USO DE GAFAS DE REALIDAD AUMENTADA PARA LA MEJORA DE LA AUTONOMÍA DE USUARIOS CON BAJA VISIÓN

USE OF AUGMENTED REALITY GLASSES TO IMPROVE THE AUTONOMY OF USERS WITH LOW VISION

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTORES
IULIUS GHERASIM
IÑAKI BERROCAL DÍAZ
PABLO VALDÉS BALDUZ

DIRECTORES
MARÍA GUIJARRO MATA-GARCÍA
JOAQUÍN RECAS PIORNO

GRADO EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

USO DE GAFAS DE REALIDAD AUMENTADA PARA LA MEJORA DE LA AUTONOMÍA DE USUARIOS CON BAJA VISIÓN

USE OF AUGMENTED REALITY GLASSES TO IMPROVE THE AUTONOMY OF USERS WITH LOW VISION

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTORES

IULIUS GHERASIM

IÑAKI BERROCAL DÍAZ

PABLO VALDÉS BALDUZ

DIRECTORES

MARÍA GUIJARRO MATA-GARCÍA

JOAQUÍN RECAS PIORNO

CONVOCATORIA: JUNIO 2023

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

12 DE MAYO DE 2023

DEDICATORIA

A mi familia y amigos, que siempre han estado ahí para apoyarme y ayudarme cuando lo más lo necesitaba.

Iñaki

A todos mis seres queridos.

Iulius

A mis padres, por estar ahí siempre que lo he necesitado. A Amaya, por ser un gran ejemplo a seguir y un gran apoyo, y a Adriana por creer siempre en mí y animarme siempre.

Pablo

AGRADECIMIENTOS

Los tres integrantes del grupo queremos agradecer a nuestros tutores Joaquín y María el apoyo y la guía que nos han proporcionado a la hora de realizar este trabajo. También queremos mencionar la gran ayuda proporcionada por el equipo de la empresa Plusindes prestándonos el material necesario para el correcto desarrollo del proyecto.

RESUMEN

Uso de gafas de Realidad Aumentada para la mejora de la autonomía de usuarios con baja visión

La realidad aumentada es un campo en auge en el sector de la tecnología con mucho futuro en un amplio espectro de aplicaciones, como la medicina, la industria o el entretenimiento. Esta agrupa un conjunto de tecnologías que permiten que un usuario visualice la realidad a través de un dispositivo electrónico que añade información virtual para que ambos elementos se combinen.

Un uso más concreto de esta tecnología es el de mejorar la calidad de vida de las personas con baja visión. Para ello existen productos específicos que incorporan esta tecnología, como son las gafas Retiplus [1] que proporciona la empresa Plusindes [2]. Este tipo de gafas permiten capturar imágenes del entorno, tratarlas y posteriormente proyectar dichas imágenes ya modificadas en sus cristales.

En este trabajo, hemos desarrollado una aplicación compatible con las gafas Moverio BT-40S [3] de la empresa Epson. El objetivo de esta aplicación es la obtención de imágenes por la cámara del controlador propietario, el procesamiento de estas y su posterior proyección en la pantalla de las gafas.

Palabras clave

Realidad Aumentada, Moverio, Gafas, Baja visión, Pantalla montada en la cabeza, Android, Java, Procesado de imágenes.

ABSTRACT

Use of Augmented Reality glasses to improve the autonomy of users with low vision.

Augmented reality is a booming field in the technology sector with a great future in a wide range of applications, such as medicine, industry, or entertainment. It brings together a set of technologies that allow a user to visualize reality through an electronic device that adds virtual information so that both elements are combined.

A more specific use of this technology is to improve the quality of life of people with low vision. For this purpose, there are specific products that incorporate this technology, such as the Retiplus glasses [1] provided by the company Plusindes [2]. This type of glasses can capture images of the environment, treat them, and then project these images already modified in their lenses.

In this work, we have developed an application compatible with Epson's Moverio BT-40S [3] glasses. The objective of this application is the acquisition of images by the camera of the owner-controller, the processing of these images and their subsequent projection on the screen of the glasses.

Keywords

Augmented Reality, Moverio, Glasses, Low vision, Head-mounted Display, Android, Java, Image Processing

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

Capítulo 1 - Introducción

Esta memoria trata el Trabajo de Fin de Grado <<Uso de gafas de Realidad Aumentada para la mejora de la autonomía de usuarios con baja visión>>. Se detalla el proceso de desarrollo de una aplicación para dispositivos móviles de sistema operativo Android, con la que ayudar a personas con la condición de baja visión a incrementar su autonomía.

La patología anteriormente mencionada, la baja visión [4] [5], es una afección ocular provocada por distintas enfermedades tales como: degeneración macular asociada a la edad, cataratas, glaucoma, retinopatía diabética, etc. Los principales defectos en la visión que causa son: pérdida de visión central, visión periférica, ceguera nocturna y visión borrosa. La particularidad de la baja visión es que no se puede tratar con métodos convencionales, ya sean gafas, lentillas o tratamientos médicos.

Este proyecto surge de una colaboración de la empresa Plusindes, SL con la Universidad Complutense de Madrid. Esta empresa surge en 2014 de la mano de ingenieros industriales e informáticos con la misión de mejorar la vida de las personas con baja visión. El principal producto que ofrecen son Gafas Inteligentes con tecnología de Realidad Aumentada (AR, por sus siglas en inglés), estas gafas además de un software propio que se utiliza en conjunto con ellas constituyen el sistema RETIPLUS®. Este sistema funciona aplicando unos filtros, escogidos y ajustados por un especialista optometrista, a las imágenes que captura la cámara integrada de las gafas y mostrando el resultado en las pantallas de las gafas, en el área específica de la pantalla que el paciente pueda ver. La necesidad de utilizar un modelo específico de gafas provistas de un dispositivo controlador propietario ha llevado a la empresa a buscar alternativas, una aplicación que se pueda usar en una amplia gama de dispositivos móviles que facilite a los pacientes el uso del sistema RETIPLUS®, ahorrándoles dinero y aumentando su comodidad pudiendo utilizar estos su propio teléfono.

El trabajo realizado por los estudiantes que figuran en esta memoria ha dado como resultado una aplicación para teléfonos Android que permite mostrar la previsualización de vídeo, captada por la cámara del dispositivo en tiempo real, en las gafas, modificar diversos ajustes de la cámara y, además, aplicar filtros a lo que se ve

en las gafas utilizando la pantalla del dispositivo móvil. La principal diferencia con otras aplicaciones ya existentes es que la que hemos desarrollado nosotros permite ver en las gafas una cosa y en el teléfono otra, permitiendo así separar la pantalla con los controles de la pantalla de visualización.

1.1 Motivación

Como hemos mencionado antes, la realidad aumentada es un campo en auge el cual tiene una gran cantidad de aplicaciones. Esta tecnología todavía tiene mucho camino por delante, ya que ofrece un sinfín de posibilidades a la hora de investigar, desarrollar y realizar proyectos con ella. Muchos de estos proyectos podrían mejorar nuestra calidad de vida significativamente en un futuro. Ámbitos tan importantes como la medicina, el entretenimiento, la industria o la publicidad ya están introduciendo la realidad aumentada con resultados bastante prometedores.

Sin embargo, la realidad aumentada es relativamente reciente y por tanto conlleva una serie de problemáticas. Algunos de estos inconvenientes son la escasa disponibilidad y el alto precio de los dispositivos que se necesitan para su funcionamiento o la falta de documentación sobre cómo desarrollar aplicaciones en estos dispositivos. También es importante el hecho de que por el momento es una tecnología de nicho, por lo que su uso no está muy extendido y se han investigado relativamente poco.

En cuanto al motivo por el cual escogimos este proyecto, a los tres miembros del grupo nos movieron unos objetivos similares. Por una parte, trabajar con una tecnología tan novedosa y con tanto futuro por delante nos parecía un reto emocionante y de alto valor educativo. Por otro lado, este es un proyecto con el que esperamos poder mejorar la calidad de vida de las personas que tengan baja visión, lo cual fue un factor muy importante para los tres a la hora de escoger este trabajo.

1.2 Objetivos

El objetivo inicial del proyecto es el de realizar una aplicación que pueda ser ejecutada en el dispositivo móvil que se nos ha sido proporcionado junto con las gafas. Esta aplicación debe hacer uso de la cámara de dicho dispositivo y transmitir las

imágenes que capture la cámara a la pantalla de las gafas, las cuales estarán conectadas mediante un cable al controlador. Por otro lado, en la pantalla del dispositivo móvil estará situada una interfaz intuitiva y fácil de usar desde la que se podrá modificar las imágenes que se muestran en las gafas según una serie de filtros o ajustes.

1.3 Estructura de la memoria

En este apartado se hará un resumen de cada uno de los capítulos de la memoria, indicando qué se trata en cada uno de ellos:

- **Capítulo 1: Introducción.**

Se exponen las motivaciones que nos han llevado a realizar este desarrollo, los objetivos a alcanzar en el mismo, la metodología de trabajo elegida, y las diferentes herramientas que hemos utilizado para implementar la aplicación.

- **Capítulo 2: Estado de la cuestión.**

Se hace un estudio de la situación actual de la realidad aumentada en distintos sectores laborales, y se comenta cuál es el estado actual de los dispositivos y software que ofrece Plusindes.

- **Capítulo 3: Desarrollo de la aplicación.**

Se comenta como se ha llevado a cabo la implementación de la aplicación, dividiendo los hitos obtenidos en cada una de las distintas iteraciones que se han realizado.

- **Capítulo 4: Conclusiones y trabajo futuro.**

Concluimos nuestra investigación y comentamos posibles mejoras o implementaciones complementarias a la nuestra que se podrían llegar a desarrollar en un futuro.

- **Bibliografía.**

Se enumeran las referencias a las distintas fuentes que hemos utilizado para informarnos e investigar sobre los temas concernientes al proyecto y así poder realizar las distintas partes de este.

1.4 Plan de trabajo

La metodología empleada a la hora de afrontar el desarrollo de este proyecto ha sido el Extreme Programming (XP, por sus siglas en inglés) [6]. Esta metodología propuesta por Kent Beck surge a finales de los años 90 y tiene como pilares fundamentales las iteraciones de desarrollo cortas y la realización constante de pruebas.

Este último aspecto, la adaptabilidad, nos ha resultado extremadamente útil en la realización del proyecto ya que a lo largo de este han aparecido multitud de problemas y ha habido cambio en los objetivos constantemente.

Hemos implementado esta metodología realizando un desarrollo iterativo e incremental de la aplicación implementando pequeñas mejoras con cada iteración. Así, en la primera versión de la aplicación se realizó una implementación mínima esta y en las siguientes se fueron añadiendo las distintas funcionalidades. Con esto conseguimos responder de manera rápida a los cambios en los requisitos y tener retroalimentación constante de los tutores del trabajo y del personal de Plusindes al poderse mostrar todos los cambios realizados durante cada iteración.

Respecto a la mayor parte de la programación la hemos realizado entre los tres a la vez, y de no poder ser así como mínimo en pareja. Esto con el propósito de que todos conozcamos la labor de todos y el código nunca quede sin revisar, ya que era indispensable que se conociese la implementación de cada aspecto de la aplicación por cada miembro del proyecto debido a las características de este.

En la figura 1.1 se muestra un diagrama de Gantt el cual representa cada una de las etapas del proyecto y el tiempo que se ha necesitado para completarlas.

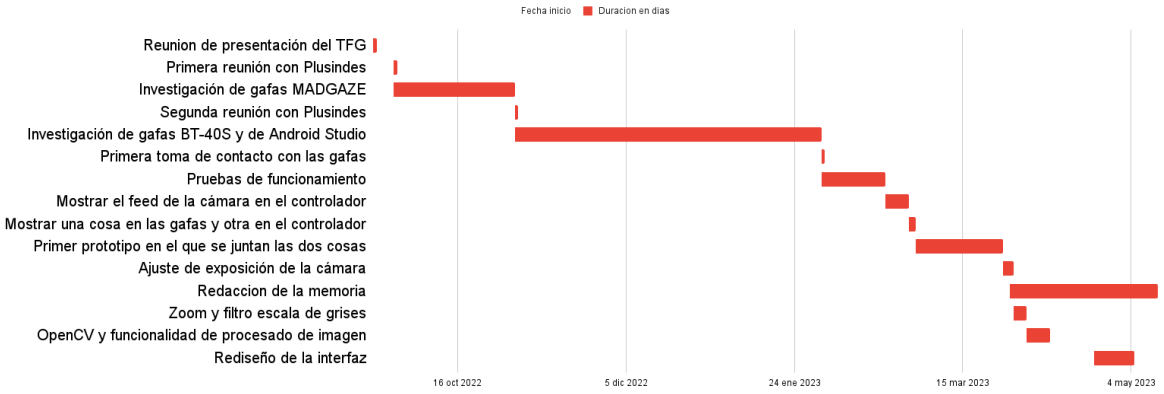

Figura 1.1 Diagrama de Gantt resultante del proyecto

Como se ve en el diagrama empezamos el proyecto reuniéndonos con el equipo de Plusindes varias veces e investigando sobre los productos que nos prestaron, así como sobre las tecnologías que íbamos a utilizar durante el desarrollo. Más adelante, y una vez tuvimos acceso libre a las gafas, realizamos pruebas e investigación de su funcionamiento. Una vez ya nos habíamos familiarizado con ellas, empezamos a desarrollar las diferentes iteraciones del desarrollo hasta llegar a la iteración final con todas las funcionalidades y la interfaz rediseñada y mejorada.

1.5 Entorno y herramientas de trabajo

1.5.1 Herramientas software

A continuación, vamos a hablar de las herramientas software utilizadas para el desarrollo de nuestro proyecto.

1.5.1.1 Android Studio

Android Studio [7]es un entorno de desarrollo para Android que utiliza el software IntelliJ IDEA, perteneciente a JetBrains, y se anunció en 2013 como reemplazo de Eclipse. Se ha creado específicamente para la creación de aplicaciones Android. Este IDE tiene como lenguaje predeterminado Kotlin, aunque también permite utilizar otros, como C++ y Java [8].

Toda la parte visual de nuestra aplicación ha sido creada utilizando este entorno, que nos permite crear y editar las vistas de una manera sencilla, arrastrando los

elementos que queremos introducir en ellas. De igual manera, toda la parte funcional también la hemos desarrollado haciendo uso de Android Studio, junto con la librería OpenCV.

1.5.1.2 Biblioteca OpenCV

OpenCV [9] es una biblioteca libre de visión por computador que apareció en 1999 y hasta día de hoy sigue siendo una de las bibliotecas más famosas en este ámbito. Algunos ejemplos de aplicaciones en las que se utiliza OpenCV son, entre otras: reconocimiento de objetos, tracking, reconocimiento facial y de gestos o realidad aumentada. Destaca porque es una librería ampliamente documentada, es de uso libre y multiplataforma.

El uso de esta librería ha sido lo que ha permitido lograr algunas de las partes más importantes del proyecto. Su uso concreto se detalla más adelante en la sección de desarrollo.

1.5.2 Herramientas hardware

A continuación, se explican los dispositivos físicos que nos ha proporcionado la empresa Retiplus para el desarrollo del proyecto, con los cuales hemos ido comprobando el funcionamiento de la aplicación a medida que íbamos avanzando en las diferentes iteraciones.

1.5.2.1 Gafas inteligentes Moverio BT-40S

Una de las herramientas hardware principales con las cuales hemos trabajado son las gafas **Moverio BT-40S,** cuyo diseño se puede ver a continuación en la figura 1.2.

Figura 1.2 Gafas Moverio BT-40S con el controlador BO-IC400

Estas son unas gafas inteligentes, creadas por Epson, una empresa japonesa fundada en 1942 que se dedica a la fabricación de impresoras, proyectores, televisores, y gafas de realidad aumentada, entre otros componentes electrónicos.

Las gafas consiguen implementar una realidad aumentada proyectando dos imágenes diferentes sobre cada uno de sus cristales, los cuales las redirigen hasta nuestros ojos. Estas se combinan con el fondo y se unen en nuestro cerebro para dar lugar a la imagen final que vemos nosotros [10].

1.5.2.2 Controlador Inteligente Moverio BO-IC 400

La otra herramienta hardware clave para el desarrollo del proyecto ha sido el Controlador Inteligente Moverio BO-IC 400 [11], el cual también podemos ver en la anterior figura 1.2.

Este es un dispositivo móvil, también de la marca Epson, basado en Android y diseñado para usarse junto con unas gafas inteligentes de Moverio. Tiene una pantalla táctil integrada y varios botones que permiten al usuario interactuar con él.

También dispone de varios sensores avanzados, como una brújula, sensores de movimiento de 9 ejes, GPS o un acelerómetro. Además de estos, tiene incorporados también altavoz, micrófono y cámara de alta resolución, la cual tiene enfoque automático y una linterna.

Para conectar el dispositivo a las gafas, existe un puerto USB tipo C con compatibilidad DisplayPort Alt, y otro más que se utiliza para cargar el dispositivo o conectar periféricos externos. A parte, las gafas también incluyen compatibilidad con un reducido abanico de teléfonos móviles, los cuales se pueden conectar a ellas de igual manera que el controlador.

1.5.3 Otras herramientas

1.5.3.1 Discord

Discord [12] es un servicio multiplataforma y gratuito de chat de voz y mensajería instantánea creado en 2015. Éste está dividido en servidores, que a su vez están divididos en canales de voz y de texto. Discord permite a los usuarios realizar llamadas o intercambiar mensajes independientemente de si estos utilizan la versión web o la aplicación, e independientemente del dispositivo que estén usando.

Lo hemos utilizado para realizar la mayoría de las llamadas y trabajo en equipo, gracias a las funciones anteriormente mencionadas de chat de voz y de texto, además de la función de compartir pantalla que ha resultado muy útil a la hora de programar en conjunto.

1.5.3.2 Google Drive

Google Drive [13] es un servicio utilizado para alojar archivos creado por Google en 2012. Permite a los usuarios almacenar, descargar, editar y sincronizar archivos en la nube y compartirlos con otros usuarios. También cuenta con un sistema de control de versiones y con un sistema de respaldo automático de los archivos.

Lo hemos utilizado para llevar una carpeta compartida con distintos archivos informativos, la memoria, así como las actas de las reuniones que hemos ido llevando a cabo.

1.5.3.3 GitHub

GitHub [14] es un servicio creado para almacenar proyectos haciendo uso del sistema de Git para el control de versiones, perteneciente ahora a Microsoft. Este permite realizar un seguimiento de todos los cambios que se realizan en el código de

una aplicación en desarrollo. Además, esta plataforma también nos permite trabajar en distintas versiones, llamadas ramas, del código de manera simultánea y después unirlas en una sola versión, así como trabajar de manera remota de manera sencilla.

Hemos utilizado GitHub para crear el repositorio de nuestro proyecto, en el que hemos ido guardando todas las versiones de nuestro código y las distintas ramas. De esta manera podíamos compartir el código y controlar todos los cambios que se hacían sobre el mismo.

Capítulo 2 - Estado de la cuestión

2.1 Realidad aumentada en la actualidad

Hoy en día, la realidad aumentada es una tecnología que ha visto incrementado su uso notablemente en diferentes sectores del mundo laboral, y cuyo estudio continúa haciendo avances novedosos que nos traen herramientas cómodas con las que ayudarnos en nuestro día a día.

2.1.1 Realidad aumentada en logística

En el sector logístico, los avances logrados en AR han permitido introducir cambios que agilizan el trabajo, haciendo las tareas de recogida y transporte de materiales más sencillas [15]. Los sistemas de gestión actuales existentes en los almacenes, basados en papel y en programas antiguos o poco intuitivos, provocan un trabajo costoso y lento: se tarda mucho tiempo en organizar y contabilizar el inventario, puede ser a veces pesado, lo cual puede provocar que los trabajadores se ralenticen o se equivoquen y por último, estos sistemas requieren un gasto extra tanto de tiempo como de dinero en mantenerlos.

Es aquí donde entra la realidad aumentada, introduciendo cambios como gafas que permiten a los trabajadores visualizar dónde se localizan los objetos a recoger, identificarlos y posteriormente saber dónde se tienen que depositar. En la figura 2.1 podemos ver a un trabajador en un almacén que hace uso de unas gafas de AR, así como de un escáner de códigos de barras para obtener información sobre los paquetes a enviar. También existen sistemas de recogidas de objetos basados en frecuencias de radio, en el que los trabajadores llevan un dispositivo que, por medio de estas frecuencias, les informa de cuántos objetos tienen que recoger, dónde están, y les permite escanear códigos de barras, para notificar rápidamente de que han llevado a cabo una recogida. Con esto, se puede prescindir del papel, caro y dañino para el medioambiente, y se establecen unos modelos de trabajo más intuitivos para los trabajadores y menos propensos al error humano.

Figura 2.1 Un trabajador utilizando gafas de AR, Fuente: https://www.istockphoto.com/es

2.1.2 Realidad aumentada en industria

El sector de la industria también se ha visto beneficiado por el uso de la realidad aumentada [16]. Avances como la introducción de los robots industriales ha significado un gran aumento en la eficiencia de trabajo y un abaratamiento de costes. Y es que la robótica está ampliamente relacionada con la realidad aumentada, ya que esta tecnología permite a los usuarios interactuar con el mundo virtual, modificando los objetos a reconocer por los robots, las tareas que van a realizar y cómo las van a llevar a cabo, etc.

2.1.3 Realidad aumentada en videojuegos

Aunque en el mundo de los videojuegos es ampliamente más conocido el uso de las gafas de realidad virtual, como las Oculus Rift, es cierto que en los últimos años ha aumentado el uso de la AR. De hecho, el primer videojuego que utilizó AR se remonta al año 2000: ARQuake [17]. Esta fue una versión del famoso juego Quake que utilizaba esta tecnología para mostrar en el mundo real escenarios, enemigos y otros elementos virtuales, ver figura 2.2.

Figura 2.2 Interfaz del juego ARQuake, Fuente: https://ultimatehistoryvideogames.jimdofree.com/arquake

Un ejemplo del uso de la AR en el mundo del videojuego en la actualidad es Pokémon Go [18]. Este juego, lanzado a nuestros móviles en julio de 2016, nació con un objetivo claro: fusionar el mundo real y el virtual para crear una aplicación móvil que permitiera a la gente capturar Pokémon allá donde fueran. El juego consta de dos partes importantes: por una parte, **el desplazamiento**: a medida que tú te desplazas, tu personaje se va moviendo por un mapa creado mediante geolocalización, y descubre nuevas localizaciones y obtiene objetos, y por otra, **la captura**: cuando vas a combatir o capturar a un Pokémon, tienes la opción de activar la AR para visualizar al Pokémon en el mundo real tal y como se puede observar en la figura 2.3:

Figura 2.3 Ejemplo de la interfaz de captura de un Pokémon con AR, Fuente: https://www.imore.com/best-places-use-ar-mode-pokemon-go

Entre los beneficios que traen los juegos que implementan realidad aumentada destaca sobre todo que promueven el movimiento y la actividad física [19]: la mayoría de ellos basan parte de su contenido en desplazarte por distintos lugares, ya sea para conseguir objetos nuevos o capturar Pokémon, derrotar enemigos o cumplir misiones. Además, al jugarse la mayoría en espacios abiertos, y con posibilidad de modalidades online y cooperativas, impulsa a los jugadores a interactuar y socializar con otros jugadores, frente a los videojuegos tradicionales, donde los jugadores pueden a veces sentirse solos.

2.1.4 Realidad aumentada en educación

En el sector de la educación también se ha hecho cada vez más común el uso de esta tecnología *[20]*. Destacan entre otros el uso de las pizarras inteligentes, que permiten al profesor proyectar diferentes elementos en la pizarra y utilizar un lápiz para poder navegar por ella o escribir. También cabe mencionar pequeños dispositivos, como tabletas o dispositivos móviles que, al visualizar unos libros especiales, muestren pantallas con información, imágenes e incluso vídeos, como se puede observar en la figura 2.4. El

uso de la AR en esta rama puede hacer que ciertas asignaturas o partes de ellas, que pueden ser más complejas de entender, o que necesiten de alguna capacidad de abstracción, puedan ser explicadas mejor mediante elementos virtuales, que además logran que la información sea más amena y fácil de entender para los alumnos.

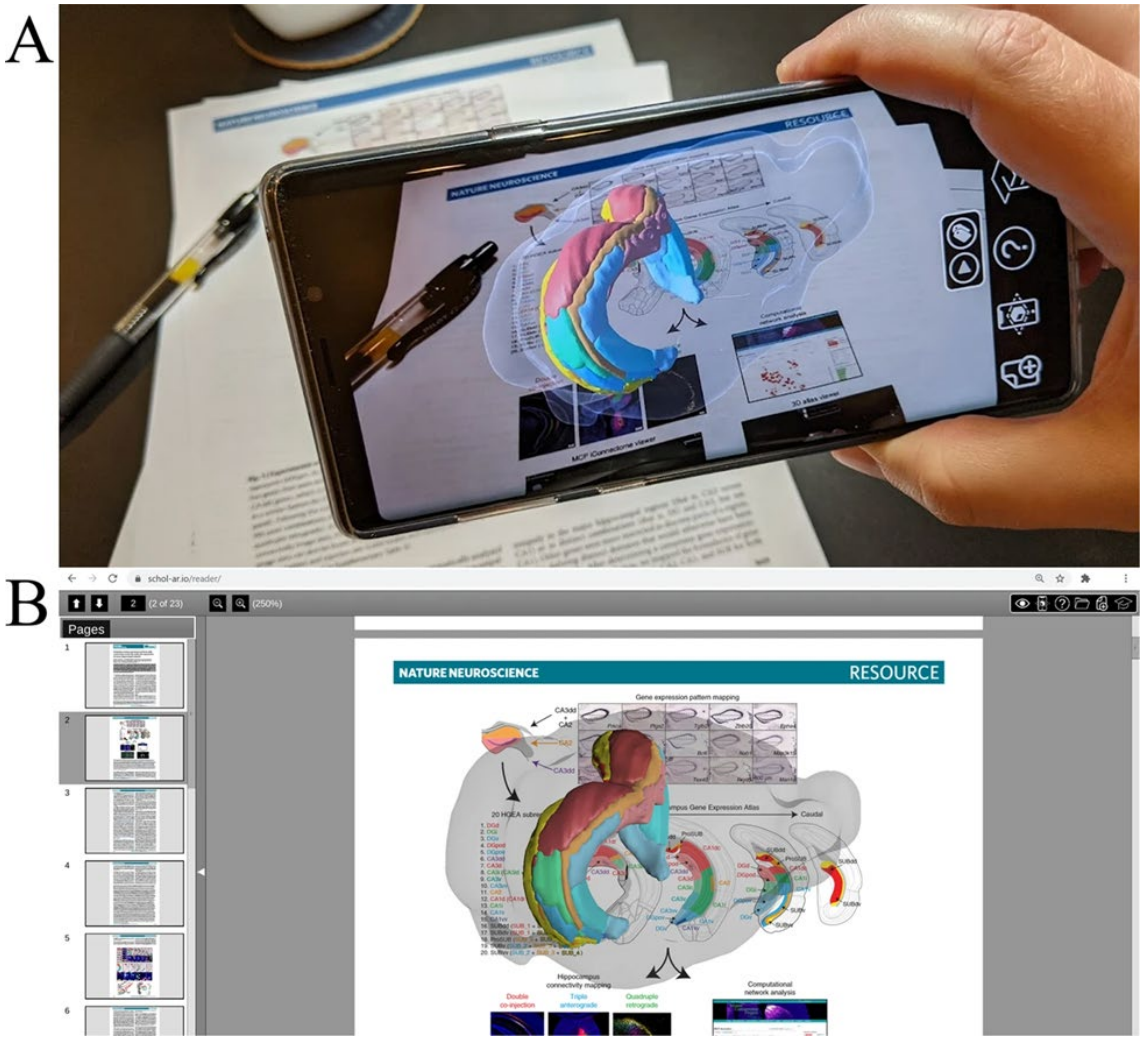

Figura 2.4 Ejemplo de datos visualizados a través de (A) una aplicación móvil de realidad aumentada y (B) un visor PDF basado en web, Fuente: Integrating Data Directly into Publications with Augmented Reality and Web-Based Technologies – Schol-AR

2.1.5 Realidad aumentada en medicina

Por último, en medicina también ha destacado el uso cada vez mayor de la realidad aumentada [21]. En los quirófanos es muchas veces indispensable poseer una gran precisión, pues en una operación se debe saber en todo momento dónde se debe operar, en que zona se encuentra un tumor u otros elementos que se deban retirar, etc.

En la actualidad se han logrado implementar herramientas que permiten a los cirujanos ver en tiempo real dónde están ubicados los órganos de los pacientes, los tumores, que muchas veces están ocultos o permiten a otros médicos visualizar la operación desde otro lugar mediante gafas. Esto permite operar con extrema exactitud, reduciendo al mínimo las posibilidades de que se produzca un error humano, y potencialmente mejorando la eficiencia de muchas operaciones que antes eran demasiado arriesgadas.

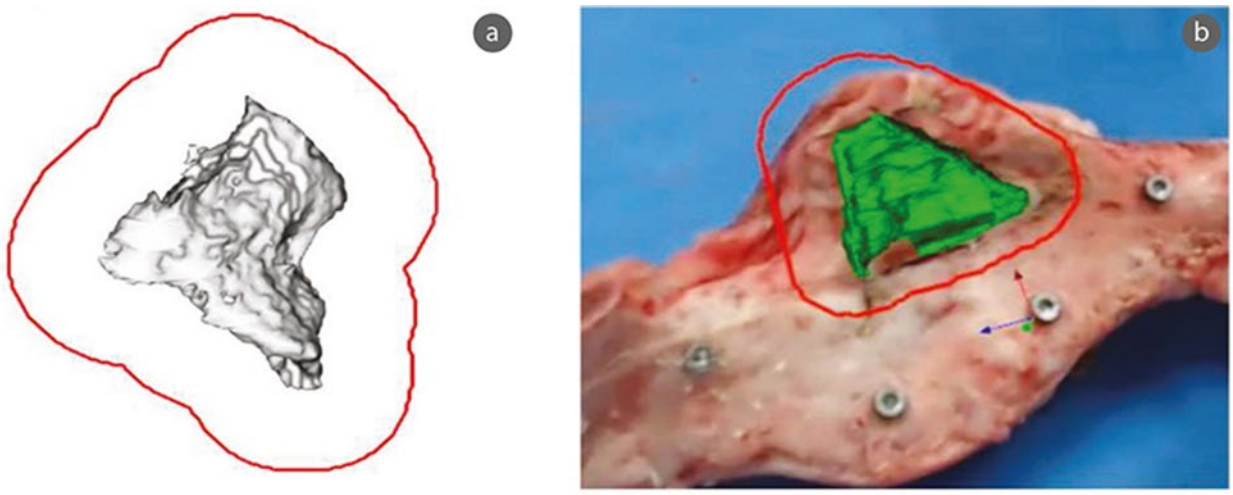

Figura 2.5 Modelo 3D de un tumor óseo (a) e imagen capturada utilizando el sistema de AR (b), Fuente: https://synapse.koreamed.org/articles/1044286

Pero no solo destaca la realidad aumentada en la cirugía. Otro sector importante es el de salud y bienestar, y aquí es donde destaca Plusindes.

2.2 Plusindes

Plusindes se dedica al desarrollo de esta tecnología centrado en ese último punto: gafas de realidad aumentada diseñadas para ayudar a las personas con problemas de baja visión como glaucomas, Retinosis Pigmentaria, Usher, etc. Mediante las gafas y su cámara, los pacientes son capaces de visualizar las imágenes que captura ésta, pero calibradas por técnicos para que sean adecuadas a las capacidades visuales que tienen según sus patologías. También les sirven a los médicos especializados, para saber cómo ven los pacientes, lo cual les sirve de ayuda para investigar las distintas afecciones y cómo calibrar las gafas para conseguir la mayor eficiencia al usarlas con ellos.

Con los últimos modelos del mercado de gafas de AR Retiplus ha conseguido implementar gafas que permiten proyectar en las lentes las imágenes que captan la cámara integrada, junto a un mando que sirve de controlador, para modificar las imágenes visualizadas para que sean aptas para el paciente.

Sin embargo, para favorecer la comodidad de los usuarios, era buena idea realizar un desarrollo que combinara dichas gafas con un dispositivo móvil personal, no necesariamente con el controlador. Por tanto, comenzamos a realizar nuestro desarrollo: combinar la cámara del dispositivo con las gafas, de manera que proyectemos en las lentes las imágenes capturadas por el dispositivo, añadiendo además en la pantalla del dispositivo un menú sencillo con algunos parámetros de la cámara para modificar, como el brillo o escala de grises.

Capítulo 3 - Desarrollo

Como hemos mencionado anteriormente, la aplicación ha sido desarrollada mediante la metodología **Extreme Programming** [6], es decir, mediante iteraciones incrementales en las que se añaden las distintas funcionalidades de la aplicación.

Por este motivo hemos decidido estructurar este capítulo en diferentes partes, cada una de las cuales corresponde al ciclo de vida completo de una iteración. Además, en la sección 3.1 se explica el planteamiento inicial de la aplicación, con los casos de uso propuestos y una descripción general de los objetivos a cumplir en el proyecto. En la sección 3.9 se explican en detalle las problemáticas que han surgido a lo largo del desarrollo.

3.1 Concepto de la aplicación

El concepto de la aplicación estuvo claro desde un principio: una aplicación que, al conectar las gafas AR permitiese ver en estas el vídeo en directo que se esté captando por la cámara integrada, por un lado, y, por otro, tuviese en la pantalla del teléfono conectado una sección de ajustes con parámetros modificables y filtros aplicables. En la figura 3.1 se ilustra el concepto de la aplicación.

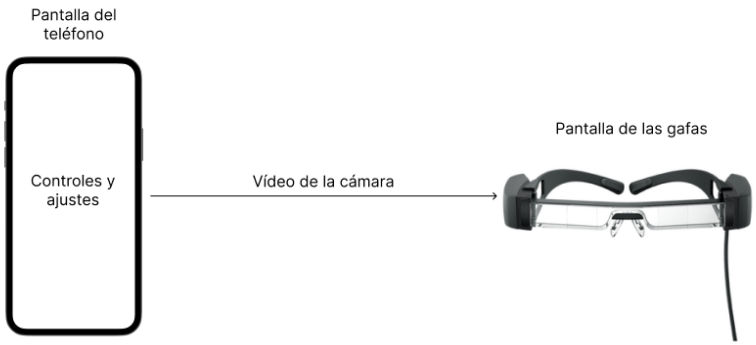

Figura 3.1 Diagrama del concepto de la aplicación

Teniendo el concepto de la aplicación definido se comenzó por estudiar el funcionamiento de las gafas Moverio. Se empezó por probar la aplicación de ejemplo que proporciona Epson en su página web. Esta aplicación de prueba permite ver

información variada sobre las gafas Moverio que estén conectadas en ese momento. Se puede ver la información sobre el modelo de las gafas, los sensores integrados en estas e incluso se puede visualizar una demostración de subtítulos. A la vez que aprendíamos cómo funcionaban las gafas empezamos a recabar información sobre el funcionamiento de Android y cómo programar aplicaciones para este sistema operativo. Para programar en Android hay que dividir todas las funcionalidades que se desean implementar en Activities [22]. Una Activity es la forma que tiene el usuario de interactuar con la aplicación, suelen representar cada una de las pantallas visibles en la aplicación. Una cosa de la que nos dimos cuenta al principio fue que no podríamos utilizar la cámara en Android de manera sencilla, utilizando los Intents [23]. Los Intents en Android permiten realizar acciones sencillas simplemente llamando a estos Intents en una Activity. Un ejemplo de Intent es el de iniciar una llamada o compartir un archivo. Para utilizar la cámara se pueden utilizar Intents para tomar una fotografía o grabar un vídeo, pero, lo malo de estos, es que se ciñen solo a la actividad que describen y no se pueden modificar. Por ello nos vimos obligados a utilizar la cámara de manera manual y personalizar el resultado que queríamos obtener.

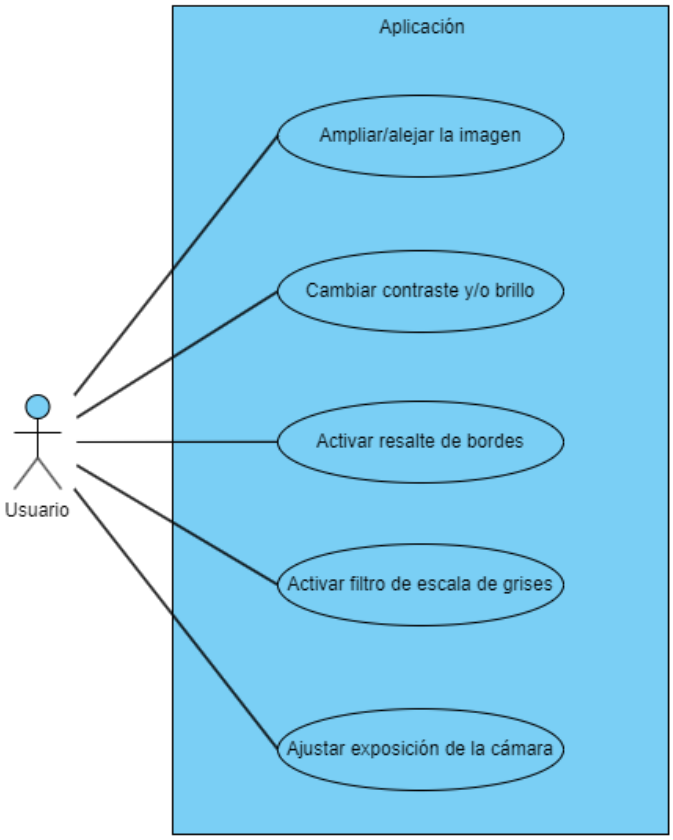

Figura 3.2 Diagrama de casos de uso de la aplicación

En la figura 3.2 se representan los distintos casos de uso que se podrían dar mientras un usuario utiliza la aplicación. Todos estos casos se han tenido en cuenta a la hora de desarrollar e implementar las funcionalidades finales de la aplicación.

3.2 Pruebas de funcionamiento y primeros prototipos de las funcionalidades

La primera iteración del desarrollo tenía como principal objetivo poder compilar una aplicación Android que permitiese al usuario visualizar una vista previa de la cámara dentro de la propia aplicación. Durante esta parte del proceso aprendimos a utilizar las herramientas que proporciona Android para facilitar la comunicación entre la capa de aplicación y la capa de hardware del dispositivo móvil, el paquete *camera2* [24], que da acceso a la funcionalidad de la cámara.

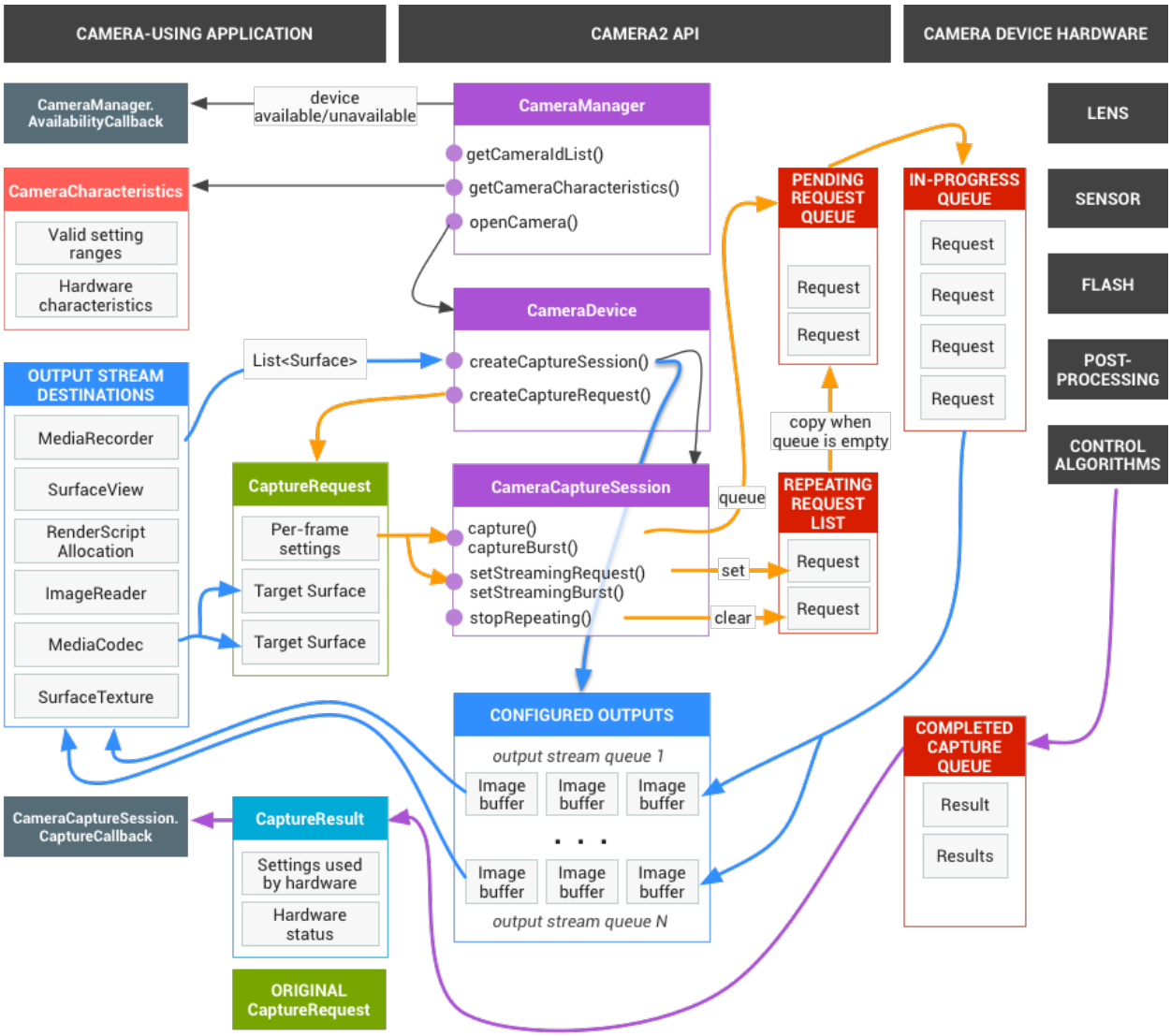

Figura 3.3 Diagrama del modelo de cámara camera2, Fuente: https://source.android.com/docs/core/camera/camera3_requests_hal?hl=es-419#requests

En el diagrama anterior, figura 3.3, aparece representada la arquitectura de comunicación entre la aplicación, la API de camera2 y el hardware de la cámara. Para poder utilizar la cámara en una aplicación sin tener que recurrir a los Intents y en su lugar, poder acceder directamente a los datos que proporciona el hardware, es necesario emplear las clases proporcionadas por camera2. El manejo de la cámara por parte de esta API requiere de una clase CameraManager [25], esta clase se encarga de obtener el listado de todas las cámaras disponibles en el dispositivo, las características específicas de cada una de estas cámaras y, finalmente, de crear una instancia para cada una de estas cámaras para después abrirlas. Una vez completado este paso,

desde la aplicación se puede comprobar la disponibilidad de las cámaras, así como las características de estas y ajustar las funcionalidades que ofrecerá la aplicación al usuario en base al nivel que ofrezca el hardware [26].

El anteriormente mencionado nivel del hardware de la cámara establece cuáles son las características a las que camera2 puede acceder y manipular, incluyéndose entre ellas la cantidad y eficiencia de fotogramas procesados y también la posibilidad de modificar los ajustes de cada fotograma individual, pudiéndose alterar, entre otros: la exposición, el balance de blancos o el ruido de la imagen. La aplicación, después de obtener toda la información sobre las características físicas de la cámara, deberá proporcionar una <<vista>>, una matriz de píxeles en pantalla en los cuales se mostrará la información que se obtiene a través de la API de camera2. Existen distintos tipos de vista, cada uno útil para distintos casos de uso concretos, pero los dos principales más usados para renderizar vídeo en tiempo real son TextureView [27] y SurfaceView [28] , estos dos tipos de vista se explorarán más en detalle posteriormente en la memoria. Tras proporcionar una vista en la cual renderizar, se debe crear una sesión de captura, o CameraCaptureSession [29], a la que se debe asignar una función de retrollamada, (en inglés callback) que se llamará una vez finalice alguna de las peticiones que se realizarán en la sesión de captura.

Mientras esta sesión de captura esté activa se pueden realizar las siguientes peticiones de acción a la cámara: captura de un fotograma, captura de fotogramas en rápida sucesión, capturas periódicas de fotogramas y cancelación de capturas periódicas. Todas estas peticiones se mandan a una cola o una lista dependiendo de si es una petición atómica o, por el contrario, de una petición periódica. Las peticiones periódicas no son más que sucesiones de peticiones atómicas por lo que estas, cada vez que la cola de peticiones pendientes está vacía, se envían a la misma cola que las anteriores. Tras llegar a la cola de peticiones pendientes, cada petición se va enviando por orden a la capa de hardware de la cámara, para que esta, con los ajustes previamente establecidos en el momento de su instanciación, procese el fotograma y devuelva el resultado una vez este se haya completado. Esta parte del proceso es síncrona y no bloqueante, es decir, puede haber varios fotogramas esperando a ser

capturados pero el orden en el que se devuelven es el orden en el que entraron, esto asegura que el resultado sea coherente y no se intercambien fotogramas de posición.

Una vez los fotogramas han sido procesados por la cámara estos se asignan a una cola de peticiones completadas de la cual se obtiene el resultado de la captura, o CaptureResult [30], el cual ya se puede gestionar por cualquiera que sea la callback que se asignó en el momento de declarar la sesión de captura. Es en esta función de callback donde se podrá manipular el resultado que ha devuelto la cámara y realizar el post procesado pertinente. En la figura 3.4 se puede observar la escena virtual que simula Android Studio cuando se utiliza la cámara en el emulador.

Figura 3.4 Escena virtual que se ve al iniciar la cámara en Android Studio

El otro objetivo propuesto para este hito fue poder conectar una pantalla secundaria y visualizar contenido solamente en esta última y así poder utilizar el teléfono como controlador. La forma de mostrar contenido en pantallas distintas de la principal es mediante la clase Presentation [31]. Esta clase de Android es una extensión de otra

clase llamada Dialog [32], la cual implementa la funcionalidad de mostrar diálogos al usuario. Estos diálogos son pequeñas ventanas que se muestran por encima de la vista principal de la aplicación y sirven para mostrar información al usuario o instarle a tomar una decisión, tal y cómo se puede observar en la figura 3.5.

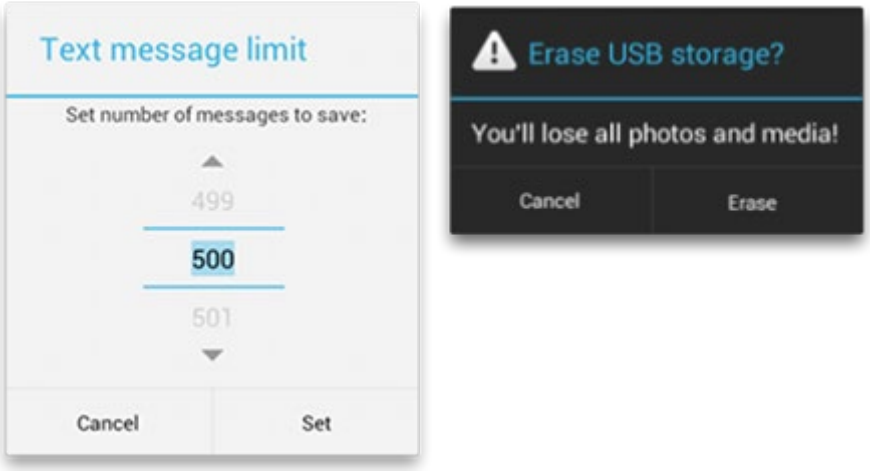

Figura 3.5 Ejemplos de cuadros de diálogo en Android

La clase Presentation hereda de la clase Dialog la idea de mostrar al usuario una ventana con información, pero como ahora esta ventana se muestra en otra pantalla no hay límite para el espacio que puede ocupar, por lo que generalmente se tiende a ocupar todo el espacio disponible. La filosofía que se recomienda seguir al utilizar la clase Presentation es que se utilice meramente para mostrar contenido y que si se desea interactuar con este se realice desde la actividad principal, es por esto por lo que esta clase se adecúa completamente con el propósito para el que se va a utilizar en este proyecto, ya que lo que se muestre en la presentación es lo que se verá en las gafas y para realizar cambios o ajustes al contenido se empleará el dispositivo móvil. Para poder utilizar una presentación es necesario primero asignarle una pantalla en la que se mostrará el contenido, esto se puede asignar manualmente utilizando la clase DisplayManager [33]. Esta clase tiene una lista de las pantallas conectadas al dispositivo y permite acceder a cada una escogiendo el índice correspondiente de la lista. También existe otra forma de administrar las pantallas conectadas, la clase MediaRouter [34], que permite administrar de forma dinámica los dispositivos conectados al teléfono ya sean de audio o de vídeo. Utilizando esta clase se puede distinguir cuando hay una pantalla secundaria conectada y cuando no, modificando lo que se muestra en la

pantalla principal. Por defecto, al conectar pantallas adicionales a un dispositivo el contenido que hay en la pantalla principal se duplica en la secundaria, por esto es necesario definir todo lo que se quiera mostrar cuando se cree la presentación. En esta primera aproximación utilizamos el ejemplo proporcionado en la documentación de Android sobre cómo implementar la clase Presentation utilizando un MediaRouter, este ejemplo permite visualizar un vídeo renderizado en tiempo real de unos cubos de colores girando. En la figura 3.6 se muestra lo que se ve en la pantalla del teléfono al ejecutar la aplicación sin estar conectado a una pantalla secundaria y en la figura 3.7 se puede observar la diferencia al conectar una pantalla adicional.

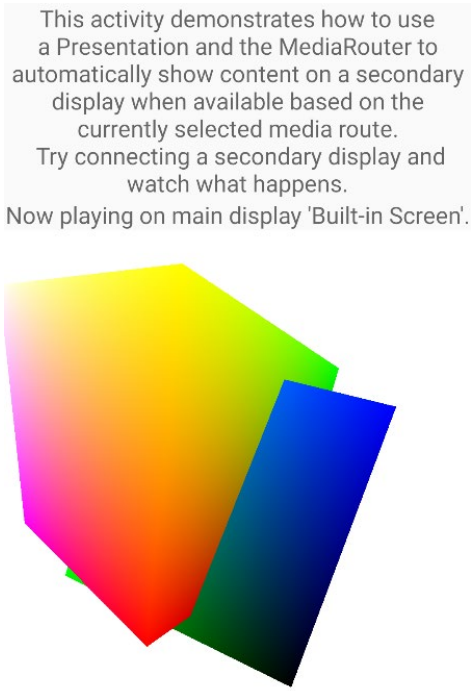

Figura 3.6 Demo de los cubos girando en la pantalla principal

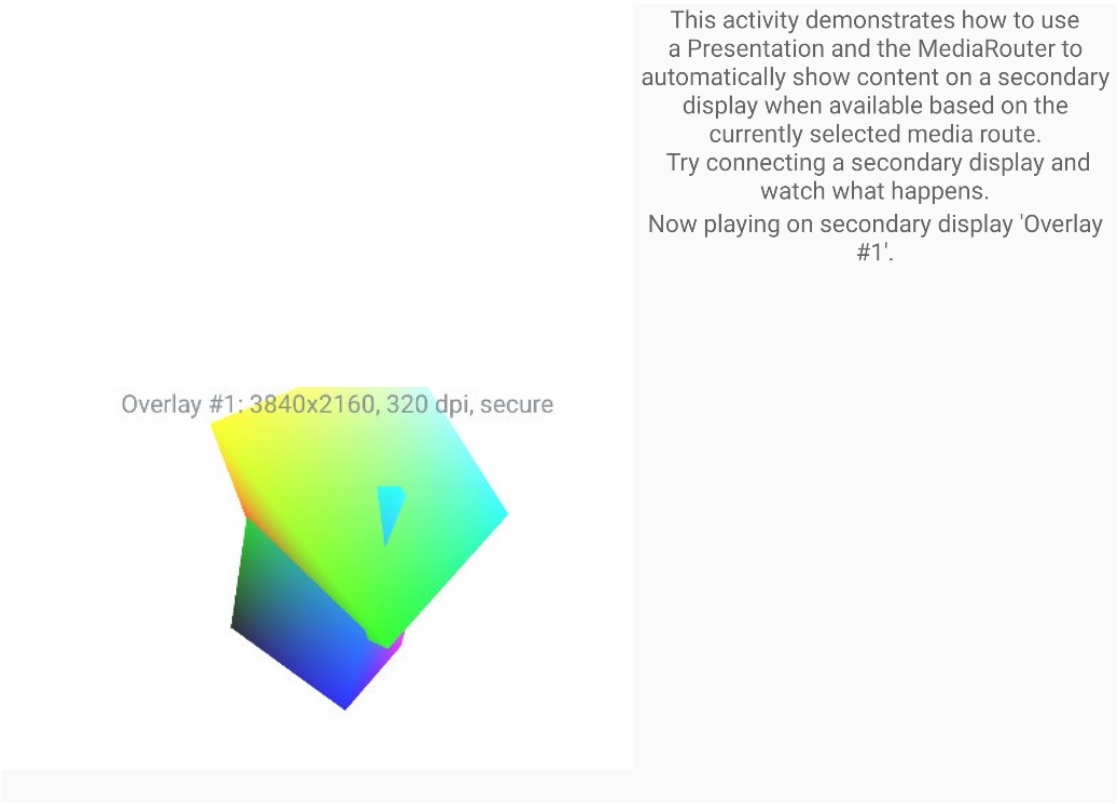

Figura 3.7 Demo de los cubos girando en la pantalla secundaria

Una vez implementadas estas dos funcionalidades por separado, el siguiente reto era poder unir ambas y mostrar en la pantalla de las gafas el vídeo captado por la cámara en tiempo real.

3.3 Primer prototipo en el que se logra mostrar lo captado por la cámara en las gafas

Habiendo logrado implementar las funcionalidades anteriores simplemente siguiendo las instrucciones proporcionadas por la documentación de Android, no esperábamos tener mayor dificultad para integrar ambas. Inesperadamente, esto resultó ser más complicado y llevar más tiempo del que inicialmente se pensó; esto se debe al modelo de ejecución de las aplicaciones Android. En este sistema operativo la manera sugerida de programar es utilizar el hilo principal del sistema operativo para ejecutar una sola actividad. Esta limitación surge del propio medio en el que se concibió Android, un sistema operativo para pequeños dispositivos portátiles que utilizan batería. Es por esto por lo que al programar aplicaciones móviles hay que tener siempre en

mente las restricciones de consumo de energía y el tiempo de respuesta de la aplicación, no se puede tener al usuario esperando mucho tiempo ya que puede pensar que el teléfono se ha bloqueado ni tampoco se pueden ejecutar procesos multihilo muy intensivos ya que esto provocaría un elevado consumo de batería y sobrecalentamiento del dispositivo. También se tiene que recordar que el objetivo final del proyecto es poder utilizar la aplicación en conjunto con las gafas Moverio, las cuales ya de por sí consumen batería del dispositivo al que van conectadas porque carecen de batería propia.

Teniendo todo esto en mente, fue necesario cambiar la aproximación que teníamos hasta el momento. En primera instancia la idea principal era tener una clase dedicada a la obtención de imágenes de la cámara y otra clase que se encargase de la presentación, aparte de la clase principal que contendría los distintos ajustes de imagen que se querían implementar. Como se describió en la sección anterior, la presentación necesita tener el contenido que se va a mostrar en ella declarado en el momento de su instanciación. La primera aproximación fue obtener el vídeo en directo de la cámara en una clase separada y después enviar este a la presentación para así poder mostrarlo en la pantalla de las gafas. Esto, sin embargo, no es posible de esta manera ya que la cámara necesita estar vinculada a una vista a la que enviar la información que procesa, por lo que no se podía utilizar una vista declarada en la presentación ya que esta necesita tener los datos que va a mostrar antes de ser creada. Debido a la tardanza en obtener resultados aceptables empleando esta idea y el cada vez menor tiempo disponible, se decidió separar las líneas de trabajo en dos: por una parte, se seguiría intentado mostrar la cámara en la presentación y por otro se mostraría la cámara en la pantalla principal, pero se crearía una vista deslizante que permitiría acceder a los ajustes. Este segundo planteamiento de la aplicación se sugirió como un plan de contingencia para poder seguir con el desarrollo de la aplicación, viendo la demora que estaba generando el poder utilizar la presentación de manera correcta. Durante este período en el que existían dos modelos de la aplicación, se descubrió la existencia de CameraX [35], una librería más moderna de Android que simplificaba enormemente el uso de la cámara. Esta librería fue clave para el desarrollo ya que permitía utilizar la cámara enfocándola a casos de uso definidos y reducía

drásticamente el código necesario para poder crear una instancia de cámara y utilizarla. Esta librería se explicará con más detalles en la siguiente sección.

Después de varios intentos fallidos de interconectar clases y sus distintas funcionalidades, surgió la idea de combinar las clases de cámara y la de presentación en una sola, es decir, hacer que la presentación se conectase a la cámara en el momento de su creación y obtuviese el contenido a presentar directamente de esta. Esto también supuso el reemplazo de la clase MediaRouter por la de DisplayManager ya que el código se modificó para que la cámara se inicializase solamente al haber una pantalla externa conectada donde se iniciaría la presentación. Esta clase, MediaRouter, se puede prescindir ya que se puede lograr el mismo objetivo empleando el DisplayManager. Además, utilizar DisplayManager disminuye drásticamente la cantidad de código necesario; esto se debe a que las gafas Moverio necesitan estar conectadas a la entrada USB del dispositivo para poder funcionar, la cual es siempre la primera posición en la lista del DisplayManager. También hay que tener en cuenta que, a excepción del controlador propietario de las gafas, los teléfonos móviles no suelen tener más de un puerto USB, por lo que la lista de pantallas conectadas está limitada a una. Finalmente, esta fue la idea que triunfó y obtuvo resultados, se consiguió por fin mostrar la cámara en la pantalla de las gafas.

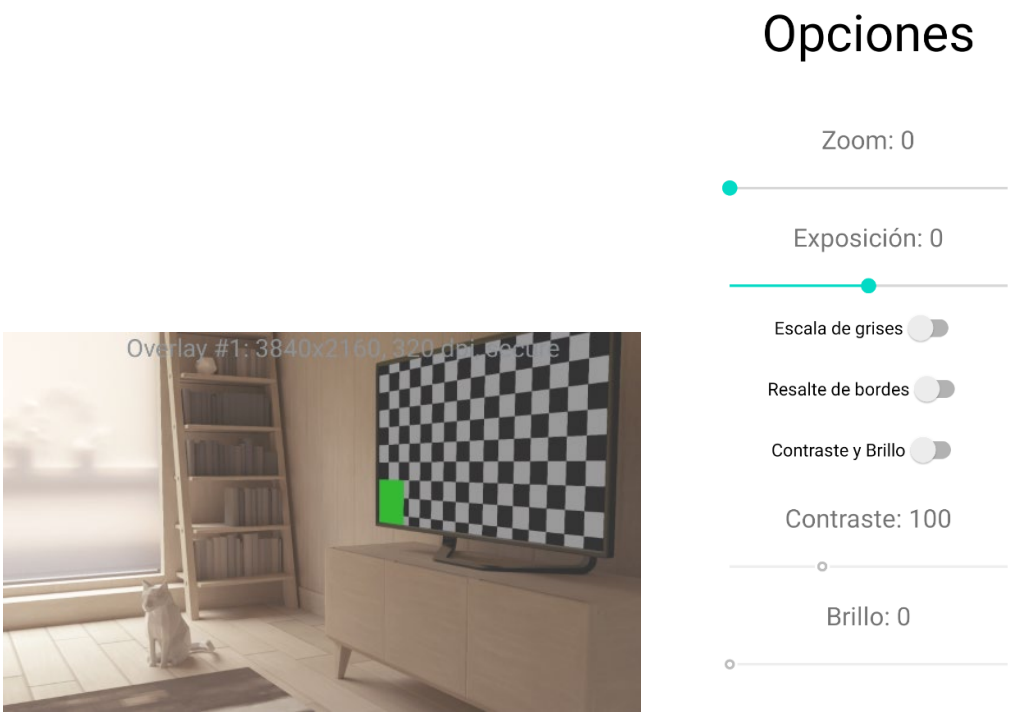

Figura 3.8 Escena virtual de la cámara mostrándose en la segunda pantalla

De esta forma se consiguió separar el visionado de la cámara en las gafas y los ajustes de imagen en el teléfono, ver figura 3.8. El siguiente objetivo fue añadir la funcionalidad de todos los ajustes de imagen que se pueden ver en la pantalla principal en la imagen superior.

3.4 Adición de manipulación de zoom y exposición

Tras haber superado el mayor desafío planteado en este proyecto, mostrar en la pantalla de las gafas lo que se capta por la cámara en tiempo real, ahora había que plantearse qué ajustes de imagen se añadirían. Estos ajustes se pueden dividir en dos grupos, los ajustes propios del hardware y el postprocesado de la imagen captada. El primer grupo contiene todos los parámetros modificables relativos a las gafas y a la propia cámara como son: el brillo de la pantalla de las gafas, la exposición y el zoom de la cámara. Por otro lado, el segundo grupo es más abierto, ya que se pueden realizar diversos tratamientos a la imagen captada; en este proyecto se ha optado por incluir un filtro de escala de grises, un filtro de detección de bordes, un filtro de corrección de contraste y un último filtro de corrección de brillo de la imagen.

Como se puede leer en la anterior sección, la implementación final de la cámara se realizó empleando la librería CameraX. Las ventajas que ofrece esta librería respecto a camera2 son, principalmente: la facilidad de uso, el enfoque a casos de uso de la cámara y el modelo de implementación con un ciclo de vida. CameraX permite obtener una instancia de cámara funcional de manera mucho más sencilla que camera2, con esta nueva librería ya no hace falta declarar e inicializar cada una de las clases necesarias para el funcionamiento de camera2, sino que es el propio CameraX quien se encarga de realizar todo esto por debajo.

Para utilizar CameraX es necesario declarar un ListenableFuture [36], un resultado futuro de una operación de cómputo asíncrona, para poder obtener los resultados de la cámara una vez estos estén listos. A esta clase se le debe añadir un Listener, función que se ejecutará cuando esté disponible el resultado anteriormente mencionado, que será ejecutado por un MainExecutor [37], una clase que se encarga de gestionar cuándo y en qué hilos se ejecutará el Listener, al tratarse del ejecutor principal se emplea el hilo principal de Android que se encarga de manejar la interfaz de usuario. Una vez hecho esto hay que especificar lo que debe hacer la función llamada en el Listener, es aquí donde se crean los objetos y configuran los ajustes necesarios para CameraX. Los casos de uso que se han utilizado en esta aplicación son el de selección de cámara, el de vista previa y el de análisis de imagen. El caso de uso de selección de cámara podría haberse omitido si se trabajase directamente con la cámara de las gafas Moverio, ya que estas disponen de una sola cámara orientada al frente, pero, al vernos obligados a utilizar la cámara del dispositivo móvil, es necesario especificarle a CameraX cual de todas las cámaras del dispositivo debe usar, en este caso la cámara trasera. Para el caso de uso de la vista previa basta con especificar la resolución objetivo de la vista donde se va a mostrar el resultado y establecer dicha vista, que en este caso se trata de un nuevo tipo de vista introducido con CameraX, la PreviewView [38]. Esta vista PreviewView fue creada para mostrar específicamente el resultado del caso de uso de vista previa y se ajusta de manera automática para mostrar correctamente este resultado. Por dentro PreviewView utiliza las previamente mencionadas TextureView y SurfaceView y elige la que mejor se adapte a cada situación dependiendo del valor de su atributo ImplementationMode. Por último, el caso de uso de análisis de imagen es el que después servirá para realizar el postprocesado. Para poder declararlo se necesita

especificar una resolución objetivo y una estrategia de contrapresión, que método utilizar para controlar el flujo productor-consumidor entre las imágenes enviadas a analizar y el resultado del análisis, así como un Executor para el análisis, en este caso el hilo principal nuevamente. En cuanto a la estrategia de contrapresión, se ha optado por mantener únicamente la última imagen enviada a analizar para evitar tener que usar una cola de imágenes que analizar, esto se debe a que el objetivo de la aplicación es mostrar vídeo en tiempo real y utilizar una cola para analizar todas las imágenes recibidas podría causar una latencia indeseada. Una vez declarados todos los casos de uso es necesario vincularlos al ciclo de vida de la cámara para que se ejecuten durante todo el tiempo que esta esté activa.

Teniendo la cámara funcionando en la presentación, era necesario diseñar una interfaz de control en la pantalla principal. Se optó por implementar dos tipos de elementos de control, barras deslizantes y conmutadores. Las barras deslizantes sirven para controlar los ajustes que admiten un rango de valores distintos, como el zoom o el contraste, y los conmutadores para activar o desactivar alguno de los filtros de postprocesado de la imagen.

El primer ajuste que se intentó implementar fue el del brillo de la pantalla de las gafas Moverio, este ajuste se puede cambiar desde la aplicación de Moverio Link Pro instalada de base en el controlador de las gafas, pero, como para los dispositivos que no sean el controlador propio de Moverio esta aplicación no está disponible, se consideró oportuno poder modificar el brillo desde la aplicación desarrollada para este proyecto. Esto, sin embargo, no resultó posible ya que para poder acceder a los ajustes de las gafas es necesario emplear la clase DisplayManager proporcionada por el SDK de Epson. Aunque sí que se pueden acceder y llamar a las funciones que modifican los ajustes del brillo en las gafas, utilizar estas funciones devuelve un código de error y no modifica ningún ajuste en las gafas. Se probaron varias de las funciones que ofrece el SDK, pero ninguna dio resultados apreciables por lo que se decidió investigar el funcionamiento interno de las mismas. Para ello fue necesario descompilar las clases y realizar ingeniería inversa para deducir su funcionamiento. Esto se realizó utilizando una funcionalidad que viene incluida en Android Studio, que permite mostrar el origen de una clase. Esta herramienta busca la definición de la clase en la biblioteca de la que

proviene y la muestra al usuario para que este pueda documentarse sobre su funcionamiento. Al tratarse de una librería externa propietaria de Epson las clases están compiladas y la única información que puede mostrar Android Studio sobre su definición es la clase descompilada.

Tras realizar esta investigación se determinó que los ajustes de las gafas solo eran modificables si el modelo de estas se ajustaba con los modelos que vienen definidos en las funciones del SDK, al no estar el modelo concreto de las gafas de las que disponíamos para el proyecto se asumió que esta funcionalidad no se podía implementar. Aun así, al estar disponible en el propio software de Moverio, los usuarios siguen teniendo acceso a ella, ver figura 3.9.

```
if (var4 != 0) {
    if (var4 != 1) {
        Log.e(varó.a, msg: "Unknown model.");
        var2 = -1;
    } else if (!varó.d.b(varó)) {
        Log.e(varó.a, msg: "Not execute DisplayManager#open yet.");
    } else {
        var10000 = varó.c;
        var11 = var3 = h.b();
        var4 = -1;
        switch (var11.hashCode()) {
            case 63499735:
                if (var3.equals("BT-40")) {
                    var4 = 1;
                }
                break;
            case 697962723:
                if (var3.equals("Debug device")) {
                    var4 = 3;
                }
                break;
            case 1968490891:
                if (var3.equals("BT-30C")) {
                    var4 = 4;
                }
                break;
            case 1968491048:
                if (var3.equals("BT-35E")) {
                    var4 = 0;
                }
                break;
            case 1968492007:
                if (var3.equals("BT-45C")) {
                    var4 = 2;
                }
```

Figura 3.9 Extracto del código descompilado del DisplayManager de Epson. Se puede observar un switch en el que se comprueba el modelo del dispositivo conectado y se actualiza el valor de una variable que presumiblemente después sirve para permitir la modificación del brillo

Después de determinar que el brillo de las pantallas no se implementaría se pasó a trabajar en la funcionalidad de modificación del zoom. El zoom es la capacidad de acercar o alejar una imagen ya sea durante su captura o, posteriormente, durante su edición. Se pueden distinguir dos tipos de zoom, el óptico y el digital. El zoom óptico es

el resultado de modificar la posición de las lentes de la cámara con el objetivo de crear distintas distancias focales. Por otro lado, el zoom digital se logra mediante un algoritmo que amplía una sección de la imagen completa capturada por la cámara, haciendo que se pierdan más detalles cuanto mayor es el zoom, ya que la foto original no cambia. Por lo general, en los dispositivos Android, para mantener su reducido tamaño, los módulos de cámara utilizados son muy pequeños lo cual no permite tener muchas lentes y conlleva una reducción significativa de las capacidades de zoom óptico del dispositivo. Es por esto por lo que en la mayoría de los dispositivos Android y en especial el controlador de las gafas Moverio, el zoom que se emplea es el digital.

CameraX permite controlar los ajustes de la cámara y modificarlos en directo mediante su clase CameraControl [39]. Esta clase contiene todos los parámetros modificables de la cámara y permite mediante una sencilla función setLinearZoom ajustar el zoom de la cámara mientras esta se está utilizando. La cantidad de zoom a aplicar se puede seleccionar utilizando la barra deslizante de la pantalla principal. Las figuras 3.10, 3.11 y 3.12 son ejemplos de imágenes tomadas con tres niveles distintos de zoom: 0%, 50% y 100%.

Figura 3.10 Imagen tomada con el zoom de la cámara al 0%

Figura 3.11 Imagen tomada con el zoom de la cámara al 50%

Figura 3.12 Imagen tomada con el zoom de la cámara al 100%

Durante la implementación del zoom, viendo que CameraControl disponía de una función para modificar la exposición de la cámara, se decidió que esta sería la próxima funcionalidad a implementar. La exposición se refiere a la cantidad de luz que entra en el sensor de la cámara a través de la lente del objetivo durante el período que se tarda en tomar la fotografía. Este parámetro se ajusta controlando la apertura del diafragma de la cámara, un mecanismo que actúa de manera análoga al iris del ojo humano, aumentando o disminuyendo la misma para así dejar entrar más o menos luz.

Así como con el zoom, la exposición se controla tomando el valor de la barra deslizante de exposición y llamando a la función de la clase CameraControl setExposure. En las figuras 3.13 y 3.14 se puede observar el resultado de tomar una foto disminuyendo la exposición al 100% y 50% respectivamente. Por otro lado, las figuras 3.15 y 3.16 muestran lo obtenido al hacer lo contrario, aumentar la exposición un 50% y un 100% respectivamente.

Figura 3.13 Imagen tomada con la exposición de la cámara al -100%

Figura 3.14 Imagen tomada con la exposición de la cámara al -50%

Figura 3.15 Imagen tomada con la exposición de la cámara al +50%

Figura 3.16 Imagen tomada con la exposición de la cámara al +100%

Se decidió incorporar estos ajustes teniendo en mente a las personas que padecen retinitis pigmentaria [4]. Este grupo de enfermedades genéticas se caracteriza por la pérdida de visión nocturna o la pérdida de visión lateral. Para el primero de estos problemas una persona que padezca la enfermedad podría aumentar la exposición de la imagen y así poder ver mejor en un entorno oscuro o de noche. Por otro lado, también se puede usar este ajuste para conseguir el efecto contrario, ya que hay enfermedades

como la enfermedad de Stargardt o la uveítis [4] en las que uno de sus síntomas es la sensibilidad a la luz. Una persona con alta sensibilidad a la luz podría disminuir el brillo de las imágenes que se le muestran con el fin de evitar las molestias causadas por objetos que sean demasiado brillantes. También hay que tener en cuenta que, aunque la opción de ampliar la imagen para poder ver objetos distantes puede ser beneficiosa para la gran mayoría de los usuarios, es especialmente práctica en el caso de las personas que tienen dificultades para ver los objetos que se encuentran lejos. Este es el caso de las personas que tienen miopía [4], una enfermedad causada por un defecto en la refracción del ojo que causa el síntoma descrito anteriormente, así como fatiga ocular o la necesidad de cerrar los ojos para ver claramente. En estos casos, una persona con problemas para ver objetos distantes solo tiene que ampliar el zoom en la aplicación para poder ver estos objetos sin mayor problema.

3.5 Adición del filtro de escala de grises

Habiendo implementado el grupo de ajustes relacionados con la modificación de parámetros hardware de la cámara, ahora les tocaba el turno a los filtros de postprocesado. El primero que se implementó fue el filtro de escala de grises. Una escala de grises es una manera de organizar los colores, de forma que a éstos se les asigna un valor, desde el negro hasta el blanco, pasando por distintas tonalidades de grises. Es distinto a las imágenes monocromas ya que estas otras únicamente utilizan estos dos colores, el blanco o el negro. Utilizar una escala de grises a la hora de visualizar imágenes pueden ser de gran utilidad para personas con daltonismo [4], ya que las personas con esta enfermedad pueden sufrir distintas patologías que les reduzcan o incluso inhabiliten la capacidad de ver tonos rojos, verdes o azules, así como confundir distintos colores, llegando algunos al punto de solo distinguir los grises. Debido a esto, una escala de grises puede conseguir que las personas que padecen daltonismo puedan percibir de mejor manera las distintas tonalidades de las imágenes que visualizan a través de las gafas. Este ajuste puede utilizarse cuando el usuario tenga problemas a la hora de percibir los colores, como puede ser el caso anteriormente mencionado de las personas con daltonismo o el de algunas personas con cataratas [4]. En estos casos en los que el usuario no es capaz de distinguir entre dos o más colores el filtro puede ser útil, ya que los transforma todos a una escala entre el blanco y el negro. Una vez la imagen se

encuentra en blanco y negro, el usuario podría observar la diferencia entre los colores y así saber que, aunque los vea de la misma manera, son colores diferentes.

La primera aproximación para implementar esta funcionalidad fue la de usar el caso de uso de análisis de imágenes de CameraX. Este caso de uso permite obtener los fotogramas que va procesando la cámara uno por uno para después analizarlos. Este caso de uso está pensado para aplicaciones que tiene como objetivo el reconocimiento de objetos en tiempo real, la obtención de información relacionada con las imágenes que capta la cámara o, el uso que se le da en esta aplicación, el procesado de los fotogramas captados. Al poder obtener cada uno de los fotogramas se puede crear una función que altere dichos fotogramas aplicando el filtro que sea necesario y los devuelva para poder mostrarlos en una vista. Se puede pensar en reutilizar la PreviewView que ya se está utilizando para mostrar la vista previa y mostrar en ella el resultado, pero desafortunadamente esto no es posible ya que PreviewView no permite sobrescribir arbitrariamente las imágenes que recibe para mostrar. La solución a este inconveniente fue instanciar una nueva vista SurfaceView, la cual se inicializa invisible y solo se vuelve visible cuando alguno de los filtros de postprocesado está activo. Solo una de las vistas puede ser visible al mismo tiempo para evitar problemas entre las dos. Una vez que se tiene la vista a la que mostrar el resultado del análisis faltaba implementar el algoritmo para pasar de una imagen a color a una en escala de grises. Para lograrlo se toma el fotograma actual de la PreviewView en formato Bitmap, una matriz que contiene los valores rojo, verde y azul de cada píxel de la imagen junto con su transparencia, y se le aplica una ColorMatrix [40], matriz utilizada para la modificación de colores, con un valor de saturación igual a cero. Una vez se tiene el mapa de bits en escala de grises este se debe pintar en un Canvas [41], el "lienzo" en el que se va a pintar, que en este caso se obtiene de la SurfaceView previamente declarada. La figura 3.17 ilustra el procedimiento a seguir para lograr mostrar una imagen en escala de grises.

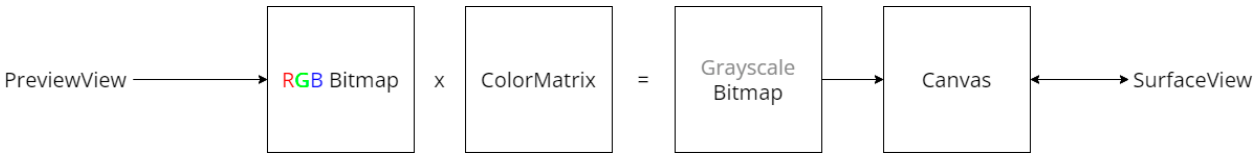

Figura 3.17 Diagrama de alto nivel del proceso de aplicación del filtro de escala de grises

Este filtro se activa cuando el usuario pulsa el conmutador *Escala de grises* en la pantalla principal del dispositivo. La figura 3.18 muestra el resultado de aplicar al filtro de escala de grises a lo que está captando la cámara en ese momento.

Figura 3.18 Imagen tomada con el filtro de escala de grises activado.

3.6 Optimización del filtro de escalas de grises y adición del filtro de detección de bordes

Teniendo el primer filtro ya implementado el siguiente objetivo era añadir un filtro de detección de bordes, una funcionalidad de la que se habló en las primeras reuniones con la gente de Plusindes y se determinó muy importante para la aplicación. Un filtro de bordes es una técnica en el procesamiento de imágenes que consiste en resaltar los bordes o contornos de una imagen. Estos filtros se utilizan con el objetivo de enfatizar las transiciones abruptas en la intensidad de la luz. Con esto podemos lograr una imagen con los bordes más nítidos y resaltados, lo que sirve para que se puedan detectar mejor los cambios profundidad, la textura o la forma de los diferentes objetos que se encuentran en la imagen. Existen varios algoritmos para la detección de bordes, como el de Roberts [42], el de Prewitt [43], el de Canny [44] o el de Sobel [45]. En general, todos ellos funcionan comparando los píxeles de la imagen con los píxeles adyacentes. Cuando se producen cambios bruscos en la intensidad o el color en la imagen, el algoritmo detecta la existencia de un borde y lo resalta. Investigando sobre la detección

de bordes nos topamos con OpenCV, una biblioteca de software libre de visión por computador. Como se mencionó al principio de la memoria esta biblioteca ha sido clave para el desarrollo de la aplicación. En ella hay definidos una gran cantidad de algoritmos de tratamiento de imágenes, todos ellos optimizados para distintos lenguajes de programación y enfocados en su uso en tiempo real. En esta biblioteca se pueden encontrar distintos métodos para conseguir detectar bordes, los más destacables son Sobel [46], Laplacian [47] y Canny [48].

Para ilustrar el resultado de la aplicación de los distintos filtros de detección de bordes se ha utilizado la imagen *Mandrill* [49]. Esta imagen, junto con el resto del paquete de imágenes de test proporcionadas por la Universidad del Sur de California, se utiliza con frecuencia en el campo del procesado de imágenes debido a que tiene gran cantidad de detalles, variación en niveles de luminosidad y sombras y bordes acentuados. La figura 3.19 representa la imagen utilizada.

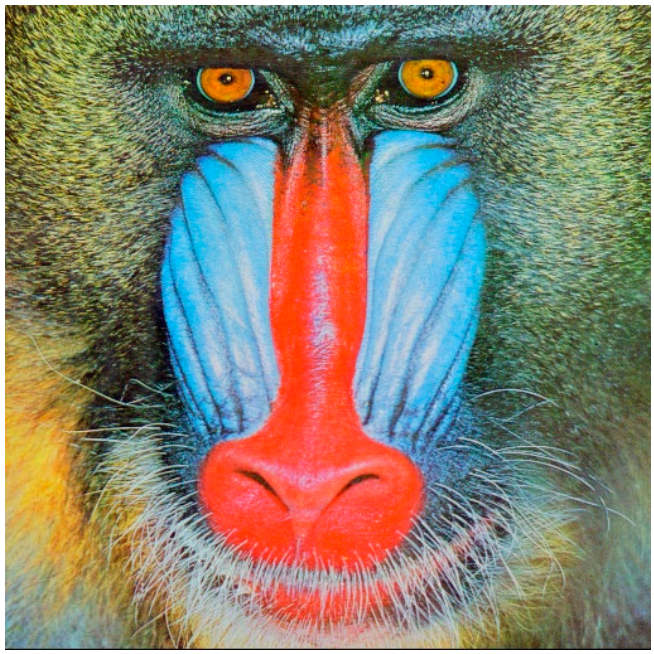

Figura 3.19 Mandrill, imagen utilizada para ilustrar los resultados de los distintos algoritmos

El método de Sobel calcula el gradiente de la intensidad de cada píxel en la imagen para así reconocer los posibles bordes y su orientación, para lograr esto se debe aproximar la primera derivada para un grupo de 3x3 píxeles y buscar los máximos, los cuales representan un borde. Estos cálculos deben realizarse en para ambos ejes de la imagen, en el eje de abscisas o eje X, para obtener los bordes verticales, y en el eje de

ordenadas o eje Y, para obtener los bordes horizontales, ver figura 3.20. Finalmente, estos dos resultados se combinan para obtener la detección de bordes en ambas direcciones, ver figura 3.21.

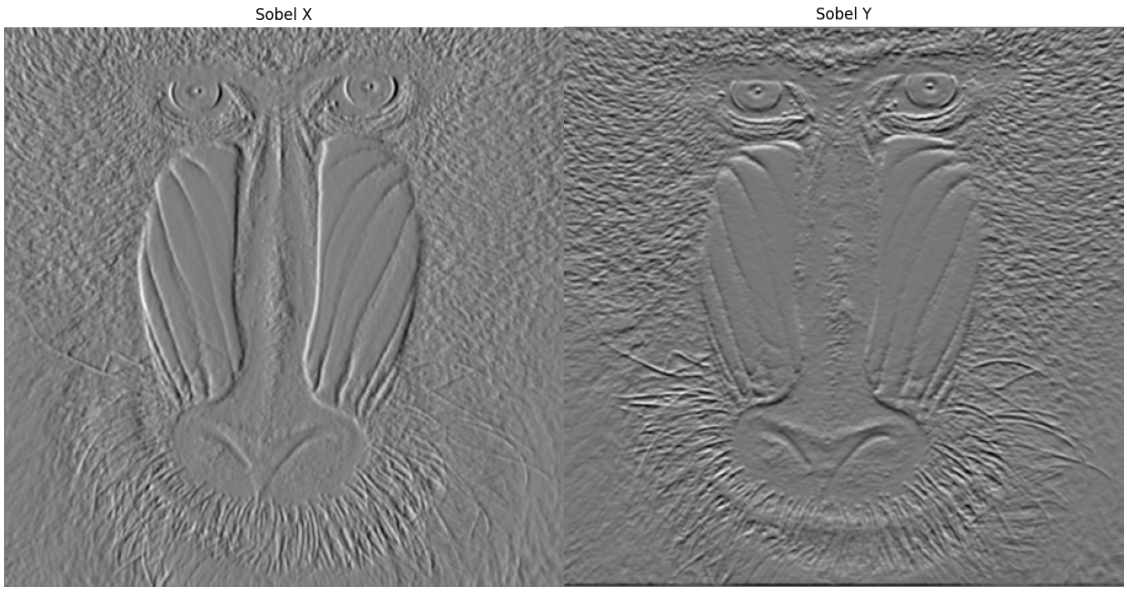

Figura 3.20 Resultado de aplicar la función Sobel de OpenCV en el eje X (izquierda) y el eje Y (derecha) sobre Mandrill

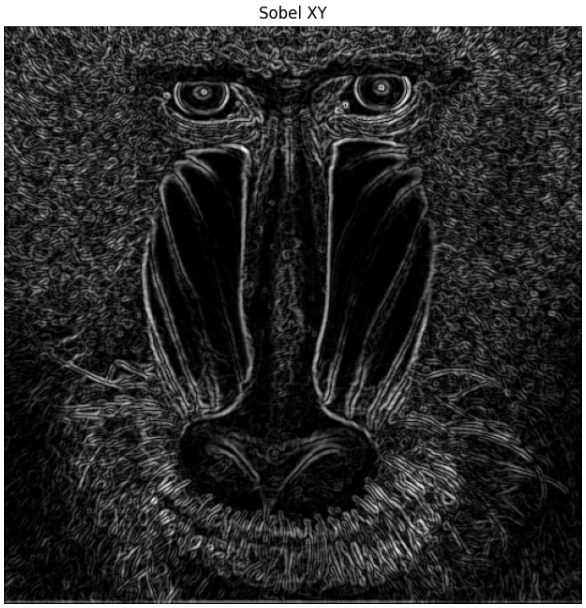

Figura 3.21 Resultado de combinar las imágenes anteriores, ahora se distinguen los bordes en ambas direcciones

El método Laplacian [50] utiliza una implementación discreta del operador de Laplace para calcular la segunda derivada. Esta forma de detección de bordes utiliza

el método de Sobel por debajo para calcular la primera derivada y después aplica el operador de Laplace sobre este resultado. En la segunda derivada los máximos locales de la primera derivada, que indican cambios rápidos en la intensidad de color de la imagen, junto con los mínimos locales se convierten en ceros. Estos ceros después se filtran para eliminar las regiones que son mínimos locales ya que estas representan regiones de la imagen en las que la intensidad del color no varía demasiado. Tras la ejecución del método Laplacian se obtiene una imagen con los bordes resaltados con menor ruido y mayor distinción entre las zonas que son bordes y las que no, aunque se pierden algunos pequeños detalles en la imagen. En la figura 3.22 se puede ver el resultado de aplicar este algoritmo sobre la imagen de prueba.

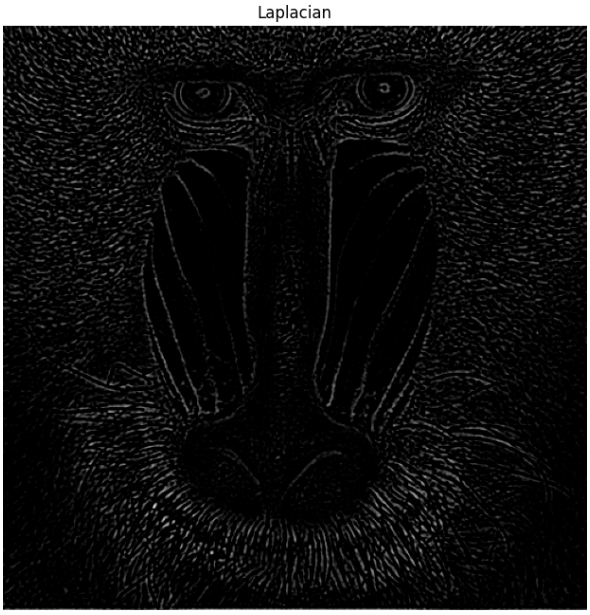

Figura 3.22 Resultado de aplicar la función Laplacian de OpenCV sobre Mandrill

Finalmente, el método de Canny es una mejora respecto al método Laplacian ya que combina las derivadas de Sobel en X e Y para después juntarlas en derivadas cuatridireccionales. Tras hacer esto se buscan los máximos locales de estas últimas derivadas y se asignan como candidatos a ser bordes. Otro aspecto único al algoritmo de Canny es que busca contornos, es decir bordes que están conectados y forman el perímetro de un objeto. Estos contornos se encuentran aplicando un umbral de histéresis a los píxeles. Este umbral de histéresis consta de un límite superior y uno inferior, los píxeles que superan el umbral superior son considerados como parte del borde y los que no alcanzan el inferior son descartados. Por otro lado, los píxeles que se encuentran entre

los límites del umbral sólo pueden llegar a ser considerados bordes si están conectados a otros píxeles borde. Este umbral de histéresis se declara al llamar a la función *Canny* de OpenCV y el ratio recomendada entre los límites está entre dos y tres veces mayor el superior que el inferior, en la implementación de la práctica se ha optado por una ratio de 2:1 ya que el resultado es de mejor calidad utilizando esta proporción. El resultado final es una imagen binaria donde los bordes son los píxeles blancos, ver figura 3.23.

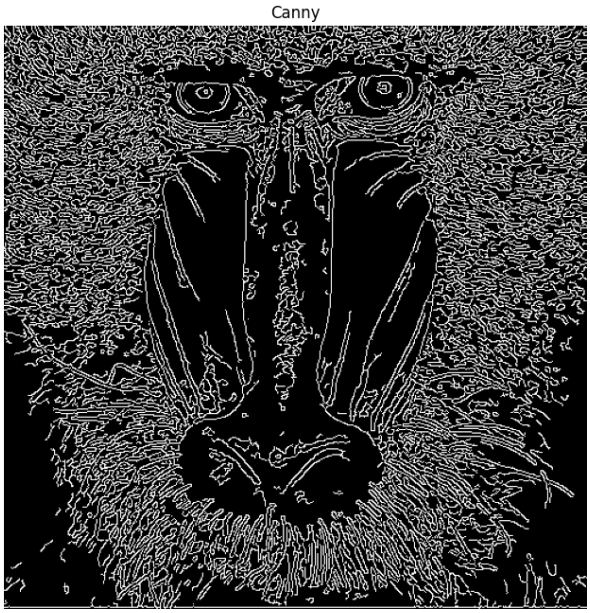

Figura 3.23 Resultado de aplicar la función Canny de OpenCV sobre Mandrill

Habiendo probado estos tres métodos de detección de bordes, el que finalmente se implementó en la aplicación fue Canny. Los motivos de esta decisión tienen que ver con los escenarios en los que se utilizará esta funcionalidad, se deben detectar bordes de imágenes en tiempo real y muchas veces en movimiento, es por esto por lo que el filtro de Canny, que muestra una imagen fácilmente distinguible y con alto contraste es la mejor opción. En la figura 3.24 se pueden comparar los tres resultados obtenidos con los tres algoritmos probados.

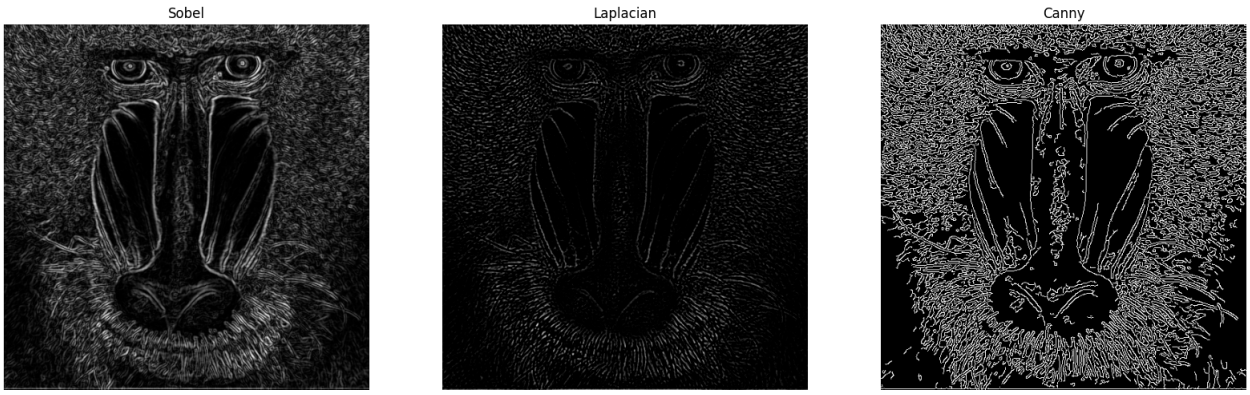

Figura 3.24 Comparativa entre los tres tipos de métodos de detección de bordes

En la pantalla principal del dispositivo el usuario puede activar esta funcionalidad presionando el conmutador *Resalte de bordes*. La figura 3.25 muestra el resultado de la implementación final del filtro de detección de bordes.

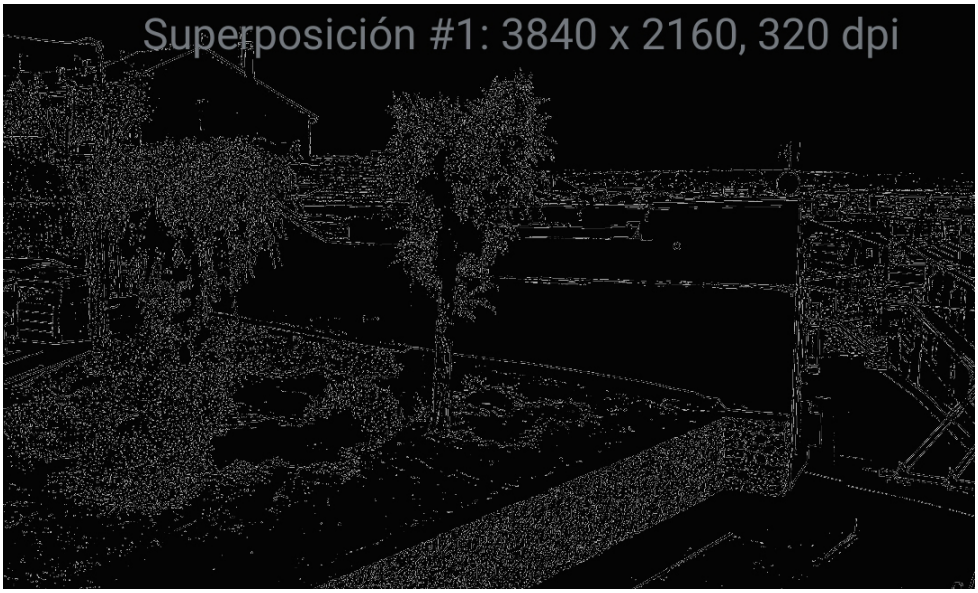

Figura 3.25 Imagen tomada con el filtro de detección de bordes activado

Un requisito esencial para el correcto funcionamiento de estos filtros es que la imagen que se les debe suministrar tiene que estar en escala de grises. Para conseguir esto OpenCV dispone de una función muy útil que transforma una imagen de color a escala de grises. Teniendo esto en cuenta se decidió modificar la funcionalidad del filtro en escala de grises previamente utilizado por el que ofrecía OpenCV, ya que este último ofrecía el mismo resultado en menos líneas de código y seguramente con mejor rendimiento.

El filtro de detección de bordes puede ser útil en caso de que el usuario tenga dificultades para distinguir los objetos que tiene delante. Esto puede ser debido a muchos factores, uno de ellos es la visión borrosa, provocada por una gran cantidad de enfermedades como la miopía, hipermetropía, astigmatismo o presbicia [4] entre muchas otras. Otro posible motivo sería la dificultad para ver la diferencia entre los colores, como es el caso de las personas con daltonismo [4]. Un usuario que tenga dificultad para detectar los objetos podría utilizar este filtro y ayudarse de la detección de bordes para mejorar su percepción de lo que tiene delante. Además, cualquier otro usuario que necesite distinguir algún objeto o que considere que le puede venir bien en alguna situación específica también lo tiene disponible, ya que este ajuste se puede utilizar para fines muy variados.

3.7 Se añade ajuste de brillo y contraste

Después de haber logrado uno de los objetivos del proyecto, la detección de bordes, el siguiente paso fue buscar que otros filtros pudieran ser de utilidad, además de lo ya implementado anteriormente. Surgió la idea de poder modificar el contraste, una idea que estaba presente desde el principio pero que a menudo se confundía con la exposición. El contraste de una imagen es la diferencia entre sus zonas más claras y sus zonas más oscuras. Esta diferencia permite apreciar los colores, las texturas, las sombras o los detalles de la imagen con más facilidad. Modificar este ajuste permite adaptar las imágenes a los ojos de cada persona y así mejorar la calidad de las imágenes que perciben. Para lograr la modificación de este parámetro en una imagen basta con utilizar la función de OpenCV *convertTo* [51]. Esta función permite convertir un array a otro tipo de dato añadiendo opcionalmente escalado, la fórmula empleada es la siguiente [52]:

$$Matriz(i,j) = saturate\_cast < Tipo > ((\alpha * Píxel\ original_{ij}) + \beta)$$

Como se mencionó en secciones anteriores, las imágenes con las que se trabaja son mapas de bits, los cuales se representan con matrices que no son más que arrays de arrays. Teniendo esto en cuenta, para modificar una imagen es suficiente con llamar la función de la figura XX para cada píxel de la imagen. En la fórmula cada píxel de la imagen original se multiplica por un parámetro alfa y después se le suma una beta.

Como el contraste es el ratio entre lo más brillante de la imagen y lo más oscuro este parámetro alfa se puede utilizar para modificar este ratio aplicándolo a todos los píxeles de la imagen. De esta forma se consigue modificar el contraste de la imagen de manera sencilla, pero, debido a problemas de rendimiento causados por calcular esta transformación para cada fotograma que se recibe, la implementación final se realizó con una tabla de consulta, una tabla que contiene todos los valores resultantes de multiplicar el valor alfa por cada valor posible de cada píxel. Como los píxeles están restringidos entre 0 y 255 basta con crear la tabla una vez se recibe por primera vez el parámetro alfa y después comprobar qué valor se debe asignar a cada píxel resultante según esta tabla. El valor del parámetro alfa se controla con una barra deslizante desde la pantalla principal, una vez se ha activado el conmutador *Contraste y Brillo*. En las figuras 3.26 y 3.27 se puede observar el resultado de reducir y aumentar el contraste respectivamente.

Figura 3.26 Imagen tomada con el contraste con $\alpha = 0.5$

Figura 3.27 Imagen tomada con $\alpha = 2.5$

Cuando se modifica el contraste de una imagen es habitual querer modificar también el brillo de esta, pues los dos ajustes van de la mano. El brillo de una imagen es la cantidad de luz que se encuentra en dicha imagen. A mayor brillo, mayor luminosidad de la imagen, es decir, la imagen se hace más clara. Por el contrario, a menor brillo, la imagen se oscurece. El brillo se suele confundir con la exposición, sin embargo, son cosas diferentes. Por un lado, la exposición se ajusta a la hora de tomar la imagen y por el otro el brillo se modifica una vez tenemos la imagen ya tomada. Con esto en mente y viendo la fórmula de anterior es fácil adivinar cómo se puede modificar el brillo de la imagen. Como se dijo previamente los valores RGB de un píxel indican la intensidad del color que representan, cuanto más alto el valor más intenso el color. De esta forma se puede ver que para aumentar el brillo basta con sumar una cantidad a cada uno de los píxeles y esto se consigue mediante el parámetro β, que se suma después de haber realizado la multiplicación por el parámetro α. El valor que toma este parámetro se obtiene de la otra barra deslizante que se activa al pulsar el conmutador *Contraste y Brillo*. La figura 3.28 muestra el resultado de haber aplicado el filtro de brillo con un valor de 150.

Figura 3.28 Imagen tomada con β = 150

3.8 Rediseño de la interfaz de usuario

Habiendo implementado todas las funcionalidades que se han ido describiendo a lo largo de este capítulo, llegó la hora de mostrar el resultado a los tutores y al personal de Plusindes. La aplicación fue bien recibida y se dio el visto bueno a todos los ajustes implementados, no obstante, se señaló que la interfaz de usuario de la pantalla principal debía ser mejorada. La interfaz inicial se puede observar en la figura 3.8, en ella se pueden apreciar que todos los elementos interactuables tienen el diseño predeterminado. La principal modificación que había que realizar era aumentar el tamaño de los conmutadores y barras deslizantes ya que eran demasiado pequeños para ser manipulados de manera correcta. Esto sumado a que la pantalla del controlador Moverio es, a su vez, muy pequeña hacía aparatoso controlar la aplicación. Es por esto por lo que se aumentó el tamaño de toda la interfaz para hacerla más legible y fácil de usar. Además, se crearon barras deslizantes y conmutadores personalizados para poder modificar su tamaño y adecuarlos al tamaño del dispositivo. El resultado del rediseño de la interfaz se puede observar en la figura 3.29.

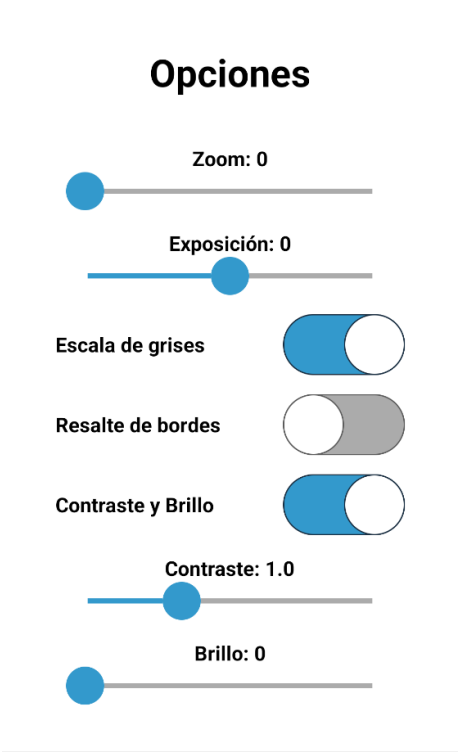

Figura 3.29 Nuevo diseño de la interfaz

3.9 Dificultades sobrevenidas durante el desarrollo

A lo largo del desarrollo de este trabajo de fin de grado han surgido diversos contratiempos, algunos los hemos podido solventar mientras que otros no han sido posibles. El primero de ellos se presentó nada más comenzar con el proyecto, el primer modelo de gafas AR que se nos iba a prestar, el modelo Glow Plus de la marca MadGaze [53] ya no ofrecía la descarga de su SDK. Tras contactar por correo electrónico con la empresa e incluso tratar de hablar con el CEO de la empresa, se decidió cambiar de gafas debido a la falta de respuesta por parte de MadGaze. Seguidamente se nos anunció a existencia de un nuevo modelo de gafas Moverio, las BT45 C [54]. Este modelo más novedoso incluía características nuevas y mejoradas, pero, como era de reciente aparición, la oferta de unidades era reducida y cara. Es por esto por lo que finalmente las gafas con las que desarrollaríamos el TFG serían las BT 40S, a la espera de noticias sobre una posible mejora a las BT 45C. Este modelo tiene unas características notablemente inferiores al modelo previamente mencionado, lo cual motivó algunos cambios en el concepto de la aplicación. Por un lado, las gafas no están provistas de una cámara, con lo que las imágenes que captamos para mostrar en ellas

han de ser obtenidas con el dispositivo móvil al que estén conectadas las gafas. Además, estas gafas no tienen ningún altavoz para emitir sonido, por lo que no se puede añadir ninguna funcionalidad a la aplicación que necesite esta característica, como por ejemplo una guía por voz, o indicar los parámetros que se modifican en el momento en el que se hace el ajuste. Otro contratiempo surgido a raíz de usar este modelo de gafas es que muchas de las funcionalidades del SDK no están disponibles para estas gafas concretas, algunas de las cuales se encontraban dentro de los objetivos iniciales del proyecto. Aun así, hemos sido capaces de implementarlas por otros medios. El último problema que limita nuestros objetivos del proyecto es el extremadamente escaso número de dispositivos que pueden conectarse a las gafas, los cuales además son bastante costosos. Por este motivo se optó por usar el dispositivo controlador propietario que es proporcionado con las gafas en lugar de un teléfono móvil personal.

Además, esta área de desarrollo para pantallas montadas en la cabeza es muy nuevo, por lo que no hemos podido encontrar un amplio número de desarrollos similares, o que por lo menos relacionados con el tema. Otro de los grandes contratiempos que tuvimos que superar durante el desarrollo de la aplicación fue el desconocimiento del lenguaje Android. No ha sido hasta el segundo cuatrimestre de este curso cuando los tres integrantes del grupo hemos cursado la asignatura <<Programación de aplicaciones para dispositivos móviles>> en la que finalmente obtuvimos conocimientos para el desarrollo en Android. Aun así, gracias a la asignatura previamente mencionada y a labores propias autodidactas se consiguió desarrollar una aplicación móvil funcional y acorde con los objetivos del proyecto.

El último gran problema que se ha vivido durante la mayor parte del desarrollo del código para la aplicación ha sido ni más ni menos que el entorno de desarrollo ofrecido por los mismos desarrolladores que mantiene Android, Android Studio. Este entorno de programación ha sido una lacra común para todos los integrantes del grupo, la incompatibilidad entre versiones, el gran consumo de recursos del ordenador o problemas de emulación han sido algunos de los múltiples problemas ocasionados por este IDE. Aún con todos estos problemas no ha habido más remedio que utilizar Android Studio ya que las capacidades que ofrece siguen siendo necesarias para el desarrollo en esta plataforma.

Capítulo 4 - Conclusiones y trabajo futuro

4.1 Conclusiones

Para los tres integrantes del grupo, esta ha sido la primera vez que hemos realizado un proyecto de estas dimensiones. Durante el desarrollo, hemos tenido que enfrentarnos a una gran cantidad de retos de todo tipo. Sin embargo, los que más se han repetido y sobre todo los más difíciles se han debido a dos factores principalmente. Por un lado, han estado siempre presentes las limitaciones del hardware proporcionado por Plusindes, como la falta de cámara en las gafas o la imposibilidad de conectar ningún teléfono que tuviéramos disponible. Por el otro las dificultades que se plantean al no haber mucha documentación disponible al ser la realidad aumentada una tecnología nueva y de nicho.

Los miembros del equipo estamos bastante orgullosos de haber conseguido desarrollar una aplicación que cumple con la mayor cantidad posible de los requisitos propuestos sorteando las barreras de las limitaciones de los componentes físicos. El producto final es una aplicación compatible con las gafas Moverio BT-40S y el controlador BO-IC 400, que muestra en las gafas las imágenes que captura la cámara del controlador mientras que en este último se muestra una interfaz con un diseño simple e intuitivo desde la cual se pueden modificar todos los ajustes de la aplicación. Entrando más en detalle, los ajustes que el usuario puede modificar son el zoom, el contraste, el brillo, el modo de resalte de bordes, el filtro de escala de grises y la exposición de la cámara.

También estamos bastante contentos de haber podido encontrar soluciones a todos aquellos problemas que han surgido desarrollando el código de la aplicación, muchos de los cuales no estaban documentados, gracias al trabajo en equipo y la perseverancia de todos.

Esperamos que con nuestro trabajo y el del equipo de Plusindes se pueda ayudar a tantas personas como sea posible, sin dejar de innovar y buscar soluciones a los problemas con los que conviven diariamente las personas con baja visión y los cuales desgraciadamente son desconocidos para una gran cantidad de personas.

4.2 Trabajo futuro

Aunque hemos sido capaces de desarrollar una gran cantidad de funcionalidades, algunas se nos han quedado en el tintero ya sea por falta de tiempo o por la falta de disponibilidad del hardware adecuado. A continuación, se detallarán varias mejoras y nuevas funcionalidades que se podrían implementar en futuras versiones de la aplicación:

- Utilización de un modelo de gafas con cámara integrada que permita prescindir de la cámara del controlador.
- Introducción de un modo de uso por comandos de voz, sin necesidad de usar la pantalla táctil. De esta manera los usuarios podrían utilizar la aplicación sin tener que mirar la pantalla, lo cual puede generar distracciones y molestias, en especial a los usuarios con baja visión.
- Adición de ajustes para personalizar la fuente de la aplicación, el tamaño y los colores del fondo y de los botones. Cada usuario con baja visión es distinto, por lo que la aplicación debería poder adaptarse a las necesidades de cada uno.
- Utilización de un teléfono móvil personal en vez del dispositivo proporcionado por Epson. El principal objetivo propuesto por Plusindes era el de poder utilizar las gafas con un teléfono móvil en vez de con el controlador propietario, para así hacer más fácil su uso a sus clientes.
- Mejora del rendimiento y estabilidad de la app. Al haber sido desarrollada solo para un dispositivo desconocemos el rendimiento que puede tener la aplicación en caso de ejecutarse en otro entorno y los problemas que podría ocasionar en cuanto a conectividad con otros modelos de gafas y teléfonos. Durante las pruebas que se realizaban para comprobar el funcionamiento de las distintas funcionalidades implementadas, en ocasiones se recibía una notificación por parte de las gafas indicando que se estaban sobrecalentando. También se observó que la batería del controlador disminuía rápidamente durante las sesiones de pruebas, aunque esto puede deberse a que la capacidad de la batería es solamente de 3400 mAh.

Introduction

This report is about the Final Degree Project <<Use of Augmented Reality glasses to improve the autonomy of users with low vision>>. It details the development process of an application for mobile devices with the Android operating system, to help people with low vision condition to increase their autonomy.

The aforementioned pathology, low vision [4] [5], is an ocular condition caused by different diseases such as: age-related macular degeneration, cataracts, glaucoma, diabetic retinopathy, etc. The main vision defects it causes are loss of central vision, peripheral vision, night blindness and blurred vision. The particularity of low vision is that it cannot be treated with conventional methods, whether glasses, contact lenses or medical treatments.

This project arises from a collaboration of the company Plusindes, SL with the Complutense University of Madrid. This company arises in 2014 from the hands of industrial and computer engineers with the mission to improve the lives of people with low vision. The main products they offer are Smart Glasses with Augmented Reality (AR) technology, these glasses in addition to their own software that is used in conjunction with them constitute the RETIPLUS® system. This system works by applying filters, chosen, and adjusted by an optometrist specialist, to the images captured by the glasses' integrated camera and displaying the result on the glasses' screens, in the specific area of the screen that the patient can see. The need to use a specific model of glasses fitted with a proprietary driver device has led the company to look for alternatives, an application that can be used on a wide range of mobile devices that will make it easier for patients to use the RETIPLUS® system, saving them money and increasing their convenience by allowing them to use their own phone.

The work carried out by the students in this report has resulted in an application for Android phones that allows showing the video preview, captured by the device's camera in real time, on the glasses, modifying various camera settings and, in addition, applying filters to what is seen on the glasses using the screen of the mobile device.

Motivation

As we have mentioned before, augmented reality is a booming field which has many applications. This technology still has a long way to go, as it offers endless possibilities when it comes to researching, developing, and realizing projects with it. Many of these projects could significantly improve our quality of life in the future. Major areas such as medicine, entertainment, industry, and advertising are already introducing augmented reality with promising results.

However, augmented reality is recent and therefore it comes with several problems. Some of these drawbacks are the limited availability and high price of the devices needed for its operation or the lack of documentation on how to develop applications on these devices. Also important is the fact that for the moment it is a niche technology, so its use is not very widespread and little research has been done.

As for why we chose this project, the three members of the group were driven by similar objectives. On the one hand, working with such a novel technology and with such a bright future ahead seemed to us an exciting challenge and of high educational value. On the other hand, this is a project with which we hope to improve the quality of life of people with low vision, which was an especially crucial factor for all three of us when choosing this work.

Goals

The initial objective of the project is to create an application that can be executed on the mobile device that has been provided with the glasses. This application must make use of the camera of the device and transmit the images captured by the camera to the screen of the glasses, which will be connected via a cable to the controller. On the other hand, an intuitive and easy-to-use interface will be located on the screen of the mobile device, from which the images displayed on the glasses can be modified according to a series of filters or settings.

Memory structure

This section will summarize each of the chapters of the report, indicating what is covered in each one of them:

- **Chapter 1: Introduction.**

The motivations that have led us to conduct this development, the objectives to be achieved in it, the work methodology chosen, and the different tools we have used to implement the application are exposed.

- **Chapter 2: State of the art.**

A study of the current situation of augmented reality in different work sectors is made, and the status of the devices and software offered by Plusindes is commented.

- **Chapter 3: Application development.**

It discusses how the implementation of the app has been conducted, dividing the milestones obtained in each of the different iterations that have been carried out.

- **Chapter 4: Conclusions and future work.**

We conclude our research and comment on possible improvements or complementary implementations that could be developed in the future.

- **Bibliography.**

We list the references to the various sources we have used for information and research on the topics related to the project to carry out the various parts of the project.

Work plan

The methodology used in the development of this project has been Extreme Programming (XP) [6]. This methodology, proposed by Kent Beck, emerged at the end of the 90's and its main pillars are short development iterations and constant testing.

This last aspect, adaptability, has been extremely useful to us in the realization of the project since throughout the project a multitude of problems have appeared and there have been constant changes in the objectives.

We have implemented this methodology by performing an iterative and incremental development of the application, implementing small improvements with

each iteration. Thus, in the first version of the application a minimal implementation was made and in the following ones the different functionalities were added. With this we were able to respond quickly to changes in the requirements and have constant feedback from the tutors of the work by being able to show all the changes made during each iteration.

Most of the programming has been done by the three of us at the same time, and if not, at least in pairs. This is so that we all know each other's work and the code is never left unchecked, as it was essential to know the implementation of each aspect of the application by each member of the project due to the characteristics of the project.

Figure 4.1 shows a Gantt chart which represents each of the stages of the project and the time required to complete them.

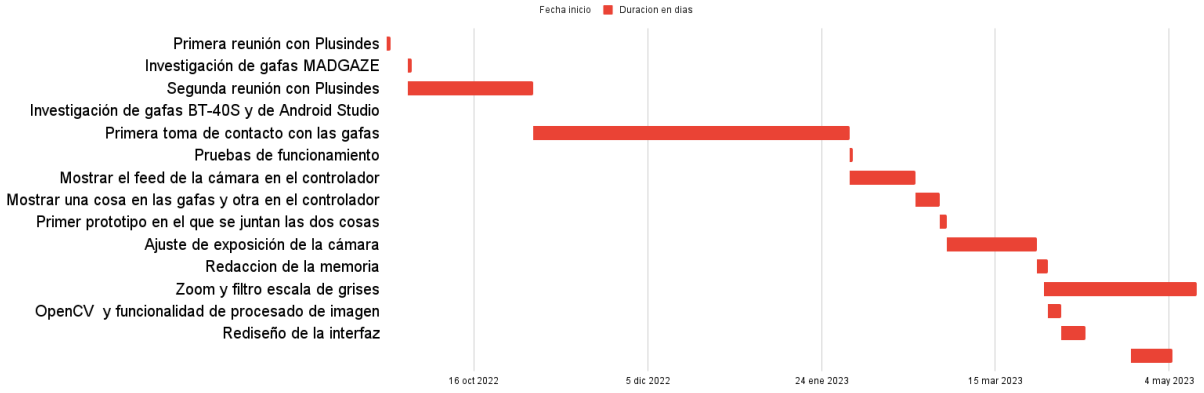

Figure 4.1 Resulting Gantt Diagram of the project.

As you can see in the diagram, we started the project by meeting with the Plusindes team several times and researching the products they lent us, as well as the technologies we were going to use during the development. Later, once we had free access to the glasses, we conducted tests and research on how they worked. Once we were familiar with them, we started to develop the different iterations of the development until we reached the final iteration with all the functionalities and the redesigned and improved interface.

Development environment and work tools

Software tools

We will now discuss the software tools used for the development of our project.

Android Studio

Android Studio [7] is an Android development environment that uses IntelliJ IDEA software, owned by JetBrains, and was announced in 2013 as a replacement for Eclipse. It has been created specifically for the creation of Android applications. This IDE has Kotlin as its default language, although it also allows the use of others, such as C++ and Java [8].

All the visual part of our application has been created using this environment, which allows us to create and edit the views in a straightforward way, dragging the elements that we want to introduce in them. In the same way, all the functional part has also been developed using Android Studio, together with the OpenCV library.

OpenCV Library

OpenCV [9] is a free library for computer vision that appeared in 1999 and to this day remains one of the most famous libraries in this field. Some examples of applications in which OpenCV is used are, among others: object recognition, tracking, facial and gesture recognition, or augmented reality. It stands out because it is a widely documented library, it is free to use and multiplatform.

The use of this library has allowed us to achieve some of the most important parts of the project. Its concrete use is detailed later in the development section.

Hardware tools

The following section contains an explanation of the physical devices provided by the company Plusindes for the development of the project, with which we have been evaluating the operation of the application as we progressed through the different iterations.

Moverio BT 40S Smart Glasses

One of the main hardware tools we have been working with are the **Moverio BT-40S glasses**, whose design can be seen below in Figure 4.2.

Figure 4.2 Moverio BT-40S glasses with the BC-IC400 controller

These are smart glasses, created by Epson, a Japanese company founded in 1942 that manufactures printers, projectors, televisions, and augmented reality glasses, among other electronic components.

The glasses manage to implement augmented reality by projecting two different images on each of their lenses, which redirect them to our eyes. These are combined with the background and merged in our brain to give rise to the final image that we see [10].

Moverio BO-IC 400 Smart Controller

The other key hardware tool for the development of the project has been the Moverio BO-IC 400 Smart Controller [11], which we can also see in Figure 4.2 above.

This is a mobile device, also from the Epson brand, based on Android and designed to be used together with Moverio smart glasses. It has an integrated touch screen and several buttons that allow the user to interact with it.

It also has several advanced sensors, such as a compass, 9-axis motion sensors, GPS or an accelerometer. In addition to these, it also has built-in speaker, microphone, and high-resolution camera, which has autofocus and a flashlight.

To connect the device to the glasses, there is a USB Type-C port with DisplayPort Alt compatibility, and another one that is used to charge the device or connect external

peripherals. In addition, the glasses also include support for a small range of cell phones, which can be connected to them in the same way as the controller.

Other tools

Discord

Discord [12] is a free cross-platform voice chat and instant messaging service created in 2015. It is divided into servers, which are further divided into voice and text channels. Discord allows users to make calls or exchange messages regardless of whether they use the web version or the app, and regardless of the device they are using.

We have used it for most calls and teamwork, thanks to the aforementioned voice and text chat features, as well as the screen-sharing feature that has been very useful when scheduling together.

Google Drive

Google Drive [13] is a service used to host files created by Google in 2012. It allows users to store, download, edit and synchronize files in the cloud and share them with other users. It also has a version control system and an automatic file backup system.

We have used it to keep a shared folder with different informative files, memory, as well as the minutes of the meetings we have been holding.

GitHub

GitHub [14] is a service created to store projects using the Git system for version control, now owned by Microsoft. It allows us to keep track of all the changes made to the code of an application under development. In addition, this platform also allows us to work on different versions, called branches of the code simultaneously and then merge them into a single version, as well as to work remotely in a straightforward way.

We have used GitHub to create the repository of our project, where we have been saving all the versions of our code and the different branches. This way we could share the code and control all the changes made to it.

Conclusions and future work

Conclusions

For the three members of the group, this was the first time we have conducted a project of this size. During the development, we have had to face many challenges of all kinds. However, the most repeated and especially the most difficult ones were mainly due to two factors. On the one hand, the limitations of the hardware provided by Plusindes have always been present, such as the lack of a camera in the glasses or the impossibility of connecting any phone we had available. On the other hand, the difficulties posed by the lack of documentation available as augmented reality is a new and niche technology.

The members of the team are quite proud to have managed to develop an application that meets as many of the proposed requirements as possible while overcoming the barriers of the limitations of the physical components. The final product is an application compatible with the Moverio BT-40S glasses and the BO-IC 400 controller, which displays on the glasses the images captured by the controller's camera, while the controller displays an interface with a simple and intuitive design from which all the application settings can be modified. Going into more detail, the settings that the user can modify are zoom, contrast, brightness, edge highlight mode, grayscale filter and camera exposure.

We are also quite happy to have been able to find solutions to all those problems that have arisen while developing the application code, many of which were not documented, thanks to the teamwork and perseverance of all.

We hope that with our work and that of the Plusindes team we can help as many people as possible, while continuing to innovate and seek solutions to the problems that people with low vision live with on a daily basis and which unfortunately are unknown to a large number of people.

Future Work

Although we have been able to develop a large number of functionalities, some of them have been left behind either due to lack of time or lack of availability of the appropriate hardware, and we will now detail several improvements and new functionalities that could be implemented in future versions of the application:

- Use of a model of glasses with integrated camera that allows to dispense with the controller's camera.
- Introduction of a mode of use by voice commands, without the need to use the touch screen. This would allow users to use the application without having to look at the screen, which can be distracting and annoying, especially for users with low vision.
- Addition of settings to customize the application font, size and colors of the background and buttons. Each user with low vision is different, so the application should be able to adapt to the needs of each user.
- Use of a personal cell phone instead of the device provided by Epson. The main objective proposed by Plusindes was to be able to use the glasses with a cell phone instead of the proprietary controller, in order to make them easier to use for their customers.
- Improved app performance and stability. As it has been developed only for one device, we do not know the performance of the application in case of running it in another environment and the problems it could cause in terms of connectivity with other models of glasses and phones. During the tests that were conducted to check the performance of the various implemented functionalities, sometimes a notification was received from the glasses indicating that they were overheating. It was also observed that the battery of the controller was rapidly decreasing during the testing sessions, although this may be due to the battery capacity of only 3400 mAh.

CONTRIBUCIONES PERSONALES

Iulius Gherasim

Durante el desarrollo de este Trabajo de Fin de Grado me he encargado de escribir gran parte del código de la aplicación. He implementado todas las funcionalidades de la aplicación: uso de la cámara, vista en la pantalla de las gafas, ajustes de zoom y exposición, filtro de escala de grises, filtro de resalte de bordes y filtro de contraste y brillo. También hice la primera versión de la interfaz de usuario, aunque esa versión solo se utilizó para demostrar las funcionalidades.

Para poder implementar todo lo anterior tuve que documentarme a fondo sobre el funcionamiento de Android y posteriormente de la biblioteca OpenCV. Para ello consulté multitud de artículos de la web para desarrolladores que ofrece Android y también consulté los tutoriales disponibles en la web de OpenCV.

En la memoria me he encargado de escribir la introducción del capítulo 1 y gran parte del capítulo 3, mayoritariamente la parte relacionada con el desarrollo de la aplicación. También me he encargado de tomar las fotos y realizar los distintos diagramas que aparecen en las figuras de la memoria. Para obtener las figuras de los resultados de las distintas opciones de la aplicación realicé capturas de pantalla con mi teléfono. Por otro lado, las figuras que ilustran los algoritmos de detección de bordes las obtuve utilizando Python, ya que con pocas líneas de código permite ejecutar los algoritmos y guardar los resultados en imágenes. El diagrama de Gantt lo creé en Hojas de cálculo de Google, el diagrama del filtro de escala de grises lo hice utilizando las herramientas de la web Visual Paradigm y por último el diagrama del concepto de la aplicación lo cree utilizando Figma.

Iñaki Berrocal Díaz

Este ha sido un proyecto en conjunto en el que los tres integrantes hemos tenido igual parte de contribución y responsabilidades. Gran parte del proyecto lo hemos realizado los tres juntos al mismo tiempo, y por tanto se vuelve más difícil hacer una diferenciación del trabajo particular de cada uno por separado.

A lo largo de la elaboración de este Trabajo de Fin de Grado me he encargado de recoger la información de las reuniones en actas para su posterior uso en la memoria. También he investigado y recopilado otros tipos de información útil junto con mis compañeros para la elaboración de la misma, como plantillas, tutoriales, consejos y otros trabajos y artículos con el fin de ayudarnos a redactar el documento de la mejor manera posible.

También he reunido toda la información posible sobre las gafas, el controlador, el SDK de Moverio, información sobre Android y Android Studio y sobre las demás aplicaciones y herramientas que hemos usado durante el proyecto, además de buscar gran parte de la bibliografía usada en la memoria y con la que hemos obtenido información útil para el proyecto.

A parte, me he encargado de la organización del equipo y la metodología de trabajo, de proponer fechas límite para los diferentes hitos, proponer reuniones y de decidir en conjunto con mis compañeros cuales eran los objetivos a corto plazo que debíamos completar en cada iteración. Agregando a lo anterior, he ayudado también a mis compañeros en la elaboración del código de la aplicación en las distintas fases del proyecto.

En lo que respecta a la memoria he realizado las partes correspondientes al resumen inicial, la motivación, objetivos, plan de trabajo, parte del entorno y herramientas de trabajo, parte del capítulo 3 y los problemas del desarrollo, el capítulo 4 y la bibliografía.

Pablo Valdés Balduz

Durante la realización de este proyecto, me he encargado de la elaboración de la parte front-end de la aplicación, sobre todo de la versión final: la idea era crear un menú sencillo, visual e intuitivo, donde las distintas configuraciones fueran fácilmente reconocibles y ajustables, por eso elegimos botones y textos grandes, así como una paleta de color con unos colores diferenciados. Hice diversos prototipados, pero al final elegimos la interfaz que hemos terminado implementando ya que el contraste del blanco del fondo con el azul y gris de los botones, así como el negro de los textos nos parecía que quedaba bien y que era agradable a la vista.

Además, durante las fases previas de desarrollo, colaboré buscando diferentes documentos que nos pudieran servir de guía para abordar el desarrollo de una aplicación utilizando un hardware de estas características, buscando en páginas oficiales de modelos parecidos a las BT-40S, así como preguntando a conocidos que trabajaban en el campo de la oftalmología En cuanto a esto último, también me informé sobre las distintas patologías que pueden padecer los pacientes que utilizan estas gafas, para así poder dar ideas sobre qué ajustes podrían ser de utilidad para ellos en nuestra aplicación.

También busqué diferentes desarrollos que se hubieran realizado en el que se planteara también el separar la aplicación en dos partes: un menú o controlador, y una segunda pantalla donde se mostrase otra vista, a poder ser, la cámara de un dispositivo Android, para poder hacer un primer boceto de cómo abordar el proyecto y entender la lógica detrás de separar en "dos" una aplicación.

En la memoria, me he dedicado sobre todo a redactar el segundo capítulo, que trata sobre el estado del arte, pero también he ayudado en el capítulo uno, en la parte del entorno y las herramientas de trabajo que hemos utilizado en nuestro desarrollo.

Para el capítulo del estado del arte, también estuve recabando información, buscando distintos informes y estudios acerca de los avances de la AR en distintos campos, como en el sector de educación, industria, videojuegos, etc.

Para la parte de las herramientas de trabajo, me reuní con mis compañeros y fuimos enumerando cada uno de los entornos, programas, bibliotecas y elementos de software y hardware que hemos utilizado a lo largo de nuestro desarrollo, tanto para la implementación y pruebas de la aplicación, como para alojar nuestro código, llevar a cabo las distintas reuniones que hemos tenido, etc.

BIBLIOGRAFÍA

[1] "Retiplus," [Online]. Available: https://retiplus.com/.

[2] Plusindes S.L., "Plusindes," [Online]. Available: https://www.plusindes.com/.

[3] Epson, "Moverio BT-40S," [Online]. Available: https://www.epson.es/productos/gafas-inteligentes/visualizador-m%C3%B3vil-transl%C3%BAcido/moverio-bt-40s/p/31096.

[4] National Eye Institute, [Online]. Available: https://www.nei.nih.gov/espanol/aprenda-sobre-la-salud-ocular/enfermedades-y-afecciones-de-los-ojos.

[5] Sociedad Española de Especialistas en Baja Visión, [Online]. Available: https://seebv.com/baja_vision/.

[6] K. Beck, "Extreme programming explained: embrace change," in *Extreme programming explained: embrace change*, Addison-Wesley Professiona, 2000.

[7] "Android Studio," [Online]. Available: https://developer.android.com/studio.

[8] "Java," [Online]. Available: https://www.java.com/es/.

[9] "OpenCV," [Online]. Available: https://opencv.org/.

[10] "Smart Glasses Optimal Technology," [Online]. Available: https://corporate.epson/en/technology/search-by-products/other/optical-technology.html.

[11] "Epson BO-IC400," [Online]. Available: https://www.epson.es/es_ES/support/sc/epson-moverio-intelligent-controller---bo-ic400/s/s2098.

[12] "Discord," [Online]. Available: https://discord.com/.

[13] "Google Drive," [Online]. Available: https://www.google.com/intl/es_es/drive/.

[14] "Github," [Online]. Available: https://github.com/.

[15] Teamviewer, "The AR Warehouse," [Online]. Available: https://www.teamviewer.com/en/ar-warehouse/.

[16] Sicma21, "Robots industriales, tecnología y aplicaciones," [Online]. Available: https://www.sicma21.com/robots-industriales-tecnologia-y-aplicaciones/.

[17] Tinmith, "ARQuake," [Online]. Available: https://www.tinmith.net/arquake/.

[18] Niantic, "Pokémon Go," [Online]. Available: https://pokemongolive.com/?hl=es.

[19] Varios, "Augmented Reality Video Games: New Possibilities and Implications for Children and Adolescents," [Online]. Available: https://www.mdpi.com/2414-4088/1/2/8.

[20] Y. O. Mehmet Kesim, "Augmented Reality in Education: Current Technologies and the Potential for Education," 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877042812023907.

[21] J. Ho-Gun Ha, "Hong KoreaMed Synapse," 2016. [Online]. Available: https://synapse.koreamed.org/articles/1044286.

[22] "Activity," [Online]. Available: https://developer.android.com/guide/components/activities/intro-activities?hl=es-419.

[23] "Intent," [Online]. Available: https://developer.android.com/guide/components/intents-filters?hl=es-419.

[24] Android Developer, "Camera2," [Online]. Available: https://developer.android.com/reference/android/hardware/camera2/package-summary.

[25] Android Developer, "CameraManager," [Online]. Available: https://developer.android.com/reference/android/hardware/camera2/CameraManager.

[26] Android Developer, "Versioning (Camera API2)," [Online]. Available: https://source.android.com/docs/core/camera/versioning#camera_api2.

[27] Android Developer, "TextureView," [Online]. Available: https://developer.android.com/reference/android/view/TextureView.

[28] Android Developer, "SurfaceView," [Online]. Available: https://developer.android.com/reference/android/view/SurfaceView.

[29] Android Developer, "CameraCaptureSession," [Online]. Available: https://developer.android.com/reference/android/hardware/camera2/CameraCaptureSession.

[30] Android Developer, "CaptureResult," [Online]. Available: https://developer.android.com/reference/android/hardware/camera2/CaptureResult.

[31] Android Developer, "Presentation," [Online]. Available: https://developer.android.com/reference/android/app/Presentation.

[32] "Dialog," [Online]. Available: https://developer.android.com/reference/android/app/Dialog.

[33] "Android developer, DisplayManager," [Online]. Available: https://developer.android.com/reference/android/hardware/display/DisplayManager.

[34] "Android developer, MediaRouter," [Online]. Available: https://developer.android.com/jetpack/androidx/releases/mediarouter.

[35] "Android developer, CameraX," [Online]. Available: https://developer.android.com/training/camerax.

[36] "Android developer, ListenableFuture," [Online]. Available: https://developer.android.com/guide/background/asynchronous/listenablefuture.

[37] "Android developer, MainExecutor," [Online]. Available: https://developer.android.com/guide/background/asynchronous/listenablefuture#adding_a_callback.

[38] "Android developer, PreviewView," [Online]. Available: https://developer.android.com/reference/androidx/camera/view/PreviewView.

[39] "Android developer, CameraControl," [Online]. Available: https://developer.android.com/reference/androidx/camera/core/CameraControl.

[40] "Android developer, ColorMatrix," [Online]. Available: https://developer.android.com/reference/android/graphics/ColorMatrix.

[41] "Android developer, Canvas," [Online]. Available: https://developer.android.com/reference/android/graphics/Canvas.

[42] L. Roberts, "Machine Perception of 3-D Solids," in *Optical and Electro-optical Information Processing*, MIT Press, 1965.

[43] J. Prewitt, "Object enhancement and extraction," in *Picture Processing and Psychopictorics*, Nueva York, B. Lipkin and A. Rosenfeld, Eds., 1970, pp. 75-149.

[44] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 8, no. 6, pp. 679-698, 1986.

[45] I. Sobel and G. Feldman, "A 3×3 isotropic gradient operator for image processing," in *Pattern Classification and Scene Analysis*, pp. 271-272.

[46] "OpenCV: Sobel Derivatives," [Online]. Available: https://docs.opencv.org/3.4/d2/d2c/tutorial_sobel_derivatives.html.

[47] "OpenCV: Laplace Operator," [Online]. Available: https://docs.opencv.org/3.4/d5/db5/tutorial_laplace_operator.html.

[48] "OpenCV: Canny Edge Detector," [Online]. Available: https://docs.opencv.org/3.4/da/d5c/tutorial_canny_detector.html.

[49] USC, [Online]. Available: https://sipi.usc.edu/database/database.php?volume=misc&image=10#top.

[50] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library, Cambridge: O'Reilly, 2008.

[51] "OpenCV: cv::Mat Class Reference," [Online]. Available: https://docs.opencv.org/4.x/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b.

[52] "converTo formula," [Online]. Available: (https://docs.opencv.org/4.x/d3/d63/classcv_1_1Mat.html#adf88c60c5b4980e05bb556080916978b).

[53] "MadGaze," [Online]. Available: https://www.madgaze.com/.

[54] Epson, "Moverio BT-45C," [Online]. Available: https://www.epson.es/es_ES/productos/gafas-inteligentes/visualizador-m%C3%B3vil-transl%C3%BAcido/moverio-bt-45c/p/36977.

APÉNDICES

Apéndice A - Enlace al GitHub del proyecto

El código desarrollado durante este proyecto se encuentra accesible en este enlace: https://github.com/luliusgh/TFG-AR