cual se lleva a cabo la incrustación de bobinas en una lente de contacto, para posteriormente ser adheridas a los ojos [59]. Y, también, la oculografía infrarroja (IROC), en la cual se hace uso de diodos emisores de luz infrarroja, dichos diodos se ubican en el marco de un par de lentes especialmente modificados, cuyo objetivo es que la luz sea reflejada en el ojo, para posteriormente obtener información y con ayuda de esta lograr inferir las posiciones de los ojos [60].

La principal ventaja de estos métodos es el área de aplicación. Mientras la EOG tiene aplicaciones clínicas, la SSC se aplica a la investigación médica y psicológica y la IROC se destina a investigación de sistemas para la medición de sacadas (movimientos oculares rápidos) [61]. De forma general se reporta una exactitud con valores medios a altos y sistemas robustos ante condiciones de iluminación variable.

En la literatura también se reporta que las técnicas invasivas presentan desventajas. Las más importantes son: complejidad en la estructura física, la cual tiene un valor de intermedio a alto, pues se requieren equipos especializados; Otra desventaja es que no pueden ser utilizadas sobre implementaciones en tiempo real como en el caso de las metodologías SSC y IROC. Finalmente, la detección de los movimientos en un solo eje, en el caso de la EOG y la SSC en donde estos solo se detectan en el eje vertical [4].

El uso de estas metodologías es habitual en el campo de la investigación médica, pues tienen una alta exactitud, lo que conlleva a obtener buenos resultados en cuanto a la estimación de las posiciones de los ojos [62]. En la literatura se reportan algunos dispositivos comerciales para estas metodologías como: BIOPAC MP 45/100 en el caso

de EOG, Chronos vision para la SSC y IntelliGaze IG-30 para la IROC [27]. Estos dispositivos son herramientas fundamentales dentro de grandes compañías y habituales en el proceso de rastreo y análisis de información.

En la clasificación de las metodologías no invasivas encontramos la video-oculografía [63], [64]. En esta se emplea una o varias cámaras para obtener la posición de la mirada, utilizando capturas, ya sea en formato de video o fotos. Posteriormente, realiza un análisis y extracción de características de interés tales como movimientos, parpadeos, entre otros. Esta metodología se utiliza en campos como marketing y publicidad [65], en el campo médico en personas con discapacidad [66], simulaciones [67], videojuegos [68], aprendizaje en el sector educacional [69] y dispositivos de uso doméstico [70].

La literatura reporta diferentes ventajas tales como la interacción humano-máquina y aplicaciones en el campo médico. También reporta que entre los parámetros más importantes en la video-oculografía se encuentran la exactitud y precisión según [71], [72]. La exactitud se define como la distancia euclidiana promedio (en píxeles) entre las estimaciones de mirada en pantalla y los objetivos de mirada reales. Por otro lado, la precisión se define como la raíz cuadrada media de las distancias euclidianas (en píxeles) entre las estimaciones anteriores y posteriores de la mirada en pantalla. Los dispositivos no invasivos con video-oculografía han reportado una exactitud alta y una baja complejidad en su estructura física y la posibilidad de una implementación en tiempo real.

Algunos dispositivos comerciales reportados son: ERICA, Tobii X3-120, Tobii EyeX, entre otros [27]. Algunas desventajas de esta metodología son la limitación debida a los movimientos de la cabeza y la vulnerabilidad ante las variaciones de las condiciones de iluminación [40].

En los últimos años, esta metodología se ha convertido en la más utilizada, ya que es la única considerada no invasiva y permite una fácil coordinación del diseño de las pruebas y la provisión de estímulos que permiten analizar automáticamente los datos [73]. La detección y SO en tiempo real de VOG pertenecen al área activa de investigación en la comunidad de visión por computadora desde hace mucho tiempo. Aunque en la actualidad no se conocen grandes organizaciones dedicadas al SO, pues es una metodología en crecimiento, si existen numerosos dispositivos de seguimiento de mirada y software en el mercado. Cada vez hay más aplicaciones en las que estas técnicas hacen una contribución importante [1]. Por estas razones se hará un enfoque más profundo dentro de esta metodología.

Debido a la gran cantidad de aplicaciones y variaciones en la plataforma física, en la video-oculografía encontramos variedad de técnicas y métodos que se clasifican en tres grandes grupos: métodos basados en modelos del ojo [34], métodos basados en apariencia [35] y métodos combinados o híbridos [36].

Las técnicas o métodos basados en el modelo del ojo son algoritmos que parten de un modelo tridimensional del ojo. En dicho modelo se aplica geometría y otras operaciones con el fin de predecir y determinar la dirección de los movimientos oculares. Esto se logra calculando el promedio de la diferencia o desviación entre el eje óptico y el eje visual para, posteriormente, estimar los puntos de mirada [13].

Entre sus principales inconvenientes se encuentra la baja resolución de imagen y como consecuencia de esto la reducción de la exactitud al realizar estimaciones erróneas. En la literatura se reportan varias configuraciones de plataformas físicas. Las más comunes son una cámara y un sistema geométrico simple como el mostrado en [74], [75]. Las configuraciones basadas en múltiples cámaras y múltiples luces de infrarrojo cercano (NIR) son presentadas como una solución porque con estas configuraciones consiguen mejorar la exactitud y agregan la capacidad de tolerar los movimientos de la cabeza del usuario, dando como resultado, en la mayoría de ellos, la posibilidad de un movimiento libre de la cabeza. Sin embargo, estas configuraciones representan un alto costo para su implementación porque son poco atractivas [4].

Los métodos basados en apariencia son algoritmos que parten del análisis de las imágenes o fotogramas de video, con el fin de encontrar ya sea destellos o el iris. Dichos algoritmos se clasifican en dos diferentes ramas: los algoritmos de visión y los basados en Redes Neuronales Artificiales (ANN, sigla en inglés). Los algoritmos de visión se encargan de detectar el iris y la pupila con ayuda de filtros. Esto lo hacen con base en la apariencia fotométrica del ojo con el objetivo de generar un mapeo de la mirada en las coordenadas de la pantalla.

En la literatura se encuentran muchos aportes en este campo. Actualmente, algunos métodos con mayor evolución utilizan un método basado en Camshift [39]. Otro método utilizado frecuentemente es el mostrado en [66], donde se utiliza la transformada de Hough con el objetivo de encontrar patrones circulares. También, el algoritmo Viola Jounes [76] utilizado para la detección de los ojos.

Dentro de la literatura se encuentran aportes del uso de ANN al tratamiento de imágenes para su desarrollo en diferentes ámbitos. Uno de ellos es el desarrollo de filtros que permiten identificar de forma menos compleja los componentes de la misma. Estos algoritmos obtienen una clasificación por el tipo de tarea realizada, tareas como segmentación y detección [77], [58], [78], [14], [79], [80], además de métodos de regresión para la parte de calibración [73], entre otras.

Cabe destacar que los algoritmos basados en apariencia poseen algunas deficiencias notables como baja resolución de imagen, efectos de ruido, variación de iluminación sensible a distancia del usuario y movimientos. Esta situación implica regiones de ojo indistinguibles y reducción de la exactitud al no encontrar el objetivo claramente. Además, estas técnicas son muy vulnerables a los movimientos de la cabeza y requieren que los usuarios mantengan la cabeza muy quieta usando un reposacabezas, montonera o barra de mordida.

Los métodos combinados o híbridos son una combinación de los métodos 3D y de apariencia en los cuales se aprovechan los beneficios de ambos métodos para la estimación de la mirada [36]. Las deficiencias de estos métodos son la baja resolución de imagen, efectos de ruido y variación de iluminación. Esto ocasiona una reducción de la exactitud al realizar estimaciones erróneas.

2.1. Selección de hardware

Para realizar la adquisición y procesamiento de los datos del usuario, fue necesario realizar una selección de hardware y software. El sistema hardware que se escogió para

esta investigación esta compuesto por una plataforma tipo estática mono-cámara para procesar un método de modelo geométrico tridimensional implementado con base en la literatura. Para la estructura física se debió escoger qué tipo de plataforma se iba a implementar, los tipos y características de la cámara adecuada para realizar la grabación del usuario, la pantalla donde se debe realizar el procedimiento y un equipo de cómputo con las condiciones necesarias para soportar el aprendizaje profundo para realizar todo el procesamiento con ANN .

2.1.1. Tipos de plataformas

Las plataformas de SO son las configuraciones físicas que permiten adquirir la información necesaria del usuario para realizar el proceso de detección y rastreo. Los sistemas de seguimiento de la mirada basados en video comprenden comúnmente una o más cámaras digitales, LED de infrarrojo cercano (NIR) y una computadora con pantalla que muestra una interfaz gráfica de usuario donde se rastrea la mirada del usuario [4]. La interfaz gráfica de usuario para el seguimiento de la mirada puede ser activa o pasiva, simple o multimodal [81], [82]. La interfaz de usuario activa permite rastrear la mirada del usuario para activar/desactivar una función y la información de la mirada se toma como una función de entrada.

Al contrario de una interfaz de usuario activa, una interfaz pasiva no posee comando, esta solo recopila datos de la mirada para realizar análisis de atención del usuario. Las interfaces gráficas de seguimiento de mirada también se dividen en mono-modo y multimodal.En las mono-modo se usa la mirada como la única variable de entrada, mientras que las multi-modal combinan la entrada de mirada junto con las entradas de ratón, teclado, toque o parpadeo para el comando. Hay dos clases principales de sistemas de estimación de la mirada: sistemas montados en la cabeza y sistemas remotos (sistema estático) [83].

Plataformas montadas en la cabeza: La configuración general incluye dos cámaras, una señalando el ojo del usuario para detectar la pupila y la otra que captura el punto de vista del usuario como se ilustra en la Figura 2.1. Suelen usarse componentes adicionales como fuentes de luz NIR y espejos calientes. Los rastreadores de mirada montados en la cabeza se han implementado como dispositivos libres de accesorios móviles, de bajo costo y livianos con hardware y software simples. También se sabe que proporcionan información de mirada de alta precisión en entornos sin restricciones. La estimación de la mirada en 3D con rastreadores montados en la cabeza se ha informado en varios documentos tales como [84], [85], [86].

Plataformas remotas: La plataforma más sencilla es la estática, pues en esta se hace uso de una estructura física estática, con un soporte para la cabeza. Estas se componen comúnmente por una pantalla, una o varias cámaras y un sistema de luces (ver Figura 2.2). En algunos casos poseen mentonera y un soporte para la frente, lo cual permite al usuario mantenerse a una distancia adecuada en el momento de realizar el proceso de detección y seguimiento, aunque existen algunas plataformas que soportan variaciones de movimiento de cabeza. Hacer que la cabeza del usuario se mantenga a una distancia controlada con respecto a la cámara, mejora la forma en que el algoritmo realiza su procesamiento. Algunos ejemplos de aplicaciones basadas en plataformas estáticas se ven en [87], [88], [89].

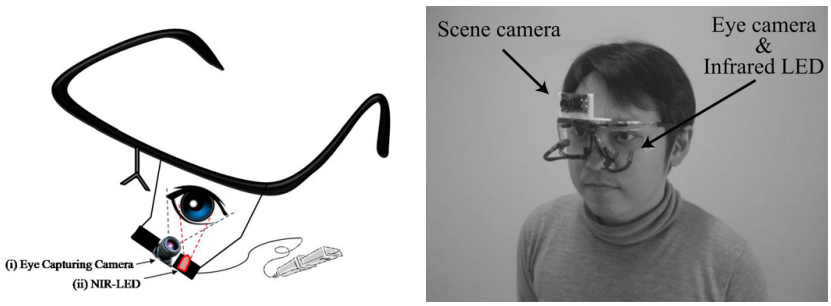


Figura 2.1: SO móvil (Tomado de [85] [86]).

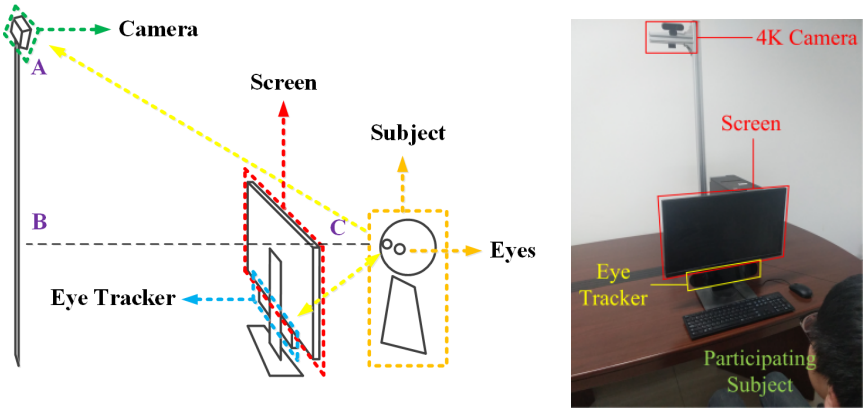


Figura 2.2: SO remoto (Tomado de [35]).

2.1.2. Tipos de cámaras de SO

Los sistemas de SO implementan 2 tipos de configuración de cámara para su desarrollo. Existe la configuración mono-cámara [90], [91], donde el sistema hace uso de una única cámara para obtener la información del usuario. Y la configuración multi-cámara, donde se utilizan 2 o más cámaras para obtener imágenes desde varias posiciones [92].

Un sistema multi-cámara posee un grado mayor de complejidad que un sistema mono-cámara, pero permite mayor versatilidad en cuanto a obtención de información [93], [94]. Yihua Cheng [95] sustenta que la mayoría de seguidores oculares usan una única cámara RGB como principal herramienta de seguimiento. Aunque existen otros estudios donde se hace uso de una cámara tipo IR (Infrarrojo) como en [96] para trabajar en ambientes con condiciones de baja iluminación y otros estudios que usan cámaras RGBD para obtener información sobre profundidad de la imagen [97]. En la Figura 2.3 se muestran los 3 tipos de cámaras, junto con su imagen capturada.

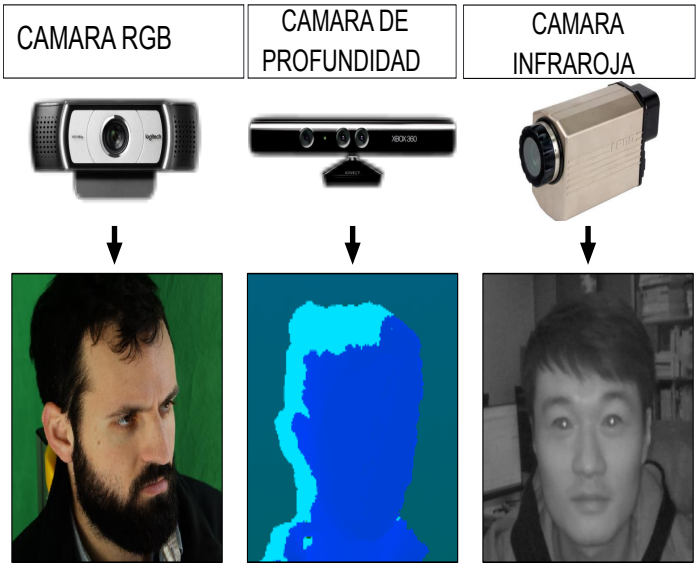


Figura 2.3: Tipos de cámaras e imágenes capturadas (Tomado de [95]).

2.1.3. Plataformas

En [4] se muestra la configuración física de variedad de algoritmos de acuerdo a su método de SO implementado. En la Tabla 2.1 se muestran configuraciones de plataformas que constan de variedad en cuanto a la cantidad de cámaras, luces infrarrojas y movimiento de la cabeza. Se pueden ver cambios significativos en exactitud y diferencias en cuanto a la capacidad de soportar pequeños o grandes movimientos de cabeza [98].

Metodologías como SO basado en regresión 2D como en [99], en el cual la configuración física está compuesta por cuatro LED y una cámara web, permitiéndole alcanzar una exactitud de 1.3 grados, además de soportar que el usuario realice marcados movimientos de cabeza. También se muestra la configuración física de métodos de SO basado en apariencia [100] y finalmente las configuraciones de métodos basadas en modelos tridimensionales, modelos de interés de esta investigación.

Artículo	Configuración	Exactitud(°)	Condiciones de funcionamiento
[101]	Una cámara pan tilt- stereo con matrices de LED	1°	Movimiento libre de cabeza
[90]	Una cámara y dos LED	1 - 3°	Movimiento de cabeza totalmente compensado
[91]	Una cámara y dos LED	1 - 3°	Movimiento de cabeza totalmente compensado
[10]	Una cámara sin LED	1 - 2°	Sin variación en posición de la cabeza
[93]	4 cámaras, 2 LED y espejos	0.6°	Movimiento de cabeza compensado
[102]	2 cámaras y 2 LED	$<$ 1 - 2°	Movimiento de cabeza compensado
[103]	2 cámaras y 2 LED	$<$ 1 - 2°	Movimiento de cabeza compensado
[104]	Modelo de rostro 3D, 2 cámaras, sin LED	1°	Movimiento de cabeza compensado
[105]	1 cámara, 1 LED y espejos	3°	Movimiento de cabeza compensado
[106]	2 anillos de luz, 2 cámaras	1.25°	Pequeña tolerancia a movimientos de cabeza
[107]	1 sensor de profundidad con LED integrados	3.78°	Movimiento de cabeza, requiere distancia del usuario
[108]	1 sensor de profundidad con LED integrados	5°	Movimiento de cabeza

Tabla 2.1: Configuraciones de plataforma (Tomado de [4]).

De acuerdo a las configuraciones de plataforma mostradas anteriormente, se entiende que se puede implementar un sistema de SO para el uso del método de modelo geométrico tridimensional, realizando una configuración de plataforma de tipo mono-cámara, competente con respecto a exactitud, además de reducir su costo. Teniendo en cuenta lo expuesto por [109] donde se resalta que la mayoría de seguidores oculares suelen usarse en gran parte para proyectos de investigación y no son asequibles para la mayoría por su costo elevado.

Los sistemas mono-cámara [90], [74], [105] en su mayoría están compuestos por una cámara y una pantalla y algunas veces se hace uso de luces infrarrojas y espejos [91]. Dentro de los sistemas mono-cámara descritos anteriormente, se encuentran diferencias notables en cuanto a factores como exactitud, costo, complejidad en desarrollo y/o procesamiento y la tolerancia de la configuración para permitir algunos movimientos de cabeza intencionalmente generados por el usuario. Es necesario resaltar que algunos algoritmos basados en 3D son tan robustos que toleran los movimientos de cabeza, aun si el sistema está diseñado estático y no móvil.

2.1.4. Selección de plataforma física

A continuación se muestra la Tabla 2.2, donde se evalúan tres artículos con modelos de algoritmo 3D diferentes. Su variación se da principalmente por la configuración de plataforma. Los parámetros evaluados son la exactitud, el costo y complejidad de

implementación. Además, se tiene en cuenta si el algoritmo permite o no realizar los movimientos de cabeza.

Artículo	**Exactitud(°)**	**Costo($)**	**Compensación**	**Complejidad de implementación**
A Single-Camera Remote Eye Tracker	1 a 3	(i)Una cámara de alta resolución calibrada única (1280x1024 píxeles) (ii) dos diodos emisores de luz infrarroja (LED) a cada lado de la cámara que iluminan la cara y generan reflejos corneales (CR) en la superficie de la córnea (iii) una pantalla ubicada sobre la cámara y los LED	Permite movimiento libre de cabeza	Usa algoritmo Starburst mejorado La plataforma física es muy sencilla, no explica implementación, se muestra en un taller y usa código abierto de Open eyes
Detecting Eye Position and Gaze from a Single Camera and 2 Light Sources	1 a 3	(i) Una cámara de alta resolución que procese los rayos infrarrojos (ii) Una pantalla (iii) 2 luces infrarrojas (iv) 1 espejo convexo	No permite el movimiento libre de cabeza	La teoría óptica de superficies esféricas para rayos paraxiales se combinan con el modelo del ojo de Gullstrand NO REQUIERE CALIBRACIÓN FRECUENTE. Muestra ecuaciones y proceso.
Eyetracking: Pupil orientation geometrical modeling	1 a 2	(i) Cámara Hamamatsu C5999 (ii)Una pantalla. (iii) Sistema óptico del ojo	No permite el movimiento libre de cabeza	Proceso un tanto más complejo que los demás porque no usa tecnología IR.

Tabla 2.2: Principales configuraciones de plataforma (Tomado de [4]).

En la tabla anterior se tuvo como parámetro la estimación del costo, aunque esta puede no ser exacta con respecto a los precios de la época y la actualidad, pero si se logra dar la información necesaria con respecto a los elementos de la plataforma, con las características que se deben tener en cuenta para tolerar el procedimiento. En la tabla se pueden observar las características que se tuvieron en cuenta de cada tipo de plataforma para visualizar cada parámetro necesario y elegir el mejor sistema de plataforma a usar en esta investigación.

De acuerdo a la tabla anterior, se construye la Tabla 2.3, creándose una matriz de decisión, donde cada parámetro se evalúa de 1 a 5 de forma ascendente. Para elegir la mejor configuración a seguir en el proyecto, se eligió la configuración con el valor más alto en la tabla:

¿Cuál es la configuración adecuada de plataforma para el desarrollo mono-cámara con algoritmo 3D?					
Parámetros					
Artículo	**Exactitud(°)**	**Costo**	**Compensación de movimientos de cabeza**	**Complejidad de implementación**	**Total**
[90]	2	4	5	3	14
[91]	2	3	5	3	13
[10]	3	4	1	2	10
-	Exactitud alta: 5 Exactitud baja: 1	Costo alto: 1 Costo bajo: 5	Si compensación: 5 No compensación: 1	Complejidad alta: 1 Complejidad baja: 5	/

Tabla 2.3: Matriz de decisión para configuración de plataforma (Fuente propia).

De acuerdo a las cámaras usadas y plataforma, [90] usa la plataforma que mejor se adecuó para un sistema mono-cámara para llevar a cabo la implementación del algoritmo 3D de SO. Adicional a esto, el sistema de detección de ojos OSCANN [110] usa un estabilizador de cabeza con un soporte para frente y mentón. Para el desarrollo de esta investigación se utilizó el proyecto de fuente abierta RestEasy [111], el cual es similar a OSCANN. El modelo tridimensional de dicho estabilizador se muestra en la Figura 2.4.

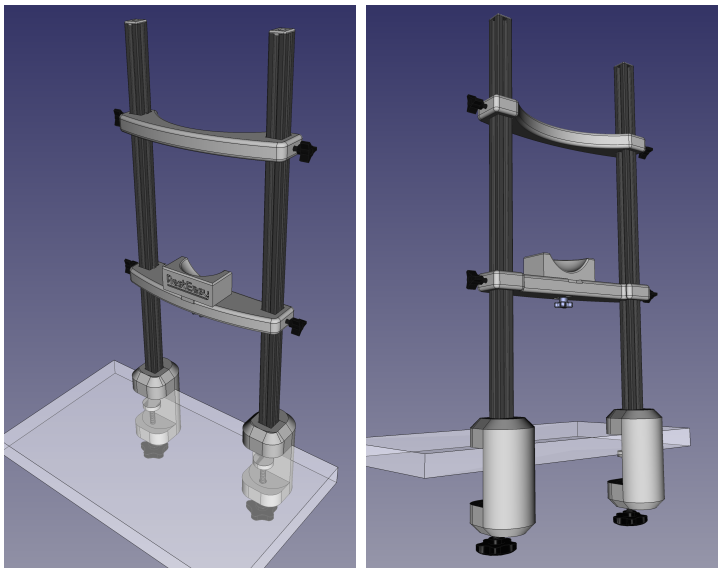


Figura 2.4: Modelo de estabilizador 3D (Tomado de RestEasy [111]).

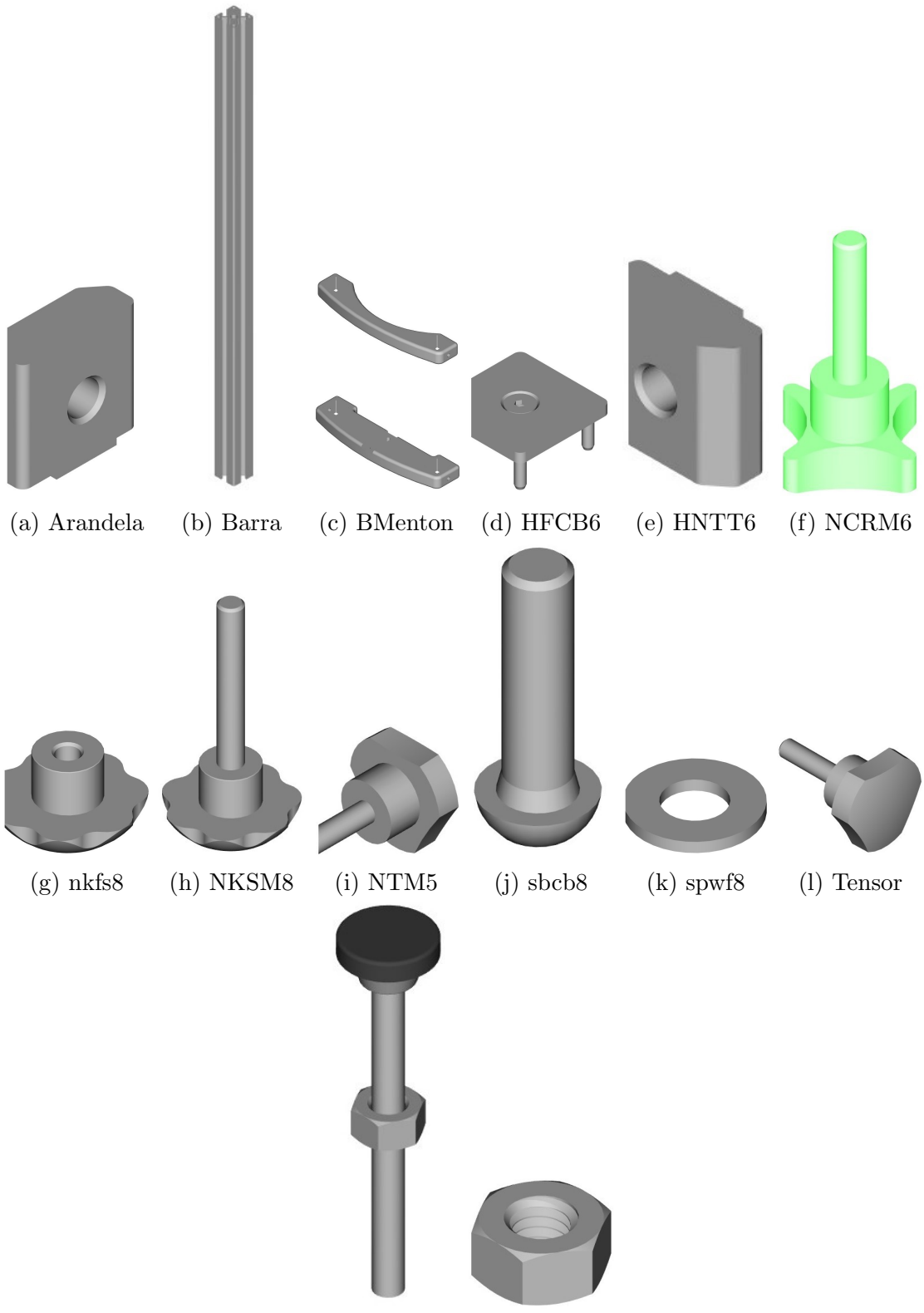
Se decidió montar un estabilizador basado en el modelo tridimensional de RestEasy y de acuerdo a eso se tuvieron en cuenta los materiales mostrados en la Tabla 2.4:

Descripción	Costo unidad($)	Cantidad	Precio($)
Perfil de aluminio de 30x30 mm x 600 mm, orificios roscados M8 de 15 mm	40.000	2	80.000
Tapas de plástico negras retenidas por pernos	10.000	2	20.000
M8, tornillos hexagonales de 30 mm	2.500	2	5.000
Arandelas M8	1.000	2	2.000
Tuercas con ranura en T, M5	2.000	4	8.000
Pomo de plástico de 3 lóbulos con rosca M5 de 20 mm de largo	30.000	4	120.000
Pomo de plástico de 4 brazos con rosca M6, 30 mm de largo	30.000	1	30.000
Perilla de plástico de 7 brazos con rosca M8 Tuerca	10.000	2	20.000
Tornillo hexagonal M8	2.000	2	4.000
Tuerca hexagonal M6	2.000	1	2.000
Varilla roscada M8, 100 mm	10.000	2	20.000
Patas niveladoras de goma M8	15.000	2	30.000
TOTAL ($)			341.000

Tabla 2.4: Materiales (Fuente propia).

El precio de dichos materiales corresponde al costo en pesos, por el cual fueron obtenidos en la zona donde se realizó dicha implementación.

A continuación se muestran los componentes del estabilizador desarrollado para esta investigación. Las piezas se tomaron igualmente del repositorio de RestEasy [112]:



(a) Arandela (b) Barra (c) BMenton (d) HFCB6 (e) HNTT6 (f) NCRM6

(g) nkfs8 (h) NKSM8 (i) NTM5 (j) sbcb8 (k) spwf8 (l) Tensor

(m) Tuercalar (n) Tuercam6

Figura 2.5: Piezas de estabilizador de RestEasy (Tomado de [112]).

Las anteriores piezas dan lugar al estabilizador mostrado en la Figura 2.6:

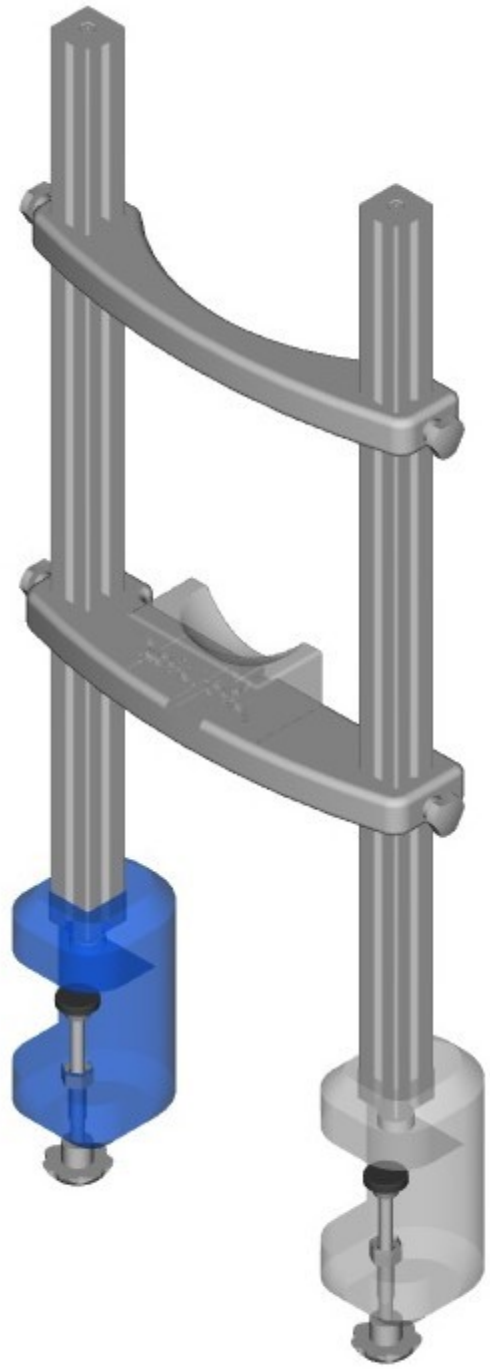


Figura 2.6: Modelo 3D del estabilizador del proyecto.

Las piezas se descargaron y se hizo uso de la herramienta FreeCAD para visualizar los modelos 3D, se cambió su formato a uno de tipo “obj” para realizar su impresión 3D. A continuación se muestra la Figura 2.7 donde se enseña el estabilizador del proyecto montado en físico:

(a) Parte frontal (b) Parte trasera

Figura 2.7: Estabilizador del proyecto.

El estabilizador implementado se ubica frente a la cámara de seguimiento y permite graduar la altura con respecto a la cámara. Posee un soporte para la frente y otro para el mentón ajustando la posición de la cabeza y permitiendo que esta mantenga una distancia fija para que la detección de su rostro y ojos sea un proceso menos complejo. Respecto a la plataforma completa implementada para este proyecto, se decide montar un sistema de SO con una configuración mono-cámara, con una cámara web, más un estabilizador con soporte para cabeza, con uso de una pantalla ubicada debajo de la cámara y sin el uso de luces infrarrojas. Las condiciones del algoritmo 3D del seguidor ocular implementado que se mostrara más adelante, permiten que el uso de IR no sea necesario.

Capítulo 3

Seguimiento Ocular basado en modelo geométrico 3D

Para entender algunos de los principales métodos de SO y estimación de la mirada, parte de este capítulo se basa en [4], donde se establece que los algoritmos de estimación de la mirada emplean principalmente cinco métodos diferentes. Estos usan modelos y técnicas tanto físicas como matemáticas para desarrollar el proceso. Técnicas tales como la reflexión corneal, para estimar la mirada, la cual se desarrolla con ayuda de luces infrarrojas NIR, en muchos casos. El uso ya sea de funciones polinomiales o de métodos donde se implementan modelos geométricos del ojo, como es el caso de los métodos basados en relación cruzada, regresión 2D y modelo 3D.

En el caso de los métodos basados en apariencia y forma, se utiliza la luz visible e información acerca del ojo, tomándose características como forma, textura y otros. En las siguientes secciones se describen estos métodos y en adelante se profundiza en el método del modelo geométrico 3D, usado para esta investigación.

3.1. Métodos de algoritmos de estimación de la mirada

En la Figura 3.1 se observa el cambio de dispositivos de SO a través de los años. En sus inicios, como se mencionó en capítulos anteriores, se hacía uso de metodologías intrusivas como la búsqueda escleral y electro-oculografía, con uso de dispositivos tales como bobinas y electrodos. Luego, estos dispositivos se cambiaron y se implementó el uso de la video-oculografía, donde se pasó a usar sistemas de SO basado en video y se utilizaron dispositivos no intrusivos, como el uso de luces infrarrojas, apoyadas en reflejos corneales del ojo, los cuales se graban para ser rastreados y medidos. Y, finalmente, se desarrollan sistemas de seguimiento que hacen uso únicamente de cámaras web del ordenador, dichos sistemas se apoyan en modelos y técnicas implementadas por algoritmos desarrollados para SO.

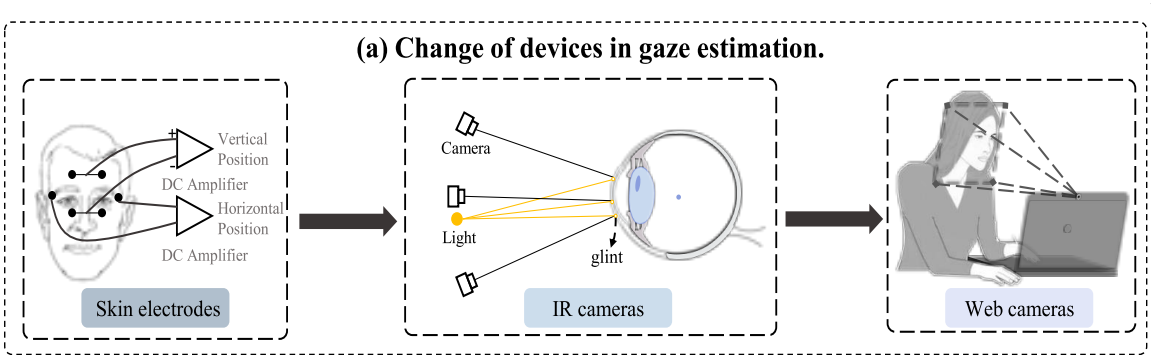


Figura 3.1: Dispositivos usados en SO a través de los años (Tomado de [95]).

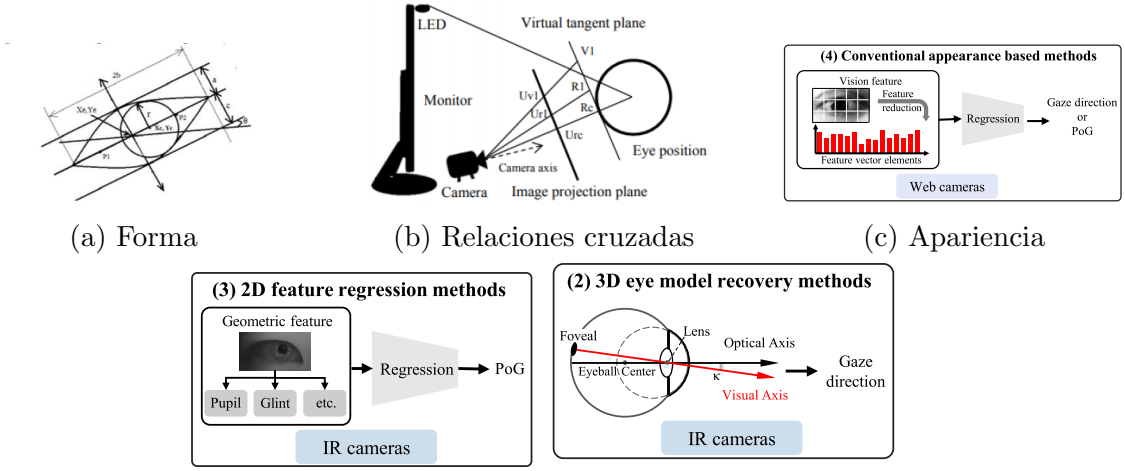
Los algoritmos de SO y/o estimación de la mirada, implementan algunos métodos y técnicas claves para realizar dicho proceso. En la Tabla 3.1, [4] proporciona la información sobre los principales métodos, usados por algoritmos de SO, mostrando algunas de sus características, ventajas y desventajas.

Método basado en	Características	Ventajas	Desventajas
Forma	Calculan la dirección de la mirada a través de la observación de los ojos [113](ver Figura 3.2a). Desarrolla plantillas de la región del ojo, usando dos parábolas para el contorno de los ojos y un círculo para el iris, y luego se comparan para hallar la similitud entre ambos. Usa geometría y ecuaciones matemáticas para su desarrollo [4]. Trabajos relacionados se ven en [114], [115].	Permite hallar la forma del ojo y obtener la posición, el tamaño de la pupila y otra información [114].	Tienen limitaciones en el seguimiento de la mirada debido a que el borde del iris es difícil de captar con luz IR/-NIR, y la apariencia del iris está sujeta a reflejos corneales bajo condiciones de luz visible variable [113].
Relaciones cruzadas	Se implementa una configuración de cámara, pantalla y luces infrarrojas (ver Figura3.2b). Se usan 4 fuentes de luz (IR) desde las 4 esquinas de una pantalla hacia el rostro del participante y se estima el cruce de la mirada en el plano por medio de la proyección y reflexión de las luces en la córnea. Se usa la propiedad invariante de la geometría para estimar la mirada [4]. Trabajos relacionados se ven en [116], [117].	No requiere conocimiento geométrico de las partes del sistema ocular. Permite hacer SO con movimiento de cabeza y consta de dispositivos sencillos para su funcionamiento [117].	"Pero se ven afectados por problemas como un mayor error con la distancia del usuario y factores dependientes del usuario" Tomado de [4].

Apariencia	Se utilizan las propiedades de forma y textura de los ojos, junto con la posición de las pupilas en relación con las esquinas de los ojos para estimar la mirada [4](ver Figura 3.2c). Por lo general hacen uso de cámaras web comunes para extraer las características de los ojos y hacen uso principalmente de información sobre su apariencia. Utilizan características como píxel de imagen o características profundas para realizar proceso de regresión de mirada; utilizando varios modelos de regresión como la red neuronal, modelo de regresión del proceso gaussiano o el modelo de regresión lineal adaptativa y la red neuronal convolucional [95]. Trabajos relacionados se ven en [118], [119].	Son más robustos y pueden operar bien con imágenes de baja calidad, ya que no se basan en características locales, sino que utilizan información de toda la región del ojo. No requieren ninguna calibración de la cámara [120].	Requieren una mayor cantidad de datos de entrenamiento específicos del usuario que los métodos basados en modelos [121]. Sus configuraciones no permiten extraer con eficacia las características de mirada de alto nivel de las imágenes. A menudo tienen una caída en el rendimiento cuando se encuentran con el movimiento de la cabeza [95]. Su precisión se degrada con los movimientos de cabeza, variación en los niveles de iluminación y para un rendimiento robusto, necesitan grandes bases de datos de imágenes de entrenamiento [4].
Regresión 2D	Utilizan características del ojo humano como su geometría, los contornos de sus pupilas y los reflejos corneales (ver Figura 3.2d). Se genera un vector entre el centro de la pupila y el brillo corneal y se asigna a las coordenadas correspondientes en pantalla frontal por medio de una función de transformación polinómica [4]. Trabajos relacionados se ven en [122], [123].	Permiten una implementación desde una sola cámara y con pocos LED [4].	Este tipo de técnicas poseen alta vulnerabilidad a los movimientos de la cabeza. Requieren que el usuario use un tercer objeto para mantener su cabeza quieta [4].

Modelos 3D	Utilizan un modelo geométrico del ojo humano (ver Figura 3.2e) para estimar el centro de la córnea, ejes ópticos y visuales de los ojos, junto con las coordenadas del globo ocular y realizan una estimación de la mirada, enfocándose en la intersección de los ejes visuales con la escena [4]. Trabajos relacionados se ven en [106], [74].	Poseen un alto nivel de precisión y la mayoría permite realizar pequeños movimientos de cabeza y algunos, inclusive, permiten el movimiento libre de cabeza [4].	Los requisitos hardware para implementar su montaje son altos [4].

Tabla 3.1: Principales metodologías implementadas por algoritmos de SO (Basado en [4]).



(a) Forma (b) Relaciones cruzadas (c) Apariencia

(d) Regresión 2D (e) Modelo 3D

Figura 3.2: Métodos de SO (Tomado de [4], [95]).

El método de apariencia ha tenido avances significativos en los últimos años. Además de la información proporcionada en la Tabla 3.1, este implementó el uso de aprendizaje profundo, mejorando considerablemente el proceso de seguimiento [95]. De acuerdo a la tabla anterior, el método de SO basado en modelo geométrico 3D, es el más robusto y posee mayor precisión que otros métodos, además de que permite utilizar una configuración tanto mono-cámara como multi-cámara y posee tolerancia ante movimientos de cabeza. En la siguiente sección se da una descripción sobre sus características.

3.2. SO basado en modelo 3D

En el desarrollo de este tipo de métodos se emplea un modelo geométrico del ojo, este permite estimar el centro de la córnea, su eje óptico y, adicionalmente, estimar las coordenadas de la mirada como puntos de intersección, donde el eje visual se encuentra con la escena (ver Figura 3.3).

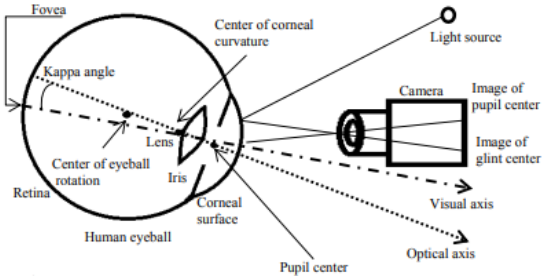


Figura 3.3: Modelo 3D del ojo humano, parámetros del ojo y elementos de configuración, utilizados generalmente en el seguimiento de la mirada del ojo 3D [74], [124]. El eje óptico se muestra como la línea que une el centro de curvatura de la córnea con el centro de la pupila. El eje visual atraviesa la fóvea y el centro de la curvatura corneal. El ángulo de Kappa es una desviación angular entre el eje óptico y el visual [4]

.

Los métodos 3D se pueden clasificar según la configuración de cámara implementada y el tipo de calibración realizada. Algunos métodos importantes usan una sola cámara, como en el caso de Martínez [125], el cual de manera adicional hace uso de LED y logra una precisión de 0.5 grados, con la calibración realizada al usuario. En Guestrin [74] se presenta un modelo matemático, el cual logra una precisión de alrededor de 0.9 grados. Su configuración de sistema de seguimiento, también usa dos fuentes de luz NIR y una cámara. En el sistema que propone Hennessey [105], se realiza el procedimiento con una sola cámara y múltiples LED, permitiéndole al usuario un movimiento libre de cabeza y el seguimiento de la mirada en 3D.

Para el desarrollo de los métodos multi-cámara se requiere una configuración más elaborada y compleja que la anterior, puesto que requieren procedimientos de calibración completa del sistema, realizando la calibración tanto en cámaras para las mediciones 3D, como un adecuado posicionamiento de los LED, además, determinar las propiedades geométricas de los monitores y su relación con las cámaras. Aunque estos métodos poseen un mayor grado de dificultad de configuración, logran una alta precisión y robustez frente al movimiento de la cabeza, permitiendo mayor utilidad en variedad de aplicaciones. Trabajos importantes que utilizan dos o más cámaras se informan en [106], [124], [93], [126]. Por su parte, Ohno [124] desarrolla un seguidor ocular con una calibración simple de dos puntos y un sistema de dos cámaras. Este sistema posee su especialidad porque comprende una unidad de posicionamiento del ojo y una unidad de detección de la mirada.

El sistema de SO implementado con base en el método de modelo geométrico 3D y uso de cámara única, posee una serie de parámetros importantes que se deben tener en cuenta para entender su funcionamiento. A continuación se realiza una descripción de las partes importantes que lo componen, como es el caso de la cámara, la cual es la base esencial dentro de toda la configuración del sistema. Por lo tanto, es necesario entender su funcionamiento, sus parámetros (matriz intrínseca y extrínseca) y la configuración necesaria (calibración) para que pueda realizar una adecuada obtención de datos.

También, dentro del sistema de seguimiento se usa un modelo canónico facial, similar al mostrado por [127] y usado para hacer reconocimiento de rostro y ojos, además de la obtención de las coordenadas de los puntos que este ubica en el rostro del participante, cuando se hace el proceso de SO. Se usan librerías y frameworks para implementar este sistema de seguimiento, basado en la información encontrada en la literatura, dichas

librerías se describen en el Anexo A de este documento. Luego se procede a dar la explicación del funcionamiento del sistema.

3.2.1. Modelo canónico

En los últimos años, se ha hecho bastante énfasis en el desarrollo de modelos faciales para el proceso de seguimiento de la mirada. En la Tabla 3.2 se hace una descripción de los principales métodos de alineamiento del rostro. Cada librería desarrolla un modelo facial con algoritmos, funciones y técnicas diferentes. La tabla proporciona un enlace hacia el repositorio donde se encuentra cada implementación. Además, provee un artículo relacionado con su uso, siendo relevante para esta investigación, el modelo facial MediaPipe proporcionado por Google.

Nombre	Año	Publicación	Enlace
Dlib [128]	2014	CVPR	Enlace
MTCNN [129]	2016	SPL	Enlace
DAN [130]	2017	CVPRW	Enlace
OpenFace [131]	2018	FG	Enlace
PRN [132]	2018	ECCV	Enlace
MediaPipe [133]	2019	–	Enlace
3DDFA_V2 [134]	2020	ECCV	Enlace

Tabla 3.2: Principales métodos de alineamiento de rostro (Basado en [95]).

MediaPipe es un framework que se cataloga como un alineador o clasificador de rostros y proporciona un modelo canónico estático del rostro humano, el cual se compone de 468 puntos de referencia faciales 3D (vértices) [135]. Emplea aprendizaje automático (ML) para inferir la superficie facial 3D, permitiendo que solo haya una entrada de cámara sin que sea necesario un sensor de profundidad.

Este modelo se define en escala de unidades métricas y utiliza las posiciones de los puntos en pantalla para calcular una transformación facial dentro de ese espacio. Cada punto de referencia facial tiene un número y posee su vértice con coordenadas establecidas en (x,y,z). El modelo canónico posee la ubicación fija específica de sus coordenadas (x,y,z) en el origen, establecidas como el punto de referencia del modelo. Este modelo usa primitivas 3D comunes, una matriz de transformación de rostros y una malla de rostro triangular (ver Figura 3.4).

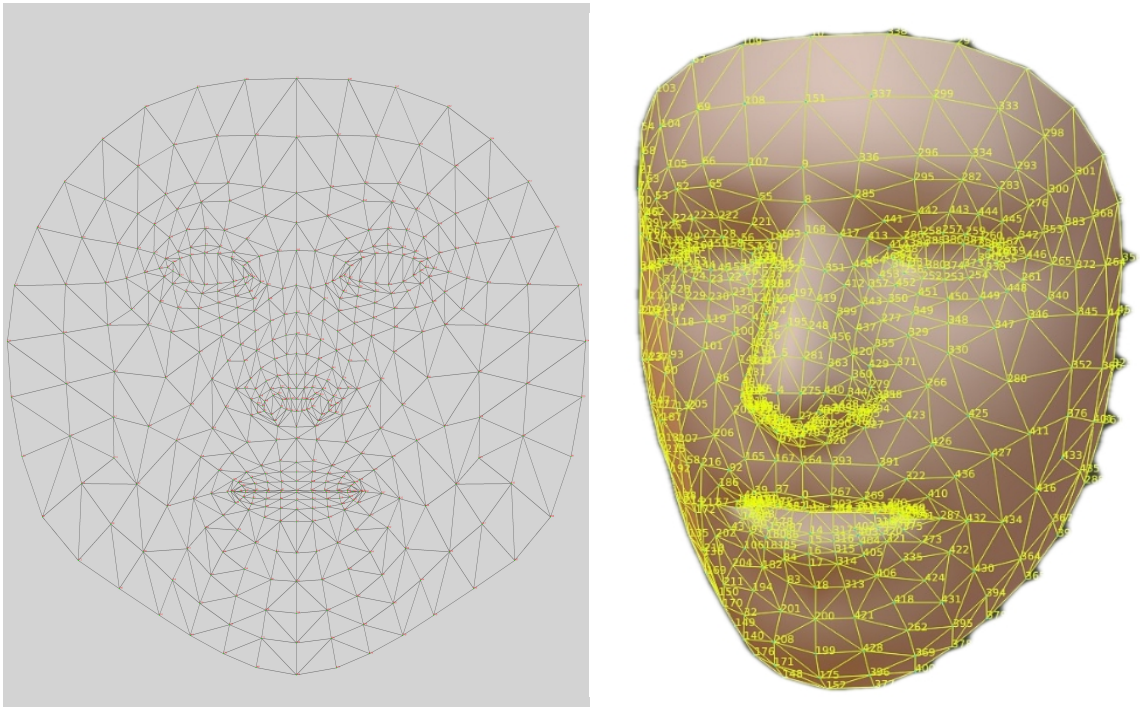


Figura 3.4: Modelo canónico de puntos faciales (Tomado de [136]).

MediaPipe permite realizar detección de rostro y ojos en la imagen obtenida por la cámara, con el uso combinado de funciones de la librería OpenCV. Luego de realizar la detección, alinea los 468 puntos del modelo canónico, en el rostro detectado en la imagen y obtiene las coordenadas 2D de dichos puntos en píxeles. De esta manera, MediaPipe se convierte en un proveedor de información de coordenadas 2D como 3D. Dicha información es importante para encontrar las matrices de rotación y traslación de la cámara, para obtener la posición y orientación de esta, con respecto a las coordenadas del rostro, como se verá en secciones posteriores.

3.2.2. Matriz intrínseca

Es una transformación de proyección que permite darle ubicación en píxeles a los rayos de los puntos proyectados que provienen del objeto del mundo. Esto se realiza por medio de la distancia focal. Dichas capturas de proyecciones de puntos son realizadas sobre el plano de la imagen. En [137], Krishna proporciona información sobre los parámetros y funcionamiento de una cámara estenopeica. Para el desarrollo de este proyecto se hace énfasis en este documento para explicar algunos parámetros muy importantes del sistema de seguimiento implementado. En la Figura 3.5 y Figura 3.6 se observa el modelo de la proyección de un punto en una cámara.

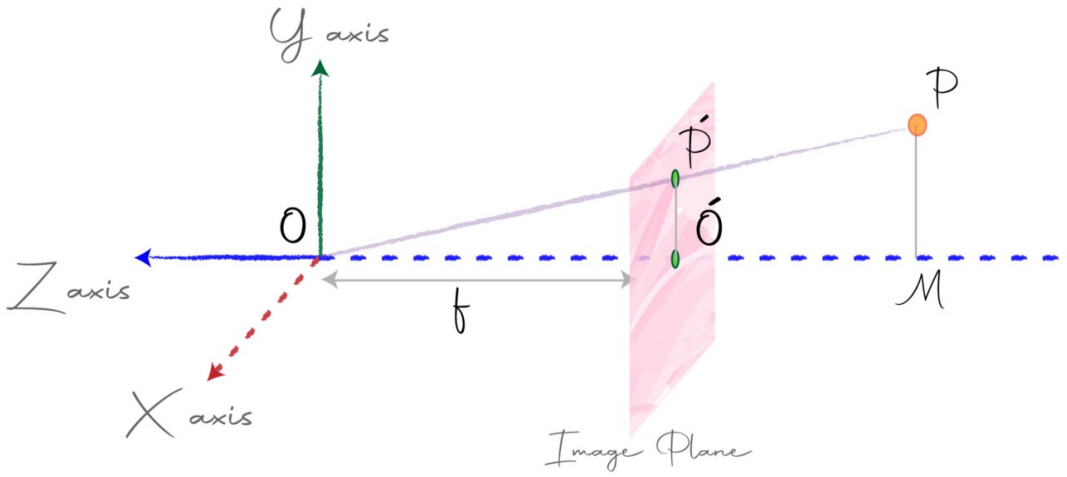


Figura 3.5: Proyección de un punto (Tomado de [137]).

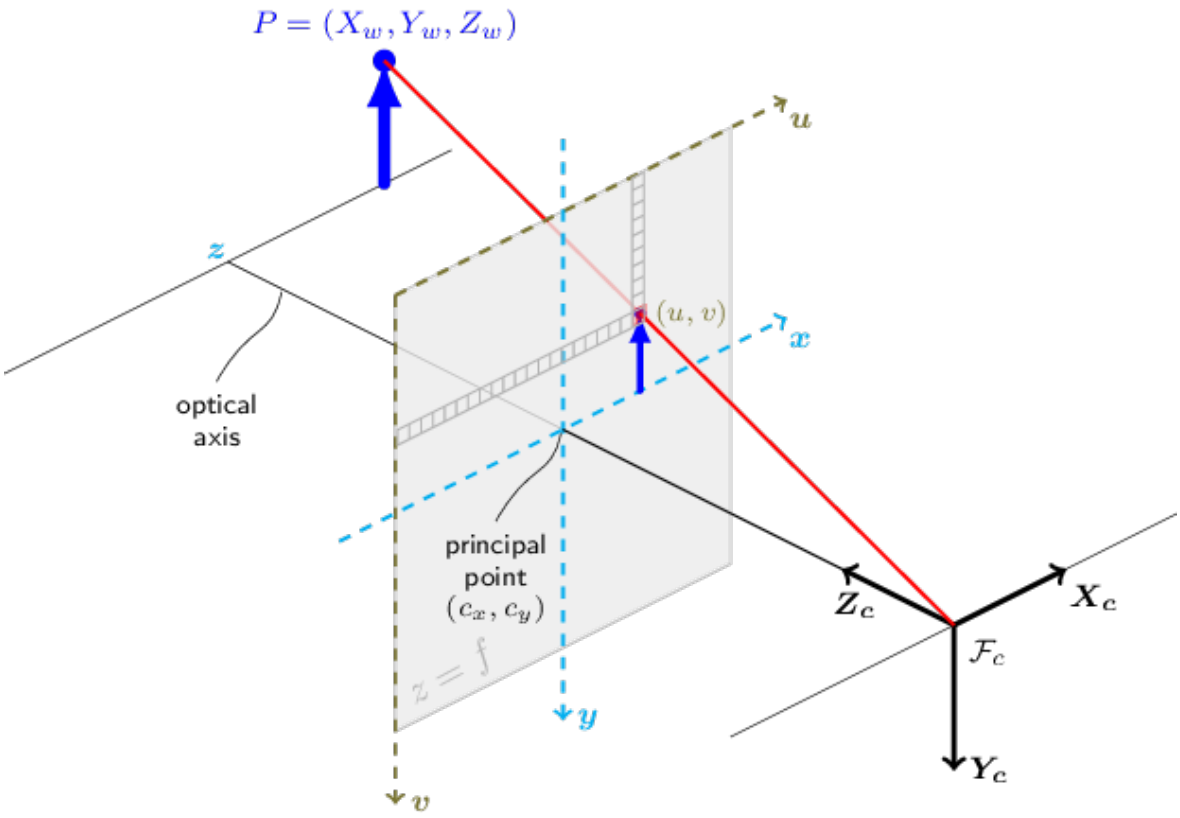


Figura 3.6: Modelo de cámara pinhole (Tomado de [138]).

En la Figura anterior, el centro de la cámara o "pinhole" está ubicado en el origen "0" y el plano de la imagen se encuentra a una distancia focal (f) hacia el eje (-ve Z). Se observa un punto P perteneciente a un punto de un objeto en el mundo. Este se proyecta sobre el plano de la imagen como P'. Las coordenadas de P son (x,y,z) y las de

P' son(x',y',z'), las cuales corresponden a las coordenadas de la proyección del punto P en el plano. Krishna proporciona el cálculo de estas coordenadas y proporciona la ecuación final de formación de imagen, información que se describe en el Anexo B de esta investigación. Dicha ecuación de formación de imagen es:

$$(u, v) = [\frac{\alpha x}{z} - (\frac{\alpha y}{z})cot(\theta) + x0, (\frac{\beta y}{zsin(\theta} + y0] \tag{3.1}$$

De esta ecuación, Krishna obtiene la matriz intrínseca de la cámara, usando coordenadas homogéneas. La matriz obtenida es la siguiente:

$$\begin{bmatrix} f & s & cx \\ 0 & \alpha f & cy \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

De la anterior matriz:

f= Distancia focal

s= Factor de sesgo

cx, cy= Desplazamiento

α= Relación de aspecto

Los parámetros encontrados de la matriz intrínseca, corresponden a factores internos de la cámara y a variables que pueden afectar la formación de una imagen. Tanto el proceso de formación de imagen como la descripción de los parámetros de la cámara intrínseca se describen en el Anexo B de esta investigación.

La matriz intrínseca junto con los coeficientes de distorsión se usan para corregir la distorsión producida por el lente de la cámara y en la práctica pueden ser obtenidos por un proceso denominado calibración de la cámara, usando un tablero de ajedrez y las funciones de la librería de visión artificial OpenCV. Dicho proceso de calibración se describe en el Anexo C de este proyecto. Es importante aclarar que cuando una cámara ha sido calibrada a una resolución de imagen, tomando como ejemplo 320 x 240 píxeles de resolución, sus coeficientes de distorsión calculados pueden ser usados para imágenes con otro tamaño de resolución de imagen, para la misma cámara, mientras que la matriz intrínseca debe ser escalada apropiadamente para cada resolución [138].

3.2.3. Matriz extrínseca

Es un cambio de transformación base de un sistema de coordenadas a otro que permite darle coordenadas a la cámara a partir de las coordenadas de los objetos en el mundo real. Esta matriz nos permite ver el mundo desde la perspectiva de la cámara. Para este proceso es necesario saber cómo se encuentra orientada la cámara y cuál es su ubicación dentro de las coordenadas del mundo (x,y,z). Por este motivo se usan dos transformaciones adicionales: la transformación de rotación para orientar la cámara y la transformación de traslación para mover la cámara. En el Anexo B de esta investigación se provee una descripción sobre los tipos de transformaciones o cambios de base que suelen usarse para determinar las proyecciones de puntos de un objeto en la cámara.

De acuerdo la ecuación 3.3 equivalente a la ecuación 3.1 sobre la formación de imagen 2D, proporcionada en [137] para obtener el vector con las coordenadas de un punto en píxeles (P_p), se debe multiplicar la matriz intrínseca de la cámara (M_i), la matriz de rotación (R) y la matriz de traslación (T), por el vector de coordenadas de los rayos emitidos por los puntos de un objeto en el mundo (P_w). Por lo tanto, para obtener R y T (matriz extrínseca de la cámara) se divide (P_p) sobre la matriz intrínseca (M_i) y las coordenadas del punto del objeto del mundo (P_w), la cual corresponde a la ecuación 3.4, de esta manera:

$$P_p = M_i * R * T * P_w \tag{3.3}$$

$$\frac{P_p * M_i^{-1}}{P_w} = R * T \tag{3.4}$$

Para el cálculo de la matriz extrínseca (rotación y traslación) se hace uso de la función solvePnP de la librería de OpenCV, descrita en el Anexo A de esta investigación. Dicha función requiere como entrada los puntos 2D y 3D, proporcionados por MediaPipe mencionados en la anterior sección, y la matriz intrínseca de la cámara, obtenida mediante el proceso de calibración. La función retorna la matriz de rotación y traslación de la cámara.

3.2.4. Funcionamiento del algoritmo 3D implementado

En la Figura 3.7 se muestra un diagrama de flujo que describe el funcionamiento técnico del algoritmo de SO. La librería FaceAnalyzer [139], permite hacer uso de las funciones de MediaPipe para realizar algunas tareas importantes dentro del proceso de seguimiento. A continuación, se hace una descripción sobre los componentes del diagrama, junto con el uso de funciones implementadas.

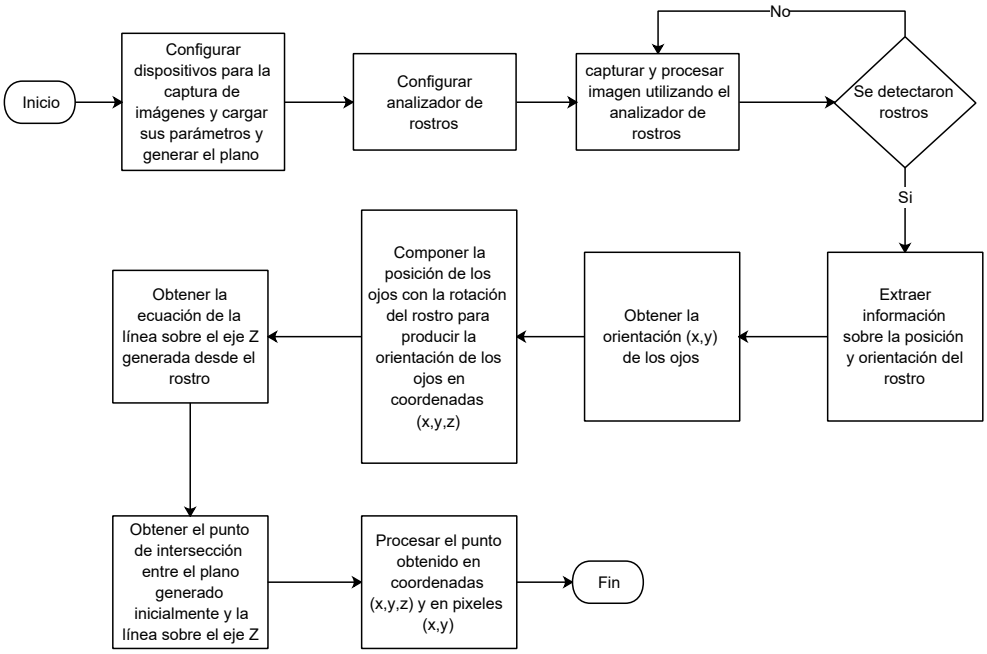


Figura 3.7: Diagrama de SO basado en modelo 3D (Fuente propia).

El algoritmo como primera medida necesita ser configurado para realizar una adecuada toma de datos. Se debe configurar la cámara, eligiendo el tamaño de captura de imagen y, para evitar distorsión en la generación de imagen, se carga la matriz intrínseca de la cámara junto con los coeficientes de distorsión. Se hace uso de la librería OpenCV para hacer lectura de cámara y video. Luego se realiza una configuración para la detección de rostro, configurando un analizador de rostros con ayuda de la librería FaceAnalyzer, que usa una función que tiene el mismo nombre **"FaceAnalyzer"**. Esta función permite establecer cuál es el máximo número de rostros que debe detectar el algoritmo junto con la configuración de tamaño de dichas capturas.

La función que permite obtener la postura de la cabeza en la imagen capturada por la cámara, haciendo uso de la proyección de N puntos (PnP), se describe como: **"get_head_posture"**. La función tiene como parámetros de entrada la matriz intrínseca de la cámara y los coeficientes de distorsión. De dicha función se obtiene una tupla con la posición (pos) y orientación (ori) de la cámara, correspondiente a las matrices R y T ya descritas anteriormente. Pos y Ori se guardan y serán usadas más adelante.

Con ayuda del analizador de rostros y la función "process" se realiza la captura de imagen. La función "process" escanea la imagen, comprueba la existencia de rostros y guarda el número de rostros detectados, toma posteriormente cada rostro y, por medio de la función de postura, le asigna posición y orientación al rostro. Luego, se hace uso de la función **"get_eyes_position"**, que permite obtener la posición de los iris en el rostro detectado. Dicha función recibe como parámetros de entrada la matriz intrínseca de la cámara y los coeficientes de distorsión y retorna las posiciones del iris dentro del ojo.

Después se genera una concatenación, donde se toma el promedio de las posiciones de los iris, simulando un ojo en la distancia promedio entre ambos ojos. Dicho ojo recibe las coordenadas de posición de los dos iris y una relación de orientación con respecto a la cara por medio de la función, **"compose_eye_rot"**. La función permite que la posición y los movimientos del ojo se ajusten y sean dependientes a los movimientos de la cabeza. Lo que servirá más adelante para generar la línea desde el centro del globo ocular a través de la pupila, para simular la vista de dicho ojo.

La función **"get_z_line_equation"** permite generar un vector normal. Esta función tiene como argumentos de entrada la posición origen del vector en el rostro y la orientación del vector generado, dada también por el rostro. Se retorna una tupla correspondiente a un vector que se compone de su posición y orientación en el eje z.

La función que usa FaceAnalyzer para crear el vector está dada por **"get_plane_infos"**. Esta función crea un vector en el espacio de coordenadas entre el modelo 3D del ojo recreado y la cámara simulada. El plano simula la pantalla del sistema físico, pero no posee las coordenadas exactas de la pantalla con respecto a la cámara, sino que se genera lo suficientemente grande para que el rayo de mirada que sale de la pupila, cruce el plano y se añade un delimitador para realizar cruces hasta ciertos límites del plano. La función recibe como parámetros de entrada tres vectores correspondientes a un plano y hace uso de pos y ori, obtenidos anteriormente. Dicha función retorna una tupla con cuatro parámetros que corresponden a la posición origen del plano, dos vectores orto-normales para el marco del plano y un vector normal al plano.

La función **"get_plane_line_intersection"** es la que genera la intersección entre la línea y el plano. Esta función recibe como parámetros la tupla que corresponde a la línea del vector normal proveniente del rostro y la tupla con la información del plano virtual. La función retorna las coordenadas 3D y 2D que corresponden al cruce del vector normal con el plano virtual.

En Höffner [140], se implementa un modelo de globo ocular 3D, el cual realiza las estimaciones de postura de cabeza junto con la detección de rostro, pupilas y globos oculares. En dicho artículo se prueba el algoritmo 3D con ayuda de la librería Dlib. Höffner usa un modelo geométrico para encontrar los puntos de mirada en la pantalla, utilizando ray cast, rayos que se lanzan desde el centro del globo ocular a través de la pupila, donde las intersecciones del plano de la pantalla y las proyecciones de dichos rayos son los puntos clave para la detección. A continuación se realiza una descripción sobre el funcionamiento del algoritmo de seguimiento implementado, tomando como base algunas exposiciones de este documento.

El algoritmo realiza la captura de imagen por medio de la cámara MediaPipe. Como primera medida detecta su rostro y las ubicaciones 2D de los iris de cada uno de sus ojos. La matriz intrínseca de la cámara debió ser obtenida previamente junto con los coeficientes de distorsión de la misma, tal como se explicó en la sección 3.2.2. Este proceso solo se realiza una vez y se guarda para aplicarse posteriormente a las imágenes tomadas por la cámara.

Cuando el algoritmo de MediaPipe ubica los puntos del modelo canónico sobre el rostro del participante, las coordenadas de los puntos de dicho modelo se igualan con

las coordenadas de su rostro y/o cabeza y, dado que el modelo está centrado sobre el origen, con coordenadas de referencia (x,y,z) en (0,0,0). Entonces, las coordenadas de la cabeza se toman también como el punto de referencia (0,0,0) en el espacio de coordenadas 3D.

Luego de plasmar el modelo sobre el rostro de la persona con sus puntos, MediaPipe toma seis puntos referencia de los 468, los cuales se denominan como puntos principales del rostro, denotados como P_w. En la Tabla 3.3 se listan estos puntos de referencia, junto con sus nombres y coordenadas, y en la Figura 3.8 se ilustra un modelo facial 3D con dichos puntos resaltados. Se hace uso de la función "solvePnP', la cual recibe las coordenadas 3D de dichos puntos y, de acuerdo a sus coordenadas generadas en píxeles por MediaPipe, soluciona el problema de proyección de n puntos (PnP), mencionado en la sección 3.2.3.

La función hace uso de las matrices de R y T, pertenecientes a la matriz extrínseca de la cámara y determina sus coordenadas 3D con respecto a las coordenadas de la cámara, además de generar los ángulos de rotación de la cabeza (pitch, roll y yaw). Después de solucionar el problema de proyección de n puntos, se puede hallar las coordenadas con respecto a la cámara de cualquier ubicación del rostro detectado, correspondiente a cualquiera de los 468 puntos pertenecientes al modelo canónico (ver Figura 3.4). De esta forma se hallan las coordenadas de los puntos que pertenecen a las pupilas detectadas en el rostro de la imagen.

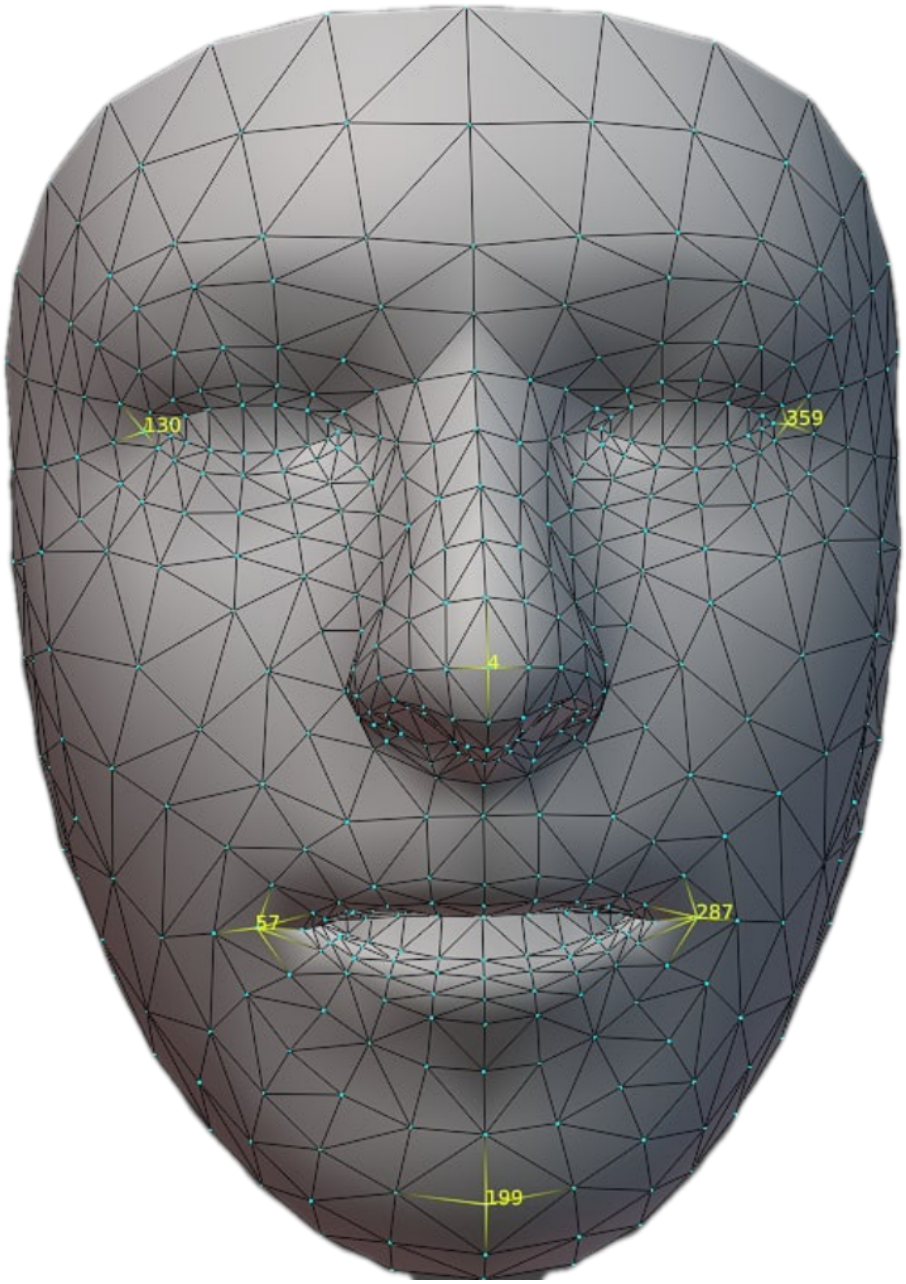


Figura 3.8: Modelo canónico con puntos de referencia (Tomado de [135]).

Nombre del punto referencia	**Abreviatura**	**No. Punto**	**Coordenadas en MediaPipe [mm]**
Pronasal	prn	4	(0, -0.004632, -0.075866)
Gnathion	gn	199	(0, -0.079422, -0.051812)
Exocanthion right	ex_r	130	(-0.04671, 0.026645, -0.030841)
Exocanthion left	ex_l	359	(0.04671, 0.026645, -0.030841)
Cheilion right	ch_r	57	(-0.031026, -0.04353, -0.040959)
Cheilion left	ch_l	287	(0.031026, -0.04353, -0.040959)

Tabla 3.3: Puntos referencia del modelo de cabeza 3D de MediaPipe (Puntos generados en FaceAnalyzer).

Con la información obtenida se tienen las coordenadas 3D de la cámara, las coordenadas 3D de la cabeza y las coordenadas 3D de las pupilas. Con esta información se encuentra

la ubicación 3D de los centros de los globos oculares. Se ha reportado que el radio de un globo ocular humano mide aproximadamente 1.2 cm [127]. De acuerdo a esto se puede definir las coordenadas 3D del centro del globo ocular con respecto a la cámara midiendo la distancia en el eje z que hay desde el centro de la cámara hasta el iris y sumándole los 1.2 cm en el mismo eje z, distancia en la que el eje z obtiene la posición 3D del centro del globo ocular del ojo.

Con las coordenadas 3D de cabeza, cámara, pupilas y centros oculares se genera un modelo esférico que simule un globo ocular con un diámetro de 24 mm (valor reportado en Davson 2017). También se genera un círculo con diámetro pequeño y se ubica en las coordenadas encontradas de la pupila. Se crea un vector normal y un plano virtual para simular el choque de la mirada en la pantalla. El vector tiene un origen en el centro del globo ocular, el cual debe pasar por el centro de la pupila y debe interceptar con el plano virtual (ver Figura 3.9). Todo esto en un espacio interno generado en torno a las coordenadas de referencia de la cabeza.

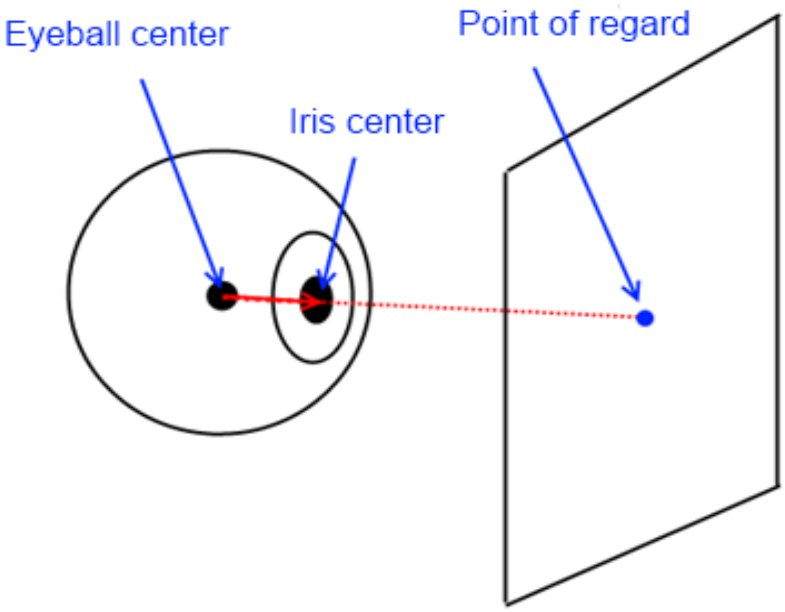


Figura 3.9: Modelo de cabeza 3D (Tomado de [127]).

En [95] se ilustra la Figura 3.10, donde se muestra un método basado en modelo 3D que realiza un proceso similar al expuesto por Höffner [140]. Aquí se ilustra la intersección entre una línea desde el ojo hasta interceptar con el plano virtual. Dicha interceptación se denomina POG y para realizar su cálculo se hace uso de coordenadas de ubicación y orientación, proporcionadas por las matrices de rotación y traslación Rs y Ts, explicadas anteriormente.

Symbol	meaning
$\boldsymbol{p} \in \mathbb{R}^2$	$\boldsymbol{p} = (u, v)$, gaze targets.
$\boldsymbol{g} \in \mathbb{R}^3$	$\boldsymbol{g} = (g_x, g_y, g_z)$, gaze directions.
$\boldsymbol{o} \in \mathbb{R}^3$	$\boldsymbol{o} = (x_o, y_o, z_o)$, origins of gaze directions.
$\boldsymbol{t} \in \mathbb{R}^3$	$\boldsymbol{t} = (x_t, y_t, z_t)$, targets of gaze directions.
$\boldsymbol{R_s} \in \mathbb{R}^{3\times3}$	The rotation matrix of SCS w.r.t. CCS.
$\boldsymbol{T_s} \in \mathbb{R}^3$	$\boldsymbol{T_s} = (t_x, t_y, t_z)$, the translation matrix of SCS w.r.t. CCS.
$\boldsymbol{n} \in \mathbb{R}^3$	$\boldsymbol{n} = (n_x, n_y, n_z)$, the normal vectors of x-y plane of SCS.

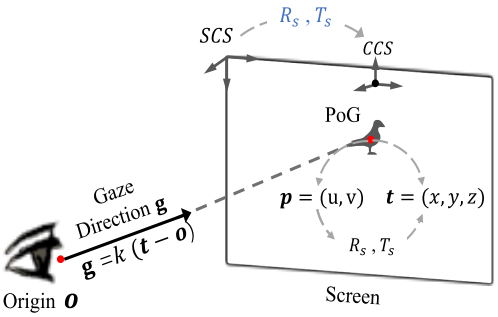


Figura 3.10: Método similar implementado por Yihua (Tomado de [95]).

De acuerdo a la Figura 3.10, "g" corresponde a la dirección de la mirada y se puede representar como un vector normal con origen cero (0). "Screen" corresponde a un plano virtual y el punto donde ambos se interceptan se denomina POG. Para generar un vector normal y plano virtual junto con la intersección de ambos, se usa la librería FaceAnalyzer, librería de Saifeddine Aloui [141] la cual proporciona funciones y herramientas necesarias para dicha implementación. También se tomó información importante del proyecto OMES [142], para el manejo de MediaPipe y OpenCV.

Para la creación del vector normal, FaceAnalyzer recibe dos parámetros de entrada, los cuales se componen de coordenadas del origen del vector y los ángulos de orientación de la postura de cabeza y ojos. La línea de la mirada es un vector normal al rostro de la persona y su módulo o longitud recorre el eje z y es también un vector normal para el plano virtual. Dicha función retorna una colección de datos en tupla con la posición y orientación del vector de mirada sobre el eje Z.

Para la generación del plano virtual, FaceAnalyzer recibe como parámetros de entrada 3 vectores comunes P1, P2 y P3, donde P1 corresponde a la posición del origen del plano con coordenadas (0,0,0), P2 con coordenadas (100,0,0) y P3 con coordenadas (0,100,0). FaceAnalyzer retorna a su salida una tupla con 4 parámetros, donde se hace uso del producto cruz de vectores. Los parámetros obtenidos corresponden a:
n: Vector normal al plano virtual.

$$\vec{n} = (\vec{P_2} - \vec{P_1}) \times (\vec{P_3} - \vec{P_1}) \tag{3.5}$$

P: Posición origen en coordenadas 3D del plano virtual.

$$P = (X, Y, Z)Coordenadas \tag{3.6}$$

e_1, e_2: Vectores orto-normales que definen un marco de referencia del plano virtual.

$$\vec{e_1} = (\vec{P_2} - \vec{P_1}) \tag{3.7}$$

$$\vec{e_2} = (\vec{n}) \times (\vec{e_1}) \tag{3.8}$$

Para convertir dichos vectores a vectores unitarios se dividen las coordenadas de cada vector sobre su respectiva norma:

$$\vec{n} = \frac{n}{\|n\|}, \vec{n} = \frac{(\vec{P_2} - \vec{P_1}) \times (\vec{P_3} - \vec{P_1})}{\|(\vec{P_2} - \vec{P_1})\|} \tag{3.9}$$

$$\vec{e_1} = \frac{e_1}{\|e_1\|}, \vec{e_1} = \frac{(\vec{P_2} - \vec{P_1})}{\|(\vec{P_2} - \vec{P_1})\|} \tag{3.10}$$

$$\vec{e_2} = (\vec{n}) \times (\vec{e_1}), \vec{e_2} = (\frac{(\vec{P_2} - \vec{P_1}) \times (\vec{P_3} - \vec{P_1})}{\|(\vec{P_2} - \vec{P_1})\|}) \times (\frac{(\vec{P_2} - \vec{P_1})}{\|(\vec{P_2} - \vec{P_1})\|}) \tag{3.11}$$

De esta forma se obtienen la posición origen del plano, 2 vectores orto-normales para la generación del plano y un vector normal al plano generado.

En la intersección, FaceAnalyzer obtiene como entrada las dos tuplas pertenecientes a la ecuación del vector normal de la mirada y la ecuación del plano virtual, retornando a su salida "$P2D''$ correspondiente a las coordenadas 2D del punto del plano, donde se intercepta la línea virtual y "P'', correspondiente a la coordenada 3D del vector normal donde se produce el cruce con el plano. Para entender el proceso de cruce de línea y plano, desarrollado por dichas funciones, véase [143].

Con dichas coordenadas 2D de mirada en el plano, se puede realizar el seguimiento de la mirada, usando puntos de referencia en la pantalla. Una vez el detector ubique las pupilas y sus coordenadas en pantalla, se puede evaluar qué tan exacto y preciso es con respecto al seguimiento. Luego se toman los fotogramas, se procesan por medio de super-resolución (ver Capítulo 4) y se compara el seguimiento de acuerdo a tres posiciones, la posición real de los puntos en pantalla, la posición detectada en los fotogramas enviados al algoritmo de SO y la posición obtenida con aplicación de super-resolución a dichos fotogramas.

En el trabajo desarrollado por Höffner, se observa que para implementar un seguidor ocular basado en un modelo 3D no es necesario hacer uso de luces IR, pues el algoritmo se ajusta a las condiciones de luz bajas, haciendo que el uso de luces IR no sea determinante. De la misma forma, al realizar pruebas en esta investigación, se determinó que el algoritmo usado por MediaPipe es robusto ante condiciones bajas de iluminación, haciendo que el uso de luces IR no sea fundamental. Por lo tanto, en la implementación desarrollada en esta investigación se decidió no optar por esta configuración adicional, en el sistema de seguimiento llevado a cabo. En la figura (tal) se muestra la toma de datos realizada con condiciones de baja iluminación, en la que el algoritmo realiza un seguimiento de mirada sin ningún tipo de alteración.
(imagen toma de datos)

En la siguiente sección se muestra el desarrollo de una interfaz gráfica con los componentes mencionados a lo largo de este capítulo. Dicha interfaz posee una ventana con múltiples pestañas como prueba y ajuste de imagen, detección de pupila 2D, aplicación de modelo canónico en el rostro y otros.

3.3. Interfaz gráfica

En el diagrama de la Figura 3.11 se muestra el proceso general de SO implementado en el proyecto y a continuación se muestra la interfaz gráfica implementada para llevarlo a cabo.

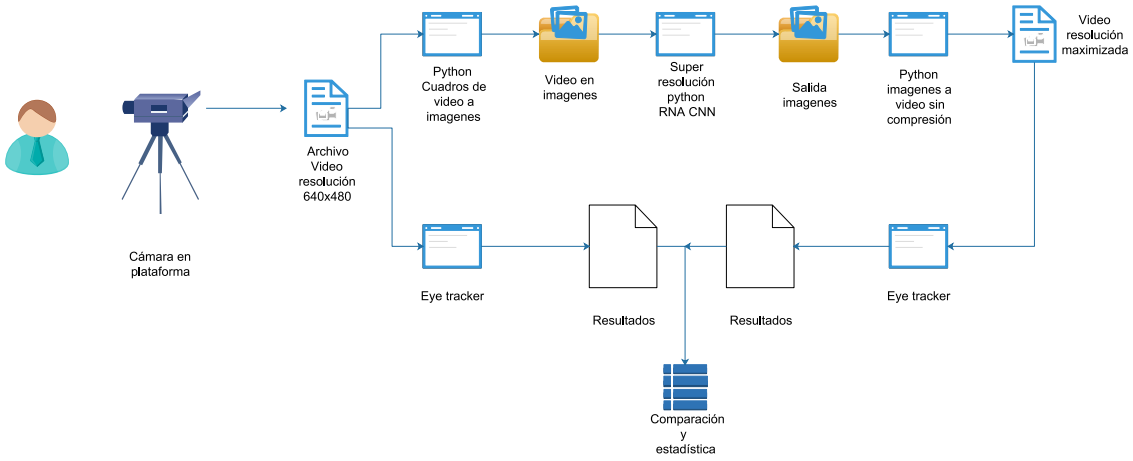


Figura 3.11: Diagrama de flujo del SO implementado (Fuente propia).

Para realizar el proceso de SO se generó una interfaz gráfica que combina todos los algoritmos, frameworks y librerías mencionadas. En la Figura 3.12 se muestra el diagrama de clases implementado en la interfaz gráfica del proyecto. Esta interfaz permite obtener y procesar la información necesaria, con ayuda de sus botones. El diseño gráfico y visual de la interfaz se realiza mediante QT nativo en una versión pyside2, pues esta última posee licencia gratuita de uso.

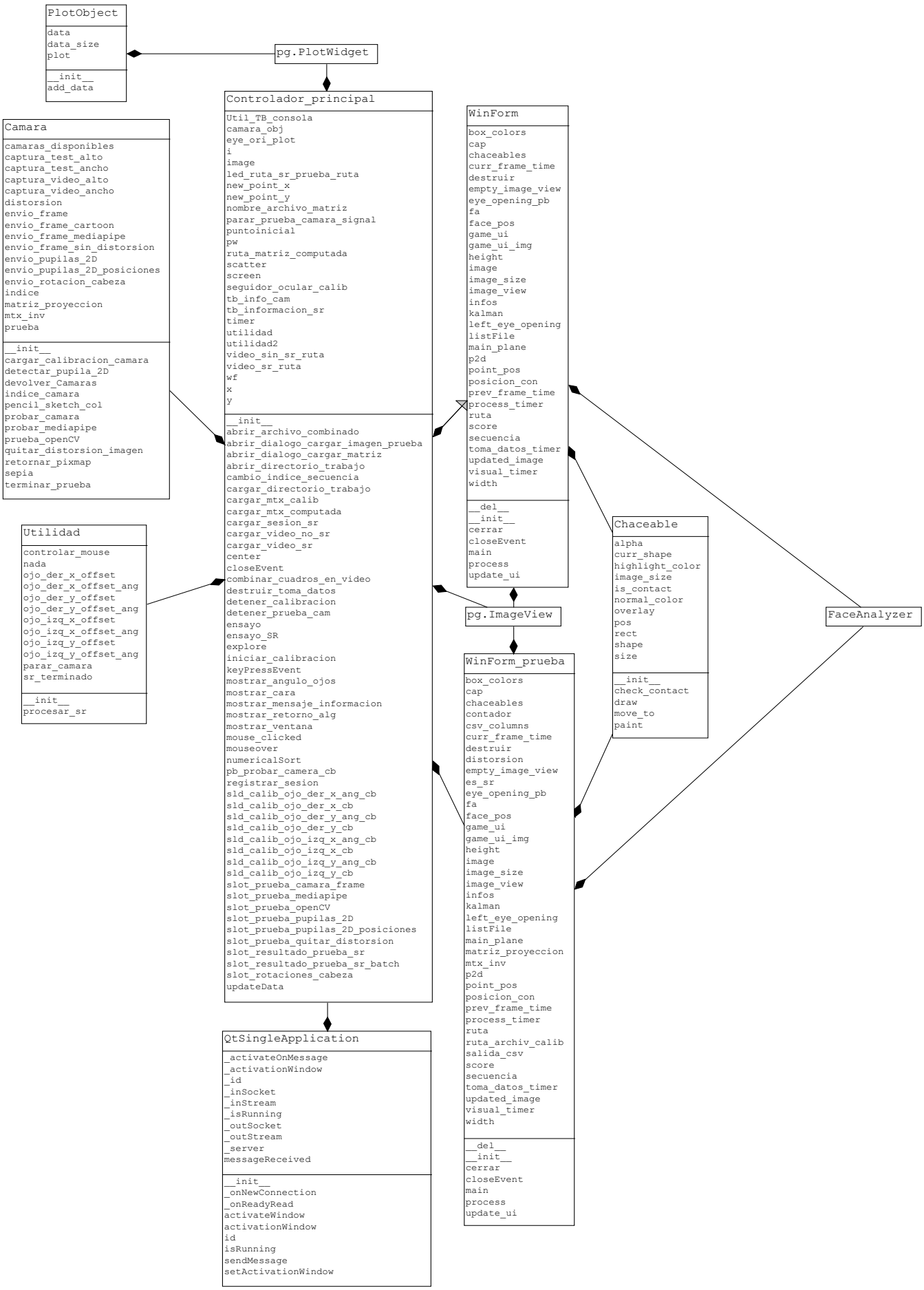


Figura 3.12: Diagrama de clases (Fuente propia).

Es necesario aclarar que previo al uso a la interfaz gráfica, esta debe ser cargada con los coeficientes de distorsión y la matriz de calibración de la cámara (matriz intrínseca), presionando en el botón cargar y buscar el archivo de la calibración correspondiente. La interfaz posee cuatro pestañas principales, las cuales se explicarán a continuación.

1. Pestaña de registro de datos. Se obtiene la información de usuario para guardar sus datos personales (nombre, identificación) y luego generar la carpeta con el nombre, la fecha y la hora en que se generaron sus datos de seguimiento. En dicha carpeta se guardarán los fotogramas capturados durante el proceso de seguimiento.

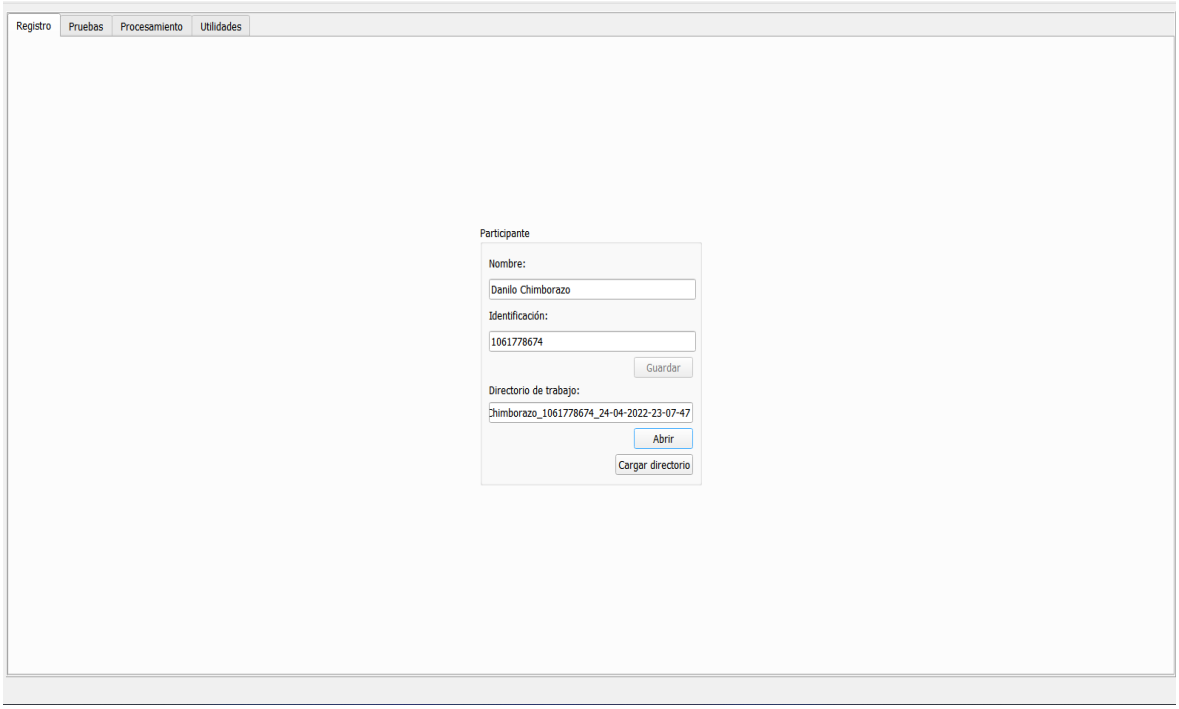


Figura 3.13: Pestaña de registro de datos.

2. Pestaña de pruebas. Se prueba que la cámara esté disponible y capturando el video, que OpenCV y MediaPipe funcionen adecuadamente realizando las detecciones. En esta pestaña se prueba el correcto funcionamiento de todo el sistema de seguimiento junto con sus librerías, frameworks, funciones y algoritmos. De esta manera se evita que quien controla el sistema no tenga que estar haciendo revisiones en todo el código de programación, para ver si están cargados todos los paquetes y correr cada algoritmo o pestaña por cada vez que se desee completar un proceso. Esta ventana hace más fácil la verificación de funcionamiento.

Esta pestaña posee cinco pestañas internas, con nombre: Cámara, OpenCV, MediaPipe, MediaPipe pupila 2D y super-resolución. Esta última se muestra en el Capítulo 4. En la pestaña cámara, se prueba cuántas cámaras están disponibles y también se encuentra el botón para cargar la matriz intrínseca de cámara.

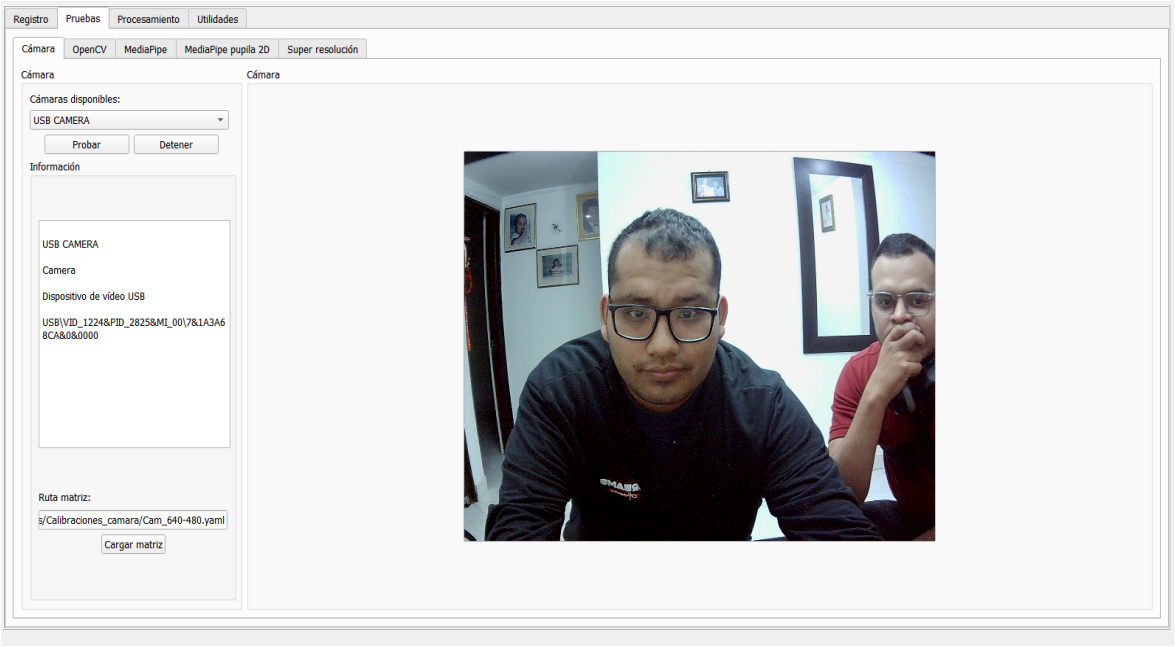


Figura 3.14: Funcionamiento de la cámara.

En la pestaña OpenCV se muestran dos imágenes, donde la toma de imagen de la izquierda es imagen tomada por la cámara sin distorsión y la imagen de la derecha posee aún distorsión y tiene aplicado un filtro de imagen. De esta forma se pueden comparar las dos diferentes tomas y se puede probar que OpenCV está aplicando la matriz intrínseca junto con los coeficientes de distorsión.

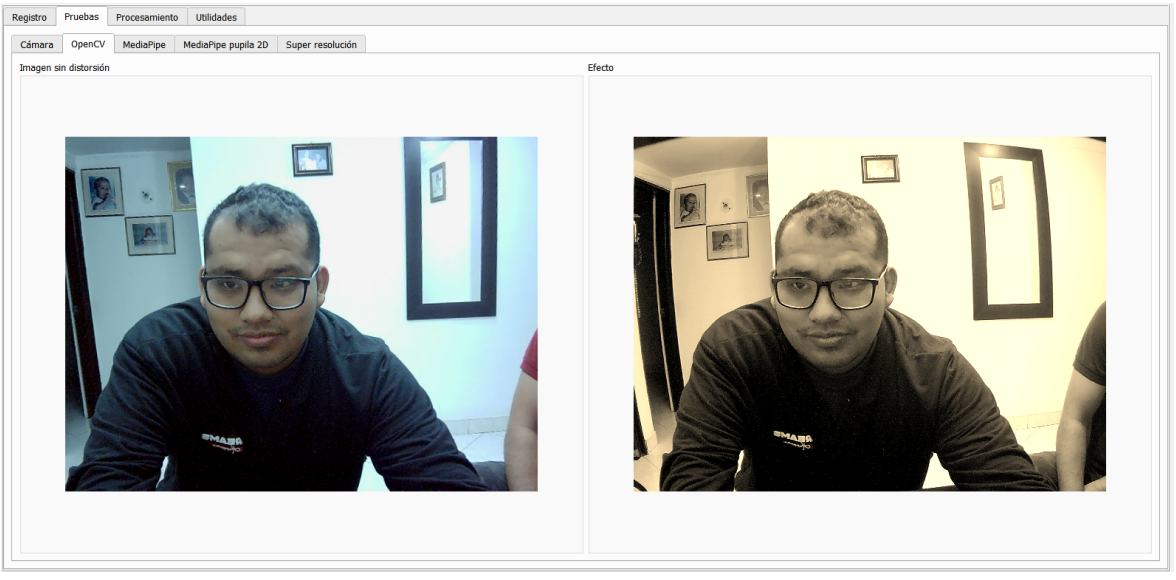


Figura 3.15: Funcionamiento de OpenCV.

En la pestaña MediaPipe se prueba que el algoritmo esté ubicando la malla del modelo canónico sobre el rostro.

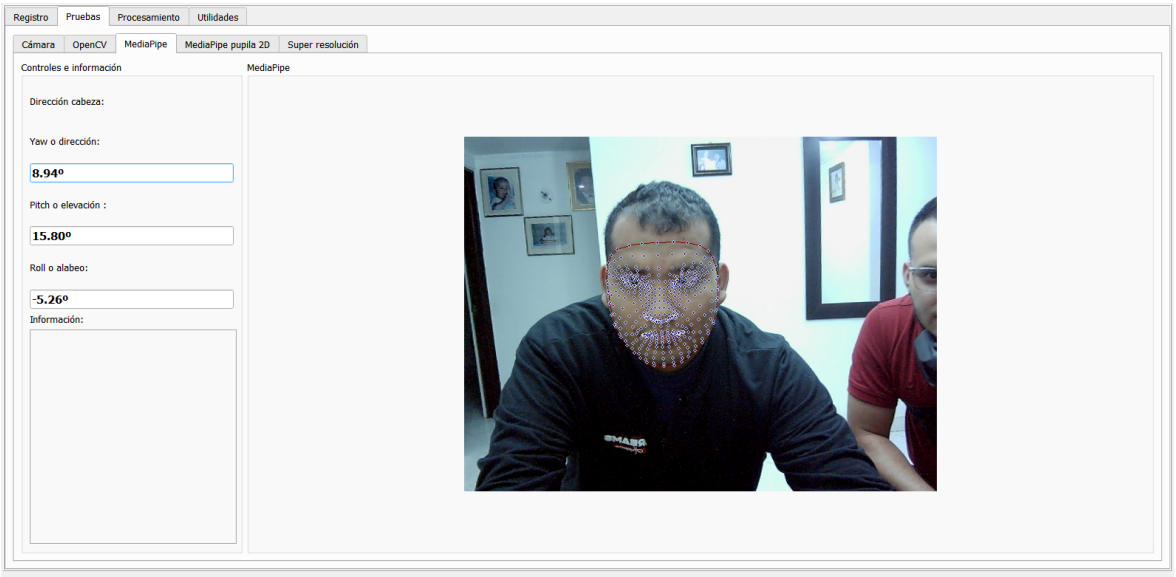


Figura 3.16: Funcionamiento de MediaPipe.

En la pestaña MediaPipe pupila 2D se prueba que el algoritmo esté haciendo el seguimiento adecuado de la pupila. Se verifica que se muestre las ubicaciones de las pupilas en píxeles en X y en Y.

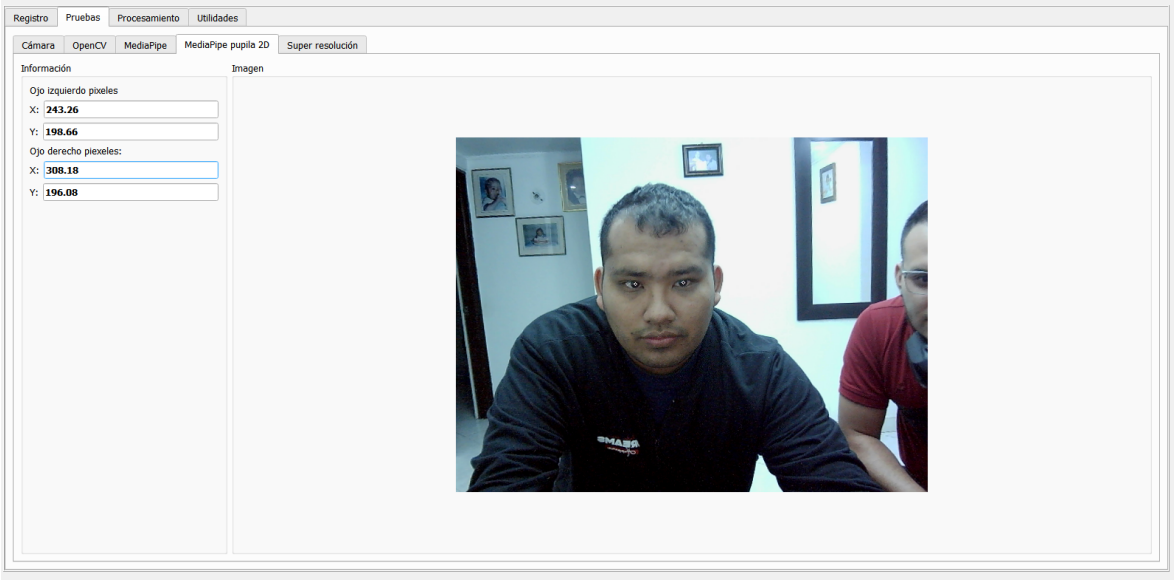


Figura 3.17: Funcionamiento de MediaPipe Pupila.

3. Pestaña de calibración y procesamiento. Posee dos pestañas internas, una para realizar una calibración sencilla con el participante y la otra para realizar el proceso de obtención de información del usuario.

El sistema utiliza una calibración genérica, en la cual se usa todo el ancho de la pantalla. En caso de que se requiera usar una dimensión distinta dentro de la pantalla, se debe realizar una calibración extra con 2 puntos en la parte superior izquierda e inferior derecha ubicada en las nuevas dimensiones, como se ve en la Figura 3.18.

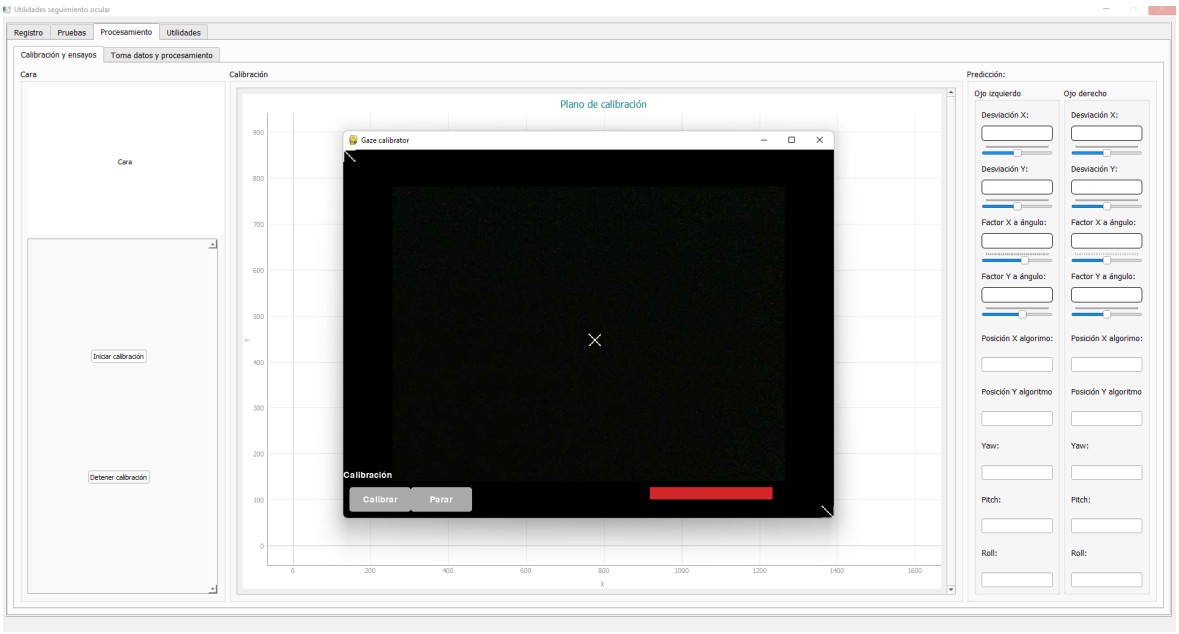


Figura 3.18: Funcionamiento de calibración extra.

La pestaña de procesamiento permite escoger cuatro tipos diferentes de secuencias. En cada secuencia se genera un punto con coordenadas distintas de pantalla. El participante se graba realizando estas cuatro secuencias y por cada una se graba un video y se obtienen sus fotogramas, los cuales se guardan en la carpeta generada en la pestaña de registro de dicho participante. En total se generan cuatro carpetas por cada participante.

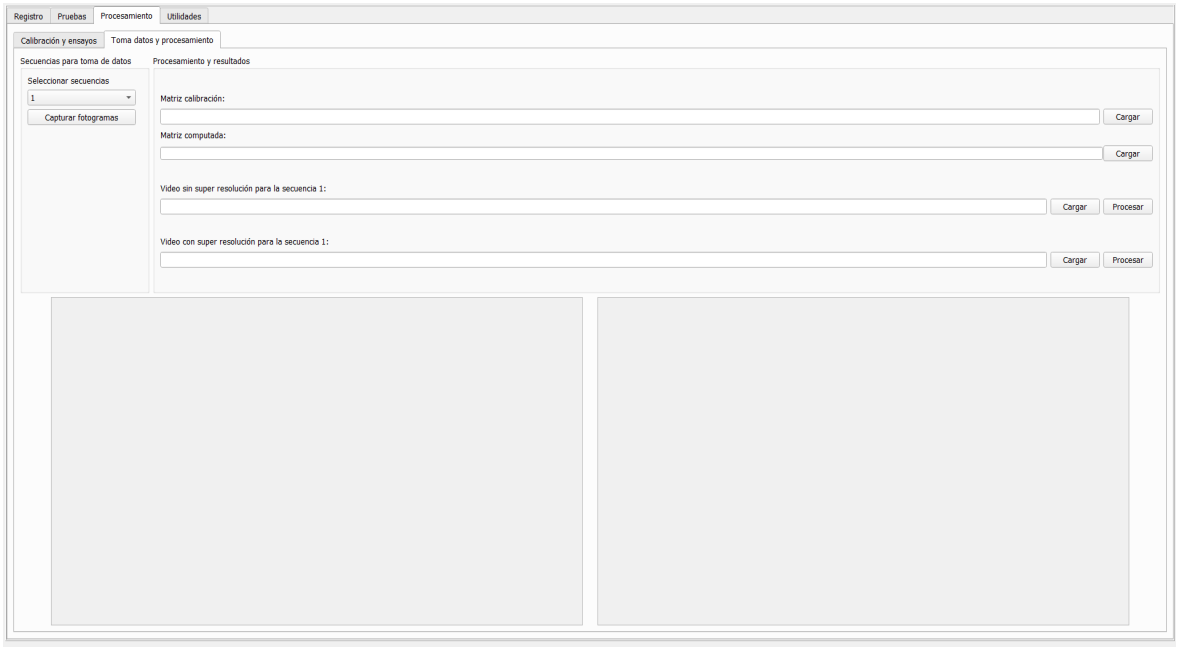


Figura 3.19: Pestaña de procesamiento.

4. Pestaña de utilidades. Posee un botón para cargar las carpetas por lotes de fotogramas para aplicarles super-resolución, generando una carpeta dentro de la carpeta de registro de cada participante con nombre S.R. y dentro de esta se ubican los fotogramas con aplicación de super-resolución. También posee el botón para cargar nuevamente los

fotogramas y generar el video nuevamente para enviar al algoritmo de SO. Se muestra una pequeña ventana donde se observa la salida o consola de Python.

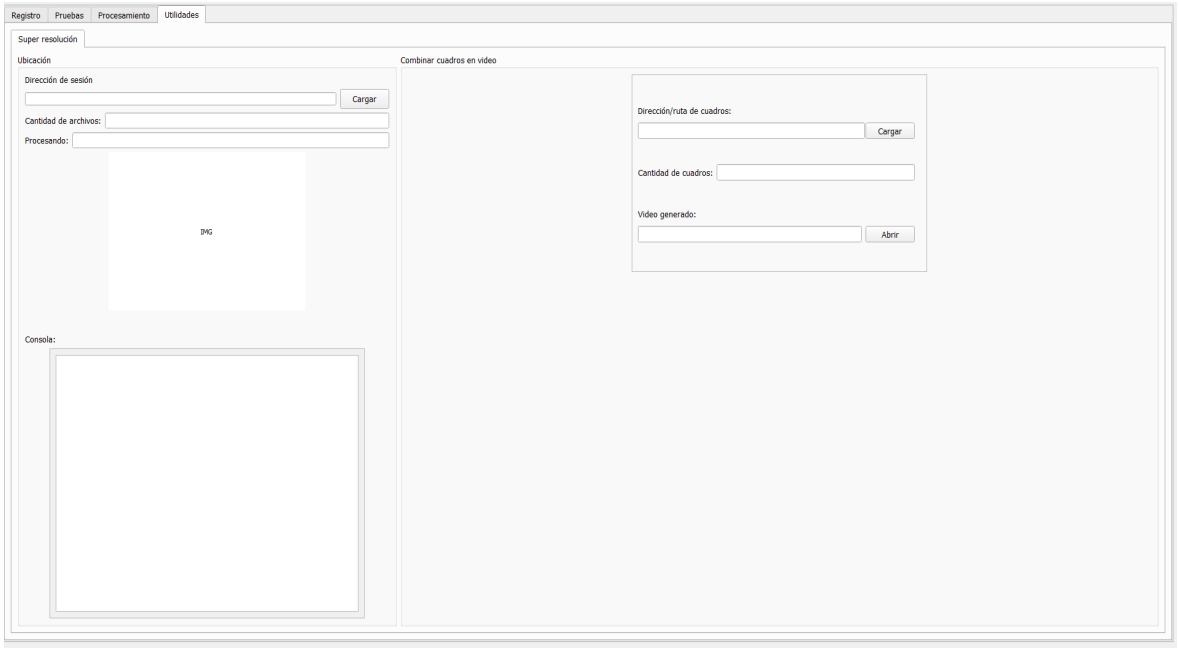


Figura 3.20: Pestaña de utilidades.

Capítulo 4

Super-resolución

El proceso de super-resolución comprende temáticas muy importantes como las Redes Neuronales Artificiales (ANN, sigla en inglés) y específicamente las Redes Neuronales Convolucionales (CNN, sigla en inglés), junto con otras técnicas dentro del campo del aprendizaje profundo.

4.1. Redes Neuronales Artificiales (ANN)

Para dar un concepto adecuado de ANN nos basamos en [144], donde se menciona que las ANN son una colección de neuronas o unidades básicas de procesamiento conectadas entre sí y que estas, al mismo tiempo, se encuentran en capas, siendo capaces de reconocer, procesar y clasificar las diversas formas de información provenientes del mundo exterior. De esta forma, una neurona artificial es considerada similar, según su comportamiento, a una neurona biológica e inclusive a una función matemática [145], pues estas poseen una información de entrada, un proceso interno y una respuesta como salida (ver Figura 4.1); información que pasa de una neurona hacia otras como en una red neuronal biológica.

Es así como una red neuronal se compone intensamente de variedad de unidades de procesamiento, con conexiones de entrada a través de los cuales reciben estímulos externos, los cuales son los valores de entrada. Con esos valores realizará cálculos internos y generará un valor de salida. Internamente, la neurona usa todos los valores para realizar una suma ponderada de ellos. La ponderación de cada una de las entradas viene dada por el peso que se asigna a cada una de las conexiones de entrada. Cada conexión que llega a la neurona tendrá asociada un valor que servirá para definir en qué porcentaje esa variable de entrada afecta a la neurona (ver Figura 4.2).

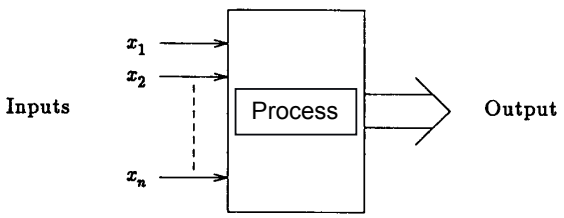


Figura 4.1: Arquitectura de una neurona artificial (Tomado de [146]).

Las redes neuronales se componen de capas ocultas [147]. Cuando la cantidad de capas ocultas que posee una red neuronal es mayor, permite adquirir conocimientos más complejos. Entre más capas se añaden, más complejo puede ser el conocimiento elaborado. Esta profundidad en la cantidad de capas es lo que se conoce como aprendizaje profundo [54], para alcanzarlo, se deben conectar múltiples neuronas de forma secuencial, como se ve en la Figura 4.2.

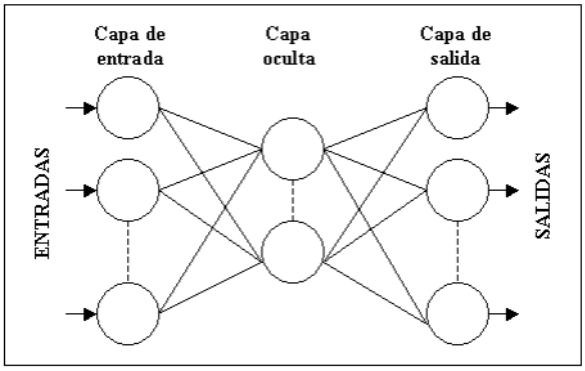


Figura 4.2: Composición interna de ANN (Tomado de [148]).

La función de activación distorsiona los valores de salida, añadiéndole deformaciones no lineales para poder encadenar de forma efectiva la computación de varias neuronas. Estas deformaciones dependen de las funciones de activación. Según [144] existen 5 tipos de funciones de activación:

- Función escalonada. Con un valor de 0-1 en función del valor de umbral. Para el valor de entrada mayor al umbral el valor es 1 y para un valor de entrada menor al umbral el valor es 0.
- Función sigmoide. Hace que valores muy grandes se saturen en 1 y valores muy pequeños se saturen en 0. La función sigmoide también funciona para representar probabilidad de por qué dan valores en rangos de 0 a 1.
- Función tangente hiperbólica. Varía similar a sigmoide y cuyo rango varía desde -1 a 1.
- Función unidad rectificada lineal o ReLu. Se comporta como una función lineal cuando es positiva y constante a 0 cuando el valor de entrada es negativo.
- Función de activación de Saha Bora (SBAF). SBAF es una función de activación que puede resolver el problema de alcanzar óptimos globales en tiempo finito, debido a su estructura matemática.

Estas 5 funciones aportan la no linealidad a las funciones resultantes de la suma ponderada de valores de entrada.

Las redes neuronales también permiten ser clasificadas bajo dos criterios. El primer criterio depende de la naturaleza de su entrada, la cual puede ser continua o binaria (discreta). Otra puede ser vista desde 5 propiedades, a saber: topología, arquitectura, modelo neuronal, algoritmo de aprendizaje y planificación [145]. En [149] podemos mirar los diferentes tipos de redes neuronales de acuerdo a sus características:

- Según su topología o estructura de la red. Podemos tomar las características de una red para clasificarlos, tales como el número de capas, el tipo de las capas, que pueden ser ocultas o visibles, de entrada o de salida y la direccionalidad de las conexiones de las neuronas.
- Según su algoritmo de aprendizaje. Como la red aprende los patrones, podemos distinguir como características, si es supervisada, no supervisada, competitiva o por refuerzo.

4.2. Redes Neuronales Convolucionales (CNN)

Las CNN [45] poseen una similitud amplia con respecto a las redes neuronales ordinarias. Este tipo de redes permiten comprender el contenido de las imágenes realizando tareas de reconocimiento, segmentación, detección y recuperación de imágenes [150], [151], [152], [153]. Cada neurona recibe información, realiza el procesamiento con el producto escalar y aplicación opcional de linealidad. La gran diferencia es que las CNN poseen características que permiten que la red reciba como entrada imágenes y no datos, motivo por la cual se ven evidenciados algunos cambios en su arquitectura y parámetros de la red [154].

El aprendizaje profundo o deep learning es la estructura que se aplica dentro de las capas de redes neuronales para que el entrenamiento sea mayormente complejo y, por ende, exista una mayor experiencia o entrenamiento por parte de la red neuronal. Cuantas más capas posea una red neuronal, mayor o más profundo será su aprendizaje. Cuando las máquinas hayan entrenado lo suficiente, entonces podrán llevar a cabo diferentes tipos de tareas, tales como conducir un vehículo, detectar hierba en un campo de cultivo, detectar enfermedades, inspeccionar maquinaria para identificar errores, etc.

El reconocimiento de características en el aprendizaje profundo es completamente autónomo. Aunque se debe realizar una adecuada y completa configuración para que el proceso y resultados sean efectivos, las redes neuronales de aprendizaje profundo memorizan por medio de la detección de estructuras complejas de los datos que reciben. Una red neuronal se puede entrenar con miles o millones de imágenes que contengan un objeto en particular. Este tipo de red aprende a identificar los píxeles en común de las imágenes que recibe como entradas, clasificándolos por grupos de acuerdo a las partes de dicho objeto. Cada parte con unas características de píxeles diferentes, pero que al unirlas, poseen el objeto principal como dato en común. Esto hace que la red identifique a ese objeto en el millón de imágenes de entrada y es de esta forma como la red puede identificar ese mismo objeto en cualquier otra imagen que se le dé como entrada, gracias a su entrenamiento y experiencia obtenidas inicialmente [155].

Las CNN son un tipo de modelos de aprendizaje profundo [156] y se ha demostrado que este tipo de redes son relativamente insensibles a algunas variaciones en sus entradas, sensibilidad notable en otro tipo de ANN [157]. En la Figura 4.3 se observa la arquitectura de una red neuronal convolucional.

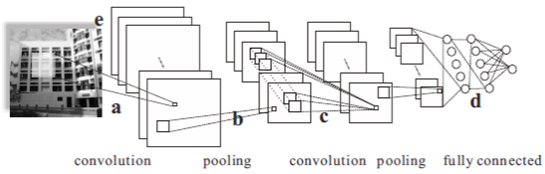


Figura 4.3: Arquitectura de Red Neuronal Convolucional (Tomado de [158]).

Convolución, agrupación, agrupación de convolución y conexión completa, hacen parte del proceso de vectorización para diferentes capas en las CNN.

4.3. La super-resolución

Es una técnica de procesamiento de imágenes y videos que permite obtener imágenes de alta resolución (HR) a partir de imágenes de entrada de baja resolución (LR), utilizando CNN y otras técnicas de aprendizaje profundo [159]. Es así que esta técnica permite aumentar la resolución de una imagen con una pérdida de información mínima. En Wang [159], se implementa la super-resolución con el uso de CNN y un modelo auto-encoder.

Dentro de las aplicaciones de la super-resolución, se encuentran los videojuegos. Este campo se está beneficiando en gran manera con estas técnicas al permitir aumentar su resolución sin dañar su contenido, además de realizar la remasterización de videojuegos antiguos, recomponiendo sus texturas a mayor resolución, aumentando la calidad del videojuego de una forma considerable [160]. En la siguiente sección se muestra el funcionamiento del algoritmo de super-resolución.

4.4. Funcionamiento de la super-resolución

Para explicar el proceso de aplicación de super-resolución a una imagen, se tomó como referencia el artículo [161], donde se propone un método de super-resolución de imagen única (SISR). Este modelo es una CNN profunda con redes residuales, salto de conexión y red en red (DCSCN). Además, emplea una técnica de muestreo ascendente, llamada interpolación bicúbica, método por el cual se realiza la interpolación horizontal y vertical con los 16 píxeles más cercanos, lo que mejora la resolución de la imagen [162]. Dicho modelo propuesto se basa en el algoritmo mostrado en la Figura 4.4 de super-resolución, el cual cumple dos importantes funciones en las que divide sus CNN.

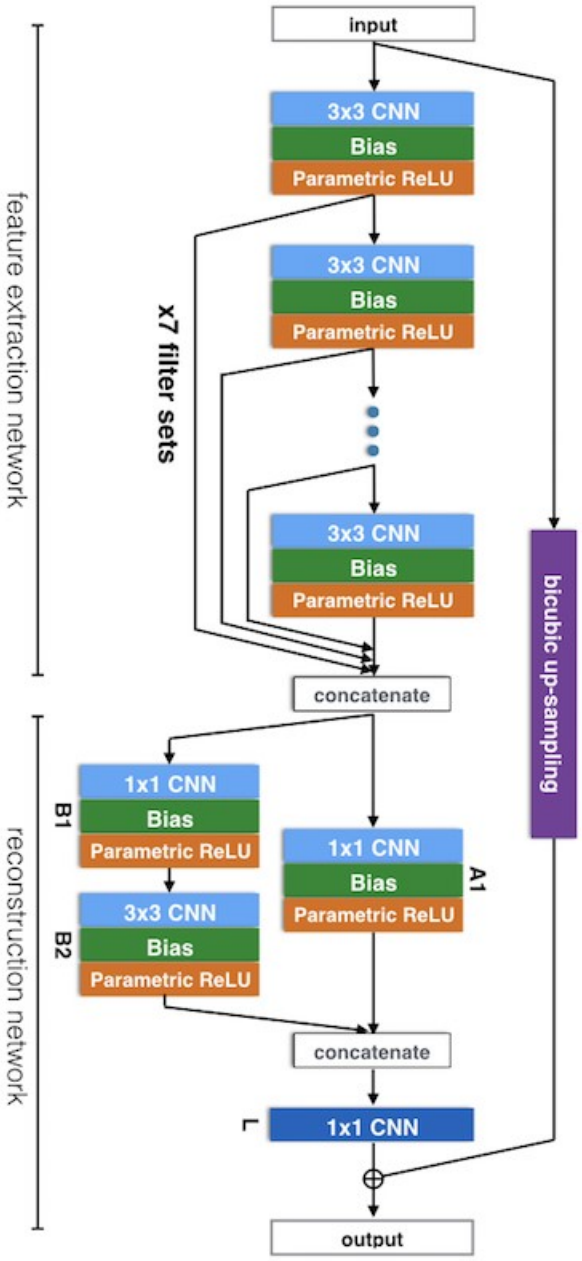


Figura 4.4: Diagrama de funcionamiento del algoritmo (Tomado de [161]).

En el primer grupo se encuentra la red de extracción de características y el segundo grupo se encarga de la reconstrucción de imagen. Como primera medida, el algoritmo recibe la imagen a procesar, que a diferencia de otros modelos de super-resolución no es una imagen muestreada sino que usa la imagen original como entrada para que la red pueda captar las características de forma eficiente. En la primera red de extracción de características se conectan en cascada siete conjuntos de CNN de 3 x 3, con polarización y unidades ReLU paramétricas. La salida de cada red se pasa a la siguiente unidad y saltan simultáneamente a la red de reconstrucción de imagen por medio de una conexión de salto, gracias a que el algoritmo también se compone de redes residuales [163].

Antes de pasar al siguiente conjunto de reconstrucción, se concatenan todas las funciones establecidas por el conjunto de redes artificiales de extracción de características y luego procede a pasar a la red de reconstrucción de imagen que se compone de un conjunto de una red de 3 x 3 y tres redes de 1 x 1. En el conjunto de redes de re-

construcción se usan dos redes de 1 x 1 después de recibir la información concatenada, pues al estar concatenada la información, su dimensión es bastante grande, razón por la que se usan estas redes (1 x 1) para reducir la dimensión de entrada antes de generar los píxeles de alta resolución. Aquí se aplican las CNN en paralelo (red en red [48]), las cuales cumplen con la tarea de reconstruir los detalles de imagen (en forma) y la última capa de CNN genera la imagen de cuatro canales (tres canales de RGB y un canal de transparencia) obteniéndose una imagen muestreada detallada en forma.

El algoritmo, por su parte, toma la misma imagen inicial de baja resolución y le aplica interpolación bicúbica, la cual se encarga de rellenar los espacios vacíos resultantes con píxeles al aumentar el tamaño físico de la imagen. Estos píxeles se rellenan con información y/o color de los píxeles más cercanos. Cabe resaltar que la interpolación bicúbica no realiza ningún proceso de alucinación, solo aumenta o disminuye la cantidad de píxeles de una imagen. Finalmente, la imagen obtenida por las capas de reconstrucción se une a la imagen muestreada construida por la interpolación bicúbica. Dicha interpolación, entonces, brinda un mejor arreglo de colores de cada píxel. Al final se obtendrá una imagen mejorada en detalles de forma y color como se ve en el ejemplo de Yamanaka en la Figura 4.5.

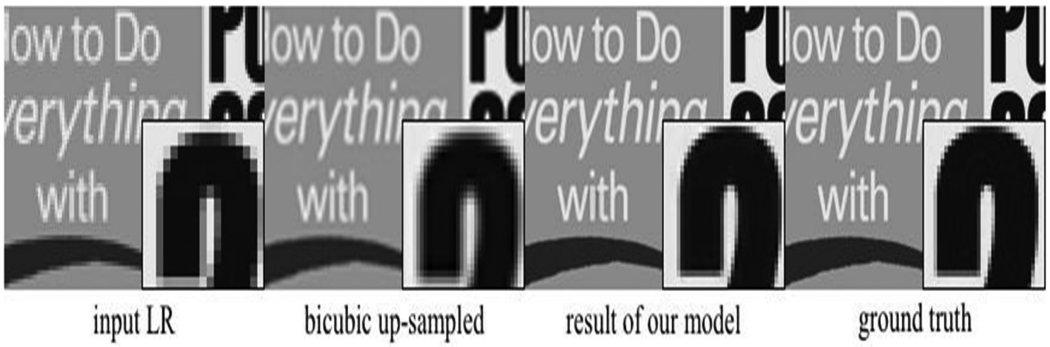


Figura 4.5: Aplicación de método DCSCN con técnica de interpolación bicúbica (Tomado de [161]).

4.5. Ejemplo de super-resolución

A continuación se muestra un ejemplo de aplicación de super-resolución a una imagen utilizando CNN para extraer características de la imagen para posteriormente aumentar su resolución. El proceso de aplicación de ANN para super-resolución en el ejemplo, se basa en la información proporcionada por Yamanaka [161].

- Algoritmos y métodos usados. El modelo implementado mostrado por Yamanaka [161] hace uso de un modelo de aprendizaje profundo conocido como CNN, junto con una red residual, conexión de salto y un sistema de red en red [48] (DCSCN).

- Procedimiento. Se usa una imagen de entrada random de baja resolución, se le aplica el método mencionado anteriormente, el cual da como resultado una imagen de alta resolución notable, con una mínima pérdida de información o características. El procedimiento realizado se muestra en la explicación de la Figura 4.4.

Este modelo (DCSCN) es una red neuronal totalmente convolucional. DCSCN realiza el trabajo de extracción de características y reconstrucción de imagen por medio de redes internas. Para su funcionamiento se conecta en cascada un conjunto de pesos CNN, sesgos y capas no lineales a la entrada. Luego, se extraen las características de la imagen local y global. Todas las salidas de las capas ocultas se conectan a la red de reconstrucción por medio de salto de conexión. Después de concatenar todas las funciones, las CNN en paralelo (red en red [48]) se utilizan para reconstruir los detalles de la imagen. La última capa de CNN genera la imagen de 4 canales y, finalmente, la imagen original con muestreo superior se estima agregando estas salidas a la imagen con muestreo superior construido por interpolación bicúbica [161].

En la Figura 4.6 se muestran algunas líneas de código donde se implementa la super-resolución:

```
1
2import os
3
4import tensorflow.compat.v1 as tf
5
6import DCSCN
7from helper import args
8os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
9
10args.flags.DEFINE_string("file", "image.jpg", "Target filename"
    )
11FLAGS = args.get()
12
13
14def main(_):
15    model = DCSCN.SuperResolution(FLAGS, model_name=FLAGS.
          model_name)
16    model.build_graph()
17    model.build_optimizer()
18    model.build_summary_saver()
19    model.init_all_variables()
20    model.load_model()
21    model.do_for_file(FLAGS.file, FLAGS.output_dir)
22
23
24    def build_conv(self, name, input_tensor, cnn_size,
          input_feature_num, output_feature_num, use_bias=False,
25                   activator=None, use_batch_norm=False,
                         dropout_rate=1.0):
26        with tf.variable_scope(name):
27            w = util.weight([cnn_size, cnn_size,
                  input_feature_num, output_feature_num],
28                            stddev=self.weight_dev, name="
                                  conv_W", initializer=self.
                                  initializer)
29
30            b = util.bias([output_feature_num], name="conv_B")
                  if use_bias else None
31            h = self.conv2d(input_tensor, w, self.cnn_stride,
                  bias=b, use_batch_norm=use_batch_norm, name=name)
32
33            if activator is not None:
34                h = self.build_activator(h, output_feature_num,
                      activator, base_name=name)
35
36            if dropout_rate < 1.0:
37                h = tf.nn.dropout(h, rate=1 - self.dropout,
                      name="dropout")
38
39...
```

Figura 4.6: Código de ejemplo (Tomado de [164]).

A continuación se muestra una imagen de entrada hacia el modelo de super-resolución de imagen única (SISR) con sus propiedades de resolución (ver Figura 4.7) y enseguida se muestra la imagen de salida resultante del aumento de resolución con sus nuevas propiedades y/o características (ver Figura 4.8):

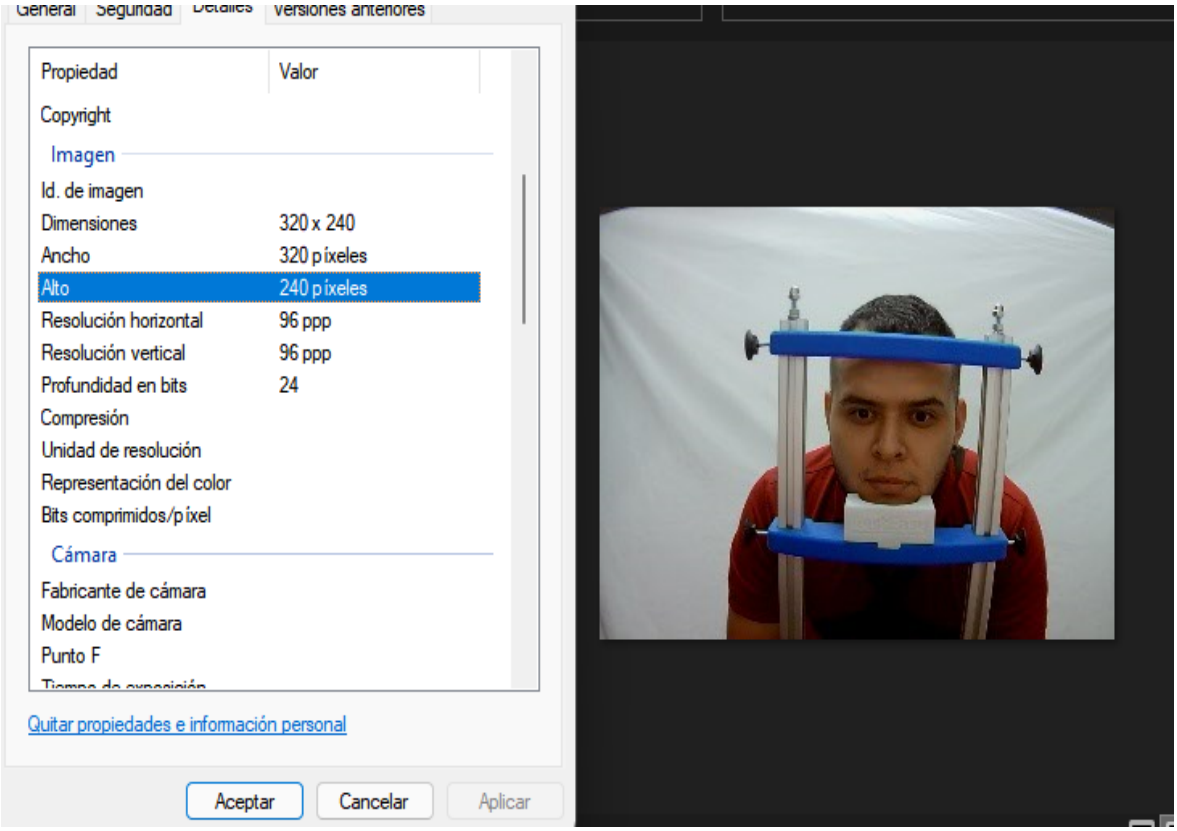


Figura 4.7: Imagen de entrada.

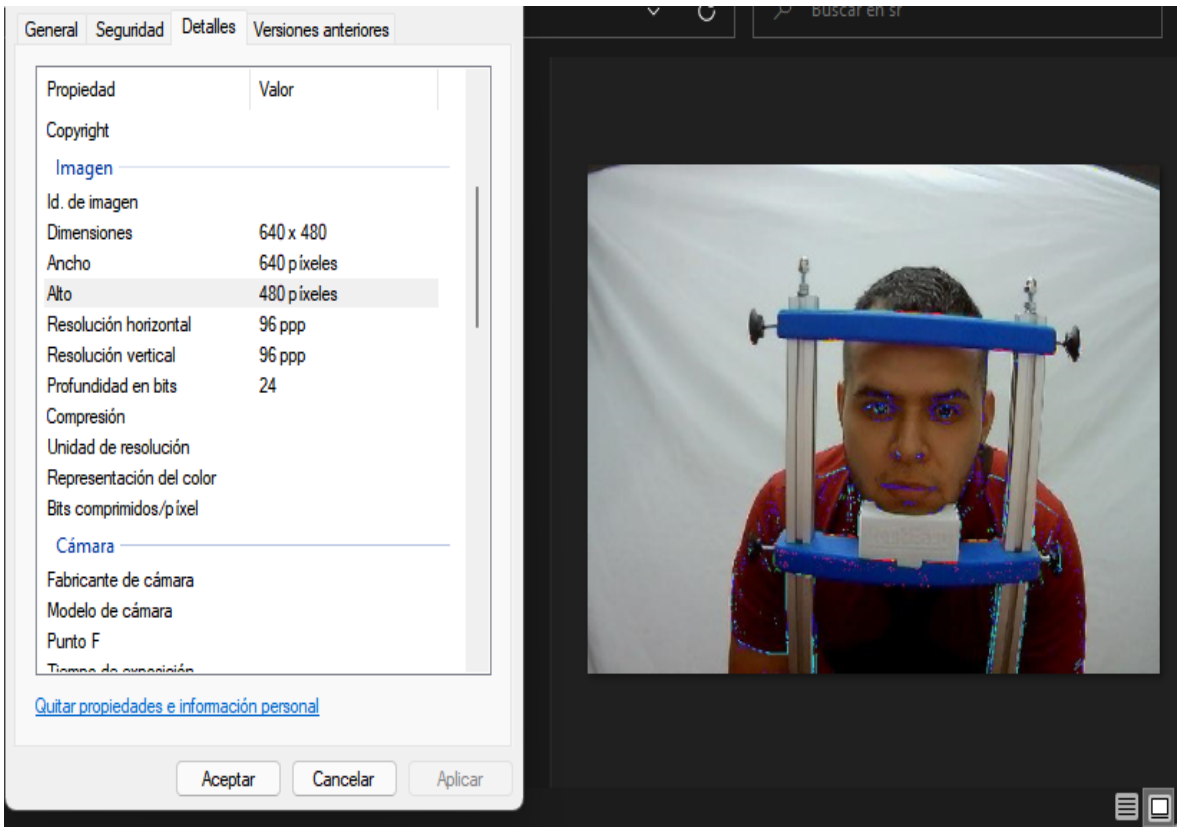


Figura 4.8: Imagen de salida.

El objetivo es comparar los dos resultados obtenidos al aplicar y no, super-resolución en las imágenes que van como entrada hacia el seguidor ocular y ver si existe un cambio notable en la exactitud y precisión en el proceso de SO de una imagen o video.

La interfaz gráfica, mencionada en el Capítulo 3, también posee una pestaña para realizar el proceso de super-resolución. En dicha pestaña se carga un archivo, se invoca a Tensorflow, se carga una imagen y se aplica super-resolución. En esta pestaña se muestra una imagen de entrada con una resolución inicial de 320 x 240 y una imagen de salida procesada con el algoritmo de super-resolución, con una resolución final de 640 x 480 como se mostró en el ejemplo anterior.

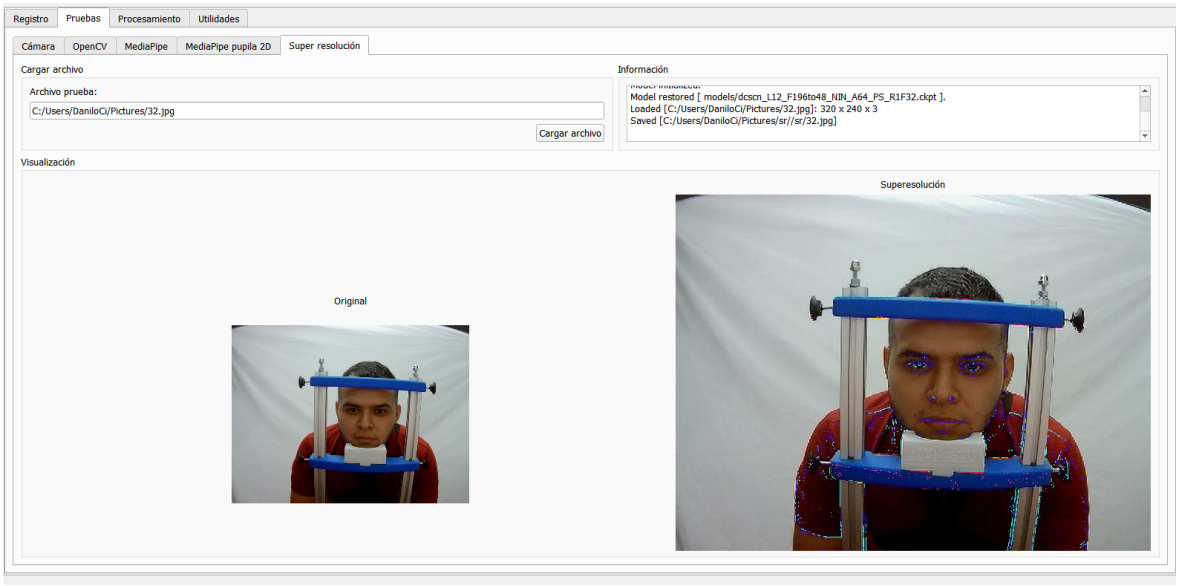


Figura 4.9: Funcionamiento de super-resolución.

Capítulo 5

Pruebas, comparaciones y resultados

En este capítulo se realiza una descripción sobre el proceso de SO llevado a cabo en esta investigación, donde se realiza la toma de pruebas a cinco participantes, mostrando su respectiva obtención de datos, el seguimiento llevado a cabo, junto con las métricas para el cálculo de exactitud y precisión y, finalmente, las diferencias en los resultados del seguidor ocular con y sin la implementación de super-resolución.

5.1. Participantes

Se incluyeron cinco personas para la toma de pruebas de la plataforma, los cuales estuvieron compuestos por compañeros del programa de ingeniería en automática industrial, familiares y amigos; personas con sentido de visión normal. En la Tabla 5.1 se muestran algunos datos importantes de cada participante.

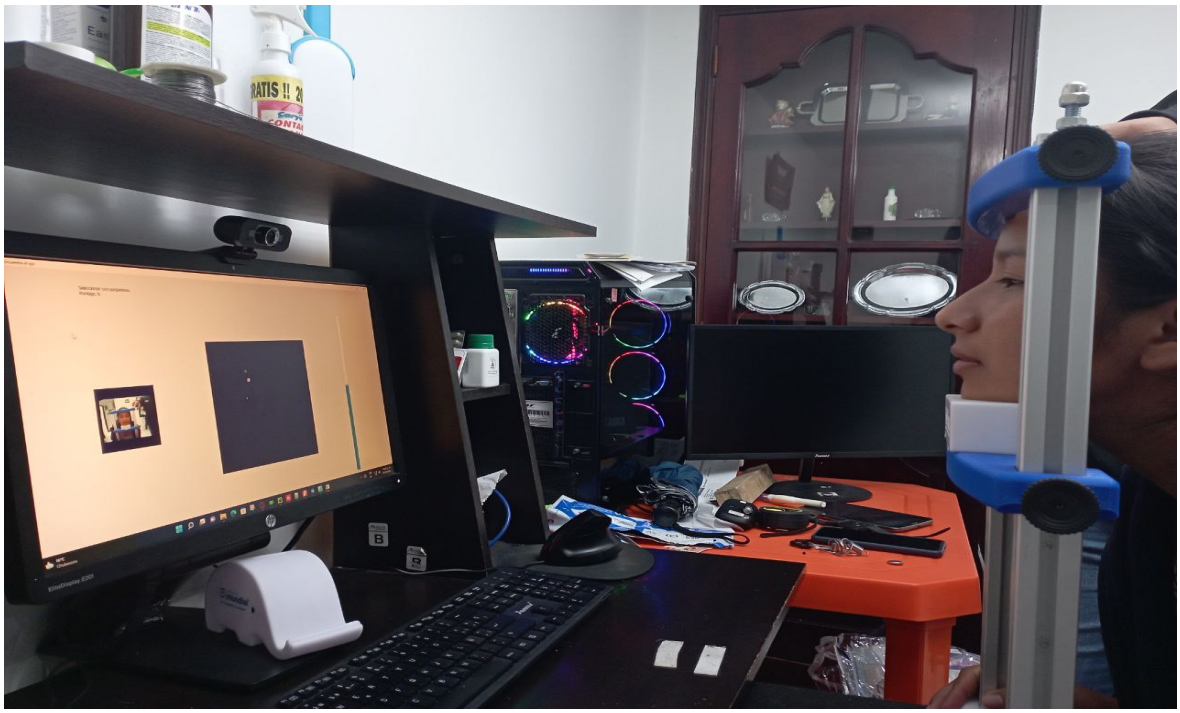


Figura 5.1: Toma de datos de la participante tres.

No. Participante	Género	Edad
1	Masculino	25
2	Femenino	15
3	Femenino	26
4	Femenino	50
5	Masculino	22

Tabla 5.1: Participantes de la toma de pruebas.

5.2. Plataforma de desarrollo

La plataforma desarrollada es tipo estática mono-cámara con soporte para frente y mentón con cámara web. Se implementó un modelo de video-oculografía basado en método de modelo 3D y se realizó la programación en Python con uso de librerías del kit de herramientas de Python llamado MediaPipe y también la librería FaceAnalyzer. Se hace uso de la biblioteca de código abierto llamada OpenCV, visión artificial, la cual ayudó a realizar las detecciones de la pupila, del framework multi plataforma orientado a objetos QT para el desarrollo de la interfaz. Phyton como lenguaje base. Además de programas para el desarrollo de piezas en 3D como FreeCAD y Autodesk Inventor, con el propósito de desarrollar los planos de la plataforma física del proyecto.

Esta plataforma permite realizar pruebas a una persona a la vez. El participante no podrá realizar movimientos de cabeza, razón por la cual se implementa el soporte para frente y mentón. Dichas partes se pueden graduar o mover de acuerdo al tamaño de la cabeza del participante. Es necesario resaltar que inicialmente se propuso usar tecnología infrarroja para el desarrollo de la plataforma, pero luego se implementó un método de toma de datos que se ajusta a las condiciones de luz sin afectar el proceso de SO, dejando el uso de esta tecnología de forma opcional.

De acuerdo al diagrama de flujo mostrado en la Figura 3.11 del Capítulo 3, se realizan los siguientes pasos para hacer SO con y sin super-resolución, en los fotogramas obtenidos por la cámara:

1) El participante ubica su cabeza entre la mentonera y el soporte para frente del estabilizador a una distancia de 90 cm de la cámara. La imagen de su rostro es capturada. 2) El participante debe mirar fijamente un punto por cada secuencia mostrada en pantalla. En total son 4 secuencias diferentes en las que se graba al participante por un lapso de 6 segundos en cada punto. 3) Los videos obtenidos por cada participante se dividen en fotogramas y son guardados en una carpeta con el nombre del participante. 4) Se genera una copia de la carpeta con los fotogramas y se envía posteriormente al algoritmo de super-resolución. 5) Se aplica super-resolución a los fotogramas recibidos y se guardan en una carpeta llamada S.R dentro de la carpeta que posee el nombre del participante. 6) Se envían los fotogramas obtenidos inicialmente ubicados en la carpeta con el nombre del participante al algoritmo de seguimiento y se realiza el proceso de seguimiento de mirada. 7) Se guardan los resultados de seguimiento con fotogramas comunes. 8) Se envían los fotogramas guardados en la carpeta S.R al algoritmo de seguimiento y se realiza el proceso de seguimiento de mirada. 9) Se guardan los resultados de seguimiento con fotogramas de super-resolución. 10) Se hace la comparación de ambos resultados obtenidos y se generan sus gráficos correspondientes.

5.3. Resultados

A continuación se muestran los resultados de los cinco participantes, donde se escoge la información del participante cuatro para realizar una muestra de resultados, explicación sobre las gráficas de seguimiento generadas y los resultados de exactitud y precisión evaluados con y sin super-resolución. Cada participante realizó cuatro secuencias de seguimiento y cada secuencia posee sus gráficas correspondientes. Mostrar gráficamente todos los resultados de las secuencias podría ser muy extenso, por lo tanto, solo se presentan los gráficos obtenidos para la secuencia dos del participante cuatro, se genera una tabla con los valores de exactitud y precisión proporcionados por el algoritmo sin uso de super-resolución. Luego, se muestran las gráficas obtenidas por el seguidor ocular con uso de super-resolución y su tabla con los datos obtenidos en precisión y exactitud. Al final se realiza la comparación de datos teniendo en cuenta la diferencia de error en la exactitud y precisión mostrada por el seguidor ocular con y sin aplicación de super-resolución.

5.3.1. Métricas

Para realizar el cálculo de precisión y exactitud se tomaron como base los artículos [165], [166] donde se aplican fórmulas de distancia euclidiana para el cálculo de la exactitud del seguidor ocular y la fórmula para el cálculo de la precisión que hace uso de la desviación estándar, la varianza y la distancia euclidiana entre cada punto. Se hace uso de las coordenadas 2D que proporciona el seguidor ocular y las coordenadas reales del punto referencia ubicado en pantalla, las cuales son diferentes para cada secuencia. En la ecuación 5.1, se muestra el cálculo de la exactitud con la fórmula de la distancia euclidiana (D), corresponde a:

$$D = \sqrt{(x - x_r)^2 + (y - y_r)^2} \tag{5.1}$$

Donde x corresponde a la coordenada en x del punto en el plano obtenido por el seguidor ocular y x_r corresponde a la coordenada real en el mismo eje de dicho punto. De la misma forma para y y y_r. Para el cálculo de la precisión se hace uso de la fórmula de varianza, dada por:

$$\sigma^2 = \frac{1}{N} * \sum_{i=1}^{N} di^2 \tag{5.2}$$

Donde la varianza (σ^2) de un conjunto de muestras de datos es una medida de la dispersión alrededor del valor de referencia. Donde N es el numero de muestras y di es una medida de distancia entre la muestra individual obtenida, en este caso, por el seguidor ocular y la medida real correspondiente a las coordenadas del punto de referencia ubicado en cada secuencia. La desviación estándar (σ) de las muestras obtenidas, se define como:

$$\sigma = \sqrt{\frac{1}{N} * \sum_{i=1}^{N} d^2} \tag{5.3}$$

La cual corresponde igualmente a la fórmula de precisión implementada en esta investigación y está dada por:

$$\sigma = \sqrt{\frac{(\sigma_x)^2 + (\sigma_y)^2)}{2}} \tag{5.4}$$

Donde,

$$\sigma_x^2 = \frac{1}{N} * \sum_{i=1}^{N} (x - x_r)^2 \tag{5.5}$$

Y,

$$\sigma_y^2 = \frac{1}{N} * \sum_{i=1}^{N} (y - y_r)^2 \tag{5.6}$$

Finalmente, cuando se tiene el sumatorio de la distancia euclidiana entre los valores obtenidos por el seguidor ocular y los valores reales del punto de referencia, se calcula dicha fórmula de precisión.

Para calcular en grados visuales, los valores obtenidos en píxeles, se hace uso de la función proporcionada en [167], dada como:

$$\text{Fc} \ = \frac{\tan^{-1}\left(\frac{0{,}5*h}{d}\right)}{0,5 \cdot r} \tag{5.7}$$

Donde h corresponde a la altura vertical de la pantalla, d a la distancia entre el monitor y el participante y r a la resolución vertical de la pantalla. Al hacer el cálculo de este, se obtiene el factor de conversión (Fc), el cual se multiplica por cada valor en píxeles para obtener los valores en grados.

5.3.2. Datos obtenidos

En la Tabla 5.2 se muestran los resultados de seguimiento realizados a los participantes, incluyendo los resultados del participante cuatro, dichos resultados se dan en forma de precisión y exactitud. A continuación se muestran los dos gráficos correspondientes a la secuencia dos del participante cuatro.

- Punto de Referencia. En la Figura 5.2 se obtiene el punto de referencia que se generó para la secuencia dos. Dicho punto posee coordenadas 2D reales ubicadas en coordenadas (220, 220). Se muestran los diferentes puntos detectados en pantalla por el algoritmo de seguimiento ocular. Se observa cómo al inicio del seguimiento, el algoritmo realiza la captura de datos con la mirada lejana al punto de referencia y a medida que el participante va encontrando el punto proporcionado en pantalla, el algoritmo realiza dicha detección.

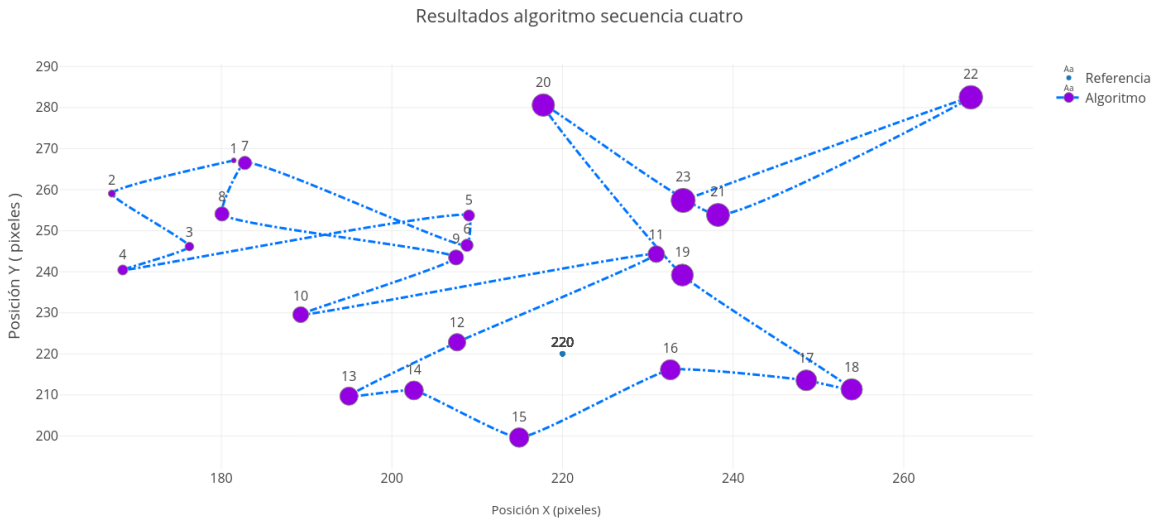


Figura 5.2: Punto de referencia y valores obtenidos en cada muestra para la secuencia dos del participante cuatro.

- Comparación de seguimiento con respecto a la referencia. La Figura 5.3 describe el proceso de seguimiento, las coordenadas (en grados) de los puntos obtenidos por el algoritmo, se acercan al valor del punto de referencia. La figura muestra cómo la evolución del error tiende a cero (0). Se quitan los primeros puntos de muestreo, donde el participante aún no tiene fija la mirada sobre el punto de referencia.

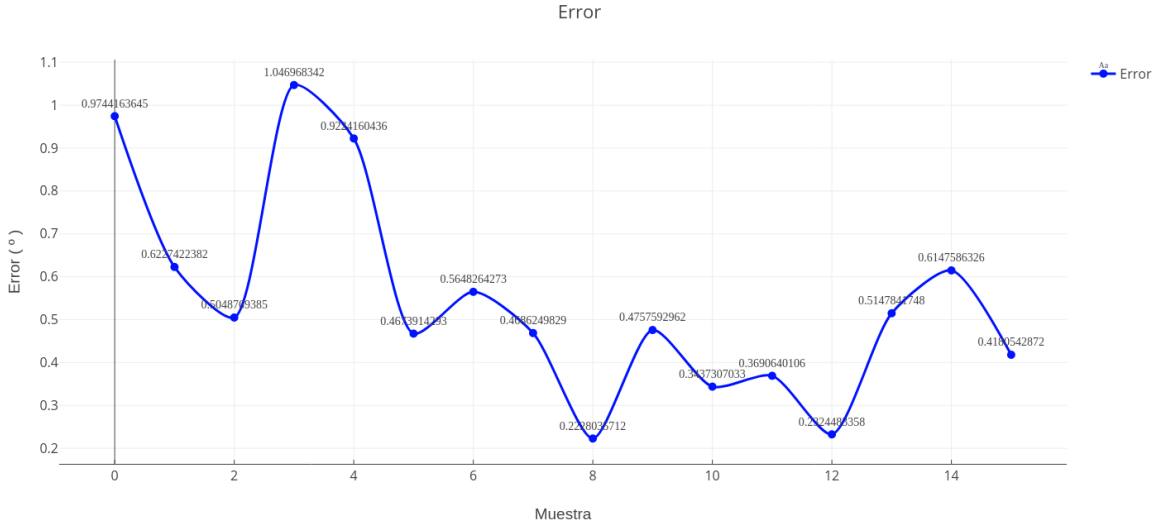


Figura 5.3: Valores obtenidos.

De acuerdo a la ecuación 5.1 mostrada al inicio de esta sección, para calcular la exactitud, se reemplaza en la ecuación de distancia euclidiana los valores en X y en Y, correspondientes a los puntos detectados por el seguidor ocular y se calcula el promedio de dichas distancias para los 16 puntos obtenidos en cada secuencia, tanto con como sin super-resolución. A continuación se muestra la Tabla 5.2 con los valores en exactitud y precisión obtenidos por el seguidor ocular sin el uso de super-resolución. En dicha tabla se pasan los valores con coordenadas en píxeles a valores en grados, haciendo uso de la función mostrada en la ecuación 5.3.1.

Seguimiento sin super resolución								
	Secuencia 1		Secuencia 2		Secuencia 3		Secuencia 4	
Participante	Exactitud	Precisión	Exactitud	Precisión	Exactitud	Precisión	Exactitud	Precisión
1	1,065	0,851	1,222	0,960	1,612	1,348	3,005	2,449
2	1,564	1,504	2,450	1,887	1,562	1,689	2,377	2,174
3	2,289	1,915	4,000	3,572	3,472	2,710	5,412	4,617
4	2,910	2,217	0,973	0,915	1,533	1,181	2,805	2,045
5	2,915	2,234	1,870	1,606	2,075	1,534	2,485	1,948

Tabla 5.2: Resultados de SO sin super-resolución (Fuente propia).

A continuación se muestran las gráficas de SO del participante cuatro, donde se realiza la aplicación de super-resolución a sus imágenes de entrada al algoritmo 3D.

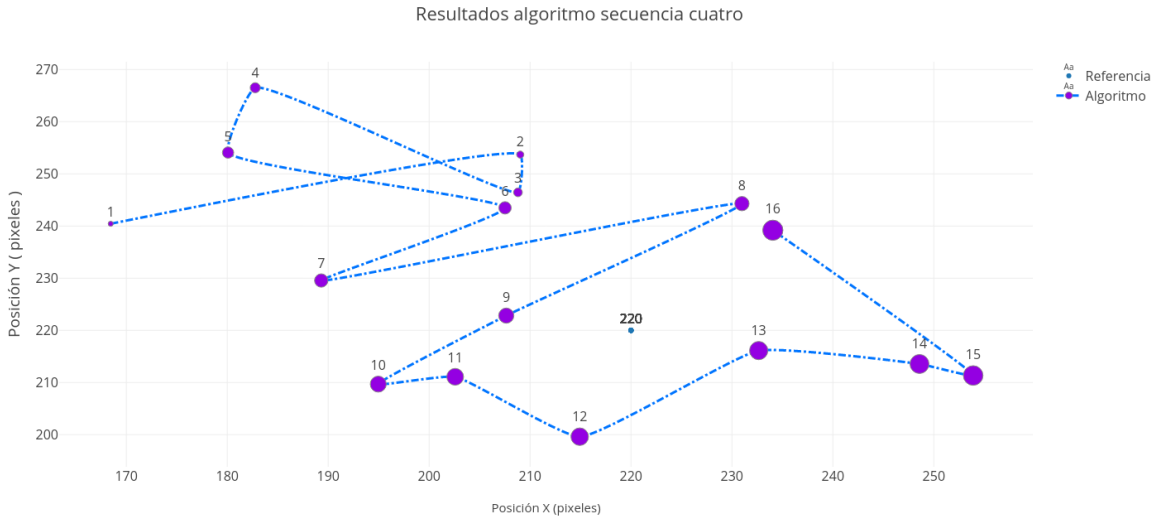


Figura 5.4: Punto de referencia de la secuencia dos del participante cuatro junto con las muestras de seguimiento con uso de super-resolución.

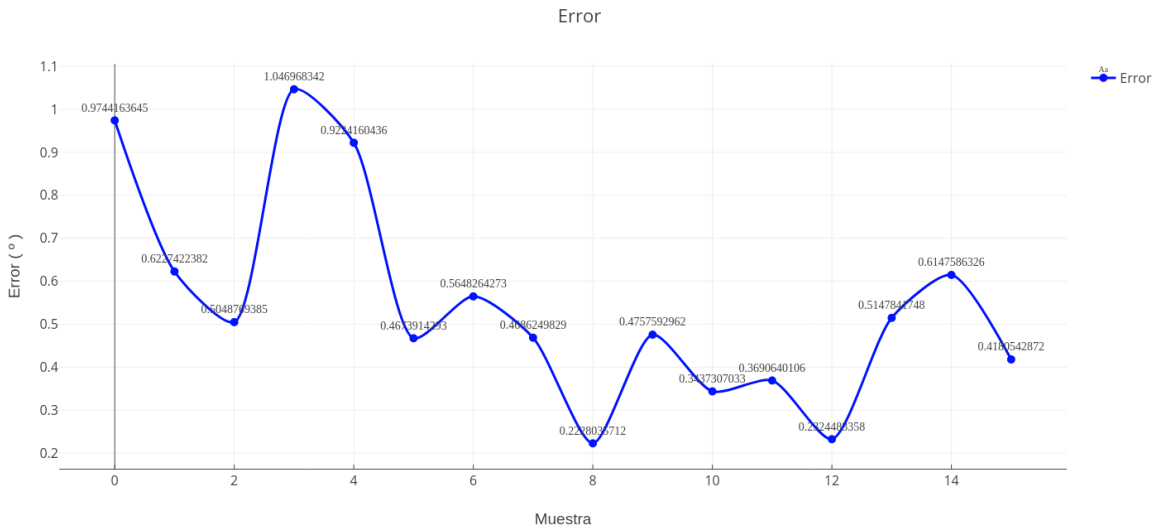


Figura 5.5: Valores obtenidos de la secuencia dos del participante cuatro con super-resolución.

A continuación se muestra la Tabla 5.3 con los valores en exactitud y precisión obtenidos por el seguidor ocular con el uso de super-resolución:

Seguimiento con super resolución								
	Secuencia 1		**Secuencia 2**		**Secuencia 3**		**Secuencia 4**	
Participante	**Exactitud**	**Precisión**	**Exactitud**	**Precisión**	**Exactitud**	**Precisión**	**Exactitud**	**Precisión**
1	1,072	0,867	0,952	1,018	1,612	1,444	3,005	2,624
2	1,795	1,504	3,476	1,869	1,462	1,689	2,645	2,155
3	2,441	2,022	4,000	3,828	3,472	2,903	5,111	4,951
4	3,048	2,288	0,976	0,944	1,512	1,173	2,805	2,120
5	3,031	2,396	1,758	1,665	2,209	1,615	2,531	1,988

Tabla 5.3: Resultados de SO con super-resolución (Fuente propia).

5.3.3. Comparación de resultados

De acuerdo a los valores obtenidos en la Tabla 5.2 y en la Tabla 5.3, se genera la Tabla 5.4 donde se muestra el error o diferencia entre la precisión y exactitud obtenidos por el seguidor ocular con y sin la implementación de super-resolución.

Participante	Secuencia	Medición	Valor sin super resolución	Valor con super resolución	Diferencia
1	Secuencia 1	Exactitud	1,065	1,072	0,007
		Precisión	0,851	0,867	0,016
	Secuencia 2	Exactitud	1,222	0,952	0,270
		Precisión	0,960	1,018	0,057
	Secuencia 3	Exactitud	1,612	1,612	0,000
		Precisión	1,348	1,444	0,096
	Secuencia 4	Exactitud	3,005	3,005	0,000
		Precisión	2,449	2,624	0,175
2	Secuencia 1	Exactitud	1,564	1,795	0,232
		Precisión	1,504	1,504	0,000
	Secuencia 2	Exactitud	2,450	3,476	1,025
		Precisión	1,887	1,869	0,018
	Secuencia 3	Exactitud	1,562	1,462	0,101
		Precisión	1,689	1,689	0,000
	Secuencia 4	Exactitud	2,377	2,645	0,268
		Precisión	2,174	2,155	0,018
3	Secuencia 1	Exactitud	2,289	2,441	0,152
		Precisión	1,915	2,022	0,107
	Secuencia 2	Exactitud	4,000	4,000	0,000
		Precisión	3,572	3,828	0,255
	Secuencia 3	Exactitud	3,472	3,472	0,000
		Precisión	2,710	2,903	0,194
	Secuencia 4	Exactitud	5,412	5,111	0,301
		Precisión	4,617	4,951	0,334
4	Secuencia 1	Exactitud	2,910	3,048	0,138
		Precisión	2,217	2,288	0,072
	Secuencia 2	Exactitud	0,973	0,976	0,003
		Precisión	0,915	0,944	0,029
	Secuencia 3	Exactitud	1,533	1,512	0,020
		Precisión	1,181	1,173	0,008
	Secuencia 4	Exactitud	2,805	2,805	0,000
		Precisión	2,045	2,120	0,074
5	Secuencia 1	Exactitud	2,915	3,031	0,116
		Precisión	2,234	2,396	0,162
	Secuencia 2	Exactitud	1,870	1,758	0,112
		Precisión	1,606	1,665	0,059
	Secuencia 3	Exactitud	2,075	2,209	0,134
		Precisión	1,534	1,615	0,080
	Secuencia 4	Exactitud	2,485	2,531	0,046
		Precisión	1,948	1,988	0,040

Tabla 5.4: Comparación de SO con uso de super-resolución (Fuente propia).

La tabla anterior muestra que existe una variación mínima con respecto a la precisión y exactitud de SO con respecto al uso de super-resolución.

Capítulo 6

Discusión, conclusiones y trabajos futuros

En este capítulo se realiza una comparación entre los datos obtenidos durante el experimento y los reportados en la literatura, posteriormente se presentan las conclusiones y algunas recomendaciones para trabajos futuros.

6.1. Discusión

El sistema mostró una precisión, sin uso de super-resolución con valores entre 0,851° y 4,617°. Y con uso de super-resolución se obtuvieron valores entre 0,867° y 4,951°. Para la parte de exactitud, el sistema mostró una exactitud sin uso de super-resolución con valores entre 0,973° y 5,412°. Y con uso de super-resolución se obtuvieron valores entre 0,952° y 5,111°.

De acuerdo a la exactitud mostrada en los artículos sobre tipos de plataformas que descritas en el Capítulo 2. Estos alcanzan una exactitud entre 1 y 2 y hasta 3°, lo que indica que el algoritmo de seguimiento implementado maneja un nivel similar de exactitud e inclusive mostrando niveles mayores que estos, pero también posee problemas con variaciones considerables al obtener bajos niveles de exactitud en comparación con los encontrados en la literatura, pues la exactitud puede llegar hasta los 5°.

Se entiende que los altos niveles de exactitud pueden deberse al uso de las funciones proporcionadas por MediaPipe, ya que esta proporciona funciones bastante robustas que permiten realizar adecuados procesos de seguimiento y detección. El hecho de que haya variaciones muy altas en cuanto a exactitud podría deberse al problema de proyección de N puntos y a la capacidad de la cámara para tomar imágenes por cada fragmento de tiempo, características que posee una cámara u otra de acuerdo a sus capacidades y su precio.

El seguidor ocular implementado posee niveles de exactitud, cercanos a los mostrados por Höffner [140], esto puede deberse a que en esta investigación se realiza un proceso con algunas similitudes a las mostradas en el método de Höffner [140], pues este, también hace uso de un modelo geométrico 3D y usa un modelo canónico que a diferencia del implementado en esta investigación, usa solo 68 puntos. El proceso de seguimiento de Höffner [140] difiere también de esta investigación en que implementa un proceso de tratamiento de imágenes para la obtención de la pupila, por medio de un método de gradiente alto, y en este proyecto se hace uso de MediaPipe, usada en esta investigación

para encontrar la pupila, pues se cree que esta posee mayor robustez en cuanto a la detección.

Se obtuvo que las diferencias entre el algoritmo sin super-resolución y con super-resolución estuvieron en rangos desde 0° hasta 1,025°, lo cual podría dar a entender que la resolución llega a un punto donde no incide en la precisión y en la exactitud, lo cual implica entonces que quizá no es necesario hacer inversiones grandes en cámaras de alta resolución, sino buscar mejorar otros aspectos en el seguimiento.

También cabe la posibilidad de que haya influencia en la aplicación de super-resolución para el mejoramiento en términos de seguimiento, aumentando la capacidad de cómputo, pues en el desarrollo de esta investigación, MediaPipe proporcionó aproximadamente 30 fotogramas por segundo (fps), pero al subir la resolución a 1280 x 720, MediaPipe proporcionó solo 10 cuadros por segundo, dando a entender que si se sube la resolución, se necesita mayor potencia de cómputo.

Dentro del desarrollo del proyecto, la aplicación de super-resolución no tuvo un efecto positivamente notable, pero quizá al usar una resolución considerablemente grande y con una capacidad de cómputo equivalente, se obtengan resultados diferentes. En este caso, para la investigación se realizó el trabajo con dos resoluciones controladas que no requirieron una potencia computacional alta.

Dentro de la investigación, se tenía inicialmente una resolución de 320 x 240 y se obtuvieron 30 fotogramas. Al subir la resolución a 640 x 480, disminuyeron algunos fotogramas porque el algoritmo debe procesarlos y mientras procesa no puede realizar la captura de otros fotogramas. Entonces, si a medida que aumentamos resolución, no aumentamos capacidad computacional, mientras que el algoritmo está procesando un fotograma, la persona puede cambiar la dirección de mirada y se puede dar una pérdida de información importante.

Para términos de esta investigación, la mayor resolución con la que se trabajó fue de 640 x 480, pero en este momento existen resoluciones hasta de 2K. Entonces queda mucho por probar en futuros trabajos de investigación. Otro aspecto importante a tener en cuenta es que es difícil acceder a una tarjeta de vídeo que permita hacer el procesamiento con resoluciones altas, pues estas tarjetas mundialmente presentan un costo elevado.

6.2. Conclusiones

- El uso de una plataforma estática facilita el procesamiento de la información, ya que el usuario, al mantener su cabeza sin movimiento, permite al algoritmo realizar de una forma mucho más sencilla la detección de rostro, ojos y pupila y, por ende, el seguimiento de la mirada. Sin embargo, el problema de este es que el usuario no tendría la libertad de movimiento con la cabeza, creando una limitante para algunas aplicaciones específicas. A pesar de que el algoritmo de seguimiento 3D implementado es robusto en cuanto a movimientos fuertes de cabeza, la plataforma tipo estática tuvo un papel más importante en el momento de realizar las pruebas de super-resolución, evitando que la persona mueva su cabeza hacia atrás, cambiando la distancia a la que se encuentra el rostro de la persona con respecto a la cámara y evitando afectar el punto del eje óptico, pues

entre más alejada este la pantalla, más desplazamiento va a haber con respecto a los grados del ojo.

- Se debieron realizar análisis o procesamientos de videos cortos, con aproximadamente 5 segundos de duración, ya que la implementación de la super-resolución necesita una potencia de cómputo elevada, ya que al ser una red convolucional, esta comienza a expandirse y necesita bastante espacio, velocidad y capacidad de cómputo para aplicar el proceso a los fotogramas obtenidos en cada video. De acuerdo al proceso llevado a cabo en esta investigación al aplicar super-resolución, se procesa un fotograma aproximadamente cada 40 segundos a un minuto. En esta investigación se tomaron aproximadamente 140 fotogramas por secuencia, siendo 4 secuencias por cada participante y 5 participantes en total. De esta forma, la aplicación de super-resolución a los vídeos de los 5 participantes equivaldría aproximadamente a un día de procesamiento. Se evidencia entonces que el proceso requiere bastante potencia computacional y tiempo, además de que podría generar un desgaste significativo en los equipos si se practica frecuentemente.

- De acuerdo a los resultados obtenidos al realizar una implementación basada en SO y la aplicación de super-resolución a las imágenes procesadas, se permite establecer que la super-resolución no genera una incidencia significativa con respecto a precisión y exactitud del SO basado en modelo geométrico 3D, dado que las diferencias de seguimiento con y sin aplicación de super-resolución tienden a tener valores pequeños.

- A pesar de la existencia de una amplia documentación respecto a seguidores oculares, en este trabajo de grado se observó que pocos de estos explicaban en detalle el proceso de implementación del mismo, lo que hizo necesario una búsqueda intensiva y uso de bastantes artículos relacionados para entrelazar datos que permitieran llevar a cabo una adecuada implementación. Existen grandes cantidades de artículos relacionados con el SO, pero la mayoría hace énfasis en la descripción teórica del seguimiento ocular y en los resultados obtenidos, dando menor importancia a la implementación física y técnica del seguidor en cuestión.

6.3. Trabajos futuros

1. Comparación de la exactitud y precisión con Redes Neuronales Convolucionales (CNN) en un sistema para movimiento libre de la cabeza.

2. Implementación de seguimiento ocular buscando un modelo o tipo de cámara con características diferentes y que principalmente ofrezca otro tipo de beneficios en cuanto a toma de datos.

3. Profundización en la implementación de seguidores oculares con la combinación de características de los diferentes métodos como modelo 3D, apariencia, forma u otros.

4. En la literatura también se reportan seguidores oculares basados en redes neuronales implementadas en métodos de apariencia. Se recomienda ver si el uso de estas, proporciona mejoras o no, en el proceso de seguimiento ocular.

Bibliografía

[1] A. J. Larrazabal, C. E. García Cena, and C. E. Martínez, "Video-oculography eye tracking towards clinical applications: A review," *Computers in Biology and Medicine*, vol. 108, no. February, pp. 57–66, 2019. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2019.03.025

[2] R. V. Kenyon, "a Soft Contact Measuring Lens Search Coil for," *Main*, pp. 0–4, 1985.

[3] S. Hewson and R. White, "Win-win organisation for psychology services: Why service managers really need the professional group," *Clinical Psychology*, no. 34, pp. 11–14, 2004.

[4] A. Kar and P. Corcoran, "A review and analysis of eye-gaze estimation systems, algorithms and performance evaluation methods in consumer platforms," *IEEE Access*, vol. 5, no. c, pp. 16 495–16 519, 2017.

[5] T. Berger, "Using eye-tracking to for analyzing case study materials," pp. 304–315, 2019. [Online]. Available: https://doi.org/10.1016/j.ijme.2019.05.002

[6] D. Kupas, B. Harangi, G. Czifra, and G. Andrassy, "Decision support system for the diagnosis of neurological disorders based on gaze tracking," *International Symposium on Image and Signal Processing and Analysis, ISPA*, no. Ispa, pp. 37–40, 2017.

[7] V. Sundstedt, "Gazing at games: An introduction to eye tracking control," pp. 1–114, 2012.

[8] C. H. Morimoto and M. R. Mimica, "Eye gaze tracking techniques for interactive applications," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, 2005.

[9] C. C. Lai, S. W. Shih, and Y. P. Hung, "Hybrid method for 3-D gaze tracking using glint and contour features," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 1, pp. 24–37, 2015.

[10] A. Villanueva, R. Cabeza, and S. Porta, "Eye tracking: Pupil orientation geometrical modeling," *Image and Vision Computing*, vol. 24, no. 7, pp. 663–679, 2006.

[11] D. B. B. Liang and L. K. Houi, "Non-intrusive eye gaze direction tracking using color segmentation and hough transform," *ISCIT 2007 - 2007 International Symposium on Communications and Information Technologies Proceedings*, pp. 602–607, 2007.

[12] A. Villanueva and R. Cabeza, "Models for gaze tracking systems," *Eurasip Journal on Image and Video Processing*, vol. 2007, 2007.

[13] Y. Sugano, Y. Matsushita, and Y. Sato, "Calibration-free gaze sensing using saliency maps," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2667–2674, 2010.

[14] E. T. Wong, S. Yean, Q. Hu, B. S. Lee, J. Liu, and R. Deepu, "Gaze Estimation Using Residual Neural Network," *2019 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019*, pp. 411–414, 2019.

[15] R. Fatima, A. Usmani, and Z. Zaheer, "Eye movement based human computer interaction," *2016 3rd International Conference on Recent Advances in Information Technology, RAIT 2016*, pp. 489–494, 2016.

[16] J. Kuruvilla, D. Sukumaran, A. Sankar, and S. P. Joy, "A review on image processing and image segmentation," *Proceedings of 2016 International Conference on Data Mining and Advanced Computing, SAPIENCE 2016*, pp. 198–203, 2016.

[17] K. Singh, A. Seth, H. S. Sandhu, and K. Samdani, "A comprehensive review of convolutional neural network based image enhancement techniques," *2019 IEEE International Conference on System, Computation, Automation and Networking, ICSCAN 2019*, pp. 1–6, 2019.

[18] N. Aloysius and M. Geetha, "A review on deep convolutional neural networks," *Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017*, vol. 2018-Janua, pp. 588–592, 2018.

[19] A. Ramírez-Vega, "Eye-tracking: una técnica se seguimiento de la mirada utilizada en la validación de unidades de aprendizaje," no. May, 2012. [Online]. Available: http://reposital.cuaed.unam.mx:8080/jspui/handle/123456789/3473

[20] B. T. Carter and S. G. Luke, "Best practices in eye tracking research," *International Journal of Psychophysiology*, vol. 155, pp. 49–62, 2020. [Online]. Available: https://doi.org/10.1016/j.ijpsycho.2020.05.010

[21] M. G. Eide, I. Heldal, C. G. Helgesen, G. Birkeland Wilhelmsen, R. Watanabe, A. Geitung, H. Soleim, and C. Costescu, "Eye-tracking complementing manual vision screening for detecting oculomotor dysfunction," *2019 7th E-Health and Bioengineering Conference, EHB 2019*, no. February, pp. 3–8, 2019.

[22] R. G. Bozomitu, A. Păsăricã, V. Cehan, C. Rotariu, and H. Costin, "Methods of control improvement in an eye tracking based human-computer interface," *2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging, SIITME 2017 - Proceedings*, vol. 2018-Janua, pp. 300–303, 2018.

[23] A. Kar, S. Member, and P. C. Fellow, "GazeVisual – A Graphical Software Tool for Performance Evaluation of Eye Gaze Estimation Systems," *2018 IEEE Games, Entertainment, Media Conference (GEM)*, pp. 1–9, 2018.

[24] L. Leveque, H. Bosmans, L. Cockmartin, and H. Liu, "State of the art: Eye-Tracking studies in medical imaging," *IEEE Access*, vol. 6, pp. 37 023–37 034, 2018.

[25] L. Goel, "A 2D illumination based adaptive gaze tracking approach for varied head orientations," *2015 International Conference on Computing for Sustainable Global Development, INDIACom 2015*, pp. 1420–1423, 2015.

[26] M. Z. C. Azemin, M. I. M. Tamrin, and A. A. Arshad, "Validation of low-cost eye tracking setup for smooth pursuit application," *ICOS 2014 - 2014 IEEE Conference on Open Systems*, pp. 123–127, 2014.

[27] X.-z. Gao, *Advances in Intelligent Systems and Computing 1086 Advances in Computational Intelligence and Communication Technology*, 2019.

[28] Y. Abdrabou, M. Mostafa, M. Khamis, and A. Elmougy, "Calibration-free Text Entry using Smooth Pursuit Eye Movements," pp. 2–6.

[29] M. Liu, Y. Li, and H. Liu, "3D Gaze Estimation for Head-Mounted Eye Tracking System with Auto-Calibration Method," *IEEE Access*, vol. 8, pp. 104 207–104 215, 2020.

[30] I. Rigas, H. Raffle, and O. V. Komogortsev, "Hybrid PS-V Technique: A Novel Sensor Fusion Approach for Fast Mobile Eye-Tracking with Sensor-Shift Aware Correction," *IEEE Sensors Journal*, vol. 17, no. 24, pp. 8356–8366, 2017.

[31] H. M. Stridh, W. Rosengren, and M. Nystr, "A robust method for calibration of eye tracking data recorded during nystagmus," 2019.

[32] J. Komulainen, "Iris and Periocular Biometric Recognition," *Iris and Periocular Biometric Recognition*, no. August, 2017.

[33] S. Wang, J. Wang, H. Peng, S. Gao, and D. He, "A New Calibration-Free Gaze Tracking Algorithm Based on DE-SLFA," pp. 380–384, 2016.

[34] C. C. Lai, Y. T. Chen, K. W. Chen, S. C. Chen, S. W. Shih, and Y. P. Hung, "Appearance-based gaze tracking with free head movement," *Proceedings - International Conference on Pattern Recognition*, vol. 1, pp. 1869–1873, 2014.

[35] W. Li, Q. Dong, H. A. O. Jia, S. Zhao, Y. Wang, L. I. Xie, Q. Pan, F. Duan, T. Liu, and S. Member, "Training a Camera to Perform Long-Distance Eye Tracking by Another Eye-Tracker," *IEEE Access*, vol. 7, pp. 155 313–155 324, 2019.

[36] F. Lu, T. Okabe, Y. Sugano, and Y. Sato, "Learning gaze biases with head motion for head pose-free gaze estimation," *Image and Vision Computing*, vol. 32, no. 3, pp. 169–179, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2014.01.005

[37] Y. Xia, H. Yu, and F. Y. Wang, "Accurate and robust eye center localization via fully convolutional networks," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 5, pp. 1127–1138, 2019.

[38] D. Lian, L. Hu, W. Luo, Y. Xu, L. Duan, J. Yu, and S. Gao, "Multiview multitask gaze estimation with deep convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 10, pp. 3010–3023, 2019.

[39] X. Li, Z. L. Li, and J. L. Qin, "An improved gaze tracking technique based on eye model," *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, pp. 7286–7291, 2014.

[40] J. R. Khonglah and A. Khosla, "A low cost webcam based eye tracker for communicating through the eyes of young children with ASD," *Proceedings on 2015 1st International Conference on Next Generation Computing Technologies, NGCT 2015*, no. September, pp. 925–928, 2016.

[41] A. George and A. Routray, "Real-time Eye Gaze Direction Classification Using Convolutional Neural Network," no. October 2018, 2016.

[42] H. Lu, Y. Li, T. Uemura, Z. Ge, X. Xu, L. He, S. Serikawa, and H. Kim, "FD-CNet: filtering deep convolutional network for marine organism classification," *Multimedia Tools and Applications*, vol. 77, no. 17, pp. 21 847–21 860, 2018.

[43] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2014.

[44] A. Toshev and C. Szegedy, "DeepPose: Human pose estimation via deep neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1653–1660, 2014.

[45] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F. F. Li, "Large-scale video classification with convolutional neural networks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[46] N. Wang and D. Y. Yeung, "Learning a deep compact image representation for visual tracking," *Advances in Neural Information Processing Systems*, pp. 1–9, 2013.

[47] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11133 LNCS, pp. 63–79, 2019.

[48] D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, "Learning a deep convolutional network for image super-resolution," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8692, no. September, pp. 184–199, 2014.

[49] Institute of Electrical and Electronics Engineers, "2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA 2017) : March 10-12, 2017, Beijing, China." pp. 721–724, 2017.

[50] E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[51] J. Xin, X. Du, Y. Shi, J. Zhang, and D. Liu, "Robust Outdoor Vehicle Visual Tracking Based on k-Sparse Stacked Denoising Auto-Encoder," *Autonomous Vehicles*, 2020.

[52] J. S. Ren and L. Xu, "On vectorization of deep convolutional neural networks for vision tasks," *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, pp. 1840–1846, 2015.

[53] L. Dong, Y. Chen, A. Gale, and P. Phillips, "Eye Tracking Method Compatible with Dual-screen Mammography Workstation," pp. 206–211, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2016.07.013

[54] R. Siegfried, "A Deep Learning Approach for Robust Head Pose Independent Eye Movements Recognition from Videos," 2018.

[55] Y. Yin, C. Juan, J. Chakraborty, and M. P. Mcguire, "Classification of Eye Tracking Data using a Convolutional Neural Network," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 530–535, 2018.

[56] C. C. Chen and Y. S. Lin, "Study on the interactive interface design of gaze input smart TV," *Proceedings of 4th IEEE International Conference on Applied System Innovation 2018, ICASI 2018*, no. Figure 1, pp. 196–199, 2018.

[57] Y. H. Yiu, M. Aboulatta, T. Raiser, L. Ophey, V. L. Flanagin, P. zu Eulenburg, and S. A. Ahmadi, "DeepVOG: Open-source pupil segmentation and gaze estimation in neuroscience using deep learning," *Journal of Neuroscience Methods*, vol. 324, no. May, p. 108307, 2019. [Online]. Available: https://doi.org/10.1016/j.jneumeth.2019.05.016

[58] J. Lemley, S. Member, A. Kar, and S. Member, "Convolutional Neural Network Implementation for Eye-Gaze Estimation on Low-Quality Consumer Imaging Systems," *IEEE Transactions on Consumer Electronics*, vol. PP, no. c, p. 1, 2019.

[59] Duchowski, *Eye Tracking Methodology Theory and Practice*, 2003, no. 1.

[60] C. Anderson, A. M. Chang, J. P. Sullivan, J. M. Ronda, and C. A. Czeisler, "Assessment of drowsiness based on ocular parameters detected by infrared reflectance oculography," *Journal of Clinical Sleep Medicine*, vol. 9, no. 9, pp. 907–920, 2013.

[61] R. G. Lupu, F. Ungureanu, R. G. Bozomitu, and V. Cehan, "Eye tracking performance improvement," *2014 18th International Conference on System Theory, Control and Computing, ICSTCC 2014*, pp. 698–701, 2014.

[62] K. Sakurai, M. Yan, H. Tamura, and K. Tanno, "Comparison of two techniques for gaze estimation system using the direction of eyes and head," *2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016 - Conference Proceedings*, pp. 2466–2471, 2017.

[63] J. Turner, A. Bulling, and H. Gellersen, "Extending the visual field of a head-mounted eye tracker for pervasive eye-based interaction," *Eye Tracking Research and Applications Symposium (ETRA)*, vol. 1, no. 212, pp. 269–272, 2012.

[64] E. Skodras, V. G. Kanas, and N. Fakotakis, "On visual gaze tracking based on a single low cost camera," *Signal Processing: Image Communication*, vol. 36, pp. 29–42, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.image.2015.05.007

[65] F. Ungureanu, R. G. Lupu, A. Cadar, and A. Prodan, "Neuromarketing and visual attention study using eye tracking techniques," pp. 553–557, 2017.

[66] S. S. Deepika and G. Murugesan, "A novel approach for Human Computer Interface based on eye movements for disabled people," *Proceedings of 2015 IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2015*, pp. 1–3, 2015.

[67] P. Biswas and J. DV, "Eye Gaze Controlled MFD for Military Aviation," in *23rd International Conference on Intelligent User Interfaces*, ser. IUI '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 79–89. [Online]. Available: https://doi.org/10.1145/3172944.3172973

[68] P. M. Corcoran, F. Nanu, S. Petrescu, and P. Bigioi, "Real-time eye gaze tracking for gaming design and consumer electronics systems," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 347–355, 2012.

[69] C.-C. Wang, J. C. Hung, S.-N. Chen, and H.-P. Chang, "Tracking students' visual attention on manga-based interactive e-book while reading: an eye-movement approach," *Multimedia Tools and Applications*, vol. 78, pp. 4813–4834, 2018.

[70] Y. L. Chen, C. Y. Chiang, C. W. Yu, W. C. Sun, and S. M. Yuan, "Real-time eye tracking and event identification techniques for smart TV applications," pp. 63–64, 2014.

[71] K. A. Dalrymple, M. D. Manner, K. A. Harmelink, E. P. Teska, and J. T. Elison, "An examination of recording accuracy and precision from eye tracking data from toddlerhood to adulthood," *Frontiers in Psychology*, vol. 9, no. MAY, pp. 1–12, 2018.

[72] M. Barz, A. Bulling, and F. Daiber, "Computational Modelling and Prediction of Gaze Estimation Error for Head-mounted Eye Trackers," *DFKI Research Reports, RR*, vol. 1, p. 10, 2015. [Online]. Available: https://www.dfki.de/web/forschung/projekte-publikationen/publikationen-uebersicht/publikation/7619/

[73] A. H. Id, V. Peysakhovich, and C. Hurter, "Improving eye-tracking calibration accuracy using symbolic regression," pp. 1–22, 2019.

[74] E. D. Guestrin and M. Eizenman, "General theory of remote gaze estimation using the pupil center and corneal reflections," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 6, pp. 1124–1133, 2006.

[75] M. Q. Khan and S. Lee, "Gaze and Eye Tracking : Techniques and Applications in ADAS," 2019.

[76] F. N. Ibrahim, Z. M. Zin, and N. Ibrahim, "Eye Center Detection Using Combined Viola-Jones and Neural Network Algorithms," *International Symposium on Agents, Multi-Agent Systems and Robotics 2018, ISAMSR 2018*, pp. 1–6, 2018.

[77] X. Fan, K. Zheng, Y. Lin, and S. Wang, "Combining local appearance and holistic view: Dual-Source Deep Neural Networks for human pose estimation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1347–1355, 2015.

[78] J. N. Stember, H. Celik, E. Krupinski, P. D. Chang, S. Mutasa, B. J. Wood, A. Lignelli, and G. Moonis, "Eye Tracking for Deep Learning Segmentation Using Convolutional Neural Networks," pp. 597–604, 2019.

[79] Y. Ni and B. Sun, "A Remote Free-head Pupillometry Based on Deep Learning and Binocular System," *IEEE Sensors Journal*, vol. PP, no. c, p. 1, 2018.

[80] M. B. Ahmad, Saifullah, M. A. Raja, M. W. Asif, and K. Khurshid, "I-Riter: Machine learning based novel eye tracking and calibration," *I2MTC 2018 - 2018 IEEE International Instrumentation and Measurement Technology Conference: Discovering New Horizons in Instrumentation and Measurement, Proceedings*, pp. 1–5, 2018.

[81] O. Špakov and D. Miniotas, "Gaze-based selection of standard-size menu items," *Proceedings of the Seventh International Conference on Multimodal Interfaces, ICMI'05*, pp. 124–128, 2005.

[82] M. Kumar, A. Paepcke, and T. Winograd, "EyePoint: Practical pointing and selection using gaze and keyboard," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 421–430, 2007.

[83] H. Hua, P. Krishnaswamy, and J. P. Rolland, "Video-based eyetracking methods and algorithms in head-mounted displays," *Optics Express*, vol. 14, no. 10, p. 4328, 2006.

[84] A. Lanata, A. Greco, G. Valenza, and E. P. Scilingo, "Robust head mounted wearable eye tracking system for dynamical calibration," *Journal of Eye Movement Research*, vol. 8, no. 5, pp. 1–15, 2015.

[85] J. W. Lee, C. W. Cho, K. Y. Shin, E. C. Lee, and K. R. Park, "3D gaze tracking method using Purkinje images on eye optical model and pupil," *Optics and Lasers in Engineering*, vol. 50, no. 5, pp. 736–751, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.optlaseng.2011.12.001

[86] K. Takemura, K. Takahashi, J. Takamatsu, and T. Ogasawara, "Estimating 3-D point-of-regard in a real environment using a head-mounted eye-tracking system," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 4, pp. 531–536, 2014.

[87] B. Yu, H. Cong, H. Yuan, X. Liu, Q. Peng, X. Zhang, X. Xu, C. Tian, R. Yang, and S. Yang, "Preparation of Doughnut-like Nanocomposite Colloidal Crystal Particles with Enhanced Light Diffraction Using Drying Self-assembly Method," *Current Nanoscience*, vol. 11, no. 2, pp. 161–165, 2014.

[88] S. Zhai, C. Morimoto, and S. Ihde, "Manual and gaze input cascaded (MAGIC) pointing," *Conference on Human Factors in Computing Systems - Proceedings*, pp. 246–253, 1999.

[89] M. U. Ghani, S. Chaudhry, M. Sohail, and M. N. Geelani, "GazePointer: A real time mouse pointer control implementation based on eye gaze tracking," *2013 16th International Multi Topic Conference, INMIC 2013*, pp. 154–159, 2013.

[90] A. Meyer, M. Bõhme, T. Martinetz, and E. Barth, "A single-camera remote eye tracker," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4021 LNAI, pp. 208–211, 2006.

[91] C. H. Morimoto, A. Amir, and M. Flickner, "Detecting eye position and gaze from a single camera and 2 light sources," *Proceedings - International Conference on Pattern Recognition*, vol. 16, no. 4, pp. 314–317, 2002.

[92] M. Tonsen, J. Steil, Y. Sugano, and A. Bulling, "InvisibleEye," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 3, pp. 1–21, 2017.

[93] D. Beymer and M. Flickner, "Eye Gaze Tracking Using an Active Stereo Head," 2003.

[94] D.-c. Cho, S. Member, W.-s. Yap, H. Lee, I. Lee, and W.-y. Kim, "Long Range Eye Gaze Tracking System for a Large Screen," pp. 1119–1128, 2012.

[95] Y. Cheng, H. Wang, Y. Bao, and F. Lu, "Appearance-based Gaze Estimation With Deep Learning: A Review and Benchmark," no. April, pp. 1–21, 2021. [Online]. Available: http://arxiv.org/abs/2104.12668

[96] Z. Wu, S. Rajendran, T. Van As, V. Badrinarayanan, and A. Rabinovich, "EyeNet: A multi-task deep network for off-axis eye gaze estimation," *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*, pp. 3683–3687, 2019.

[97] D. Lian, Z. Zhang, W. Luo, L. Hu, M. Wu, Z. Li, J. Yu, and S. Gao, "RGBD Based Gaze Estimation via Multi-Task CNN," 2017.

[98] P. A. Punde, M. E. Jadhav, and R. R. Manza, "A study of Eye Tracking Technology and its applications," *Proceedings - 1st International Conference on Intelligent Systems and Information Management, ICISIM 2017*, vol. 2017-January, pp. 86–90, 2017.

[99] C. Ma, K.-a. Choi, B.-d. Choi, and S.-j. Ko, "Robust remote gaze estimation method based on multiple geometric transforms," 2015.

[100] T. Schneider, B. Schauerte, and R. Stiefelhagen, "Manifold alignment for person independent appearance-based gaze estimation," *Proceedings - International Conference on Pattern Recognition*, pp. 1167–1172, 2014.

[101] T. Ohno and N. Mukawa, "A free-head, simple calibration, gaze tracking system that enables gaze-based interaction," *Eye Tracking Research and Applications Symposium (ETRA)*, pp. 115–122, 2004.

[102] S. W. Shih and J. Liu, "A Novel Approach to 3-D Gaze Tracking Using Stereo Cameras," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 234–245, 2004.

[103] S. W. Shih, Y. T. Wu, and J. Liu, "A calibration-free gaze tracking technique," *Proceedings - International Conference on Pattern Recognition*, vol. 15, no. 4, pp. 201–204, 2000.

[104] R. Newman, Y. Matsumoto, S. Rougeaux, and A. Zelinsky, "Real – Time Stereo Tracking for Head Pose and Gaze Estimation 2 . 3 Face Tracking System," 1998.

[105] C. Hennessey and P. Lawrence, "A Single Camera Eye-Gaze Tracking System with Free Head Motion POG on Monitor Pupil Cornea," vol. 1, no. March, pp. 27–29, 2006.

[106] C. C. Lai, S. W. Shih, and Y. P. Hung, "Hybrid method for 3-D gaze tracking using glint and contour features," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 1, pp. 24–37, 2015.

[107] X. Zhou, H. Cai, Z. Shao, H. Yu, and H. Liu, "3D eye model-based gaze estimation from a depth sensor," *2016 IEEE International Conference on Robotics and Biomimetics, ROBIO 2016*, pp. 369–374, 2016.

[108] L. Jianfeng and L. Shigang, "Eye-model-based gaze estimation by RGB-D camera," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 606–610, 2014.

[109] D. Li, J. Babcock, and D. J. Parkhurst, "openEyes," vol. 1, no. March, p. 95, 2006.

[110] E. Hernández, S. Hernández, D. Molina, R. Acebrón, and C. E. Cena, "OSCANN: Technical characterization of a novel gaze tracking analyzer," *Sensors (Switzerland)*, vol. 18, no. 2, pp. 1–16, 2018.

[111] "RestEasy: An open source chin rest for human psychophysics experiments · nimh-nif/SCNI_Toolbar Wiki · GitHub." [Online]. Available: https://github.com/nimh-nif/SCNI_Toolbar/wiki/RestEasy:-An-open-source-chin-rest-for-human-psychophysics-experiments

[112] "RestEasy open-source chin rest for psychophysics by PhenomenalCat - Thingiverse." [Online]. Available: https://www.thingiverse.com/thing:2968729

[113] Y. Li, Y. Zhan, and Z. Yang, "Evaluation of appearance-based eye tracking calibration data selection," *Proceedings of 2020 IEEE International Conference on Artificial Intelligence and Computer Applications, ICAICA 2020*, pp. 222–224, 2020.

[114] W. Wang, Y. Huang, and R. Zhang, "Driver gaze tracker using deformable template matching," *Proceedings of 2011 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2011*, pp. 244–247, 2011.

[115] I. F. Ince and J. W. Kim, "A 2D eye gaze estimation system with low-resolution webcam images," *Eurasip Journal on Advances in Signal Processing*, vol. 2011, pp. 1–11, 2011.

[116] D. W. Hansen, J. S. Agustin, and A. Villanueva, "Homography normalization for robust gaze estimation in uncalibrated setups," *Eye Tracking Research and Applications Symposium (ETRA)*, vol. 1, no. 212, pp. 13–20, 2010.

[117] D. H. Yoo and M. J. Chung, "A novel non-intrusive eye gaze estimation using cross-ratio under large head motion," *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 25–51, 2005.

[118] I. Bacivarov, M. Ionita, and P. Corcoran, "Statistical models of appearance for eye tracking and eye-blink detection and measurement," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 3, pp. 1312–1328, 2008.

[119] Y. Liang, M. L. Reyes, and J. D. Lee, "Real-time detection of driver cognitive distraction using support vector machines," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 340–350, 2007.

[120] P. Koutras and P. Maragos, "Estimation of eye gaze direction angles based on active appearance models," *Proceedings - International Conference on Image Processing, ICIP*, vol. 2015-Decem, pp. 2424–2428, 2015.

[121] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, "Appearance-based gaze estimation in the wild," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June-2015, pp. 4511–4520, 2015.

[122] Z. R. Cherif, A. Naït-Ali, J. F. Motsch, and M. O. Krebs, "An adaptive calibration of an infrared light device used for gaze tracking," *Conference Record - IEEE Instrumentation and Measurement Technology Conference*, vol. 2, no. May, pp. 1029–1033, 2002.

[123] Q. Ji and Z. Zhu, "Eye and gaze tracking for interactive graphic display," *ACM International Conference Proceeding Series*, vol. 22, pp. 79–85, 2002.

[124] T. Ohno and N. Mukawa, "A Free-head , Simple Calibration , Gaze Tracking System That Enables Gaze-Based Interaction," pp. 115–122, 2004.

[125] A. Meyer, M. Bõhme, T. Martinetz, and E. Barth, "A single-camera remote eye tracker," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4021 LNAI, pp. 208–211, 2006.

[126] J. R. Bernhagen, "Noise Control on a Heavy-Duty Mobile Crane." *SAE Prepr*, vol. 54, no. 760601, pp. 2246–2260, 2007.

[127] A. Strupczewski, B. Czupryński, J. Naruniec, and K. Mucha, "Geometric Eye Gaze Tracking," vol. 3, no. Visigrapp, pp. 444–455, 2016.

[128] V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1867–1874, 2014.

[129] Y. Zhang, X. Zheng, W. Hong, and X. Mou, "A comparison study of stationary and mobile eye tracking on EXITs design in a wayfinding system," *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2015*, no. December, pp. 649–653, 2016.

[130] M. Kowalski, J. Naruniec, and T. Trzcinski, "Deep Alignment Network: A Convolutional Neural Network for Robust Face Alignment," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2017-July, pp. 2034–2043, 2017.

[131] T. Baltrusaitis, A. Zadeh, Y. C. Lim, and L. P. Morency, "OpenFace 2.0: Facial behavior analysis toolkit," *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, pp. 59–66, 2018.

[132] Y. Feng, F. Wu, X. Shao, Y. Wang, and X. Zhou, "Joint 3d face reconstruction and dense alignment with position map regression network," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11218 LNCS, pp. 557–574, 2018.

[133] "Google Developers Blog: Object Detection and Tracking using MediaPipe." [Online]. Available: https://developers.googleblog.com/2019/12/object-detection-and-tracking-using-mediapipe.html

[134] J. Guo, X. Zhu, Y. Yang, F. Yang, Z. Lei, and S. Z. Li, "Towards Fast, Accurate and Stable 3D Dense Face Alignment," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12364 LNCS, pp. 152–168, 2020.

[135] "Home - mediapipe." [Online]. Available: https://google.github.io/mediapipe/

[136] "mediapipe/canonical_face_model_uv_visualization.png at master · google/mediapipe · GitHub." [Online]. Available: https://github.com/google/mediapipe/blob/master/mediapipe/modules/face_geometry/data/canonical_face_model_uv_visualization.png

[137] "Image Formation and Pinhole Model of the Camera — by Neeraj Krishna — Towards Data Science." [Online]. Available: https://towardsdatascience.com/image-formation-and-pinhole-model-of-the-camera-53872ee4ee92

[138] "OpenCV: Camera Calibration and 3D Reconstruction." [Online]. Available: https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html

[139] "GitHub - ishaanjav/Face_Analyzer: The purpose of this Android app is to utilize the Microsoft Face API to not only detect individual faces in an image, but also for each face provide information such as emotions, the estimated age, gender, and more. Possible applications for this app are at amusement parks, classrooms, and residential homes." [Online]. Available: https://github.com/ishaanjav/Face_Analyzer

[140] Sebastian Höffner, "(PDF) Gaze Tracking Using Common Webcams." [Online]. Available: https://www.researchgate.net/publication/326588105_Gaze_Tracking_Using_Common_Webcams

[141] "Commits · ParisNeo/FaceAnalyzer · GitHub." [Online]. Available: https://github.com/ParisNeo/FaceAnalyzer/commits?author=ParisNeo

[142] "Home » omes-va.com." [Online]. Available: https://omes-va.com/

[143] "FaceAnalyzer/euclidian.py at main · ParisNeo/FaceAnalyzer · GitHub." [Online]. Available: https://github.com/ParisNeo/FaceAnalyzer/blob/main/FaceAnalyzer/helpers/geometry/euclidian.py

[144] L. Thomas, Y. V. S. Murthy, and S. Ramesh, "Artificial Neural Networks- Theory , Concepts & Applications," no. December, 2020.

[145] E. Varela and E. Campbells, "Redes Neuronales Artificiales : Una Revisión del Estado del Arte , Aplicaciones Y Tendencias Futuras Artificial Neural Networks : A Brief Review," vol. 2, pp. 18–27.

[146] T. Ae, R. Aibara, and Y. Nishioka, "A memory-based artificial neural network," *1991 IEEE International Joint Conference on Neural Networks*, pp. 614–619, 1992.

[147] C. A. Ruiz and D. J. Matich, "Redes Neuronales: Conceptos Básicos y Aplicaciones." 2001.

[148] J. J. Montaño, "Redes Neuronales Artificiales aplicadas al Análisis de Datos," *Network*, p. 275, 2002.

[149] A. J.Serrano, "Redes Neuronales," pp. 82–86.

[150] T. F. Gonzalez, "Handbook of approximation algorithms and metaheuristics," *Handbook of Approximation Algorithms and Metaheuristics*, pp. 1–1432, 2007.

[151] X. Zhang, R. Fergus, and Y. Lecun, "OverFeat : Integrated Recognition , Localization and Detection using Convolutional Networks arXiv : 1312 . 6229v3 [cs . CV] 14 Jan 2014," no. December, 2013.

[152] C. Couprie, L. Najman, and Y. Lecun, "for Scene Labeling," pp. 1–15.

[153] D. C. Cires and A. Giusti, "Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images," pp. 1–9.

[154] "CS231n Convolutional Neural Networks for Visual Recognition." [Online]. Available: https://cs231n.github.io/convolutional-networks/#overview

[155] NetApp, "¿Qué es el aprendizaje profundo?" [Online]. Available: https://www.netapp.com/es/artificial-intelligence/what-is-deep-learning/

[156] S. Ji, W. Xu, M. Yang, and K. Yu, "3D Convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[157] Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," vol. 86, no. 11, 1998.

[158] J. S. Ren and L. Xu, "On vectorization of deep convolutional neural networks for vision tasks," *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, pp. 1840–1846, 2015.

[159] Z. Wang, J. Chen, and S. C. Hoi, "Deep Learning for Image Super-resolution: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2020.

[160] D. Learning and V. Games, "Trabajo de Fin de Grado Deep Learning en Videojuegos : Superresolución."

[161] J. Yamanaka, S. Kuwashima, and T. Kurita, "Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network," vol. 1, pp. 217–225, 2017.

[162] V. Duraisamy, P. Pro, and G. Fractals, "An Approach of Image Scaling Using DWT and Bicubic Interpolation."

[163] K. Zhang, M. Sun, S. Member, and T. X. Han, "Residual Networks of Residual Networks : Multilevel Residual Networks," vol. 14, no. 8, 2017.

[164] "GitHub - jiny2001/dcscn-super-resolution: A tensorflow implementation of "Fast and Accurate Image Super Resolution by Deep CNN with Skip Connection and Network in Network", a deep learning based Single-Image Super-Resolution (SISR) model." [Online]. Available: https://github.com/jiny2001/dcscn-super-resolution

[165] P. Blignaut and T. Beelders, "The precision of eye-trackers: A case for a new measure," *Eye Tracking Research and Applications Symposium (ETRA)*, no. March 2016, pp. 289–292, 2012.

[166] M. Thibeault, M. Jesteen, and A. Beitman, "Improved Accuracy Test Method for Mobile Eye Tracking in Usability Scenarios," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 63, no. 1, pp. 2226–2230, 2019.

[167] "Grados de ángulo visual // documentación de OpenSesame." [Online]. Available: https://osdoc.cogsci.nl/3.3/visualangle/

[168] "Acerca de Python™ — Python.org." [Online]. Available: https://www.python.org/about/

[169] "GitHub - opencv/opencv: Open Source Computer Vision Library." [Online]. Available: https://github.com/opencv/opencv

[170] "Python OpenCV — cv2.imread() method - GeeksforGeeks." [Online]. Available: https://www.geeksforgeeks.org/python-opencv-cv2-imread-method/

[171] "python: uso de cv2.VideoCapture (), read (), waitKey () en OpenCV2 - programador clic." [Online]. Available: https://programmerclick.com/article/7455936443/

[172] "OpenCV: Perspective-n-Point (PnP) pose computation." [Online]. Available: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html

[173] "cv.Rodrigues - mexopencv." [Online]. Available: http://amroamroamro.github.io/mexopencv/matlab/cv.Rodrigues.html

[174] "GitHub - google/mediapipe: Cross-platform, customizable ML solutions for live and streaming media." [Online]. Available: https://github.com/google/mediapipe

[175] Satya Mallick, "Head Pose Estimation using OpenCV and Dlib — LearnOpenCV #." [Online]. Available: https://learnopencv.com/head-pose-estimation-using-opencv-and-dlib/

[176] "Detección de líneas y círculos usando la transformada de Hough con OpenCV." [Online]. Available: http://acodigo.blogspot.com/2017/09/deteccion-de-lineas-y-circulos-usando.html?m=1

[177] "Introducción — Blender Manual." [Online]. Available: https://docs.blender.org/manual/es/dev/getting_started/about/introduction.html

[178] "Qt Framework - ¡Un marco para gobernar todo!" [Online]. Available: https://www.qt.io/product/framework

[179] "PySide2 · PyPI." [Online]. Available: https://pypi.org/project/PySide2/

[180] "FreeCAD: Tu propio modelador paramétrico 3D." [Online]. Available: https://www.freecadweb.org/?lang=es_ES

[181] "Software Inventor — Obtener precios y comprar el producto oficial Inventor 2023." [Online]. Available: https://www.autodesk.es/products/inventor/overview?term=1-YEAR{&}tab=subscription

[182] "NumPy." [Online]. Available: https://numpy.org/

[183] "Plataforma de desarrollo en tiempo real de Unity — Motor de VR y AR en 3D y 2D." [Online]. Available: https://unity.com/es

[184] "Uso del módulo yaml en python - programador clic." [Online]. Available: https://programmerclick.com/article/3930260043/

[185] "Matplotlib — Visualization with Python." [Online]. Available: https://matplotlib.org/

[186] "Introducción a TensorFlow." [Online]. Available: https://www.tensorflow.org/learn

[187] "OpenCV: Camera Calibration." [Online]. Available: https://docs.opencv.org/3.4/dc/dbb/tutorial_py_calibration.html

Anexos

Anexo A

Librerías y frameworks utilizados

Dentro de este estudio se realiza una recreación de un modelo 3D implementando como base el lenguaje de programación Python [168], porque su estructura de programación es bastante intuitiva y es un lenguaje de programación gratuito, multi-propósito, que se usa en variedad de campos como minería, inteligencia artificial, web, etc. Además, Python es un lenguaje de programación "Open source".

OpenCV [169] es una librería de visión artificial multi-plataforma que permite realizar variedad de tareas como detección de movimiento, reconocimiento de objetos y muchas otras. OpenCV dispone de un amplio repositorio con diversidad de algoritmos y funciones dedicados a varios campos. En esta investigación se implementan sus funciones para variedad de tareas. Funciones como "cv2.imread()" [170] o "cv2captureread" [171] (para uso de cámara web) realizan el proceso de tratamiento de imágenes junto con funciones como undistort para quitar distorsión a la imagen. Funciones relacionadas con hallar rotaciones y ángulos de objetos como "solvePnP()" [172], "cv.rodrigues" [173] y otras funciones útiles en este proyecto para dibujar puntos sobre un objeto o encontrar contornos, como se verá más adelante. y para realizar tareas en la cámara implementada para el modelo 3D como su lectura o realizar la calibración de la misma.

MediaPipe [174] es un marco multi-plataforma de Google que contiene un repositorio muy robusto con información ya procesada y compilada por innumerables usuarios donde se usa inteligencia artificial para ajustar rostros como lo hace la librería multi-plataforma Dlib en artículos como [140], [175]. Esta librería contiene múltiples algoritmos y funciones sobre transformación de imagen, detección de puntos y en el desarrollo de este trabajo se usan sus funciones principalmente para detección de objetos como la detección del rostro y pupilas, pues posee mayor precisión que los clasificadores cascada [176]. En MediaPipe, además, se encuentran modelos faciales útiles para nuestra investigación y funciones muy eficaces para el desarrollo de este proyecto.

La librería FaceAnalyzer [139], por su parte, organiza y condensa mucha de la información que maneja MediaPipe generando funciones con la información más importante. Contiene algoritmos que hacen detección de los puntos, medición y otras tareas muy importantes y esenciales en este trabajo de grado.

Blender [177] es un software para modelado 3D que nos permite encontrar y manipular las coordenadas 3D de los puntos del modelo canónico. Se utilizó también el framework QT [178] para la creación de interfaz gráfica y la versión utilizada en este proyecto fue pyside2 [179] por el tipo de licencia manejado por pyside2, la cual es gratuita.

FreeCAD [180] se usó para visualizar las piezas 3D del estabilizador tomado de RestEasy que hace parte de la plataforma física de este proyecto mencionada en el Capítulo 2 y Autodesk Inventor [181] para generar los planos de dichas piezas y poder realizar su impresión 3D. Python posee Una función importante implementada para el cálculo matemático de matrices llamada NumPy [182].

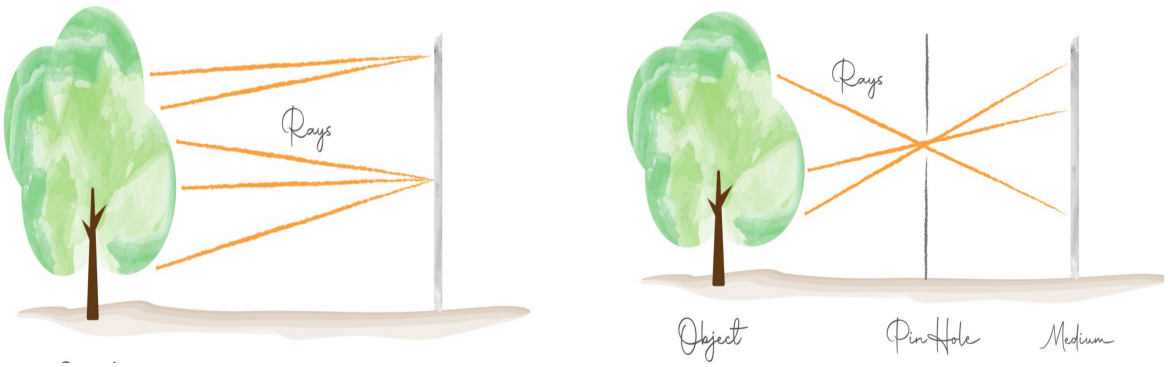
Unity [183], por su parte, es un motor gráfico multi-plataforma 2D y 3D que se usó para generar la simulación del espacio de coordenadas de nuestro sistema de SO. Se dispuso de la librería Yaml [184] para extraer datos de ejecución y guardarlos para cuando desee hacer uso de ellos. Matplotlib [185] para plotear gráficos tanto 2D como 3D en Python y Tensorflow [186] para trabajar con redes neuronales en Python.

Anexo B

Formación de imagen y parámetros de afectación

Es importante conocer el funcionamiento y los parámetros internos que hacen parte del funcionamiento de una cámara. En la Figura B.1 se muestra el funcionamiento de una cámara, tipo estenopeica, donde se observa la ubicación de un objeto y un medio, donde el medio o plano es el encargado de tomar los rayos que refleja el objeto para así capturarlos y mostrarlos posteriormente en píxeles.

En la Figura B.1a notamos que el medio no puede recibir los rayos proyectados por el objeto directamente. Estos llegarían de todas las partes del objeto y se daría una superposición de rayos impidiendo al plano realizar una adecuada captura de estos. En la Figura B.1b se ubica un obstáculo que no permita que los rayos pasen directamente del objeto al medio. Esta barrera posee un agujero muy pequeño denominado "pinhole", el cual permite que los rayos reflejados por el objeto principal pasen solo por ese espacio hacia el medio y no haya una superposición de información. La ubicación de este agujero hace que el medio tome la imagen invertida.



(a) Proyección sin barrera (b) Proyección con barrera

Figura B.1: Proyección de puntos sobre un medio (Tomado de [137]).

Para explicar adecuadamente este proceso se asume que el "pinhole" no se ubica en el espacio medio entre el objeto y el plano como es en el mundo real. Es el plano el que se encuentra entre el objeto y la barrera con el "pinhole" para poder entender mejor cómo es que se reflejan los rayos emitidos por el objeto en el mismo plano sin la

forma invertida como lo toma la cámara. El "pinhole" también es llamado centro de proyección o centro de la cámara y se asume que todos los rayos de luz emitidos por el objeto convergen en este. El objetivo de asumir esto es que la imagen de cada punto del objeto sea la proyección de ese punto en el sensor o plano de la cámara.

En la Figura B.2 se observa el modelo estenopeico de una cámara, donde se ubica el centro de la cámara o "pinhole" en el plano 3D y en frente se ubica el plano o sensor de la misma sobre el eje z ya con la suposición de que el plano está entre el "pinhole" y el objeto, solo para que los rayos que vienen del objeto intercepten en este. Aquí el objetivo es entender el proceso por el cual se proyectan los puntos con coordenadas del mundo en el plano de la imagen, comprendiendo la relación que existe entre los puntos del mundo y los píxeles correspondientes en la imagen que la cámara está capturando. Para realizar este proceso se realizan 2 importantes transformaciones llamadas cámara extrínseca y cámara intrínseca.

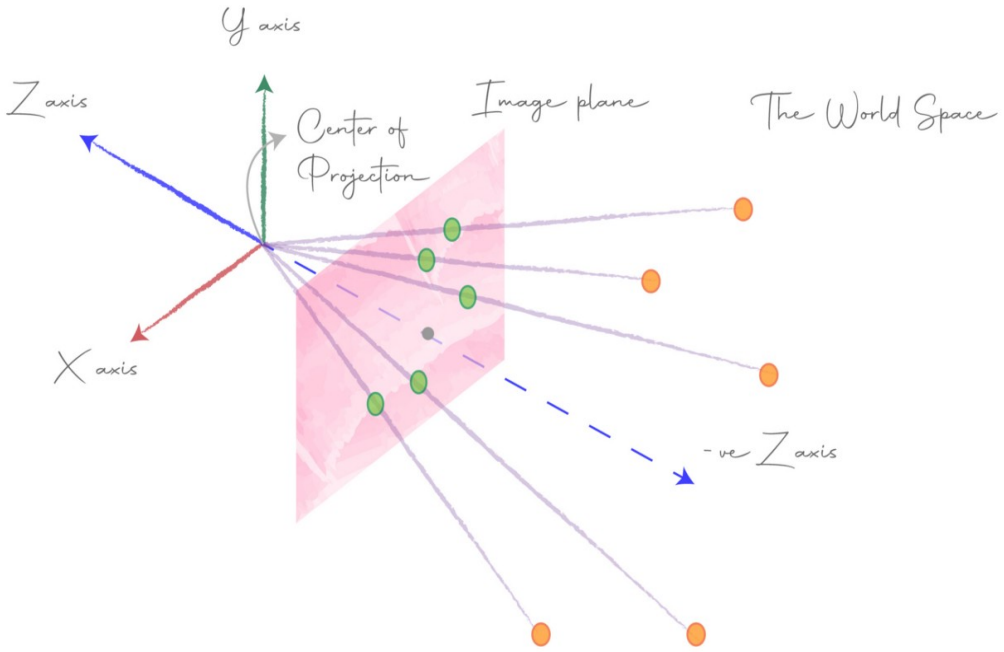


Figura B.2: Modelo estenopeico de una cámara (Tomado de [137]).

B.0.1. Matriz extrínseca

Es un cambio de transformación base de un sistema de coordenadas a otro, que permite darle coordenadas a la cámara a partir de las coordenadas de los objetos en el mundo real, nos permite ver el mundo desde la perspectiva de la cámara. Para esto es necesario saber cómo se encuentra orientada la cámara y cuál es su ubicación dentro de las coordenadas del mundo, motivo por el que se usan 2 transformaciones adicionales: la transformación de rotación para orientar la cámara y la transformación de traslación para mover la cámara. A la vez estas transformaciones se dividen en transformaciones de rotación con 2 puntos en el plano sin movimiento de ejes, transformación de rotación con un punto y movimiento de ejes, transformación de traslación con 2 puntos en el plano sin movimiento de ejes y transformación de traslación con un punto y con movimiento de ejes.

2.1. Transformación de rotación con 2 puntos en el plano sin movimiento de ejes.

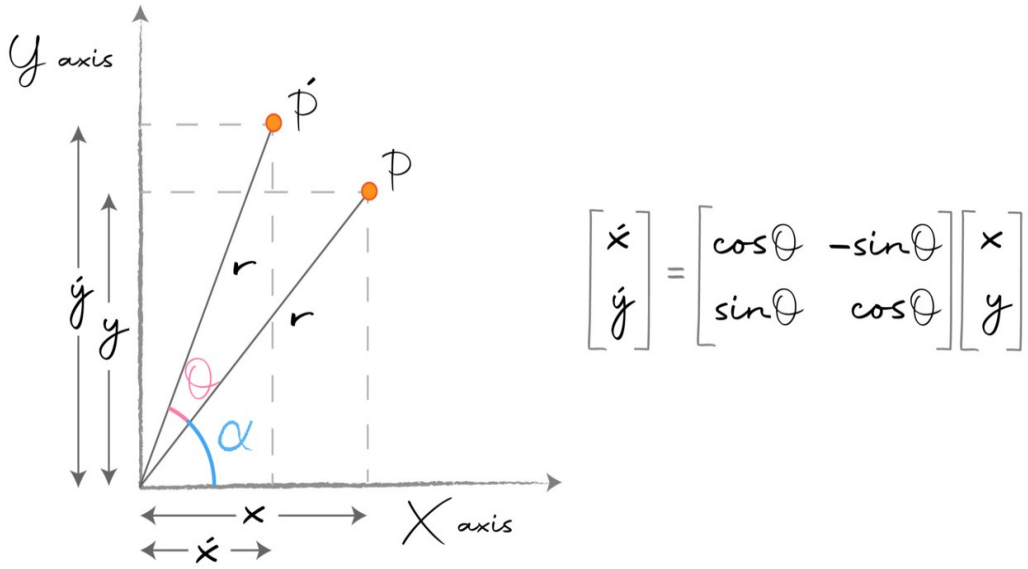


Figura B.3: Transformación de rotación (Tomado de [137]).

En la Figura B.3 se observa dentro del plano en el espacio, un punto inicial P con coordenadas (x,y) y un ángulo α que rota un ángulo θ para convertirse en un punto final P' con coordenadas (x',Y') con un ángulo $\alpha + \theta$. Debemos hallar las nuevas coordenadas del punto P', las cuales son (x',Y') así:

```
From the figure,
      sinα = y/r , cosα = x/r - [1]
⇒     xsinα = ycosα - [2]
also, x' = rcos(θ+α)
⇒     x' = (x/cosα) * cos(θ+α) (from [1])
but,  cos(θ+α) = cosθcosα - sinθsinα
⇒     x' = (x/cosα) * (cosθcosα - sinθsinα)
⇒     x' = xcosθ - xsinα * (sinθ / cosα)
⇒     x' = xcosθ - ycosα * (sinθ / cosα) (from [2])
⇒     x' = xcosθ - ysinθ

Similarly,
      y' = rsin(θ+α)
⇒     y' = (y/sinα) * sin(θ+α) (from [1])
but,  sin(θ+α) = sinθcosα + cosθsinα
⇒     y' = (y/sinα) * (sinθcosα + cosθsinα)
⇒     y' = ycosθ + ycosα * (sinθ / sinα)
⇒     y' = ycosθ + xsinα * (sinθ / sinα) (from [2])
⇒     y' = ycosθ + xsinθ
⇒     y' = xsinθ + ycosθ

Hence we have,
      x' = xcosθ - ysinθ
      y' = xsinθ + ycosθ
```

Figura B.4: Cálculo de X' y Y' (Tomado de [137]).

Al ser la rotación una operación lineal, nos permite representar estas ecuaciones como una multiplicación de matrices así:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{B.1}$$

Aquí se están transformando los puntos manteniendo fijos los ejes o la base.

Nota: Estas transformaciones se pueden llevar a R3.
2.2.Transformación de rotación con 1 punto y con movimiento de ejes.

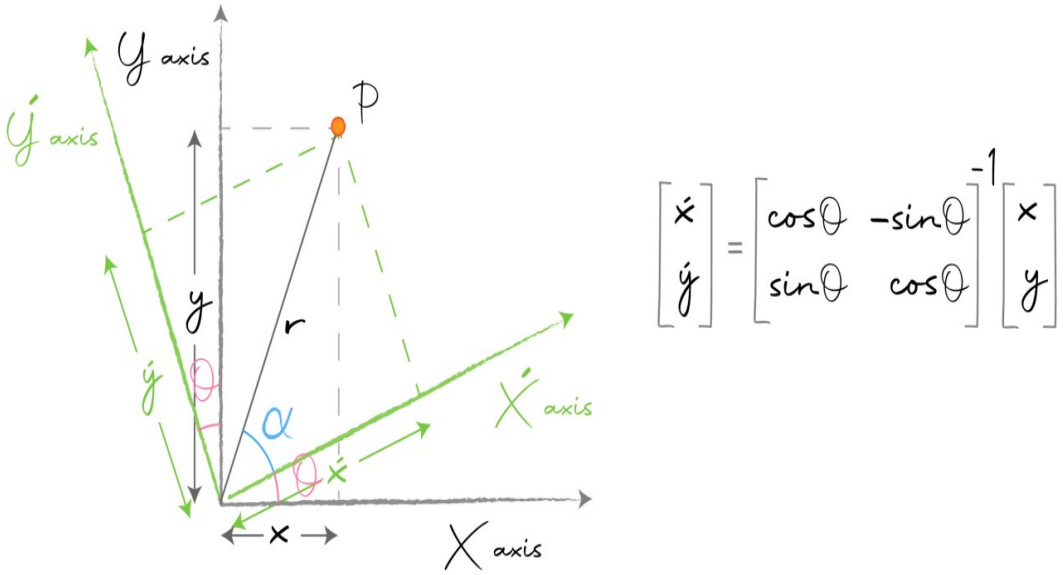


Figura B.5: Rotación de ejes XY (Tomado de [137]).

En la Figura B.5 se observa dentro del plano en el espacio, un punto inicial P con coordenadas (x,y) y un ángulo α. En este caso los ejes X y Y giraron un ángulo θ. Debemos hallar las nuevas coordenadas del punto P' con la nueva base. Estos ejes ahora son (X',Y') así:

```
From the figure,
      sinα = y'/r , cosα = x'/r - [1]
⟹    x'sinα = y'cosα - [2]
also, x = rcos(θ+α)
⟹    x = (x'/cosα) * cos(θ+α) (from [1])
but,  cos(θ+α) = cosθcosα - sinθsinα
⟹    x = (x' / cosα) * (cosθcosα - sinθsinα)
⟹    x = x'cosθ - xsinα * (sinθ / cosα)
⟹    x = x'cosθ - y'cosα * (sinθ / cosα) (from [2])
⟹    x = x'cosθ - y'sinθ

Similarly,
      y = rsin(θ+α)
⟹    y = (y'/sinα) * sin(θ+α) (from [1])
but,  sin(θ+α) = sinθcosα + cosθsinα
⟹    y = (y'/sinα) * (sinθcosα + cosθsinα)
⟹    y = y'cosθ + y'cosα * (sinθ / sinα)
⟹    y = y'cosθ + x'sinα * (sinθ / sinα) (from [2])
⟹    y = y'cosθ + x'sinθ
⟹    y = x'sinθ + y'cosθ

Hence we have,
      x = x'cosθ - y'sinθ
      y = x'sinθ + y'cosθ
```

Figura B.6: Cálculo de X' y Y' (Tomado de [137]).

En la forma matricial:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{B.2}$$

Como el objetivo no es encontrar XY, entonces movemos la matriz al otro lado tomando su inversa así:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \end{bmatrix} \tag{B.3}$$

Nota: La matriz de cambio de base es la inversa de la matriz de transformación lineal. Esto nos hace concluir que si conocemos la matriz de transformación de la cámara ya con rotación y traslación incluidas, podemos hallar su inversa para hallar la matriz de cambio de base que nos ayudará a encontrar las coordenadas de los puntos con respecto a la cámara.

2.3. Transformación de traslación con 2 puntos en el plano sin movimiento de ejes.

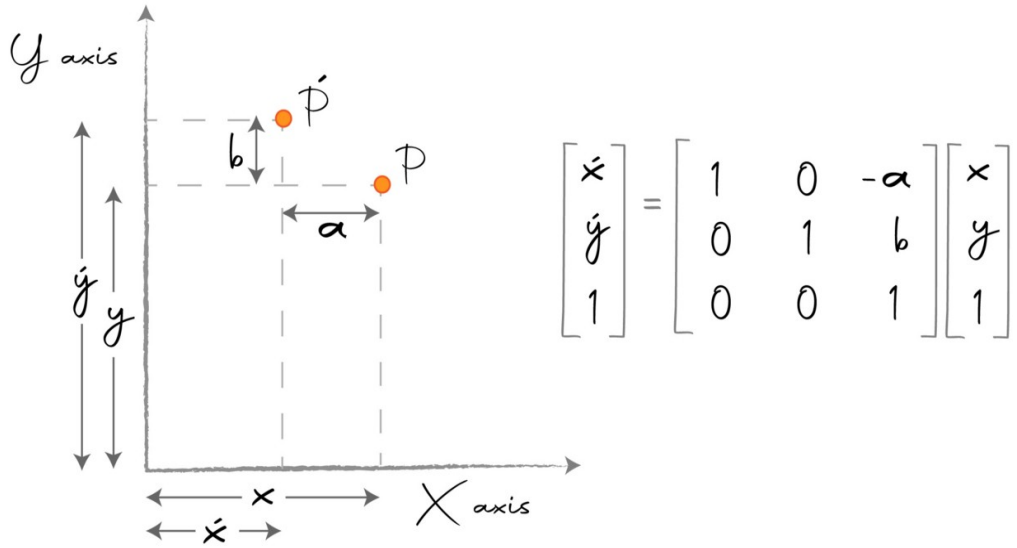


Figura B.7: Transformación de traslación (Tomado de [137]).

En la Figura B.7 se observa dentro del plano en el espacio un punto inicial P con coordenadas (x,y) y un desplazamiento (a) en X, más un desplazamiento (b) en Y, para convertirse en un punto final P' con coordenadas (x',Y'). Debemos hallar las nuevas coordenadas del punto P', las cuales son (X',Y') así:

```
from the figure,
      x' = x - a
      y' = y + b
```

Figura B.8: Cálculo de X' y Y' (Tomado de [137]).

Aquí no se pueden convertir estas ecuaciones a multiplicación de matrices, así que se agrega una dimensión adicional para expresar la traslación como una transformación lineal así:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{B.4}$$

Las coordenadas que se adicionan se denominan homogéneas, las cuales son usadas en geometría proyectiva para proyectar puntos en el infinito por coordenadas finitas y simplificar fórmulas cuando se compara a su contraparte cartesiana.

Para recuperar nuevamente las coordenadas euclidianas desde las coordenadas homogéneas, ya que es más fácil hacer operaciones y trabajar en espacio homogéneo y luego pasar a coordenadas euclidianas nuevamente, hacemos lo siguiente:

$$[x, y, 1] \cong [x/1, y/1] = [x, y] \tag{B.5}$$

2.4. Transformación de rotación con 1 punto y con movimiento de ejes.

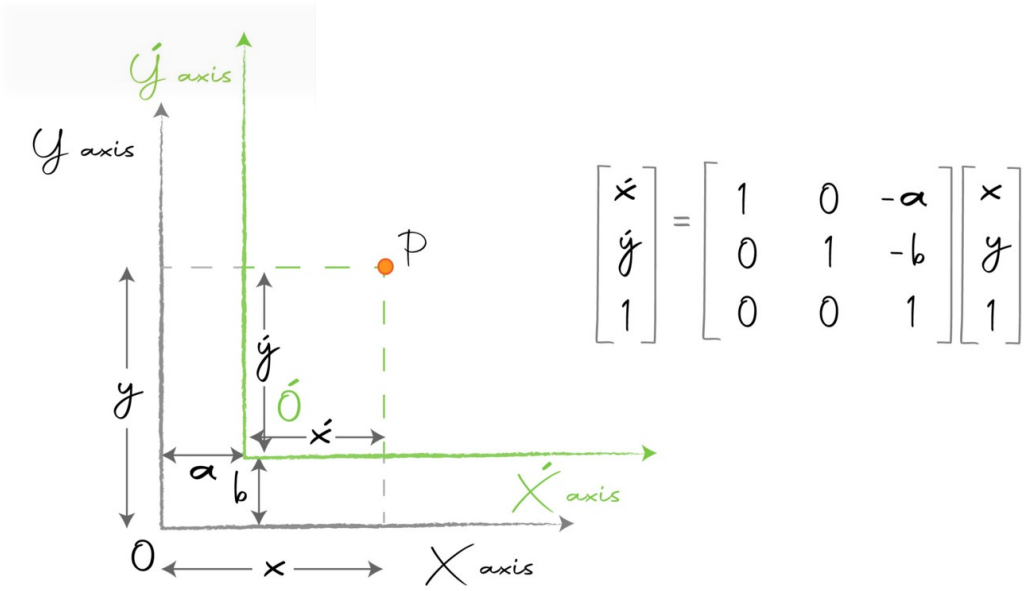


Figura B.9: Transformación de traslación (Tomado de [137]).

En la Figura B.9 se observa dentro del plano en el espacio un punto inicial P con coordenadas (x,y). En este caso los ejes XY se movieron (a) en X y (b) en Y. Debemos hallar las nuevas coordenadas del punto P' con la nueva base, estos ejes ahora son (x',Y') así:

```
from the figure,
      x' = x - a
      y' = y - b
```

Figura B.10: Cálculo de X' y Y' (Tomado de [137]).

En la forma matricial, usando coordenadas homogéneas, es así:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & -b \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{B.6}$$

En la traslación, también la transformación lineal y de cambio de base son inversas una de la otra.

Matricialmente también se puede lograr, puesto que se pueden usar composiciones de matriz para realizar rotación y traslación así:

$$\begin{bmatrix} R & O \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} 1 & O \\ 0^T & 1 \end{bmatrix} \begin{bmatrix} R & 0 \\ 0^T & 1 \end{bmatrix} \tag{B.7}$$

R es la matriz de rotación con un tamaño de 3 x 3 y O es la matriz de traslación con un tamaño de 3 x 1.

Aquí se puede realizar la matriz de cambio de base tomando la inversa de la matriz de transformación final. Esta nueva matriz con un tamaño de 4 x 4 se denomina matriz de cámara extrínseca, que denotaremos por la letra E. Con ayuda de esta matriz podremos encontrar las coordenadas de cualquier punto del mundo frente a la cámara multiplicando los puntos del punto por la matriz intrínseca y tendremos sus coordenadas en el sistema de la cámara.

$$P^c = E * P^w \tag{B.8}$$

Y llevándola a la forma extendida, quedaría así:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} r_0 0 & r_0 1 & r_0 2 & t_x \\ r_1 0 & r_1 1 & r_1 2 & t_y \\ r_2 0 & r_2 1 & r_2 2 & t_z \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ 1 \end{bmatrix} \tag{B.9}$$

La matriz extrínseca posee 6 grados de libertad por los tres ángulos de rotación y por los 3 componentes de traslación sobre los ejes x y z.

B.0.2. Matriz intrínseca

Matriz intrínseca. Es una transformación de proyección que permite darle ubicación en píxeles a los rayos de los puntos proyectados que vienen del objeto del mundo, por medio de la distancia focal. Estas capturas de proyecciones de puntos se realizan en el plano de la imagen, como se ve en la Figura B.11:
Nota: La rotación intrínseca es difícil de realizar con álgebra euclidiana porque el objeto rota sobre ejes relativos, por tal razón se enfoca en rotación extrínseca.

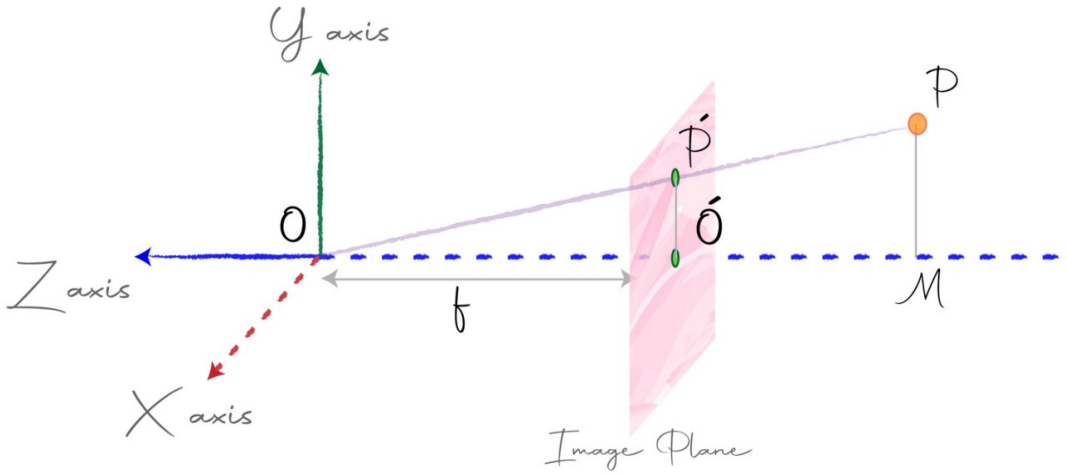


Figura B.11: Proyección de un punto (Tomado de [137]).

En la Figura B.11, el centro de la cámara o "pinhole" está ubicado en el origen O y el plano de la imagen se encuentra a una distancia f hacia el eje (-ve Z), se observa un

punto P perteneciente a un punto de un objeto en el mundo. Este se proyecta sobre el plano de la imagen como P'. Las coordenadas de P son (x,y,z) y las de P' son (x',y',z'), se deben encontrar entonces las coordenadas de la proyección del punto P en el plano, en este caso las coordenadas de P':

```
from the figure,
      ΔOMP and ΔOO'P' are similar triangles.
⟹    x'/x = y'/y = f/z
⟹    x' = x * f/z and y' = y * f/z
```

Figura B.12: Cálculo de coordenadas de P' (Tomado de [137]).

Aquí es importante resaltar cómo funciona la proyección de una imagen en un plano, ya que si vemos la Figura 3.5 notamos que a medida que un punto P o el objeto que lo proyecta, se aleja de la cámara, su coordenada en z aumenta y su proyección se hace más y más pequeña, entendiendo lo lógico, entre más se aleje un objeto de la cámara, más pequeño aparecerá en el plano de imagen.

Estas coordenadas de proyección pueden ser convertidas a píxeles, descartando la última proyección. De esa manera las coordenadas de la imagen anterior son $(xf/z, yf/z, f)$, al representar estas coordenadas como (u, v) tomamos las 2 primeras coordenadas que son $(xf/z, yf/z)$, descartando la coordenada z así:

$$(u, v) = (xf/z,\ yf/z) \tag{B.10}$$

Y estas son las coordenadas de la imagen generada.

La anterior es una formación de imagen ideal, pues en el mundo real la formación de imagen no es tan simple. A continuación se muestran los parámetros que pueden afectar la formación de una imagen:

3.1. Factor de escala. Se incorpora un factor de escala para normalizar las unidades, pues la distancia focal puede venir en mm, pero puede usar otras unidades como píxeles.

$$(u, v) = (\alpha * x/z,\ \alpha * y/z) \tag{B.11}$$

Siendo α la distancia focal escalada.

3.2. Relación de aspecto (Píxeles rectangulares). En el mundo real los píxeles no son todos cuadrados, algunos son rectangulares. Por esta razón se adicionan factores de escala diferentes tanto para el ancho como para la altura del píxel.

$$(u, v) = (\alpha * x/z,\ \beta * y/z) \tag{B.12}$$

α= Factor de escala para ancho.

β= Factor de escala para altura.

3.3. Desplazamiento. La línea perpendicular que sale desde el centro de la cámara hacia el plano de la imagen se conoce como eje óptico. El punto donde ambos se

interceptan (eje óptico y plano) se conoce como centro óptico. Pero no siempre el eje óptico intercepta con el origen del plano (ver Figura B.13), por lo que se debe de tener en cuenta esta distancia y se debe incorporar un factor de desplazamiento en la ecuación general.

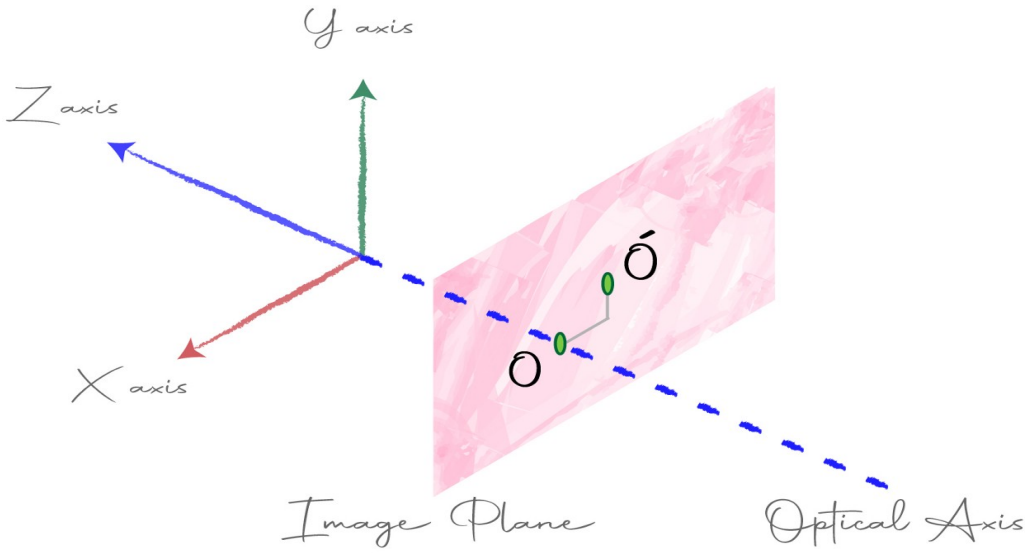


Figura B.13: Origen del plano y el centro óptico (Tomado de [137]).

$$(u, v) = (\alpha * x/z + x0,\ \beta * y/z + y0) \tag{B.13}$$

Con (x0, y0) como factores de desplazamiento.

3.4. Factor de sesgo. Un plano de imagen ideal posee una dirección de altura perpendicular a la de su ancho, como un rectángulo, pero en algunas ocasiones en la realidad el plano puede ser un paralelogramo con sus ejes en ángulo, como se ve en la Figura B.14.

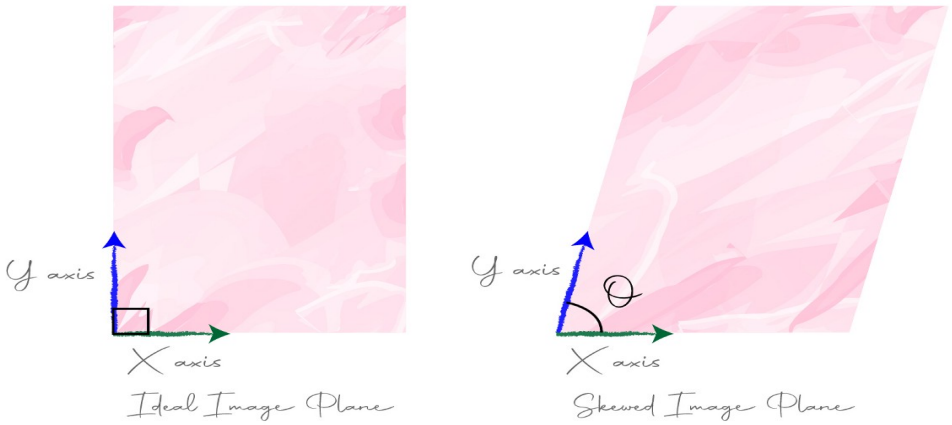


Figura B.14: Plano ideal vs. plano sesgado (Tomado de [137]).

Al existir un cambio en el ángulo de los ejes, estamos tratando con algo ya conocido anteriormente como Cambio de base, entonces dado un punto P respecto a los ejes perpendiculares iniciales, debemos expresarlo respecto a los nuevos ejes, los ejes reales de un plano de imagen (paralelogramo).

Para realizar los cálculos, sean (X,Y) las coordenadas iniciales de P y sean (x',y') sus coordenadas sobre la nueva base. Debemos encontrar (x',y').

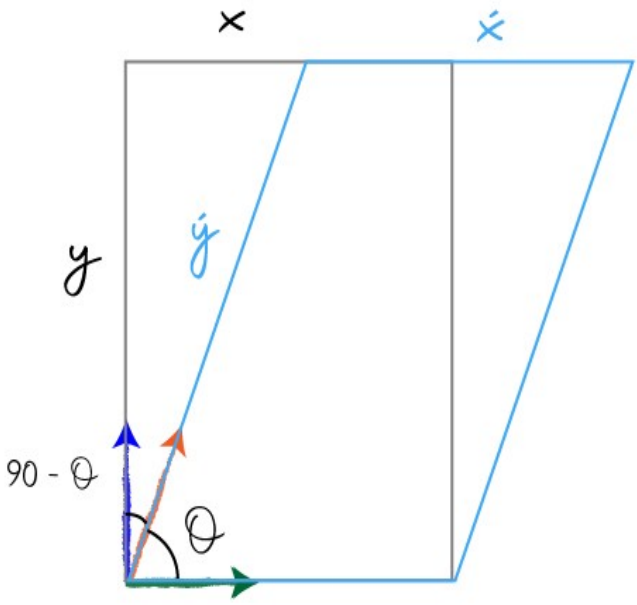


Figura B.15: (Sesgo Tomado de [137]).

```
From the above figure,
      cos(90-θ) = y/y'
⟹    sinθ = y/y'
⟹    y = y'sinθ
⟹    y'= y / sinθ

also,
      sin(90-θ) = (x - x')/y'
⟹    y'cosθ = x - x'
⟹    x' = x - y'cosθ
but,  y'= y / sinθ
⟹    x' = x - ycosθ / sinθ
⟹    x' = x - ycotθ
```

Figura B.16: Obtención de x' y y' (Tomado de [137]).

Al encontrar (x',y'), pasamos a reemplazarlas en la ecuación general:

$$u = \alpha * (x - ycot\theta)/z + x0 \tag{B.14}$$

$$v = \beta * (y/sen\theta)/z + y0 \tag{B.15}$$

$$\Longrightarrow u = \alpha x/z - (\alpha * y/z)cot\theta + x0 \tag{B.16}$$

$$\Longrightarrow v = (\beta y/zsin\theta) + y0 \tag{B.17}$$

La ecuación final con todos los parámetros incluidos quedaría finalmente así:

$$(u, v) = [\alpha x/z - (\alpha y/z)cot\theta) + x0, (\beta y/zsin\theta) + y0] \tag{B.18}$$

Esta ecuación general final se puede representar como una multiplicación de matrices usando nuevamente coordenadas homogéneas, así:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} \alpha & -\alpha cot\theta & x0 & 0 \\ 0 & \beta/sen\theta & y0 & 0 \\ 0 & -0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{B.19}$$

Esta es la llamada matriz intrínseca de la cámara y la representaremos por K. De acuerdo a esto se puede establecer que dadas las coordenadas homogéneas del mundo con respecto a la cámara, estas se pueden multiplicar por la matriz intrínseca de la cámara para obtener las coordenadas homogéneas del punto proyectado en la imagen.

$$P' = k * P_c \tag{B.20}$$

Y para obtener coordenadas euclidianas o en píxeles de estas coordenadas homogéneas, dividimos la ecuación matricial por el último elemento así:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} x'/w \\ y'/w \\ 1 \end{bmatrix} \cong \begin{bmatrix} x'/w \\ y'/w \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \tag{B.21}$$

En la ecuación B.19 de la matriz intrínseca, observamos que la última columna de la matriz intrínseca de la cámara es una columna 0, la eliminamos y simplificamos la matriz, así:

$$\begin{bmatrix} f & s & cx \\ 0 & \alpha f & cx \\ 0 & 0 & 1 \end{bmatrix} \tag{B.22}$$

Y la ecuación matricial se puede reescribir así:

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f & s & cx \\ 0 & \alpha f & cx \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{B.23}$$

Aquí no se requiere representar las coordenadas del punto en su forma homogénea.

De la anterior matriz:
f=distancia focal.

s=factor de sesgo.

cx, cy= desplazamiento.

α = Relación de aspecto.

Concluyendo que la matriz intrínseca posee 5° de libertad.

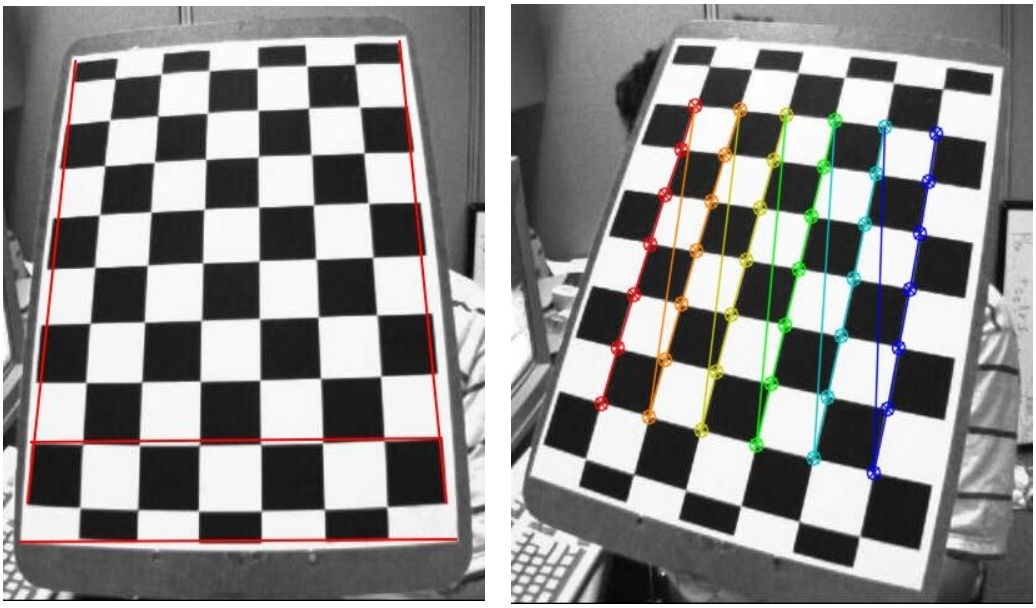
Anexo C

Matriz intrínseca computarizada y coeficientes de distorsión

Algunas cámaras estenopeicas generan imágenes con una distorsión significativa, dicha distorsión puede clasificarse en 2 tipos diferentes, siendo una de tipo radial y la otra de tipo tangencial [187]. En la distorsión radial, las lineas rectas de una imagen pueden parecer curvas, aumentando esta distorsión cuando más lejos se encuentran los puntos del centro de la imagen. En la Figura C.1a se muestra un tablero de ajedrez en el que 2 de sus bordes se han marcado con líneas rojas rectas sobre él, observando que la línea del borde del tablero no coincide con la línea roja de marca, además de no ser una línea recta.

En la distorsión tangencial algunas partes de la imagen pueden verse mucho más cercanas de lo que verdaderamente se encuentran porque la cámara no está alineada perfectamente al plano de la imagen.

Para corregir estas distorsiones, se usa un tablero de ajedrez escalado, se dibuja un patrón sobre este (ver Figura C.1b) y teniendo en cuenta la información específica de sus partes (distancias entre puntos, sus coordenadas y tamaño de cuadros), además se deben tener los parámetros extrínsecos e intrínsecos dados por cada cámara. Dicha información sirve como entrada a las funciones de OpenCV para realizar la calibración de la cámara y obtener la matriz intrínseca junto con sus coeficientes de distorsión, proceso mostrado en [187].



(a) Distorsión radial (b) Dibujo de patrón

Figura C.1: Uso de tablero de ajedrez para corrección de distorsiones (Tomado de [187]).

Después de realizar el proceso de obtención de la matriz intrínseca y los coeficientes de distorsión, estos se aplican a la imagen tomada por la cámara, solucionando la distorsión radial y tangencial de la imagen, como se ve en la Figura C.2.

Figura C.2: Tablero de ajedrez con distorsión corregida (Tomado de [187]).

Es importante aclarar que cuando una cámara ha sido calibrada a una resolución de imagen, tomando como ejemplo 320 x 240 píxeles de resolución, sus coeficientes de distorsión calculados pueden ser usados para imágenes con otro tamaño de resolución de imagen, para la misma cámara, mientras que la matriz intrínseca debe ser escalada apropiadamente para cada resolución [138].

Anexo D

Manual de puesta en marcha

D.1. Instalación de requisitos

1. Descargar el paquete de software “Anaconda”, en su versión más reciente.
2. Descargar el software “Cuda” del repositorio de “Nvidia” en su versión 10.1.
3. Clonar el respositorio `https://github.com/DaniloCi23UniC/TDGSOSR` en la carpeta deseada.
4. Instalar el software “Anaconda” tal cual se indica en la documentación oficial, una vez instalado.
5. Realizar la búsqueda tal cual se indica en la Figura D.1

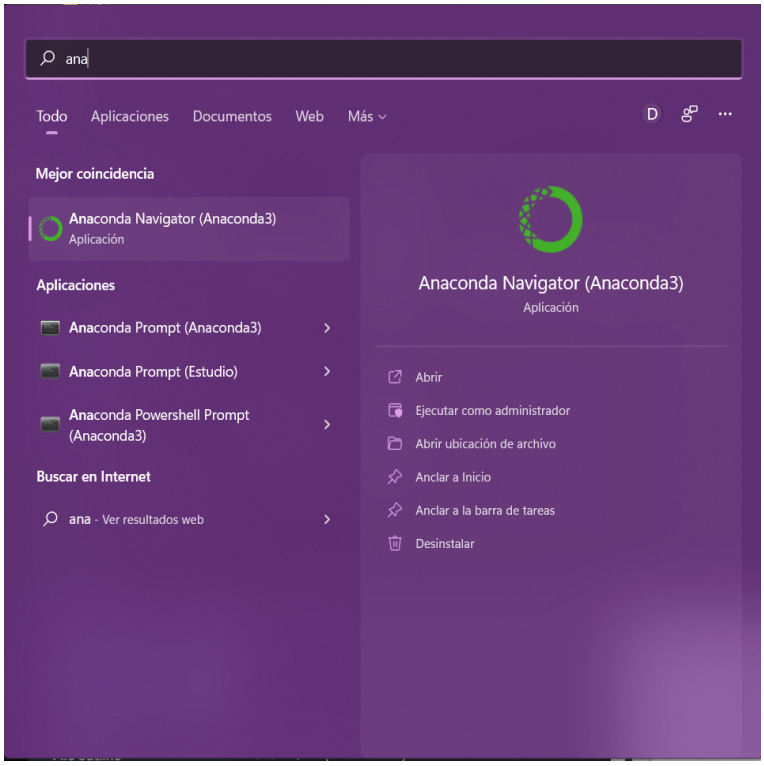


Figura D.1: Anaconda Prompt.

6. Ejecutar como administrador “Anaconda Prompt (Anaconda3)”

7. Ubicar la ventana de comandos en la carpeta donde se clonó el repositorio con el comando “Set-Location” de PowerShell.

8. Una vez ubicado en el directorio ejecutar el comando “ conda env create –file=entorno.yml ”, esto iniciaría la instalación de los paquetes necesarios y la creación de un entorno con Python en su versión 3.8, tal como se indica en la Figura D.2

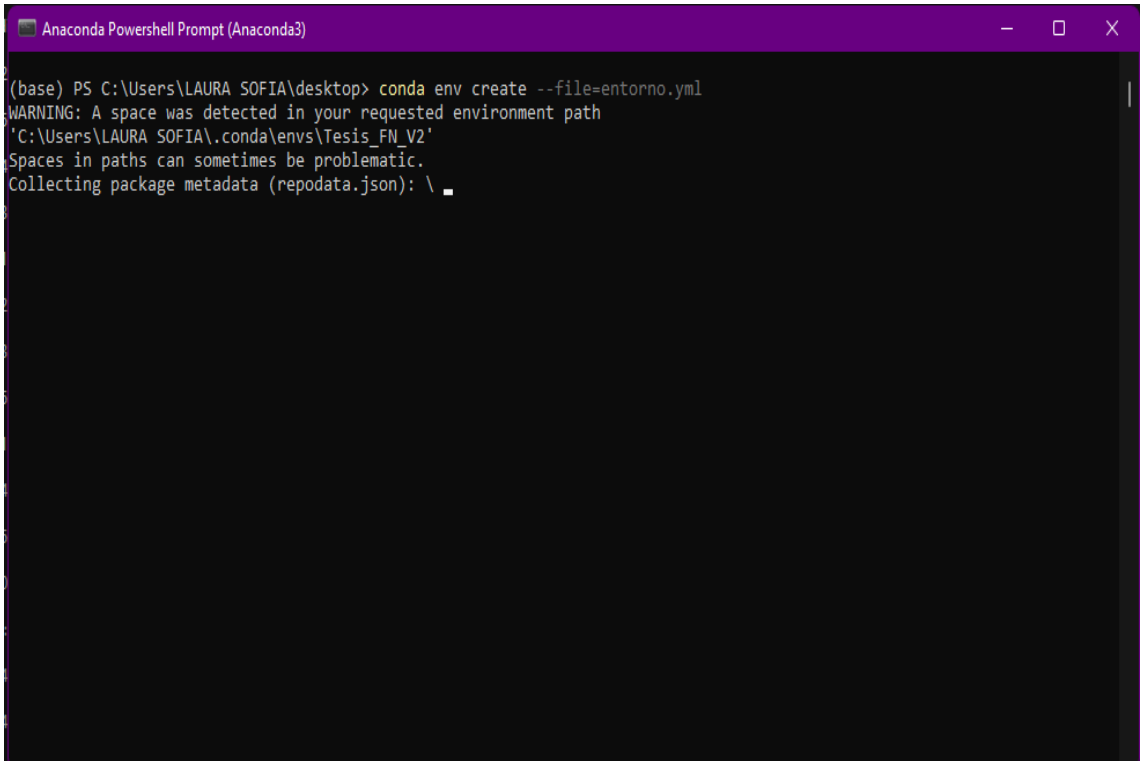


Figura D.2: Anaconda Shell.

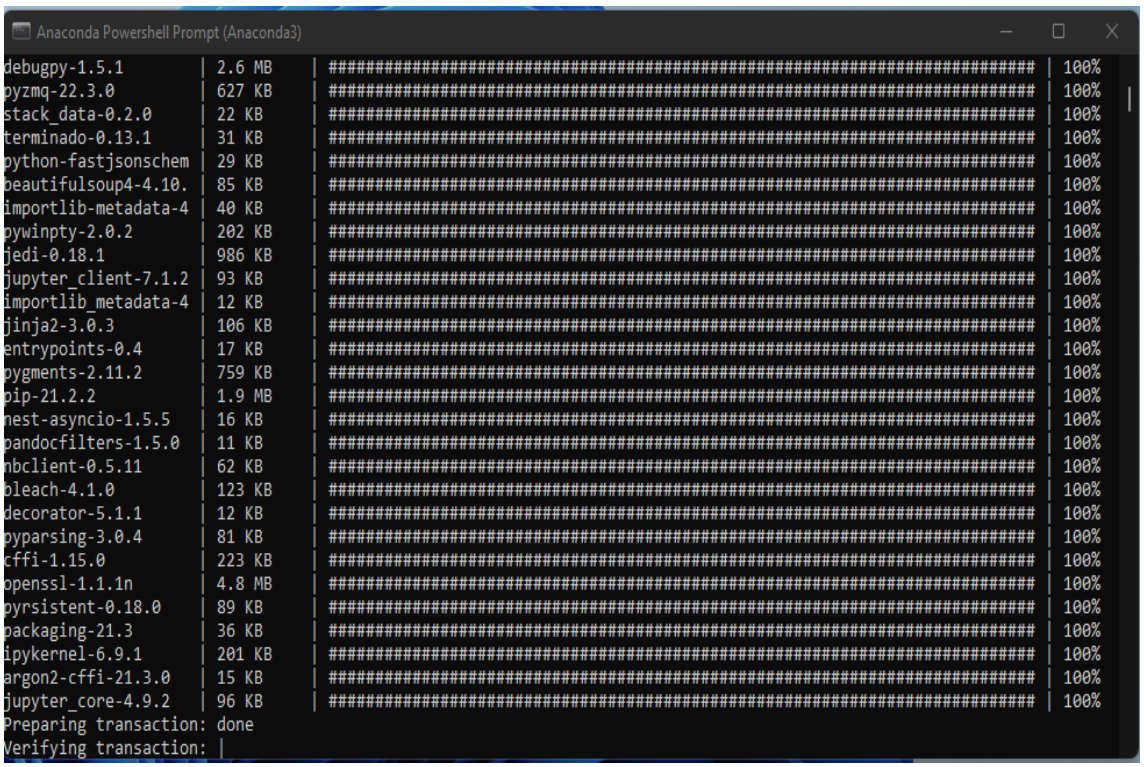


Figura D.3: Anaconda Shell instalación dependencias.

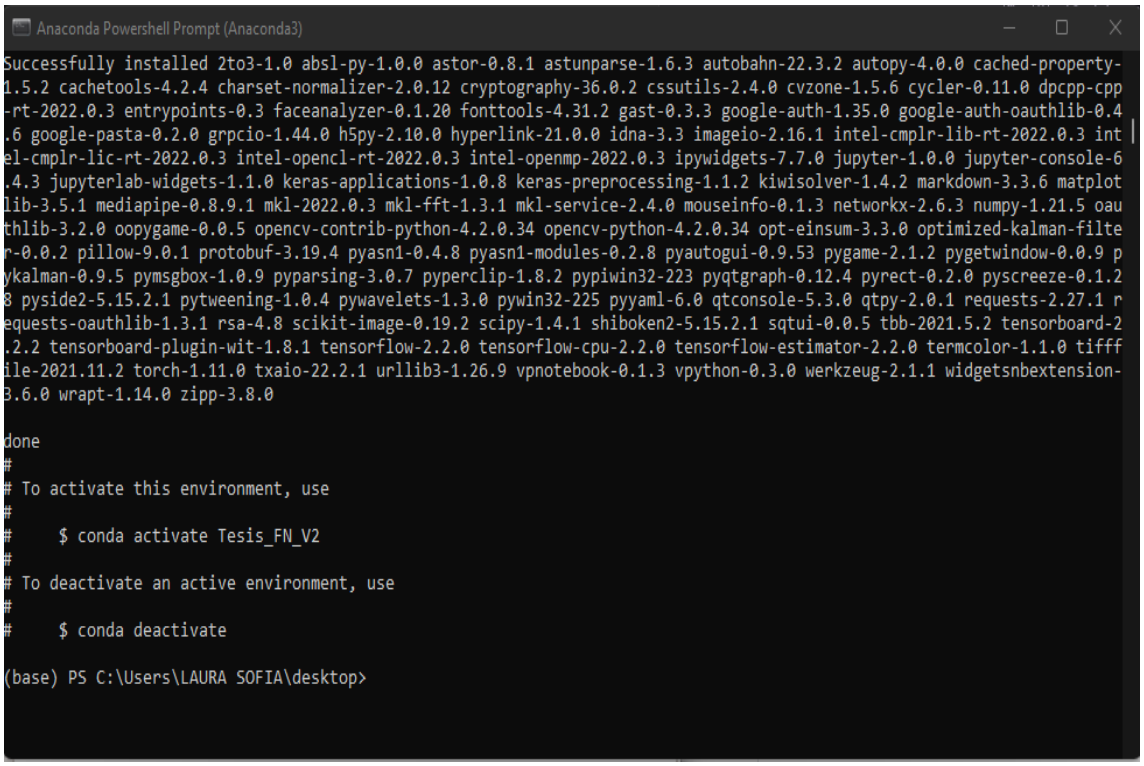


Figura D.4: Anaconda Shell activación de entorno.

9. Una vez finalizada la instalación de paquetes y creación del entorno de Python, copiar los archivos “pythoncom38.dll” y “pywintypes38.dll” a las carpetas “system32” y “syswow64” del sistema windows, tal como se muestra en la figura D.5 y D.6, respectivamente.

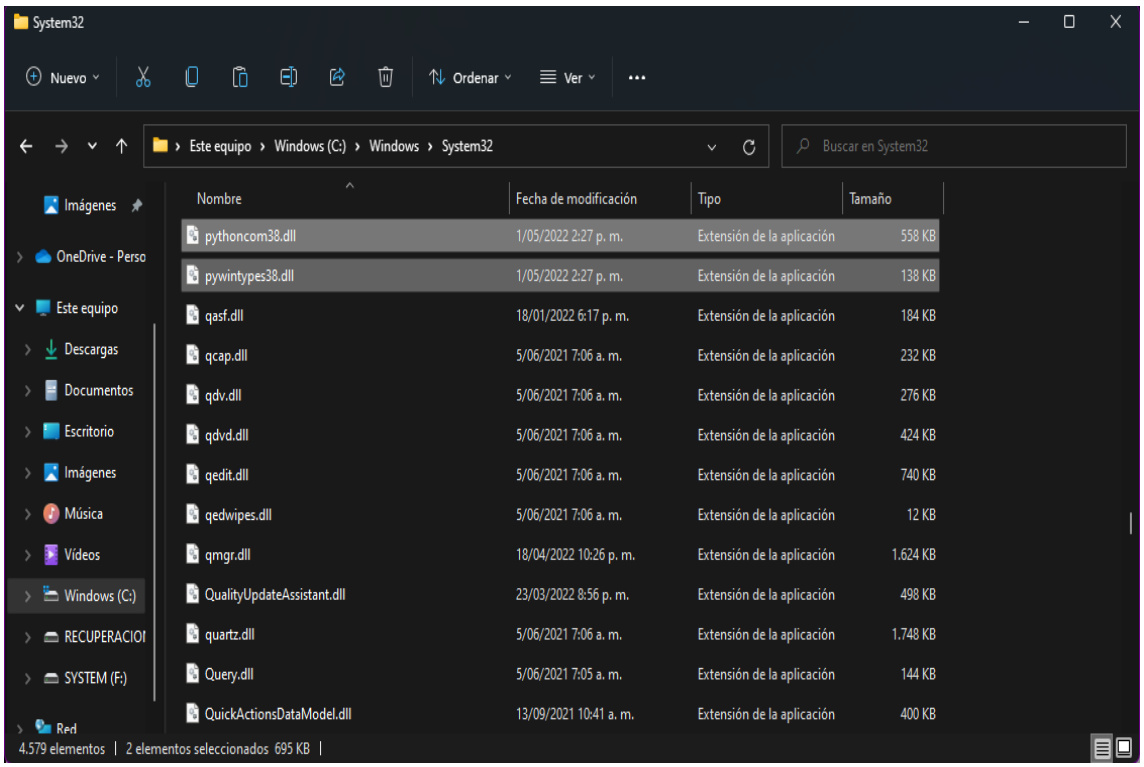


Figura D.5: Copiado de librerías System32.

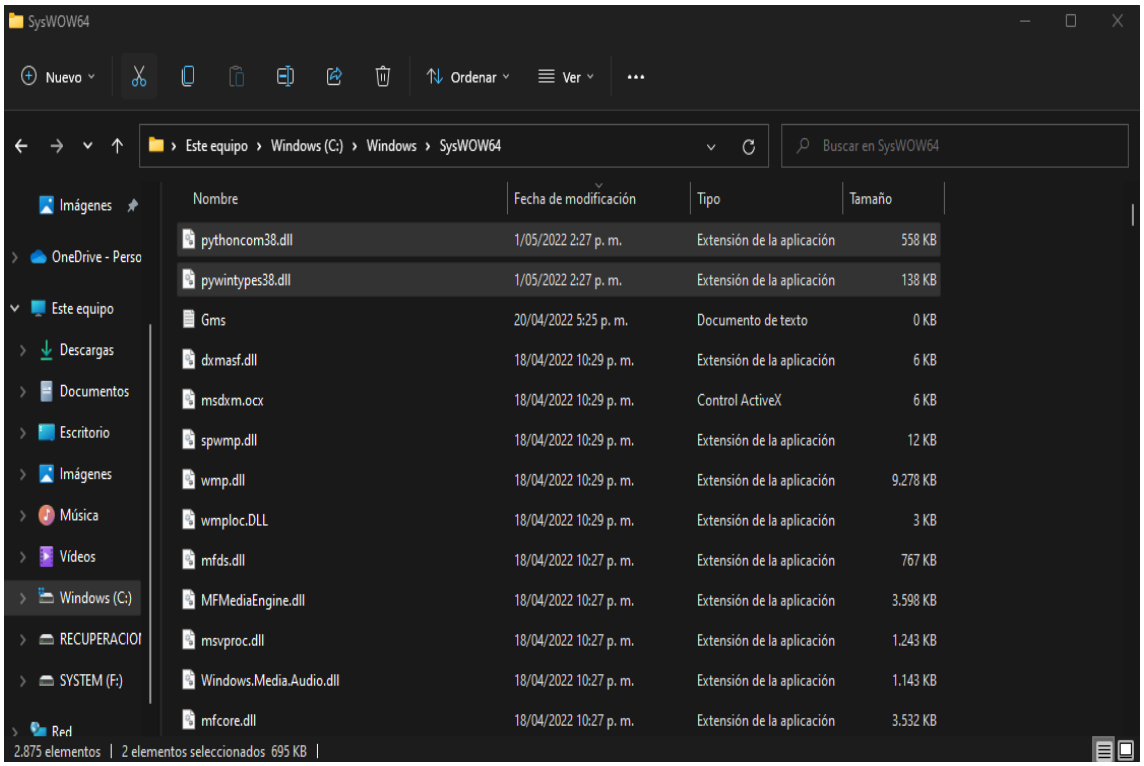


Figura D.6: Copiado de librerías SysWOW64.

10. Una vez finalizada la copia ejecutar el comando “python main.py”, seguidamente la interfaz se mostrará, ver figura D.7.

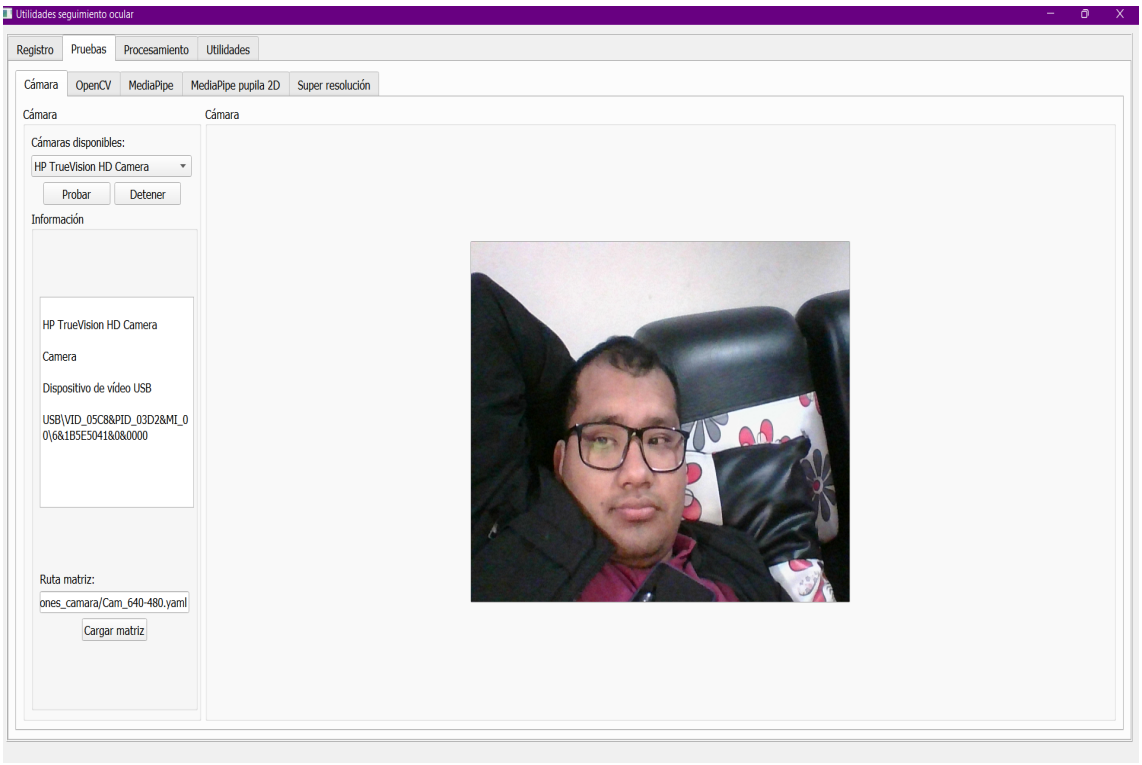


Figura D.7: Interfaz de utilidades.

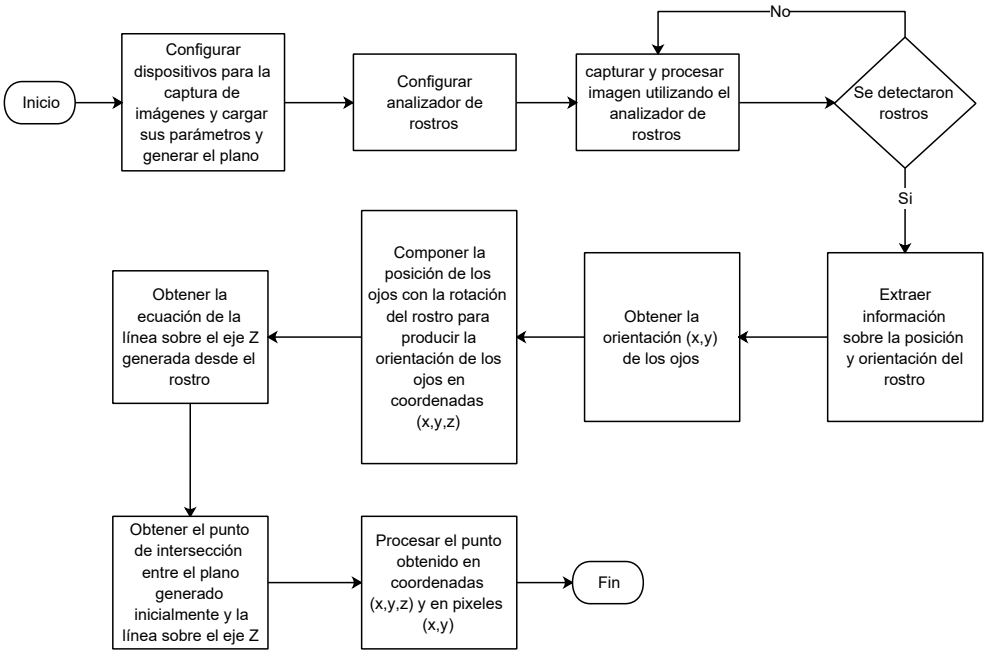


Figura D.8: Diagrama de SO basado en modelo 3D (Fuente propia).

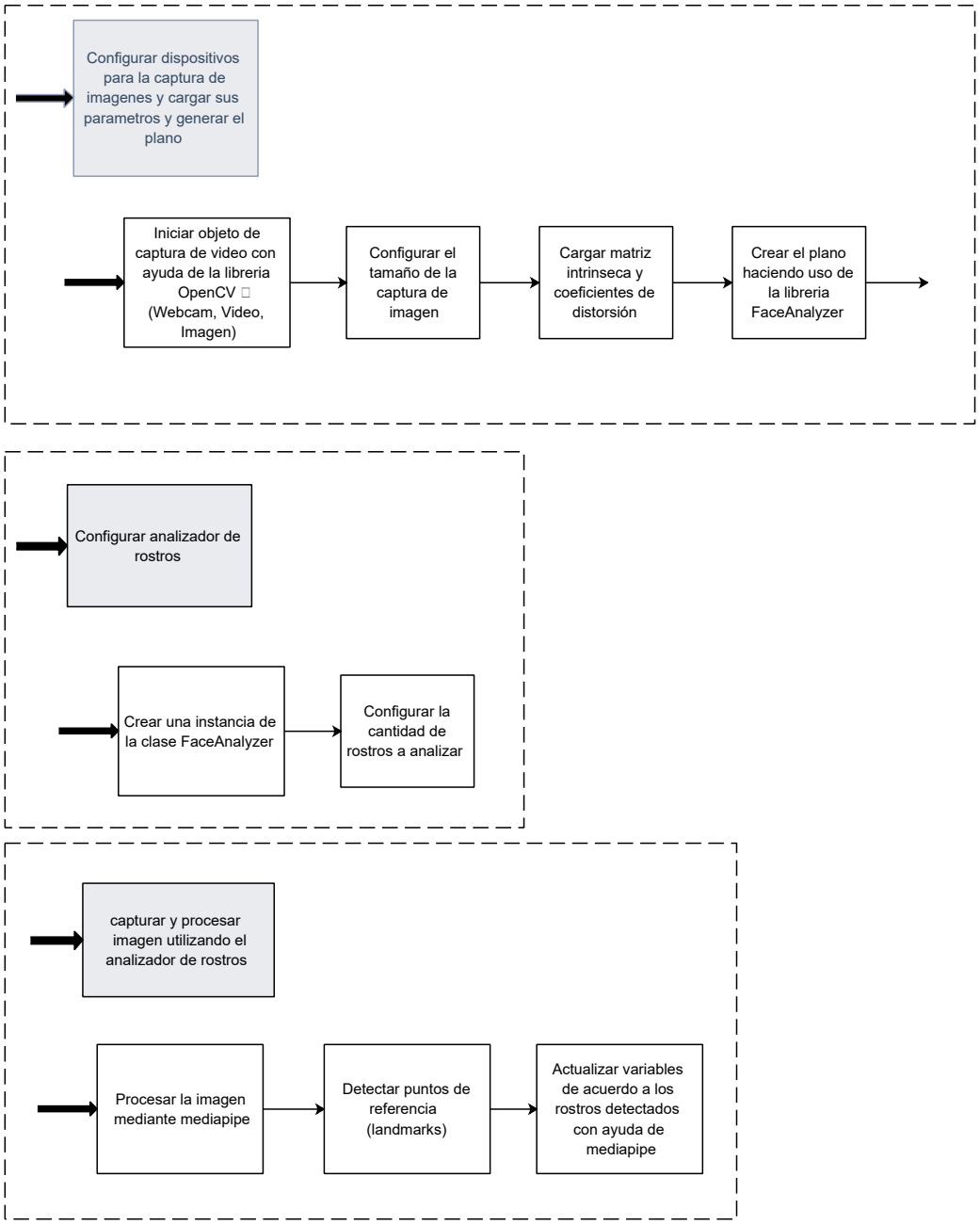


Figura D.9: Diagrama de SO basado en modelo 3D (Fuente propia).

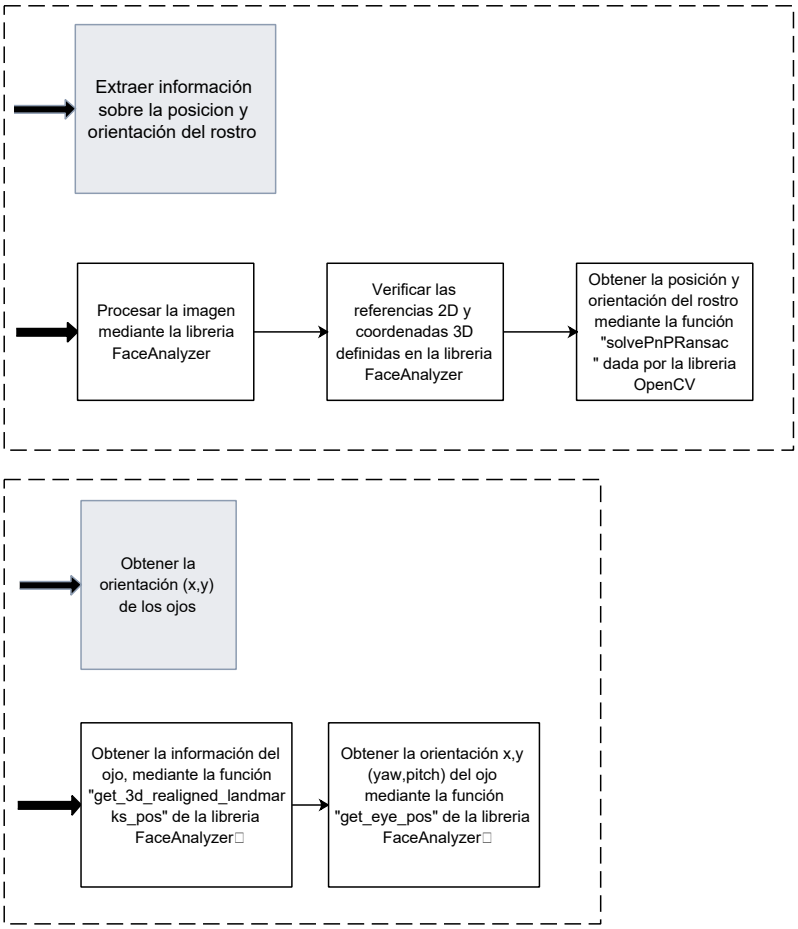


Figura D.10: Diagrama de SO basado en modelo 3D (Fuente propia).