Aplicación de la Realidad Aumentada para la asistencia en enseñanza médica

Trabajo Fin de Grado en Ingeniería de Computadores

Nicolás Bueno Mora
Ignacio Cerdá Sánchez
Ricardo Eugui Fernández

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Febrero 2017
Curso académico: 2016/2017

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Aplicación de la Realidad Aumentada para la asistencia en enseñanza médica

Trabajo de Fin de Grado en Ingeniería de Computadores

Nicolás Bueno Mora
Ignacio Cerdá Sánchez
Ricardo Eugui Fernández

Dirigida por el Doctor

Pedro Pablo Gómez Martín

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Febrero 2017
Curso académico: 2016/2017

A Delibird
por inspirarnos
en los momentos
de frustración.

¡Tú siempre con tus "NO PUEDE HACERSE"!
¿Es que escuchándome no estabas?.
Yoda a Luke Skywalker.
STAR WARS, episodio V,
El Imperio Contrataca.

Autorización

Nosotros, Nicolás Bueno Mora, Ignacio Cerdá Sánchez y Ricardo Eugui Fernández, alumnos matriculados en la asignatura Trabajo de Fin de Grado (TFG) en la Facultad de Informática de la Universidad Complutense de Madrid durante el curso 2016/2017, dirigidos por Pedro Pablo Gómez-Martín autorizamos la difusión y utilización con fines académicos, no comerciales, y mencionando expresamente a sus autores del contenido de esta memoria, el código, la documentación adicional y el prototipo desarrollado.

Agradecimientos

Ante todo queremos agradecer al Catedrático Luis Vázquez Martínez su total disposición, por habernos facilitado los medios para poder hacer realidad este proyecto; gracias a su asignatura ECTED (Escenarios Cientificos y Tecnológicos de la Defensa) nos pusimos en contacto con la unidad de telemedicina que nos dio la idea para este TFG (Trabajo Fin de Grado), sin olvidar su inestimable ayuda a la hora de solventar las trabas burocráticas en la búsqueda de director. También le agradecemos la confianza depositada en nuestro grupo en todo momento y la disponibilidad para interceder en cualquier situación en la que pudiese ayudarnos.

Mención especial al Doctor José Luis Vázquez Poletti por habernos alentado en la búsqueda de un tema tan desafiante, donde poder demostrar todo lo aprendido y con el que concluir esta etapa académica. Gracias por haber estado dispuesto a ayudarnos incluso sin ser parte de nuestro proyecto de forma oficial.

Por último quisiéramos agradecer a nuestro director, Pedro Pablo Gómez Martín, por haber aceptado llevar nuestro proyecto a pesar de no tener relación directa con su investigación, sacando tiempo de sus otras obligaciones para guiarnos de la mejor forma que ha podido.

Nicolás Bueno Mora

Individualmente, me gustaría sumar a amigos y familiares por haber estado, en la medida que les ha sido posible, cuando les he necesitado, ofreciéndome su apoyo y compresión para ayudarme a afrontar los retos que han ido surgiendo.

No quiero olvidarme de quienes me han apoyado algo más indirectamente como las bibliotecarias de la facultad, mis compañeros y profesor de deporte, así como mis vecinas de piso durante mi estancia Erasmus.

Por todo lo que me han aportado, reservo un recordado último lugar de menciones a mis compañeros de TFG, que además de haber participado en este proyecto poco convencional, han estado durante abundantes y largas horas a mi lado realizando un gran trabajo en equipo.

Ignacio Cerdá Sánchez

En primer lugar gracias a mis compañeros Nicolás y Ricardo por su apoyo y comprensión durante el arduo y prolongado camino que ha supuesto este proyecto, sin ellos no habría sido capaz de aguantar hasta el final y concluirlo de forma satisfactoria.

También me gustaría dar las gracias a mis padres, mi hermana Marta y mis familiares por haberme soportado durante todos estos años de carrera, ha sido una etapa realmente larga llena de dificultades y ellos siempre han estado ahí apoyándome en todo lo posible. Un recuerdo especial a los que no podrán presenciar este momento por no estar ya entre nosotros.

No puedo olvidarme de mis conocidos y amigos, sobre todo de Miguel, Gema y Raúl, gracias por padecer mis reiteradas quejas/lamentos y darme los ánimos necesarios para continuar cuando lo veía todo perdido. El que la sigue la consigue.

Por último, pero no menos importante, un pequeño tributo a todos esos docentes que han participado durante todo mi aprendizaje y sin los que habría sido imposible estar hoy aquí. Gracias a todo lo que la educación pública ha podido aportarme y espero que siga aportando a las generaciones venideras por muchos años.

Ricardo Eugui Fernández

Este apartado tiene dos partes claramente diferenciadas, la familia y amigos por un lado y mis compañeros de TFG por el otro. Sin estas dos partes este proyecto jamás hubiera finalizado. Por ello voy a dar las gracias, porque dar las gracias es gratis y es de bien nacidos ser agradecidos.

Empezando por la primera parte, debo dar las gracias a Laura. Sin tu apoyo, comprensión y ayuda yo seguiría dando tumbos intentando encauzar el proyecto. En estos años he aprendido más de ti y sobre lo que soy, y lo que soy es gracias a ti.

Conchi, la madre de las madres, tus ánimos y fuerza enviada desde mi Pamplona natal han sido clave para mi en momentos difíciles. De tí he aprendido todo lo que sé desde niño, sobre todo a ser mejor persona. Todos hablan que su madre es la mejor, pero no tienen ni idea, la mejor eres tú.

Ricardo, Sergio, mi padre y mi hermano. El uno con su humor y jovialidad, el otro con su saber estar y como actuar en cada situación. Vuestra fuerza es mi fuerza.

Por último debo nombrar a mi otra madre, que recientemente nos dejó y no pudo ver como su nieto se convertía en ingeniero. Filo, abuela, madre, te echo de menos tanto que hasta duele escribir estas líneas. Allí donde estés con el abuelo seguro que estás muy orgullosa de mi. Siempre conmigo.

Carlos, Dani, Ion, Gabriel, David, Leire, Tom, Iker, Bea, Idoia, Juanma,

Simona, Lidia, Nuria, todos habéis sido importantes para mi en esta etapa de mi vida ya sea a distancia o presencialmente. Gracias a todos, de corazón.

Y en la otra parte, Nico y Nacho, mis compañeros y amigos con los que nos embarcamos en este proyecto tan interesante, difícil, puntero y de frontera como diría el gran Luis Vázquez. Este proyecto que nos ha traído por la calle de la amargura y de la felicidad, como una tragicomedia griega. Gracias a los dos por vuestro esfuerzo, dedicación, ganas y por habernos sabido entendernos y trabajar en equipo, es que es una de las cosas mas difíciles de esta vida, junto con convivir con alguien. Os lleváis de este proyecto mucho mas que un aprendizaje, os lleváis en mi a un amigo para lo que necesitéis. Dicho queda.

A mis profesores, a los que me han enseñado tanto académicamente como personalmente. A los buenos y a los malos. He aprendido de todos. Gracias.

Acabaré con una frase que encaja en el proyecto y en mi vida, de André Gide: *Cree a aquellos que buscan la verdad, duda de los que la han encontrado.*

Resumen

La telemedicina es una de las áreas que mejor aprovechan las nuevas tecnologías, permitiendo a pacientes contactar con médicos a miles de kilómetros de distancia, conectando cualquier lugar del planeta. Este servicio permite hoy en día conseguir un diagnostico preciso, prescribir el tratamiento adecuado, estabilizar al paciente, decidir la necesidad de traslado y supervisar la evolución del paciente, todo esto sin tener que estar físicamente en la misma habitación.

La idea original de este TFG era solventar un problema importante que tienen las unidades desplazadas fuera de nuestras fronteras. Este problema es que existe una gran dificultad en el correcto entendimiento entre cirujanos a distancia debido a las limitaciones de la cámara cenital con la que emiten las operaciones. Dicho problema surge porque en las misiones fuera del país lo habitual es enviar a especialistas en traumatología, que es la especialidad más demandada en conflictos donde suelen desplazarse nuestras tropas. De ahí que si surgen necesidades de cirugía de otro ámbito, podría ser muy útil este tipo de tecnologías.

La idea original de poder mostrar la realidad aumentada sobre los cristales de las propias gafas es modificada porque el objetivo inicial era demasiado ambicioso, limitado por el *hardware*. Se pensó en empezar por algo mas abarcable para continuar con la idea original si se avanzaba con el proyecto a tiempo. Así surge la necesidad de orientarlo hacia docencia.

El objetivo final de este Trabajo Fin de Grado es la realización de una aplicación de realidad aumentada para ayudar en la docencia dentro de la unidad de telemedicina. Haciendo uso de unas gafas inteligentes, se busca conseguir enviar al hospital o a un aula lo mismo que ve el cirujano en tiempo real en el quirófano. A su vez se podrían proporcionar indicaciones sobre ese vídeo para los estudiantes que están presenciando la operación, con ello se conseguiría despejar el quirófano en las operaciones mas básicas, que son las que permiten alumnos alrededor y ademas poder ver las más delicadas pero sin perder la visión privilegiada que tiene el cirujano y la oportunidad de aprendizaje que tanto necesitan los futuros médicos.

Palabras clave: Gafas Inteligentes, Telemedicina, FFmpeg, Realidad Aumentada, Streaming, Icecast, ARToolKit, Unity.

Abstract

Telemedicine is a field where new technologies are extensively applied, enabling doctors communicate with their patients even when they are thousands of kilometres apart, connecting everywhere in Earth. Nowadays, this service allows to give an accurate diagnostic, prescribe the precise treatment, stabilize a patient, decide the need of moving the patient and supervise his progress, without needing to be in the same room.

The original concept was to resolve an important issue suffered by military units sent outside our borders such as surgeries where remote asistance is needed, the image is recorded by a zenithal camera, thus increasing the difficulty in seeing and guiding for the people givig advice. This situation derives from the fact that medics sent abroad are specialised in othopedic surgery because it is the most needed in conflicts where our troops are deployed. Therefore, if other specialities are required, this kind of technology is going to prove extremely helpfull.

The initial idea of displaying AR through the smart glasses has been modified because it is overambitious given the specifications of the device. Therefore, the scope has been reduced to cover a more feasible functionality and continue with the original concept in the eventuality the development would allow it. As a result, the project is focused on teaching.

The purpose of this Final Year Project is the creation of an AR application to assist in teaching for the telemedicine department.

Using smartglasses, consists in sending the live image being seen directly by the surgeon at the operating theatre to the hospital or a classroom. Moreover, some instructions could be indicated over the video footage for the students watching the surgey. This way, only the necessary personnel would be present at the operating room for most simple surgeries as well as students could learn from more critical surgeries with the privileged view of the main surgeon.

Keywords: Smartglasses, Telemedicine, FFmpeg, Augmented Reality, Streaming, Icecast, ARtoolkit, Unity.

Índice

Índice de figuras

Índice de Tablas

Parte I

Estado del Arte

Esta primera parte de la memoria presenta los conceptos básicos sobre el proyecto de fin de grado. Contiene un capítulo de introducción y una descripción de conceptos imprescindibles para la comprensión del proyecto como la telemedicina, el *streaming*, la realidad aumentada y la realidad virtual.

Capítulo 1

Introducción

1.1. Introducción

1.1.1. Antecedentes

El proyecto que se describe en esta memoria surge gracias a la asignatura ECTED, incluida en la Cátedra Juan de Borbón, impartida por el Catedrático y Académico Luis Vázquez Martínez. Fruto de una de las conferencias permitió establecer contacto con Fernando Setién Dodero profesor asociado de la Universidad Europea y miembro en aquel momento del departamento de telemedicina del Hospital Central de la Defensa Gómez Ulla de Madrid.

Entre las propuestas ofertadas, todas ellas surgidas de las necesidades diarias de la unidad en la que trabajaba, pareció interesante la enfocada a la asistencia remota mediante videoconferencia.

1.1.2. Objetivos

Para el desarrollo de este trabajo se partió de un problema que tienen en el servicio de telemedicina. Éste se da cuando se ha de prestar apoyo a un médico que se encuentra en una misión lejos del hospital.

Por lo general los cirujanos desplazados suelen tener como formación la especialidad de traumatología. Si es necesario realizar una intervención distinta y no es posible trasladar al paciente se recurre a la telemedicina como soporte.

La situación concreta se produce cuando el cirujano está operando y necesita ser supervisado desde España. Por lo general existe una cámara que proporciona una visión cenital de la intervención, pero la imagen que ésta envía se ve perturbada por las cabezas del personal; a su vez el cirujano recibe instrucciones en un monitor auxiliar situado a un lado, que le hace tener que desviar la visión del paciente para recibir asistencia.

El objetivo de este Trabajo Fin de Grado es hacer uso de unas gafas

inteligentes para conseguir enviar al servicio de telemedicina lo mismo que ve el cirujano en tiempo real en el quirófano. A su vez haciendo uso de la realidad aumentada se podrían proporcionar indicaciones sobre ese vídeo para los estudiantes que están presenciando la operación, pero liberando el quirófano de alumnos en las operaciones más delicadas sin perder la visión privilegiada que tiene el cirujano y la oportunidad de aprendizaje que tanto necesitan los futuros médicos.

Dentro de este proyecto hay varios apartados a desarrollar:

- Aplicación cliente en gafas inteligentes.
- Servidor para procesado del vídeo.
- Aplicación para el seguimiento y observación de la operación desde la sección de telemedicina.

Para poder llegar a desarrollar los apartados anteriormente mencionados del proyecto surgen una serie de retos a superar:

- Realizar una investigación para seleccionar un hardware que emita desde el quirófano, en este caso unas gafas inteligentes.
- Investigar sobre el sistema operativo que utilizan y las opciones que nos brinda el SDK que puedan incluir.
- Estudiar sobre los posible protocolos de codificación de vídeo y audio para la realización del *streaming*.
- Investigar los diferentes tipos de servidor para el procesado de audio y vídeo entre las distintas aplicaciones cliente.
- Buscar información sobre la implementación de la realidad aumentada u otras tecnologías y librerías para desarrollar la aplicación de guiado para la docencia dentro del hospital.

1.1.3. Plan de trabajo y organización de la memoria

El proyecto se compone de varias fases, una primera de investigación sobre dispositivos *hardware* capaces de grabar y retransmitir vídeo desde el quirófano y una segunda de desarrollo e implementación del *software* necesario para enviar la imagen desde las gafas inteligentes, procesar este flujo de datos en un servidor y reenviarlo. Por último se tiene una aplicación donde visualizar y manipular el contenido recibido por *streaming* desde el servidor.

1.1.4. Telemedicina

Según la OMS se puede definir la telemedicina como *aportar servicios de salud, donde la distancia es un factor crítico, por cualquier profesional de la salud, usando las nuevas tecnologías de la comunicación para el intercambio válido de información en el diagnóstico, el tratamiento y la prevención de enfermedades o lesiones, investigación y evaluación, y educación continuada de los proveedores de salud, todo con el interés de mejorar la salud de los individuos y sus comunidades.*

La telemedicina es una de las áreas más innovadoras dentro del sector sanitario en la actualidad, combina la asistencia al paciente con las nuevas tecnologías y la posibilidad de ofrecerse en cualquier lugar del planeta.

Actualmente hay múltiples tecnologías que se pueden usar y funciones que se pueden realizar a distancia, como son:

- Sistema de videoconferencia profesional.
- Signos vitales y ECG (Electrocardiogram, EKG en alemán, Elektrokardiogramm, Representación gráfica de la actividad eléctrica del corazón) en tiempo real.
- Radiología: digitalización, envío y recepción de radiografías en formato DICOM (Digital Imaging and Communication in Medicine, Estándar para el intercambio de pruebas médicas).
- Cámara de exploración externa.
- Fuente de luz fría: otorrinoscopio, dermatoscopio y oftalmoscopio.
- Ecografía: desarrollo de lenguaje convenido, investigación en reconocimiento de comandos por voz, ecografía robótica.
- Asistente quirúrgico en cirugía.

Sin olvidar que los principales objetivos de la telemedicina son similares a los de la medicina tradicional, pero a distancia:

- Conseguir un diagnóstico preciso.
- Prescribir el tratamiento adecuado.
- Estabilizar al paciente si fuese necesario.
- Decidir si existe la necesidad de trasladar al paciente.
- Supervisar la evolución del paciente.

Los orígenes de este servicio se remontan a principios del siglo XX en Australia. Haciendo uso de dos radios alimentadas por la dinamo de una bicicleta la gente que vivía en zonas remotas del país se podía comunicar con la Royal Flying Doctor Service. (Wikipedia, 2017)

Este servicio se viene prestando en nuestro país desde principios de los años 20 (Hillán García et al., 2014) en las primeras consultas de radio asistencia sanitaria en alta mar, aunque no se estableció como una unidad propia en el Hospital Central de la Defensa Gómez Ulla de Madrid hasta finales de los años 90.

Su función dentro de las fuerzas armadas es la de intentar proveer la misma atención médica que se da en territorio nacional a zona de operaciones.

En la actualidad el personal del ejército presta cobertura a buques, ONGs, otras fuerzas armadas, centros remotos, peñones y escuadrones de vigilancia aérea.

Para desarrollar dicha asistencia, el departamento viene haciendo uso de la radio, correo electrónico y sobre todo de la videoconferencia que ha mejorado su calidad sustancialmente en los últimos años.

Las tecnologías aplicadas a la telemedicina permiten además de un sistema de videoconferencia profesional, recibir datos como signos vitales, radiología, cámara de exploración, fuente fría (otorrinoscopio, dermatoscopio y oftalmoscopio), ecografía y asistencia quirúrgica del paciente de forma remota.

El servicio no se reduce solo al ámbito militar. En nuestro país se han experimentado varias iniciativas para llevar esta modalidad a pequeñas localidades de difícil acceso, acercar especialidades a centros que carecen de ellas o realizar seguimientos a pacientes crónicos o de avanzada edad, como:

- Hospital de Guadarrama y pequeñas poblaciones de la sierra conectan con el centro hospitalario Ramón y Cajal en Madrid (RTVE, 2016).
- Proyecto PALANTE sobre diabetes en Andalucía (de Andalucía, 2013).
- Iniciativa TelEPOC en País Vasco (Osakidetza, 2014).
- Plan e-tICO para enfermos de cáncer de cabeza y cuello en Cataluña (ICO, 2015).

Como líneas de investigación para el futuro se centran los esfuerzos en la tele-ecografía, asistencia quirúrgica remota, tele-microscopio, mejora de los vehículos asistenciales y monitorización de UVI's a distancia.

1.2. Introduction

1.2.1. Previous work

The described project in this report stems from the ECTD subject, part of the Cátedra Juan de Borbón and taught by the professor Luis Vázquez Martínez. From one of the seminars a relationship was established with Fernando Setién Dodero, professor at Universidad Europea and member of the telemedicine department at Hospital Central de la Defensa Gómez Ulla in Madrid

Among all the possible projects, all born from the needs of the telemedicine department, the one covering remote asistance through videoconference was the most interesting.

1.2.2. Aims

The development of this project starts with the problem at the telemdicine service when remote assistance is given to a doctor deployed far from the hospital.

In general, medics sent abroad are specialised in othopedic surgery. In case a surgery of another specality is required and moving the patient is not an option, telemedicine is used as support.

The problem arises when the surgeon needs guidance from Spain. In general there is a zenithal camera recording the surgery. The image sent is disturbed by personel heads. Moreover, instructions are received in a display situated beside the patient. In order to follow them, the surgeon needs to keep changing focus between the injured and this screen.

The purpose of this Final Year Project consists in using smartglasses for sending the live image being seen directly by the surgeon at the operating theatre to the telemedicine department.

Moreover, some instructions could be indicated over the video footage with AR for the students watching the surgey. This way, only the necessary persomnel would be present at the operating room for most simple surgeries as well as students could learn from more critical surgeries with the privileged view of the main surgeon

There are several aspects to develop in this project:

- A client application for the samrt glasses.
- A server to process the video.
- An application to follow and observe the operation from the telemedicine department.

In order to develop the previous elements, there are some challenges to overcome:

- Research about possible hardware for transmitting the video footage from the operating theatre, in this case smart glasses.
- Study about the operating system used in the selected device and the options offered by its associated SDK.
- Learn about possible protocols for video and audio codification in order to stream.
- Research about different server types to process the audio and video between clients.
- Learn how to implement augmented reality or other technologies and libraries for developping the teaching application at the hospital.

1.2.3. Workplan and report organization

This project consists of several stages. Firstly, a research about hardware devices able to record and send video footage from the operating theatre. Then, developping and implementing required software for sending the image from the smart glasses, process the stream in a server and send it to the clients. Lastly, an application for visualise and manipulate the content received via streaming from the server.

1.2.4. Telemedicine

According to WHO, telemdicine can be defined as follows: bring health services where distance is a key factor for every sanitary individual, using up to date communication technologies for a valid exchange of information about the diagnostic, treatment and prevention of sickness and injuries, investigation and evaluation, and continued education of health bringers, evrything in the benefit of improving individual and community health.

Nowadays, telemedicine is one of the most innovatives fields inside medicine, combining medical assistance with newest technologies and the possibility of being applied everywhere on Earth.

There currently are several activities that can be performed when there is distance between both parties like

- Professional videocall system.
- Vital signals and live ECG.
- Radiology: digitalization, send and reception of X-ray photograph in DICOM format.
- External exploration camera.

- Cold light source: otorrinoscope, dermatoscope y ophtalmoscope.
- Echography: convened language development, investigation in voice commands reconnaissance, robotic echography.
- Surgery assistant.

Without forgetting about the main objectives of telemedicine are similar to those of general medicine but with distance:

- Give an accurate diagnostic.
- Prescribe adecuate treatment.
- Stabilize the patient.
- Decide whether the patient need to be moved.
- Supervise the patient progress.

The beginning of the telemedicine service are at the sart of the 20th century in Australia. Making use of two radios feeded by dynamos from a bicycle, people living in remote areas of the country were able to contact the Royal Flying Doctor Service (Wikipedia, 2017).

This service has come in place in our country from the 1920s (Hillán García et al., 2014) with the firsts radio assistance at sea. It was not until the end of 1990s before it was established as an independent unit at the Hospital Central de la Defensa Gómez Ulla in Madrid.

-

Capítulo 2

Streaming

La necesidad de recibir en tiempo real lo que está sucediendo en el quirófano o en otra ubicación fuera del hospital y poder verlo a la vez que sucede era menester implementar una herramienta como el *streaming* para hacerlo posible.

El *streaming* es la distribución o transmisión digital de un contenido de forma ininterrumpida entre dos dispositivos electrónicos, su uso más extendido es el de la difusión de audio y/o vídeo. Para hacerlo, el contenido se divide en fragmentos que se van enviando al usuario del servicio de forma consecutiva.

Esta transferencia puede hacer uso de múltiples protocolos y tecnologías de codificación dependiendo de los dispositivos implicados y su ubicación.

2.1. Tipos de *streaming*

Sin entrar en tecnicismos, existen dos tipos de *streaming*:

1. *Live streaming*: es la transmisión de un evento en directo. Permite ver lo que está sucediendo en un punto del planeta simultáneamente en otra ubicación, se utiliza por ejemplo para la retransmisión de eventos deportivos o en una videoconferencia.

2. Vídeo bajo demanda: permite a los usuarios acceder a contenidos multimedia y visualizarlos en tiempo real sin la necesidad de tener que esperar a que estos hayan finalizado su descarga para su visualización. Ejemplos son Netflix y Spotify.

2.2. Actores involucrados

Una vez entendido el propósito del *streaming* caben destacar los componentes que son necesarios para que dicha transmisión sea posible.

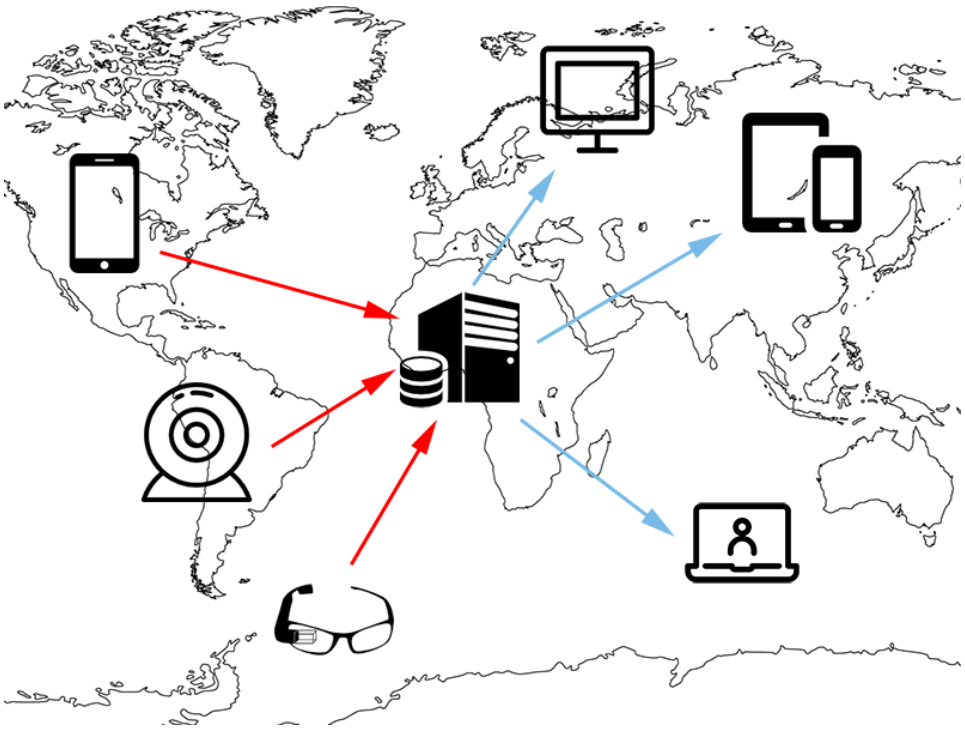

Figura 2.1: Esquema comunicación mediante servidor intermedio

- Emisor: en el *live streaming* sería la ubicación donde está teniendo lugar el evento que va a ser retransmitido en directo. En el caso del VOD el rol de este actor lo desempeñaría también el servidor.
- Servidor: es el componente capaz de procesar el vídeo y el audio recibidos desde donde se origina y el encargado de que le lleguen al cliente o clientes de la forma más optima.
- Aplicación: es el software que recibe el contenido en un dispositivo electrónico para ser visualizado, ya sea mediante una aplicación específica diseñada para este fin o desde el propio navegador web.

2.3. *Codecs*

El acrónimo *codec* proviene de los términos codificador y descodificador, que vienen a ser el conjunto de aplicaciones software o dispositivos *hardware* capaces de comprimir y descomprimir el vídeo/audio involucrado en el *streaming*. Este proceso puede implicar la pérdida de calidad de imagen o sonido dependiendo del *codec* utilizado y la compresión que este realice, en favor de reducir el tamaño en disco de los ficheros generados. Los *codecs* más

importantes son:

- H.263: es un estándar de codificación de vídeo diseñado para la compresión de videoconferencia con poco movimiento (no se captura la imagen completa en cada fotograma sino solo los cambios de posición del sujeto) y con una baja tasa de *bits* (Wikipedia, H.263).
- H.264: conocido oficialmente como MPEG-4 Part 14, es un formato de compresión de datos de audio y vídeo. Es capaz de proveer una imagen de calidad a una menor tasa de *bits* respecto al estándar anterior H.263 (Wikipedia, H.264).
- VP8: formato de compresión abierto libre de regalías creado por Google y distribuido bajo licencia BSD. La aparición de este codec motivó el paulatino abandono de Flash Player por parte de Youtube en su reproductor de vídeo *online*. El popular servicio de videoconferencia Skype lo implementa desde la versión 5.0 (Wikipedia, VP8).
- MP3: oficialmente MPEG-1 Audio Layer III o MPEG-2 Audio Layer III es un formato de compresión de audio digital que, aplicando un algoritmo de compresión con pérdida, permite conseguir un archivo de menor tamaño. Es capaz de reducir el tamaño de una pista en calidad CD entre un 75 % y un 95 % de su tamaño original. Fue desarrollado por Moving Picture Experts Group (Wikipedia, MP3).
- Vorbis: es un *codec* de audio abierto desarrollado por la fundación Xiph.Org. Se creo como replica a una oleada de cartas de infracción de derechos enviadas por parte de los creadores de MP3 a pequeños productores (Wikipedia, Vorbis)
- AAC: desarrollado por varias multinacionales como Sony, AT&T o Dolby, es un *codec* de audio que se basa también en algoritmos de compresión con pérdida. Se convirtió en un estándar de audio en 1997 y es el elegido por grandes empresas como Youtube, Apple o Sony para comprimir audio. Como particularidad y de forma opcional permite incluir protección de derechos de autor en los archivos generados (Wikipedia, AAC).

2.4. Contenedores

Con el término contenedor se hace referencia a un tipo de archivo capaz de almacenar vídeo y/o audio. Permite múltiples *streams* entrelazados. Son fáciles de distinguir gracias a la extensión que da nombre a dichos ficheros.

- 3GP: es un contenedor multimedia definido para la utilización en telefonía móvil para la transmisión de vídeo en redes 3G UMTS. La extensión oficial definida en el estándar es .3gp (Wikipedia, 3GP).

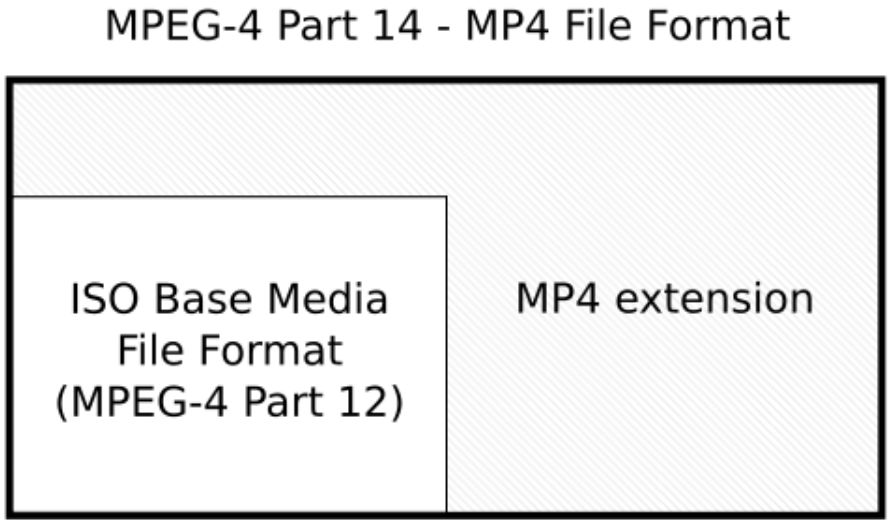

Figura 2.2: Definición gráfica de contenedor y *codec*. (Wikipedia)

- MP4: definido en el ISO/IEC 14496-14:2003, es uno de los contenedores multimedia más utilizados, permite almacenar vídeo, audio, subtítulos e imágenes, con *bitrate* (tasa de bits por unidad de tiempo) variable en función de la velocidad de transmisión. La extensión oficial definida en el estándar es .mp4 (Wikipedia, MP4).
- MP3: es un contenedor de audio utilizado para encapsular sonido. Utilizando *codecs*, se puede conseguir almacenar audio en un archivo de menor tamaño minimizando la perdida de calidad. La extensión utilizada para este contenedor es .mp3 (Wikipedia, MP3).
- MOV: es un contenedor multimedia propietario desarrollado por Apple Inc. basado en MPEG-4 Part 14. La extensión utilizada para este contenedor es .mov (Wikipedia, QuickTime File Format).
- OGG: es un contenedor libre y fuera de patentes diseñado para poder proveer *streaming* en alta calidad. El proyecto lleva en desarrollo desde 1993 por parte de la fundación Xiph.Org. La extensión utilizada para este contenedor es .ogg (Wikipedia, Ogg).
- WebM: es un contenedor abierto y libre desarrollado por Google y orientado a utilizarse con HTML5. Se compone del *codec* de vídeo VP8 y del *codec* de audio Vorbis. Dicho contenedor es el usado en la versión HMTL5 de Youtube. La extensión utilizada para este contenedor es .webm (Wikipedia, WebM).

2.5. Protocolos

Para poder transmitir el audio y el vídeo entre los diferentes actores de este proceso es necesario utilizar una serie de protocolos de transmisión/recepción que envíen y reciban encapsulado en un contenedor el audio/vídeo codificado previamente.

- RTSP/RTP: es un protocolo de control de red diseñado para el establecimiento y control de grandes flujos de datos sincronizados, como audio y vídeo. Se utiliza tanto para el control de la transmisión del servidor al cliente y viceversa. Fue desarrollado por RealNetworks, Netscape y la Universidad de Columbia a partir de 1996. Utiliza TCP para los datos de control y UDP para los datos de vídeo y audio (Wikipedia, Real Time Streaming Protocol).
- RTMP: es un protocolo desarrollado por Macromedia para la realización de *streaming* de vídeo y audio entre un servidor y un cliente con Flash Player. La sesión puede ser cifrada mediante TLS/SSL. Es capaz de transmitir audio encapsulado en MP3 o ACC y vídeo en FLV (Wikipedia, Real Time Messaging Protocol).
- SIP: es un protocolo de la capa de aplicación orientado a la gestión de la sesión en comunicaciones multimedia (Wikipedia, Session Initiation Protocol).
- Icecast: es un protocolo basado en HTTP adaptado al *streaming* mediante el servidor Icecast. La comunicación tiene lugar haciendo uso del método PUT y tiene unas cabeceras dedicadas (ePirat, 2014).

Capítulo 3

Realidad aumentada y realidad virtual

La Realidad Aumentada es la parte tecnológica que motivó este TFG. Es importante porque es el factor diferencial y que le da un plus muy interesante a nivel técnico y tecnológico.

Permite eliminar un factor clave en una operación con estudiantes de medicina, debido a que cuantas más personas se encuentren en la operación, más posibilidades de infecciones hay para el paciente. Además permite poder aprender de operaciones graves y difíciles, que hasta ahora no se podían llevar a cabo con alumnos debido a las razones antes mencionadas.

Para entender que es la Realidad Aumentada, hay que preguntarse, qué es la Realidad Virtual, ya que la primera es una variación de la segunda. Las tecnologías de VR sumergen al usuario dentro de un entorno artificial o simulado, sin tener consciencia del mundo real que lo rodea. La AR, por el contrario, permite al usuario ver e interactuar con el mundo real, en el que se superponen objetos virtuales. Se podría decir que la AR complementa a la VR. La AR y la VR tienen muchos puntos en común, lo que realmente las diferencia es la posibilidad de seguir viendo el entorno físico en el que está el usuario mediante la primera, frente a la segunda que solo permite visualizar lo que se genera para el dispositivo electrónico.

3.1. Realidad virtual

La Realidad Virtual VR es un entorno de apariencia real recreado por un computador que puede ser visualizado haciendo uso de un dispositivo electrónico, como unas gafas o casco de realidad virtual. Permite sumergirse en un mundo modelado mediante 3D donde viajar, conocer e interactuar con los elementos presentes en él.

En la actualidad este entorno se pueden encontrar en los videojuegos

Figura 3.1: Máquina Sensorama de Morton Heilig

diseñados para Playstation VR u Oculus Rift.

3.1.1. Orígenes

Los orígenes son bastante imprecisos, porque hay diferentes opiniones de cuando debería considerarse el inicio de la VR. En 1962, un director de fotografía, Morton Heilig, crea y patenta un simulador llamado Sensorama (figura 3.1) que incorpora imágenes, sonido, vibración y olores. Como curiosidad aun hoy sigue en funcionamiento.

Pocos años después, en 1968, Ivan Sutherland, profesor de la Universidad de Harvard, creó un casco de visión que permitía ver sencillos objetos 3D renderizados en tiempo real. Empleaba dos sistemas de *tracking* para calcular el registro de la cámara: uno mecánico y otro basado en ultrasonidos.

3.2. Realidad aumentada

La Realidad Aumentada (AR) redefine la visión que se tiene del mundo real a través de la cámara de un dispositivo electrónico como puede ser un *smartphone* o una *tablet*. Esta visión se puede enriquecer con elementos que se sobreponen en la pantalla del dispositivo. Uno de los ejemplos actuales de esta tecnología son el juego de Pokemon Go de Niantic (Hipertextual, 2016) o el catálogo interactivo de IKEA (Xataka, 2013).

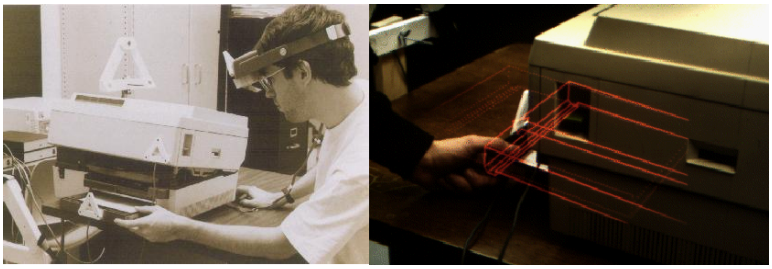

Figura 3.2: Dispositivo Karma como manual de uso de impresora

3.2.1. El nacimiento de la AR

Al mismo tiempo que el investigador de Boeing Tom Caudell inventaba el término Realidad Aumentada en 1992, la tecnología nacía propiamente dicha de otros dos trabajos. El primero surge de L.B. Rosenberg, que trabajaba para la fuerza aérea de Estados Unidos, con un dispositivo que da consejos al usuario sobre cómo realizar ciertas tareas a medida que estas se presentan, una especie de guía virtual.

La otra investigación que dio pie a la AR procedía de la Universidad de Columbia, donde un equipo de científicos inventó un HMD que interactuaba con una impresora. El dispositivo, bautizado como KARMA, proyectaba una imagen en 3D para dar instrucciones a un usuario sobre cómo recargar la impresora, en lugar de acudir al manual de uso. Figura 3.2.

Hasta 1999, la AR seguía siendo de ámbito científico. Eso cambió cuando Hirokazu Kato lanzó ARToolKit a la comunidad de código abierto. Por primera vez, permitió que el seguimiento de captura de vídeo del mundo real se combinara con la interacción de objetos virtuales y proporcionara gráficos 3D que pudieran superponerse a cualquier plataforma. A pesar de que el teléfono inteligente aún no se había inventado, permitía llevar AR a las masas usando un dispositivo portátil simple con una cámara y una conexión a Internet. Casi todos los programas que usan AR basados en Flash que se encuentre una persona a través de su navegador web, han sido posibles gracias a ARToolKit.

En 2001 se presentó Archeoguide (figura 3.3), un sistema financiado por la Unión Europea para la creación de guías turísticas electrónicas basadas en Realidad Aumentada. El sistema proporcionaba información personalizada basada en el contexto, y mostraba reconstrucciones de edificios y objetos mediante una base de datos multimedia adaptada al problema. La comunicación se realiza mediante Wi-Fi, y el sistema era altamente escalable permitiendo diferentes dispositivos como portátiles, PDAs, etc.

En el 2003, Siemens lanza al mercado Mozzies, el primer juego de Realidad Aumentada para teléfonos móviles. El juego superpone mosquitos a la

Figura 3.3: a)Estado actual del templo. b) Recreación con Archeoguide

visión obtenida del mundo mediante una cámara integrada en el teléfono. Este juego fue premiado como el mejor videojuego para teléfonos móviles en dicho año.

Años más tarde, en 2008, Mobilizy crea Wikitude, una aplicación que aumenta la información del mundo real con datos obtenidos de entradas de Wikipedia. Originalmente sólo estaba disponible para teléfonos Android, aunque actualmente puede descargarse para otras plataformas.

3.2.2. Panorama actual

Las nuevas tecnologías avanzan a pasos agigantados y con ellas van naciendo nuevos dispositivos que cubren una necesidad ya existente o la generan de cero.

La popularización de las gafas inteligentes, gracias al lanzamiento que hizo Alphabet de Google Glass, puso de actualidad un nuevo dispositivo que venía a complementar las funcionalidades que tiene un*smartphone*, pero evitando tenerlo en la mano constantemente. Aunque a Google no le salió bien económicamente la primera versión de su proyecto, animó a otros fabricantes

Figura 3.4: Wikitude permite ver información de la wikipedia sobreimpresa sobre la realidad

a redefinir el concepto y sacar sus propias versiones.

A continuación se hace una comparativa entre varios de estos dispositivos. El estudio tiene como fecha de partida el mes de junio de 2015 y se extiende hasta octubre de ese mismo año. Los modelos comparados en su mayoría ya están obsoletos pero se han tenido en cuenta como punto de partida, una vez completado el proyecto los desarrollos serían fácilmente portables a un *hardware* más moderno.

3.2.3. Requerimientos

Dado que el proyecto se va desarrollar dentro del sector sanitario y en una unidad que trabaja bajo presión en tiempo real, se marcan una serie de pautas que se deben seguir a la hora de elegir un dispositivo dentro de los disponibles en el mercado.

- Peso: ya que es un dispositivo que se va a utilizar durante largos periodos de tiempo era importante que contasen con el menor peso posible para reducir el cansancio del usuario.
- Batería: en colación al punto anterior era necesario que la batería durase lo máximo posible, ya que una vez puestas no se pueden retirar hasta finalizada la actividad del usuario que las porta.
- Comodidad: junto al peso era otro de los puntos importantes a tener en cuenta, se buscaba que el dispositivo fuese lo menos intrusivo posible, reduciendo el campo de visión lo mínimo posible y maximizando el área de visionado para la posible aplicación de realidad aumentada.
- Sistema operativo: dada la novedad del dispositivo se compararon opciones y se redujo a Android por ser el sistema más extendido y documentado en la actualidad. Se tuvo en cuenta el sistema operativo de

las HoloLens (Microsoft, 2016) de Microsoft y Android Wear (Google, 2016) de varios modelos pero se descartaron por no estar en el mercado las primeras y falta de documentación del segundo.

- Formatos vídeo/audio: dado que las aplicaciones iban a ser desarrolladas sobre *software* libre se buscó la mayor compatibilidad con formatos de audio y vídeo del mercado.

3.3. Mercado actual de la realidad aumentada

A continuación se incluye una comparativa de los tres modelos que se eligieron después del análisis de más de 15 dispositivos *hardware* (ver tabla 3.1). El primer filtro se acometió debido a que es una tecnología bastante nueva y que muchos de los dispositivos eran proyectos de *Crowdfounding* que o bien no habían llegado a buen puerto, o no había garantías de conseguir el *hardware* a tiempo. Por no mencionar que al ser proyectos minoritarios no había el suficiente apoyo desde la comunidad de desarrolladores para poder resolver dudas o problemas que pudieran surgir.

Así mismo, el dispositivo más famoso de estas características, Google Glass (Wikipedia, Google Glass), se encuentra actualmente retirado del mercado, debido al fracaso de la primera versión. Por lo que se investigó, está en un proceso de cambio radical en muchos aspectos en un proyecto que Alphabet guarda celosamente junto con la comunidad sanitaria privada norteamericana. En cuanto a las HoloLens (Microsoft, 2016) de Microsoft, aunque tienen muy buenas críticas, es un proyecto aún por desarrollar y sin acceso en su momento para desarrolladores externos a Microsoft.

En la tabla 3.1 se muestra una comparativa de los tres modelos ya citados. Como se detallará en el capítulo 4.1.1, finalmente el modelo seleccionado fue Epson Moverio BT-200

	Epson	Sony	Vuzix
Modelo	Moverio BT 200	SmartEyeglass	M100 Smart Glasses
Pantalla	0,42 pulgadas	0,23 pulgadas	4 pulgadas
Resolución	qHD 960x540	419x138	WQVGA 428x240
Ratio	16:9	N/A	16:9
Nº píxeles	518.400 Puntos	N/A	N/A
Campo Visión	aprox. 23°	20°, 19° (H) x 6° (V)	aprox. 15° (diagonal)
Color	24 bits color	Monocromo (verde)	24 bits color
Brillo	N/A	1000 cd/m2	2000 cd/m2
Frecuencia	60 Hz	10-15 fps	N/A
GPS	SI	N/A	SI
Brújula	SI	SI	SI
Giroscopio	SI	SI	SI
Acelerómetro	SI	SI	SI
Micrófono	SI	SI	SI
Tracking	N/A	N/A	SI
Cámara	VGA (640x480)	3 mpx	5mpx
Wireless	802.11b/g/n	802.11b/g	802.11b/g/n
Bluetooth	3.0	3.0	4.0
Micro USB	SI	SI (solo carga)	SI (solo carga)
CPU	TI OMAP 4460	N/A	TI OMAP 4460
RAM	1 GB	N/A	1 GB
Almacenamiento	8 GB	N/A	4 GB
Mem Externa	Micro SDHC	N/A	Micro SD
Codecs Video	MP3, MPG2	N/A	N/A
Codecs Audio	AAC, MP3, WAV	N/A	N/A
Batería	2.720 mAh	N/A	550mAh
Duración	6 horas	2-3 horas	2-3 horas
Tipo de batería	Polímero de litio	Ion de Litio	N/A
Peso Gafas	88 g	77 g	371 g
Peso Controlador	124 g	45 g	N/A
SO	Android 4.0.4	Android 4.4	Android 4.0.4
API	15	19	15
	Epson	Sony	Vuzix

Tabla 3.1: Comparativa modelos Gafas Inteligentes

Parte II

Hardware

Esta segunda parte de la memoria se centra en la parte *hardware*, en concreto en el modelo seleccionado. Se realizará un análisis completo del *hardware* seleccionado, tanto a nivel de especificaciones como las pruebas realizadas con el mismo a fin de determinar las virtudes y las limitaciones del mismo.

Capítulo 4

Análisis sobre el hardware seleccionado

Como paso inicial en el desarrollo del TFG se tuvo que realizar un profundo estudio sobre dispositivos de gafas inteligente disponibles en el mercado. Al ser una tecnología bastante reciente no se contaba con información previa al respecto.

4.1. Modelo seleccionado

Tras una exhaustiva búsqueda se eligió el modelo de Epson, debido a los motivos que se especifican a continuación:

- Trayectoria: se trata de un dispositivo ya testado, en concreto el modelo utilizado es la segunda versión de las Epson Moverio, lo que implica que el proyecto sigue adelante por parte de la marca japonesa, han corregido errores de la primera versión. Recientemente la empresa ha informado de la comercialización de una nueva versión, Epson Moverio BT300, que mejora sustancialmente una de las limitaciones más importantes del hardware, la cámara.

- Sistema operativo: desde el primer momento se tuvo acceso a las especificaciones del sistema operativo y su SDK, algo importante ya que al ser un mercado nuevo lo normal es no contar con la suficiente información.

- Visión: el campo de visión era algo extremadamente importante, eran el modelo que proporcionaba mayor visión para el usuario. Además cuenta con la opción de incluir filtros en los cristales, algo bastante útil ya que la iluminación en el quirófano es demasiado alta y esto hace que los colores se saturen, como resultado la imagen sale blanca, perdiendo

toda opción de recuperación de la imagen por métodos tecnológicos. Esto es conocido como quemar la imagen.

- Comodidad: uno de los hándicaps de las Google Glass, su ergonomía, estaba mejorado en este modelo, debido a que tienen una apariencia cercana a las gafas de ciclista. La comodidad es un factor clave para profesionales de la medicina que necesitan que el dispositivo interfiera lo mínimo en sus labores.

- Duración de la batería: se seleccionó este dispositivo debido a que la duración de la batería en modo vídeo es de 6 horas, tiempo adecuado para realizar las labores necesarias por los usuarios.

4.1.1. Especificaciones

A continuación se describe el dispositivo Epson Moverio BT200 pormenorizadamente.

- Pantalla: las gafas se componen de dos cristales con pantallas panorámicas qHD de 960x540 píxeles de resolución, permiten la visualización en un área de 0,42 pulgadas sin impedir la visión del exterior. El campo de visión es bastante amplio. Incluyen un filtro para colocar sobre los cristales para proteger la visión en zonas con una alta iluminación como se puede apreciar en las figuras 4.1 y 4.2.

- Cámara: es uno de los puntos débiles del modelo. Se encuentra localizada en la parte inferior izquierda, es capaz de capturar vídeo y realizar instantáneas a una resolución VGA de 640x480. En el apartado 4.1.2 se incluyen pruebas al respecto. Fue necesario añadir un filtro a la lente ya que la imagen se quemaba dentro del quirófano por exceso de luz.

- Audio y micrófono: las gafas son capaces de grabar audio en MP3, AAC o WAV y reproducirlo mediante los auriculares externos incluidos. Este aspecto es el menos desarrollado en este contexto ya que el quirófano cuenta con su propio sistema de audio para las comunicaciones.

4.1.2. Pruebas

Se realizaron una serie de pruebas de imagen y vídeo para comprobar la calidad y el comportamiento de la cámara en distintas situaciones:

- Prueba 1 (figura 4.4): se puede observar el comportamiento de la cámara ante una situación de abundante luz natural directa. La imagen se muestra excesivamente blanca, casi quemada en la zona donde incide el sol. Esto representó un problema ya que el quirófano presentaba una

Figura 4.1: Gafas Epson Moverio BT200 con filtro.

situación similar por lo que se tuvo que incluir un pequeño filtro sobre la lente.

- Prueba 2 (figura 4.5): la imagen fue tomada en el exterior con luz natural no directa. La figura es bastante aceptable teniendo en cuenta las limitaciones de la cámara en cuanto a su resolución.
- Prueba 3 (figura 4.6): el resultado fue bastante pobre en un interior con poca luz. La imagen muestra mucho ruido y las formas son prácticamente inapreciables en las zonas menos iluminadas.
- Prueba 4 (figura 4.7): en un interior con poca luz pero con iluminación artificial, la imagen es bastante nítida, quizás un poco blanca en las zonas donde incide la luz directa.

4.1.3. Experiencia de usuario

Las gafas de realidad aumentada permiten hacer uso de un *smartphone* manteniendo un cierto nivel de percepción del entorno físico del usuario.

Por tener como sistema operativo Android, en el campo de visión del usuario se muestra la misma información que puede estar disponible en un *smartphone* convencional bajo este sistema operativo. Las pantallas tienen un cierto nivel de transparencia ajustable para no bloquear la vista del usuario. Sin embargo, a diferencia de los terminales habituales, para manejar estas

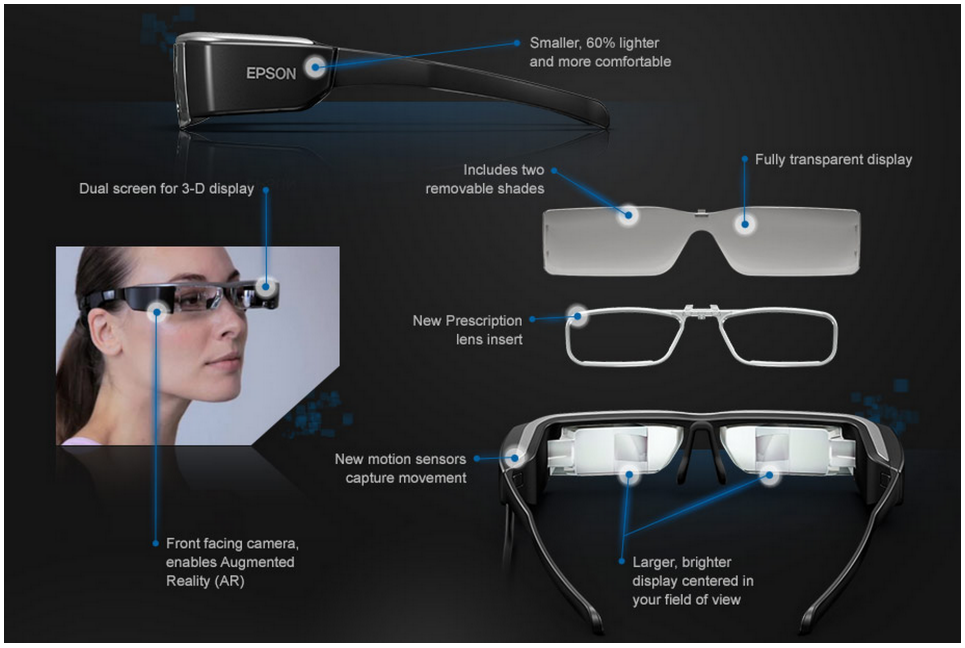

Figura 4.2: Gafas Epson Moverio BT200

Epson se necesita un mando táctil. Esta separación física entre lo que se ve y donde se selecciona requiere un cierto tiempo para acostumbrarse a su manejo.

En el contexto de la telemedicina tiene muchos usos como el que ha iniciado este proyecto de permitir al cirujano recibir información directamente mientras opera al paciente sin tener que realizar movimientos adicionales innecesarios.

Sin embargo, fuera de la telemedicina dispone de infinidad de posibles usos como realizar rápidamente un inventario en un almacén, recibir asistencia remota para reparar una máquina, o bien seguir una receta de cocina.

A nivel de experiencia del usuario las gafas son bastante cómodas, se pueden llevar durante bastante tiempo sin penalizar a la persona que las lleva.El dispositivo con el que se controla el puntero dentro de la interfaz Android que llevan las gafas responde bastante bien aunque requiere un periodo de aprendizaje para dominar su manejo. La imagen que se superpone a la realidad esta bastante bien conseguida y es posible modificar parámetros de brillo para poder verla mas nitidamente o muy sutilmente si se desea. En exteriores la grabación es bastante limitada y en interiores o zonas con poca luz la grabación se complica bastante. La duración de la batería no afectó a las pruebas, si bien es cierto que para operaciones largas debería tenerse en cuenta la duración real para evitar problemas.

Figura 4.3: Controller Epson Moverio BT200

4.1.4. Conclusiones

Entre las opciones disponibles realistas, se ha seleccionado este modelo por tener las mejores prestaciones en comparación con el resto tanto por duración de batería como por la cámara a pesar de ser uno de los puntos débiles del modelo.

En base al soporte por parte de la empresa sobre esta tecnología y a su sistema operativo se puede iniciar un desarrollo de funcionalidades sobre este modelo por existir facilidades con la portabilidad a un *hardware* más moderno y de mejores prestaciones.

Figura 4.4: Exterior con abundante luz natural directa.

Figura 4.5: Exterior con abundante luz natural no directa.

Figura 4.6: Interior con poca luz.

Figura 4.7: Interior con luz artificial.

Parte III

Implementación

Una vez razonados los conceptos básicos se puede pasar a explicar el funcionamiento de la arquitectura a un nivel más profundo. En este proyecto intervienen varios actores:

- Servidor: instalado en una ubicación remota o en el propio destino (el hospital o el aula) debe ser capaz de recibir el video, procesarlo si fuese necesario y reenviarlo a las aplicaciones que cuelgan de él.
- Cliente: se corresponde con una aplicación Android instalada en un *smartphone* o gafas inteligentes con sistema operativo Android. Su cometido es capturar el vídeo, codificarlo y encapsularlo para ser enviado a un servidor que lo procese. Dicho vídeo será codificado usando una librería para convertirlo a un formato aceptado por el servidor. Este vídeo viajará a través de un protocolo HTTP o RTSP según convenga.
- Aplicación: hace referencia a la aplicación o aplicaciones en el hospital o aula donde los usuarios recibirán el vídeo y podrán manipularlo para incrementar sus conocimientos. Este proceso se desarrollará mediante las herramientas Unity y ARToolKit.

Esta arquitectura está diseñada para que el cliente capture la escena con los marcadores y envíe la información a través del servidor a la aplicación. Ésta se encarga de mostrar el vídeo recibido y permite la realización de anotaciones sobre él aplicando la AR para que los trazos se mantengan respecto a las formas mostradas en el vídeo en lugar de la pantalla.

Capítulo 5

Servidor intermedio

Un servidor intermedio es una aplicación que permite la comunicación de uno o varios emisores con uno o múltiples receptores. Este actor es necesario para poder completar la transmisión de vídeo y/o audio y realizar la conversión entre formatos entrantes/salientes si fuese necesaria. El servidor intermedio gestiona la conexión de los clientes fuente y los receptores evitando la pérdida de datos, además de otorgar un factor de escalabilidad para en un futuro aumentar el número de clientes de sendos tipos haciendo uso de una arquitectura única.

A la hora de realizar la comunicación entre las gafas inteligentes/dispositivo emisor y la aplicación en el hospital/aula se barajaron varias posibilidades.

1. Comunicación de vídeo y audio directa entre el dispositivo emisor y la aplicación en el hospital/aula.

2. Comunicación de vídeo y audio mediante un servidor intermedio local u online que procese y lo reenvíe a la aplicación cliente.

La primera opción fue descartada por la imposibilidad de establecer la comunicación entre las dos aplicaciones, al no contar con un método para la detección del dispositivo opuesto en cada caso.

La segunda fue la elegida. En una primera fase se hizo uso de un servidor online para realizar pruebas, que posteriormente se descartó por no tener acceso al código fuente de éste por ser privativo, ni conocer los protocolos de transmisión. Debido a esto se recurrió a un servidor local de código abierto.

5.1. Funcionamiento

El esquema de funcionamiento del servidor intermedio para la comunicación es sencillo. A grandes rasgos se divide en entradas y salidas como se puede ver en la figura 2.1 . Más detalladamente contamos con:

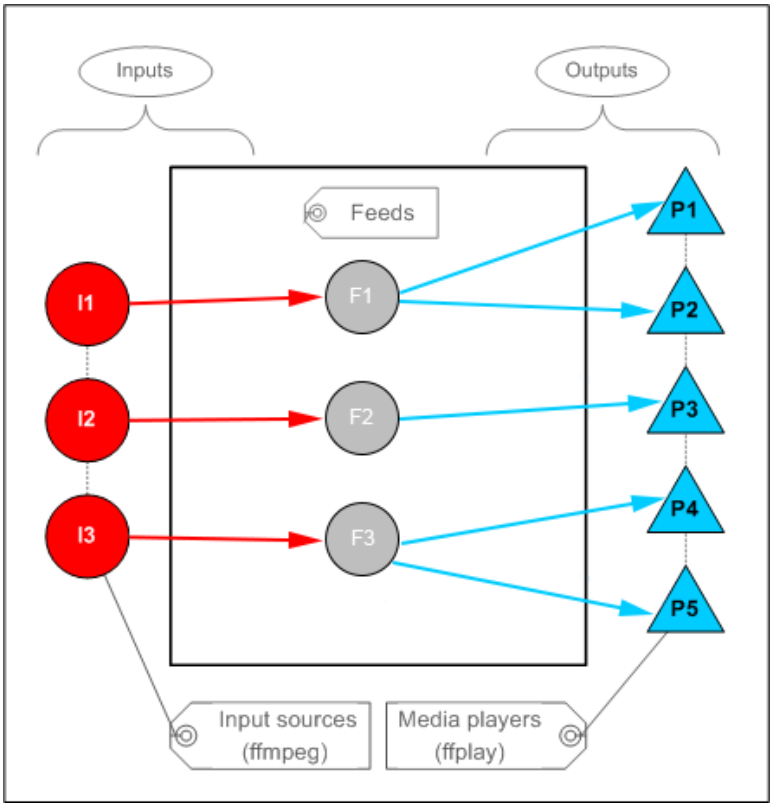

Figura 5.1: Configuración del servidor Icecast para el establecimiento de comunicaciones

- Clientes (I) en este caso el dispositivo Android u ordenador que envían vídeo y audio hasta el servidor, podrían ser una o varias fuentes de entrada según las necesidades.

- Un feed (F) es donde se almacena el flujo de datos recibido desde el cliente. Se define en el servidor y sólo puede recibir datos de un cliente a la vez.

- Cliente (P) en este proyecto sería la aplicación en el hospital/aula que se conectaría a uno de los feed disponibles para recibir el streaming de vídeo y audio, también podría serlo un reproductor multimedia tipo VLC (VideoLAN, 2017).

Todos estos parámetros se definen en un fichero de configuración del servidor junto con la codificación del audio y vídeo entre otros aspectos.

5.2. Servidor Wowza

En un primer momento se hizo uso del servidor Wowza (Wowza, 2015) de la empresa Wowza Media Systems, la finalidad era aprender como funcionaba este software y fue de bastante utilidad didáctica al contar con varios ejemplos de uso.

La conexión está asegurada mediante el protocolo RTSP desde el emisor a este servidor y haciendo uso del protocolo RTMP se garantiza el envío del vídeo a la aplicación receptora.

Se habría continuado con su utilización si realmente hubiese cumplido con las funcionalidades que prometía y sobre todo si fuese de código abierto. Preferíamos para el desarrollo del proyecto poder acceder al código fuente, disponer de información suficiente sobre como integrarlo con los otros actores y más datos sobre los contenedores y codecs utilizados.

En consecuencia se decidió buscar otras alternativas software para realizar la funcionalidad de servidor intermedio.

5.2.1. Pruebas servidor Wowza

Además de la aplicación implantada para este servidor (sección 6.1.2) se ha utilizado la aplicación propia de Wowza disponible en (Wowza, 2017). Ambas aplicaciones emisoras han tenido resultados similares. Se consigue un *live streaming*, pero las opciones de *codecs* y formatos tanto en las aplicaciones emisoras para capturar el vídeo, como en el servidor para convertir los datos recibidos, no se corresponden con las restricciones de la aplicación receptora, sección 7.1 .

5.3. Servidor FFServer

Tras los problemas con el software privativo Wowza, se estudió otra opción, en este caso de de código abierto, fue FFserver (FFmpeg, 2016b).

Debido a la incapacidad del servidor de realizar una conversión de formatos de forma ágil y a la anunciada discontinuidad del producto se exploraron nuevas opciones siguiendo la linea del software de código abierto.

5.3.1. Pruebas servidor FFServer

Se ha conseguido enviar mediante FFmpeg un *live streaming* de la *webcam* exclusivamente en formato .jpeg.

En las pruebas relacionadas con javaCV 6.1.4 se intentó la conexión con el emisor modificando varios parámetros de la configuración en los *feed* de FFserver como son los *codecs*, contenedores e incluso el tipo de *feed* (ffm y ffm2). A pesar de ello, no se ha obtenido un resultado satisfactorio.

Desde un reproductor VLC se ha simulado un *live streaming* enviando ficheros de vídeo. Este servidor ha sido capaz de aceptarlo pero no de convertirlo para enviarlo al receptor en el formato de la aplicación receptora, sección 7.1 (theora).

5.4. Elección de servidor Icecast

Una vez descartadas las otras opciones por los motivos ya comentados, se buscó un servidor alternativo antes de demorar más el progreso del proyecto con desarrollos auxiliares. El elegido en base a la documentación existente fue Icecast.

La elección final del servidor Icecast se debió a la posibilidad de tener acceso a todo el código fuente, su licencia de código abierto y las posibilidades de aprendizaje que proporcionaba la implementación.

Su funcionamiento reside en un único fichero de configuración. En éste se tienen que describir los parámetros del feed para recibir datos y de los streams asociados para el cliente. Entre ellos cabe destacar el tipo de contenedor de vídeo, siendo el idóneo en este caso .ogg. No se describirán los parámetros relacionados con el audio ya que no son un requisito del TFG (como se ha mencionado en varias ocasiones el quirófano cuenta con su propio sistema de audio para las comunicaciones).

5.4.1. Prerrequisitos

Para la correcta utilización del servidor Icecast es necesario instalar los siguientes módulos adicionales de forma nativa, para poder procesar los distintos tipos de stream.

- libxml2: es una biblioteca de código para parsear documentos XML.
- libxslt: es una biblioteca de código para transformar documentos XML.
- curl: es una biblioteca para la tranferencia de datos entre un cliente, un servidor y viceversa, soporta protocolos como RTMP, RTSP o HTTP.
- ogg/vorbis: bibliotecas para el tratamiento de los streams de audio y video en formato ogg.
- openssl (opcional): biblioteca para poder implementar el protocolo criptográfico SSL en la comunicación con el servidor.

5.4.2. Fichero de configuración

Como se ha comentado en la introducción, todos los parámetros de configuración del servidor se definen en icecast.xml

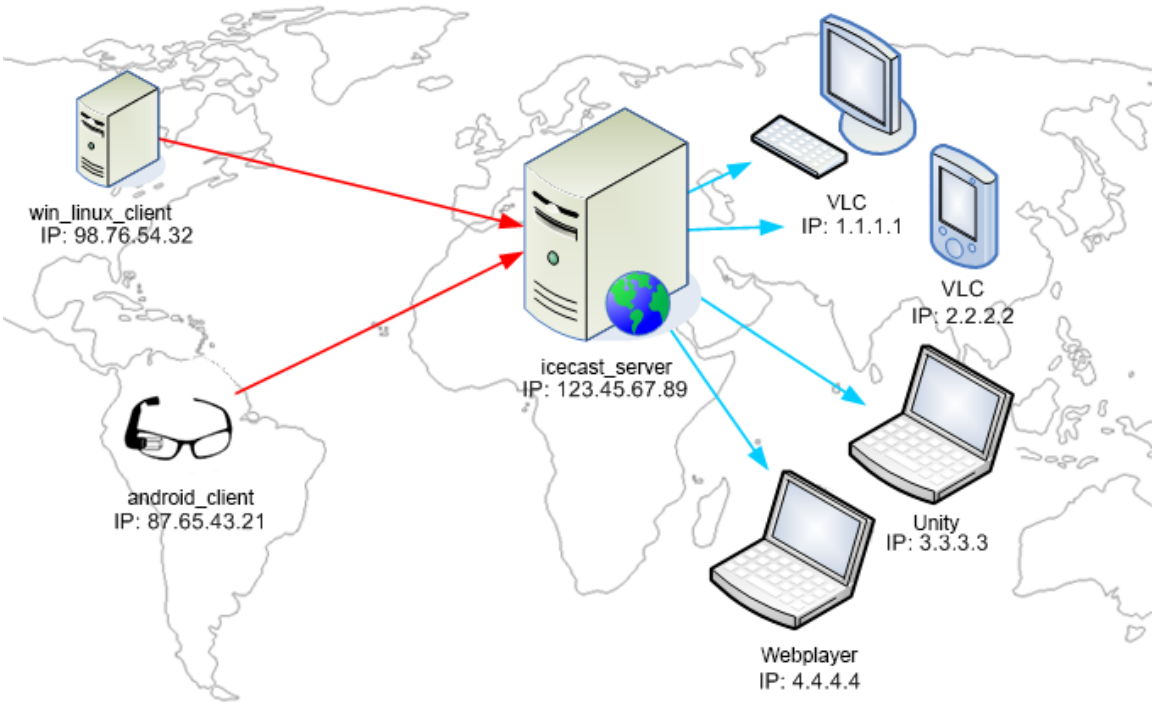

Figura 5.2: Esquema comunicación mediante icecast

```
<limits>
   <clients>100</clients>
   <sources>2</sources>
   <queue-size>524288</queue-size>
   <client-timeout>30</client-timeout>
   <header-timeout>15</header-timeout>
   <source-timeout>10</source-timeout>
   <burst-on-connect>1</burst-on-connect>
   <burst-size>65535</burst-size>
</limits>
```

- clients: número de clientes soportados por el servidor.
- sources: máximo número de fuentes simultaneas soportadas por el servidor.
- queue-size: tamaño máximo de la cola del stream, su tamaño más pequeño es 512Kb
- client-timeout: sin uso actualmente.

- header-timeout: tiempo máximo, en segundos, que espera el cliente para recibir una respuesta una vez establecida la conexión con el servidor.
- source-timeout: tiempo que tiene que pasar una fuente sin enviar datos al servidor, en segundos, para que este considere que se debe borrar esa fuente.
- burst-on-connect: es un alias del burst-size.
- burst-size: tamaño de la ráfaga de datos que se envía en cada conexión.

```
<authentication>
   <!-- Sources log in with username 'source' -->
   <source-password>1234</source-password>
   <!-- Relays log in with username 'relay' -->
   <relay-password>1234</relay-password>

   <!-- Admin logs in with the username given below -->
   <admin-user>admin</admin-user>
   <admin-password>admin</admin-password>
</authentication>
```

- source-password: contraseña para las fuentes que se conectarán al servidor para realiza el streaming.
- relay-password: contraseña para cuando los servidores esclavos se conecten al maestro solicitando la lista de streams para retransmitir.
- admin-user: usuario de acceso al panel de administración del servidor icecast.
- admin-password: contraseña de acceso al panel de administración del servidor icecast.

```
<hostname>192.168.1.21</hostname>
```

- hostname: dirección IP del servidor Icecast local.

```
<listen-socket>
   <port>8000</port>
</listen-socket>
```

- port: puerto del servidor al que entrarán y saldrán las comunicaciones.

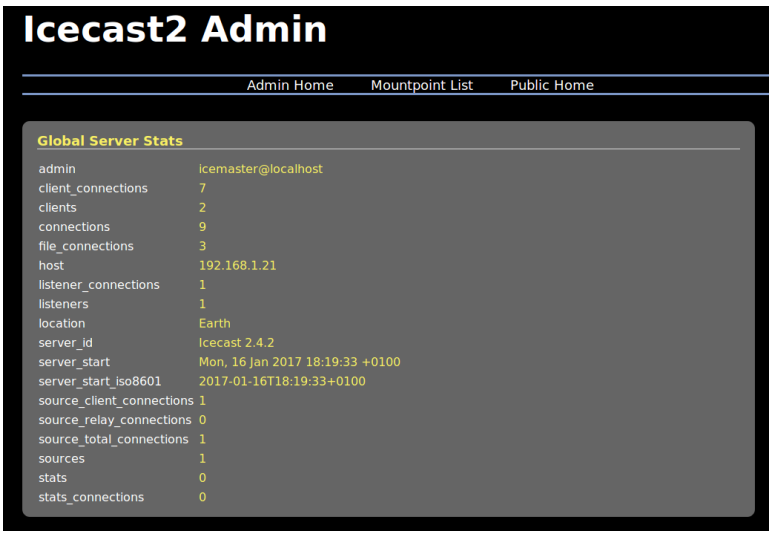

Figura 5.3: Interfaz web de Icecast, parámetros del servidor.

5.4.3. Arranque y parada del servidor

Una vez completado el fichero de configuración solo queda lanzar el servidor para su ejecución, esto se realiza mediante el siguiente comando:

```
sudo /etc/init.d/icecast2 start
```

Una vez en ejecución se puede acceder a la interfaz web, como se muestra en el punto 5.4.4. De igual forma, para detener su ejecución se utilizará:

```
sudo /etc/init.d/icecast2 stop
```

5.4.4. Interfaz de administración

Para una mejor gestión de los *streams*, fuentes y de los clientes disponibles, Icecast incluye una panel accesible desde un navegador web que muestra de forma visual toda esta información.

En la sección 'Global Server Stats', ver la figura 5.3, se pueden encontrar los parámetros relacionados con el servidor: Número de fuentes de *streaming* en uso en ese momento, número de clientes recibiendo el *streaming* enviado por la fuente...

En la sección 'Mountpoint name stream', ver la figura 5.4, se pueden observar los parámetros relacionados con el *streaming* generado, se muestran estadísticas sobre el vídeo como su resolución, bitrate o total de bytes recibidos y enviados.

Figura 5.4: Interfaz web de Icecast, parámetros del stream.

Capítulo 6

Cliente emisor

El desarrollo ha estado centrado en conseguir una aplicación Android para ser utilizada en las gafas inteligentes elegidas después del estudio previamente mencionado en la sección 3.3.

Esta decisión ha supuesto el establecimiento de una serie de restricciones sobre el proyecto, principalmente con la versión del sistema operativo, que acota en cuanto a los *codecs* y formatos disponibles a utilizar como se indica en la documentación oficial (Google, 2017b).

Sin duda ésta ha sido una de las partes más difíciles de implementar, por tener conocimientos sobre Android bastante reducidos y el momento tampoco ha sido el más adecuado, ya que Google decidió sacar su propia plataforma de desarrollo a finales de 2014 (Android Studio) en detrimento de Eclipse. Lo que supuso un esfuerzo adicional ya que la información al respecto de este nuevo entorno de desarrollo era bastante reducida.

Como consecuencia de lo anterior ha sido necesario investigar y aprender las funcionalidades y restricciones ofrecidas por este sistema operativo para obtener y realizar el envío de la imagen, así como familiarizarse con Android Studio.

6.1. Estudio

En el siguiente apartado se explican de forma pormenorizada todas las pruebas realizadas para intentar desarrollar la aplicación cliente para un dispositivo Android, en última instancia se decidió hacer uso de FFmpeg de forma nativa en Linux ante la imposibilidad de hacer funcionar una aplicación para Android.

6.1.1. Acceso directo

Con el objetivo de establecer una conexión directa entre el emisor y el receptor final, se ha explorado un enfoque de acceso a la cámara del dispo-

sitivo mediante las funciones y clases propias de Android (exclusivamente desde la versión del lenguaje Java adaptada a este sistema operativo).

6.1.1.1. Acceso a la cámara

Consistente en una aplicación sencilla que hace uso de la cámara en una actividad [1] única, realizando la grabación de vídeo y su posterior almacenamiento en local.

Para ello, la aplicación crea un *intent* [2] con la acción de capturar vídeo para luego reproducirlo una vez finalizada la actividad de grabación, como se muestra en las dos siguientes funciones respectivamente:

```
private void dispatchTakeVideoIntent(){

   Intent video = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);

   if(video.resolveActivity(getPackageManager())!= null){

      startActivityForResult(video, VIDEO_CAPTURE);

   }

}

protected void onActivityResult(int requestCode, int resultCode,
    Intent data){

   if(requestCode == VIDEO_CAPTURE && resultCode == RESULT_OK){

      Uri videoUri = data.getData();

      TextView tvaddr = (TextView) findViewById(R.id.tvAddr);

      tvaddr.setText(data.getDataString());

      video.setVideoURI(videoUri);

      video.start();
    }
}
```

[1] En Android corresponde al conjunto de acciones presentadas al usuario en una misma pantalla.

[2] En Android se utiliza para delegar una o varias funcionalidades a otra aplicación. Sirve para avisar en ejecución al sistema operativo la acción que se desea realizar, para que éste busque el conjunto de aplicaciones pudiendo completar la acción, dando a elegir al usuario en caso de existir mas de una alternativa.

6.1.1.2. Conexión directa

En el contexto de las llamadas IP, se han investigado dos enfoques:

1. La aplicación ha sido construida a partir de la documentación oficial de SIP, de Google para el audio aportando las modificaciones consideradas necesarias para la transmisión de imagen. Este envío de datos se produce mediante el protocolo RTP, cuya implementación en Android ha pasado por el uso de métodos en código nativo mediante el NDK. Esto ha presentando problemas para la gestión de ambos códigos, principalmente a nivel de la carga de las librerías en C.

   Debido a una documentación centrada sobre el audio, con una única mención al vídeo en el método "open()" (Google, 2017a) para iniciar una llamada con imagen, y siendo una arquitectura que necesita un servidor IP, no se ha conseguido completar una aplicación operativa.

2. Otra aproximación ha sido a partir de la implantación del proyecto uTox (uTox, 2017). Una aplicación de mensajería instantánea mediante conexión punto a punto con un énfasis en la seguridad permitiendo entre otras funcionalidades las vídeo llamadas.

   El estudio de viabilidad ha resultado negativo para servir en el proyecto. A pesar de tener una versión de escritorio interesante, en Android se encuentra aún en fase beta, cuyo principal inconveniente es el no haber implementado la funcionalidad de vídeo (Nachfuellbar, 2015).

6.1.2. Inclusión de un servidor intermedio

Partiendo de la prueba con un servidor intermedio Wowza (sección 5.2) se ha creado una aplicación emisora que hace uso de la librería libstreaming (Fyhertz, 2015), lo que introduce un nivel de abstracción para la codificación del vídeo y su envío.

Para el *streaming*, este emisor hace uso del protocolo RTSP adaptado a este fin. La aplicación crea una sesión con los parámetros del vídeo y los de autenticación que gestiona mientras se mantenga la conexión.

Para conseguir establecer la conexión ha sido necesario realizar un trabajo previo sobre la red donde se ejecuta la aplicación.

6.1.2.1. Modificaciones de red

A continuación se describen las modificaciones efectuadas sobre la red para garantizar la conexión entre la aplicación emisor y el servidor Wowza (Wowza, 2017). Algunas han requerido aplicaciones de terceros como un emulador de consola de comandos en Android (Palevich, 2015) o una aplicación dedicada a telnet (Corp., 2011).

Entre la batería de pruebas para determinar y solucionar el problema de conexión destacan:

- Inclusión manual de la dirección del dispositivo Android en la tabla de rutas del ordenador que hace de servidor, tanto en Windows 10 como en Linux nativo.
- Siguiendo (Wowza, 2015) se verifica que el servidor está activo y el puerto especificado abierto tanto en el *router* como en el ordenador.
- Intento de conexión mediante telnet y ping desde Android directos a la maquina de destino con resultado "host inalcanzable".
- Ping al *default gateaway* con éxito desde ambos puntos de la comunicación. Pudiendo indicar la ausencia del "forwarding" en el *router*, que no es posible modificar al no disponer de autorización legal.
- Prueba de vídeo llamada entre el ordenador y un dispositivo Android conectados exclusivamente mediante red Wi-Fi a través de la aplicación Hangouts de Google completada con éxito. Posterior uso del comando ping con resultado negativo.
- Modificación de otros parámetros de red y apertura de puertos en Windows 10 desde el panel de configuración sin mejoras.
- Creación de la aplicación en Eclipse con los mismos resultados.
- Uso de IPv6 en lugar de IPv4.
- Cambio a una red local con direcciones IP privadas [3].

Sólo el último punto ha resultado en la conexión entre el servidor Wowza y la aplicación, una vez comprobada la comunicación entre los actores mediante "ping".

6.1.2.2. Resultados

La investigación ha permitido una aplicación de la gran mayoría de los conocimientos adquiridos en redes. Estos han sido suficientes para resolver el problema de conexión al haber podido determinar su origen.

Unas pruebas adicionales sobre la red de la UCM, que está constituida de direcciones públicas en el momento de las ejecuciones, han resultado en el mismo problema, y han puesto en evidencia el requisito del tipo de red para un correcto envío de datos.

[3] Direcciones para el uso en redes internas, del estilo 192.168.X.X para IPv4, donde el acceso al exterior (*Internet*) está garantizado exclusivamente mediante un *router* haciendo traducción de direcciones.

Con la conexión establecida, esta aplicación permite el *live streaming* a un servidor Wowza. Por sólo tener implementado el protocolo RTSP y no HTTP no es posible establecer la conexión con el servidor Icecast. Se ha descartado esta opción para el emisor por no conseguir cumplir con los requisitos de contenedor y *codec* del vídeo impuestos por el receptor (sección 7.1) (vídeo en theora) al no tenerlos como opción para la captura el emisor.

6.1.3. Emisor FFmpeg sobre Android

La aplicación reposa sobre la librería FFmpeg, sección 6.2.1 y su servidor asociado FFserver, sección 5.3.

Consiste en una adaptación de FFmpeg a Android haciendo uso de código nativo en C. Desarrollada con el NDK una vez compilado el módulo de FFmpeg para Android siguiendo la guía de Roman10 (2013).

La aplicación implantada está basada en Hua (2016). En Java sólo tiene una mínima preparación de los datos necesarios como la dirección del servidor. La funcionalidad principal reside en el código nativo en C. Esta aplicación está diseñada para funcionar con un servidor intermedio FFserver mediante el protocolo RTMP.

Esta alternativa no ha resultado, principalmente por la gran cantidad de errores con el uso de código nativo, tanto en Android Studio como en Eclipse y un soporte insuficiente por parte de la comunidad para resolverlos.

6.1.4. Librería javaCV

Las siguientes tres aplicaciones han sido probadas partiendo de la librería javaCV (Audet, 2017) para obtener el emisor de la arquitectura, por contener entre otros FFmpeg.

1. La primera ha sido una implantación de una aplicación dedicada al *live streaming* a Youtube con los parámetros del servidor FFserver.

   No ha resultado por requerir la activación en una cuenta Gmail de unos parámetros para habilitar el *live streaming* al estar ligado con el servicio de Youtube. El proceso de activación de estos parámetros no se ha conseguido completar.

2. La segunda aplicación es el resultado de simplificar la segunda eliminando todas las referencias a Youtube en el código.

   En las ejecuciones, este emisor ha sido incapaz de encontrar la definición de la clase "FFmpegFrameRecorder", siendo la principal en la librería javaCV para la captura y envío de vídeo.

3. La tercera aplicación es una implantación del ejemplo (Bytedeco, 2016). Crea una instancia de "FFmpegFrameRecorder" sobre la que carga todos los parámetros de configuración necesarios entre los que destacan

la dirección para enviar y las opciones de *codec* y contenedor de vídeo. Esta aplicación es capaz de realizar un *live streaming* al servidor FFserver usando un contenedor .mkv con una baja calidad de imagen, una vídeo inestable y un retardo importante.

Las dos primeras aplicaciones se han descartado por los motivos mencionados anteriormente. No se ha tenido en cuenta la tercera aplicación por no cumplir con los requisitos de contenedor y *codec* del vídeo impuestos por el receptor (sección 7.1) en la conexión mediante FFserver. A pesar de cambiar el servidor a Icecast, la conexión se establece pero no se realiza ningún envío de datos por errores con el *codec* a pesar de haber probado todos los que figuran tanto en la documentación de Android para la versión de las gafas elegidas (sección 4.1) como en la documentación de Icecast.

6.2. Emisor seleccionado

Debido a causas externas al proyecto se ha presentado una reducción de restricciones en el emisor. Al dejar de ser obligatorio el uso de Android en su versión 4.0.4, se tuvo acceso a nuevos *codecs* y contenedores con los que probar las aplicaciones anteriores como WebM (sección 2.4) y VP8 (sección 2.3).

Siguiendo la guía Xiph (2015), se ha conseguido enviar desde la *webcam* del ordenador a Icecast mediante el comando completo de FFmpeg.

```
ffmpeg \
-f v4l2 -video_size 640x480 -framerate 30 -i /dev/video0 \
-f alsa -i default \
-f webm -cluster_size_limit 2M -cluster_time_limit 5100
    -content_type video/webm \
-c:a libvorbis -b:a 96K \
-c:v libvpx -b:v 1.5M -crf 30 -g 150 -deadline good -threads 4 \
icecast://user:password@server:port/stream_name
```

Se han modificado los parámetros necesarios para probar con distintos *codecs* y formatos obteniendo resultados satisfactorios en un receptor VLC, y en particular, en Unity con OGG.

Se ha optado por usar como emisor un ordenador con una *webcam* mediante el programa FFmpeg por permitir la transmisión al servidor Icecast con un *codec* y formato cuyo receptor es capaz de procesar y no estar limitados a un emisor bajo el sistema operativo Android.

6.2.1. Funcionamiento FFmpeg

FFmpeg consiste en un framework multimedia para el tratamiento de audio y vídeo permitiendo principalmente entre otros la conversión de formatos, *codecs*, separación de pistas, grabación, mezclas de ficheros y capturas

tanto para grabar en un fichero como para enviar a un servidor (FFmpeg, 2017).

A pesar de no existir restricciones para FFmpeg en cuanto a sistema operativo, el emisor ha sido extensamente probado exclusivamente en Linux Ubuntu 16.04 por tener previamente la configuración a raíz de las pruebas con el servidor FFserver.

Para hacer uso de FFmpeg es necesaria su instalación si no está en el equipo. Para ello, se puede descargar una versión estática de la web oficial o bien recurrir al comando

```
sudo apt-get install ffmpeg
```

En base a la documentación de Icecast y FFmpeg se ha conseguido un correcto funcionamiento en todos los elementos de la arquitectura del proyecto, que se resume en el siguiente comando:

```
ffmpeg \
-f v4l2 -video_size 640x480 -framerate 30 -i /dev/video0 \
-f alsa -i default \
-f ogg -cluster_size_limit 2M \
-cluster_time_limit 5100 -content_type video/ogg \
-c:a libvorbis -b:a 96K \
-c:v libtheora -b:v 1.5M -crf 30 -g 150 -deadline good -threads 4 \
    icecast://emisor:contraseña@ip:puerto/nombre
```

Para el que se ha creado una versión con sólo la imagen, siendo optimizado para este proyecto cuyo foco es el aspecto visual, además de tener en el cliente Unity un funcionamiento correcto excluyendo el audio:

```
ffmpeg \
-f v4l2 -video_size 640x480 -framerate 30 -i /dev/video0 \
-f ogg -cluster_size_limit 2M \
-cluster_time_limit 5100 -content_type video/ogg\
-c:v libtheora -b:v 1.5M -crf 30 -g 150 -deadline good -threads 4 \
    icecast://emisor:contraseña@ip:puerto/nombre
```

Cada uno de los parámetros se explican a continuación.

6.2.1.1. Parámetros del vídeo

En relación a la captura de imagen se tiene:

- *v4l20*: API de captura de vídeo de Linux.
- *video size*: tamaño de la imagen capturada.
- *framerate*: tasa de imágenes a enviar.

- *i*: dispositivo con el que se realiza la captura.
- *ogg*: formato de la imagen a enviar, en concreto su contenedor.
- *cluster size limit*: tamaño máximo del *buffer* de almacenamiento.
- *cluster time limit*: tiempo máximo que se da por válido el contenido del *buffer* antes de crear uno nuevo (Niedermayer, 2013).
- *content type*: tipo de datos a enviar, un vídeo ogg.
- *c: v*: librería a usar para codificar la imagen.
- *b :v*: tasa de *bits* de vídeo (Wikipedia, Streaming Guide).

6.2.1.2. Parámetros del audio

En cuanto al audio se especifica como sigue:

- *alsa*: componente para la captura de audio de Linux.
- *i*: dispositivo concreto a utilizar.
- *c: a*: librería para codificar el audio, vorbis.
- *b: a*: tasa de *bits* de audio (Wikipedia, Streaming Guide).

6.2.1.3. Parámetros generales

Para el desarrollo de la ejecución:

- *crf*: factor de la tasa de contenido del inglés *Content Rate Factor*, usado para mantener una calidad de imagen consistente variando la tasa de *bits* según los recursos disponibles de cpu, red... (Wikipedia, Streaming Guide).
- *g*: periodo máximo entre dos imágenes clave (Wikipedia, Streaming WebM to Icecast with FFmpeg).
- *deadline*: calidad de la transmisión. Se ha optado por un balance entre velocidad y calidad.
- *threads*: número de hilos a dividir las tareas necesarias para capturar, codificar y transmitir, siendo aprovechado por la ejecución en un ordenador.

6.2.1.4. Dirección del servidor

Finalmente la dirección a la que se envía:

- *icecast*: se corresponde al protocolo especial de Icecast.
- *emisor*: nombre definido en el fichero de configuración del servidor para que un emisor pueda autenticarse.
- *contraseña*: la contraseña, si se ha definido en el servidor, para poder enviar los datos.
- *ip*: dirección IP del servidor.
- *puerto*: puerto del servidor.
- *nombre*: nombre del *stream*, que va a ser utilizado por el receptor para conectarse al servidor. No es necesario que contenga extensión de fichero alguna (como puede ser .webm o .ogg).

6.3. Conclusiones

Como se ha demostrado en todas las pruebas anteriores, ha sido necesario descartar la aplicación en Android, por problemas de protocolos y *codecs*, y hacer uso de FFmpeg nativo en el emisor con un servidor intermedio Icecast.

Capítulo 7

Cliente receptor

Unity es un motor de videojuegos multiplataforma creado por Unity Technologies, disponible como plataforma de desarrollo para Windows, GNU-Linux y macOS. La plataforma de desarrollo tiene soporte de compilación para numerosos sistemas operativos y dispositivos, como Windows, Android, iOS, Smart TVs, consolas de última generación y dispositivos de realidad virtual. Dicha plataforma de desarrollo fue escogida para este TFG debido principalmente a que es un entorno de desarrollo multiplaforma que permitía múltiples posibilidades para crear la aplicación receptora, además de una integración muy buena con Vuforia para la parte de AR.

Una vez solventadas las fases de desarrollo de la aplicación para capturar el vídeo y el envío al servidor intermedio encargado de establecer la comunicación y procesar el streaming para su reenvío al hospital/aula, era necesario desarrollar una herramienta que recibiese ese flujo de vídeo y permitiese su manipulación.

Durante las pruebas iniciales sobre Unity se investigó el uso de la aplicación y sus funciones, así como varios *frameworks* para el desarrollo de la posible asistencia con realidad aumentada, en dicho proceso se encontraron múltiples inconvenientes que hicieron probar distintos *frameworks* y versiones de Unity.

7.1. Unity con *live streaming*

Uno de los primeros objetivos que se tenían que llevar a cabo en nuestro proyecto era conseguir una aplicación que recepcionase correctamente el *live streaming*.

Para ello, una de las primera pruebas que se llevaron a cabo fue hacer uso de la clase WWW (Technologies, 2017), que permite realizar una petición a una dirección local o remota que contenga un vídeo dentro de un contenedor .ogg. Véase sección 2.4.

Para llegar a esta conclusión probamos diferentes clases de Unity y múltiples contenedores como .mp4, .webm, .3gp o .mp3.

A continuación se muestra un extracto del código implementado, para la carga del vídeo:

```
1
2  private string url = "http://direccion:puerto/stream";
3
4  IEnumerator Start(){
5
6  // Start download
7     WWW retriever = new WWW(url);
8     MovieTexture mvtxt = retriever.movie;
9
10    while(!mvtxt.isReadyToPlay)
11       yield return null;
12
13    GetComponent<Renderer>().material.mainTexture = mvtxt;
14
15    mvtxt.Play();
16
17 }
```

Se crea una instancia de clase WWW para gestionar la url y poder extraer el vídeo a una textura. Antes de iniciar la reproducción, se espera a que se haya recibido la suficiente cantidad de información.

Ha sido necesario modificarlo para conseguir recibir un *live streaming* desde el servidor como se aprecia en:

```
1
2  private WWW wwwData;
3  private string url = "http://dirección:puerto/stream";
4  private MovieTexture movieObj;
5
6  IEnumerator Start() {
7     wwwData = new WWW(url);
8
9     yield return new WaitForSeconds(2.0f);
10
11    movieObj = wwwData.movie;
12
13    GetComponent<Renderer>().material.mainTexture = movieObj;
14
15    movieObj.Play();
16
17 }
```

Tal y como se puede apreciar en la linea 9 de nuestro código, se ha

reducido la espera del vídeo a 2 segundos. Se intentó eliminarlo para que fuese inmediato, pero el resultado no era satisfactorio. El código creado trata tanto el vídeo como el audio, pero se ha optado por dejar sin uso el audio porque estaba fuera de las requisitos iniciales. Se deja comentado el código para poder ser usado si fuera necesario.

Esta opción es necesaria por permitir recibir el vídeo desde otra localización pero si se quiere hacer uso de la AR es necesario aplicar un plugin adicional al que habrá que indicar como fuente de vídeo el live streaming recibido.

Una vez solventado el hecho de la recepción del *live streaming* se aborda el siguiente reto, implementar toda la parte de pintado necesaria para hacer indicaciones a la hora de operar, para que los alumnos de medicina puedan seguir de una manera mas acertada las indicaciones del profesor y médico.

Se pasa a detallar partes del código que ilustran sobre el trabajo realizado en esta sección:

Inicialmente se crea un plano lo mas cercano al cubo que retransmite el vídeo. Se crea y configura un TrailRenderer para usar como elemento de dibujo. Se decide usar este y no LineRenderer debido a que tenía mas parámetros de configuración, pero sobre todo porque con LineRenderer daba algunos problemas. Por poner un ejemplo, se puede parametrizar para que el grosor de la traza en su inicio y final sean distintos, factor que puede ser útil para la señalización de objetos.

```
1  void Start (){
2
3     // se calcula la posición del plano donde pintar para que
          coincida con el cubo
4     this.transform.position =- new Vector3(0,0,-1);
5     objPlane = new Plane (Camera.main.transform.forward,
          this.transform.position);
6     ....
7     TrailRenderer tr;
8     tr =(TrailRenderer)
          trailPrefab.GetComponent(typeof(TrailRenderer) );
9     tr.startWidth = 0.1f;
10    tr.endWidth = 0.1f;
11    ...
12 }
```

Como se puede apreciar en el código de abajo, se instancia, castea y configura tanto el trailPrefab, como la distancia entre la cámara y el cubo y la posición donde queda lo que se va pintando en la escena. Esto se pone de manifiesto en las líneas 7 y 8. Se controla tanto el hecho de pulsar el *mouse* o puntero, tanto cuando se deja de pulsar y por ultimo en la linea 23 y 24 se parametriza para que si se pincha unicamente en la escena sin arrastrar

el puntero no dibuje nada.

```
1
2  void Update (){
3
4
5     float rayDistance;
6
7     if(Input.GetMouseButtonDown(0)){
8        thisTrail = (GameObject) Instantiate(trailPrefab,
             this.transform.position, Quaternion.identity);
9        cambioColor();
10       Ray mRay = Camera.main.ScreenPointToRay (Input.mousePosition);
11       if(objPlane.Raycast(mRay, out rayDistance))
12          startPos = mRay.GetPoint(rayDistance);
13
14    ...
15    }
16    else if (Input.GetMouseButton(0)){
17       Ray mRay = Camera.main.ScreenPointToRay (Input.mousePosition);
18       if(objPlane.Raycast(mRay, out rayDistance))
19          thisTrail.transform.position = mRay.GetPoint(rayDistance);
20    }
21    //destruye el clon del trail cada vez que pinchas en el objeto y
          no arrastras
22    else if(Input.GetMouseButtonUp(0))
23       if(Vector3.Distance(thisTrail.transform.position, startPos) <
             0.1)
24          Destroy(thisTrail);
25    }
26
27    ...
28
29 }
```

Una vez finalizada esta parte se ampliaron las opciones de dibujo. Una paleta de colores básicos para que la persona que esta señalizando lo que desee en la imagen pueda cambiar entre varios colores.

Así mismo se pensó que era una buena idea habilitar que dichas líneas se pudiesen eliminar y además se controle el grosor de las lineas a dibujar, fijando unos parámetros que se creyó adecuados tanto de máximo grosor como de mínimo.

Como muestra, debajo se puede comprobar la función para que disminuya el grosor del trazo.

```
1  void minOnClick(){
2     TrailRenderer tr;
3     tr =(TrailRenderer)
```

```
        trailPrefab.GetComponent(typeof(TrailRenderer) );
    if(tr.startWidth > 0.06f){
        tr.startWidth -= 0.05f;
    tr.endWidth -= 0.05f;
}
```

Por último decir que desde esta línea de trabajo se pueden ampliar las funciones de dibujado tanto como se desee.

Para mejorar la calidad de las indicaciones se pueden añadir funcionalidades de trazado, como inclusión de símbolos predefinidos o la posibilidad del borrado de elementos individuales, de acuerdo a las necesidades de los usuarios.

Una vez llevado a cabo el segundo objetivo, se pasó al último, como es el añadir AR al proyecto. Gracias a los marcadores instalados en el cliente emisor poder pintar unas lineas coherentes con las posiciones donde se encuentra el paciente. Para ello se probaron varios *frameworks* que trabajan sobre Unity.

7.2. Frameworks de realidad aumentada

Los *frameworks* son herramientas muy útiles para desarrolladores. Facilitan soporte para la creación de la aplicación receptora, gracias a las librerías que permitían agilizar y simplificar procesos que pueden llevar un alto grado de aprendizaje. En el caso de la AR, se encargan de captura y procesado de la imagen de vídeo, creación e integración de marcadores para tener referencias físicas de donde esta el paciente y donde se pinta.

Así mismo, se pueden utilizar como módulos complementarios a programas como Unity o Visual Studio o utilizados de forma nativa. Se pasa a hablar concretamente de dos de los *frameworks* más utilizados por la comunidad de desarrolladores, así como las pruebas realizadas.

7.2.1. Framework Vuforia (Qualcomm)

Vuforia de Qualcomm es un kit de desarrollo de software que permite la creación de aplicaciones de de AR. Se pueden generar marcadores propios de manera sencilla subiéndolos a la web, y su integración con Unity la convertía en la candidata perfecta para el proyecto.

En los siguientes puntos se explica en detalle las pruebas y cambios de versión de Unity que se debieron llevar a cabo para intentar hacer funcionar Vuforia con Unity orientado siempre hacia nuestro objetivo, que no era otro que el poder recibir una imagen de *live streaming* y pintar sobre ella con realidad aumentada.

- Unity 4.6.7f1 (32 bits): las primeras pruebas con realidad aumentada

satisfactorias se dieron con esta versión, las distribuciones anteriores no eran capaces de generar un proyecto compatible con las gafas inteligentes Epson. El problema vino a la hora de intentar implementar el *streaming* de vídeo, se tuvo que buscar una versión superior ya que esta funcionalidad no era compatible con Unity 4.

- Unity 5.0.10 (64-bit): con esta versión fue posible realizar *streaming* de fotogramas .jpg, utilizando la clase WWW. También se descartó ya que no permitía utilizar una textura para mostrar el vídeo, era necesario el uso de la modalidad pro.

- Unity 5.3.4p5 (64-bit): se decidió actualizar a esta versión por el impedimento de utilizar la textura de vídeo en la modalidad gratuita. Se consiguieron avances en cuanto a la sobreimpresión de vídeo ya fuese local o mediante *streaming* sobre una textura. A la hora de probar la realidad aumentada la versión de 64 bits no es capaz de mostrarla durante el desarrollo en Windows (Vuforia no soporta los 64 bits hasta la versión 6), lo que implicaba tener que generar la aplicación cada vez que se quería probar. Por ello también fue descartada.

- Unity 5.1.1f1 (32-bit): es la versión final sobre la que se intentó desarrollar el proyecto antes de cambiar de *framework* para la AR.

Tras realizar todas estas pruebas se descartó el *framework* Vuforia ya que no permitía cambiar la fuente de vídeo para la AR impidiendo poder realizar la asistencia sobre el vídeo recibido por *streaming*. Se decidió buscar alternativas a Vuforia, y después de una ardua investigación de múltiples *frameworks* de AR se decidió trabajar con ARToolKit integrado en Unity.

7.2.2. Framework ARToolKit

ARToolKit, ya mencionado en la sección 3.2.1 es un *framework* que permite la creación de aplicaciones de realidad aumentada. Calcula en tiempo real, tanto la posición de la cámara y la orientación relativa con los marcadores físicos. Sabiendo la posición de la cámara real, se puede sobreponer la cámara virtual y los modelos en 3D sobrepuestos sobre dicho marcador. ARToolKit resuelve dos de los principales problemas en la realidad aumentada, el seguimiento de punto de vista y la interacción objeto virtual.

Problemas parecidos a los de Vuforia surgieron de las pruebas con el *framework* de ARToolKit.

En este punto se tenía que decidir si se continuaba con el *framework* de ARToolKit como tal o se buscaban otras alternativas. Se decidió crear dos vías de trabajo, una pura en Unity y otra pura en ARToolKit, sin Unity, debido a que la tercera vía antes explorada, el usar ARToolKit integrado en Unity no estaba dando los resultados esperados.

7.3. ARToolKit nativo

ARToolKit es una biblioteca de funciones para el desarrollo rápido de aplicaciones de AR. Facilita el problema del registro de la cámara empleando métodos de visión por computador, de forma que obtiene el posicionamiento relativo de 6 grados de libertad haciendo el seguimiento de marcadores cuadrados en tiempo real, incluso en dispositivos de baja capacidad de cómputo. Algunas de las características más destacables son:

- **Tracking de una cámara:** en su versión básica soporta de forma nativa el tracking de una cámara. La biblioteca soporta gran variedad de modelos.
- **Marcas negras cuadradas:** emplea métodos de tracking de superficies planas de 6 grados de libertad. Estas marcas o patrones pueden ser personalizados siempre y cuando el patrón no sea simétrico en alguno de sus ejes.
- **Rápido y multiplataforma:** funciona en gran variedad de sistemas operativos (Linux, Mac, Windows...), y ha sido portado a todo tipo de smartphones (Android, iPhone, PDAs...).
- **Licencia libre:** permite utilizar, modificar y distribuir programas realizados con ARToolKit bajo la licencia GPL v2.

ARToolKit está formado por tres módulos:

- **Video:** ese modulo permite obtener frames de video los dispositivos gracias al fichero de cabecera video.h.
- **AR:** este módulo principal contiene las funciones principales de tracking de marcas, calibración y estructuras de datos requeridas. Los ficheros de cabecera ar.h, arMulti.h (subrutinas para gestión multi-patrón) y param.h describen las funciones asociadas a este módulo.
- **Gsub:** contiene las funciones relacionadas con la etapa de representación, que se encuentran en el ficheros de cabecera gsub.h. Utiliza GLUT.

Los módulos están desacoplados, de forma que es posible utilizar ARToolKit sin necesidad de emplear los métodos de otros módulos si no es necesario, como se puede apreciar en la imagen 7.1 .

Las funciones de ARToolKit aíslan de la complejidad de tratar con diferentes dispositivos de vídeo, por lo que la captura del frame actual es una simple llamada a función.

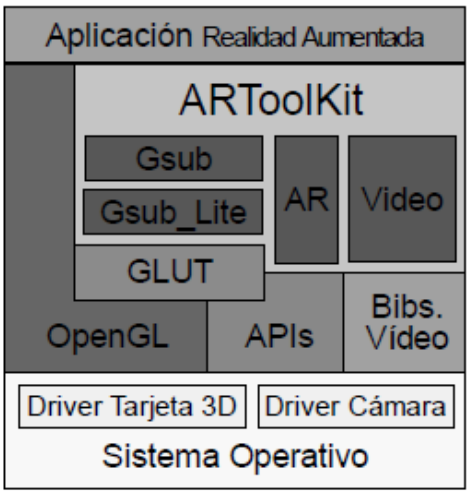

Figura 7.1: Arquitectura de ARToolKit

Si el marcador es lo suficientemente grande será detectado, la imagen es pasada a escala de grises y después se binariza, se le aplica un algoritmo de detección de los contornos y se extraen las aristas y los vértices del cuadrado que compone el marcador.

Se procede a la fase de normalización donde se extrae el contenido del marcador y se compara con marcas conocidas.

Conociendo las posiciones de las aristas y los vértices es posible estimar la posición de la cámara con respecto a la marca.

Sabiendo la proyección de cada esquina de la marca sobre las coordenadas de pantalla (xc, yc), y las restricciones asociadas al tamaño de las marcas y su geometría cuadrada, es posible calcular la matriz que permita hacer la transformación entre el sistema de coordenadas en base a la cámara y el sistema de coordenadas en base al marcador. Consigue que el trazado en AR modifique su posición a medida que se cambia la posición relativa entre la cámara y el marcador. La figura 7.2 resume esa transformación.

7.3.1. Aplicación ARToolKit nativa

Esta aplicación surge como respuesta a una necesidad fundamental de nuestro proyecto, como es el crear una aplicación que sea capaz de representar con realidad aumentada lo que se dibuja en la aplicación receptora.

Para ello se utiliza como base el libro de AR llamado Realidad Aumentada de González Morcillo et al. (2011), en castellano, realizado por 4 profesores de la Universidad de Castilla-La Mancha, a raíz de un curso que dieron sobre AR. En él se explica de manera clara y concisa los pormenores de la AR. Contiene ejemplos y ejercicios para realizar junto con el curso y la lectura

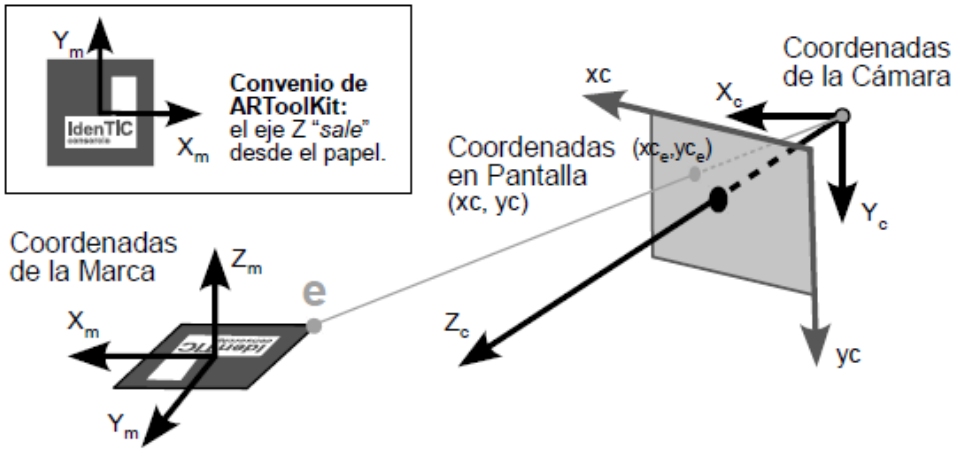

Figura 7.2: Sistema de coordenadas de ARToolKit

del libro en cuestión.

Después de leer este interesante libro y probar algunos de sus ejemplos, se dio con uno que tenía que ver con lo que nosotros buscábamos, pintar gracias a un marcador en AR sobre un plano generado con otro marcador. Era muy interesante a nivel formativo, por lo que se creyó buena idea para implementar y modificar para que se acercase al resultado final que se buscaba en el proyecto.

El proyecto estaba pensado para una versión de ARToolKit antigua, con lo que se tuvieron que hacer múltiples pruebas con el antiguo, con la versión actual, tanto en Windows como en Linux.

Por último pasamos a explicar mas concretamente el funcionamiento de ARToolKit nativo como de algunos pormenores de la aplicación creada.

- Se instala ARToolKit en Windows y se prueba el ejemplo de pintado del libro de AR. No pudiendo compilar desde la consola de comandos se cambia a Linux.

- En Linux Ubuntu 16.04 se instala primero la última versión de ARToolKit (versión 5). Como hay diferentes versiones de las funciones se intenta adaptar el código del ejemplo sin conseguir compilar.

- En el mismo sistema operativo, se intenta compilar el código original del ejemplo con la versión de ARToolKit ofrecida por el libro. Esto produce un fallo en "glDepthFunc". Por lo se decide volver a Windows y trabajar en Visual Studio.

- En Visual Studio 2015 se crea un proyecto con el código del ejemplo adaptado a la versión 5 de ARToolKit. Para ello ha sido necesario

investigar en profundidad este entorno de desarrollo para especificar las dependencias de librerías (tanto ficheros .dll como .lib) y poder enlazar todos los módulos requeridos por el código. Algunas de la librerías no estaban correctamente generadas con lo que se opta por la versión de ARToolKit (versión 2.72) disponible junto al libro.

- Previamente al código del ejemplo, se procede a recompilar las librerías necesarias de ARToolKit versión 2.72 bajo Visual Studio 2015 por haber sido creado en Visual Studio 2003, lo que implica un enlace a librerías de Microsoft no disponibles en el sistema operativo con el que se está trabajando.

- Resuelto el tema de librerías de ARToolKit en su versión 2.72, se consigue construir un proyecto con el código del pintado en AR. La ejecución se ha realizado con éxito una vez establecidos correctamente los datos de entrada necesarios (Lamb, 2007), así como la indicación de la localización de los ficheros como los de definición de los marcadores a reconocer en Visual Studio.

  Este ejemplo basa el reconocimiento de las coordenadas en un conjunto de doce marcadores teniendo inconvenientes para que sean detectados todos simultáneamente.

  A continuación se mira como cambiar la fuente de vídeo por el live streaming. En consecuencia se tiene que seguir el desarrollo con la última versión de ARToolKit por necesitar la opción quicktime para procesar una url, funcionalidad disponible a partir de la versión 4.3 (ARToolKit, 2016).

- La mayor dificultad adicional ha sido el adaptar el código de pintado en AR a la versión 5 de ARToolKit. Este proceso ha partido comparando la documentación de ambas versiones para buscar las equivalencias. Se ha podido completar con mayor facilidad gracias a los códigos de ejemplo de ARToolKit versión 5, en particular los ficheros “simpleTest.c” “multiTest.c”.

  Una vez recompilada alguna de las librerías de ARToolKit versión 5 se ha obtenido una ejecución correcta. Al igual que con el plugin de ARTolkit en Unity, la opción de vídeo mediante Quicktime sólo se consigue hacer funcionar si la url contiene un vídeo con un códec y contendor admitidos por Quicktime.

  Así pues se ha probado el modificar los parámetros en el emisor FFmpeg sin éxito siguiendo una guía para codificar que menciona el cambio del formato de pixel a yuv420p para Quicktime (FFmpeg, 2016a).

El desarrollo debería seguir por el cambio de la fuente de vídeo para recibir un live streaming. Una vez obtenido habría que proceder con asociar

los trazos realizados al live streaming y obtener la AR separada como se pretendía desde un principio.

Parte IV

Conclusiones y trabajo futuro

En esta última parte de la memoria se exponen las conclusiones y las lineas de trabajo futuras a desarrollar a partir de lo presentado.

Capítulo 8

Conclusiones y trabajo futuro

En este capítulo concluiremos la presente memoria con un análisis del trabajo realizado, así como de las decisiones tomadas y los resultados obtenidos como consecuencia.

8.1. Conclusiones

Como se pudo observar durante la investigación de las diferentes gafas de Realidad Aumentada que existen en el mercado, las aplicaciones en el ámbito de la medicina son numerosas, pero limitadas por el amplio número de empresas con prototipos y escasos productos finales que impiden llevar a la práctica todas las posibilidades que brinda la misma.

En estos momentos el *hardware* a nivel tecnológico deja bastante que desear, sobre todo cuando los avances técnicos derivados de los *smartphones* podrían aportar las especificaciones deseadas para el proyecto. Puntos clave como la calidad de la cámara y la duración de la batería son mejorables, se necesita una mayor inversión en estos aspectos. Con la llegada de las nuevas Epson Moverio BT-300 y las Hololens de Microsoft se puede paliar este gran hándicap.

A nivel software, los *frameworks* tienen una dependencia excesiva de las fuentes de vídeo *hardware*, con lo que reducen las posibilidades de trabajo en el ámbito que nos compete.

El hecho de trabajar con código libre ha llevado a encontrar una comunidad de desarrolladores muy dispersa, poco participativa y en general, proyectos y aportaciones poco actuales, con un vacío importante desde 2011 hasta la actualidad.

La realización de este proyecto nos ha llevado a explorar múltiples tecnologías, hemos podido trabajar con *software*, *hardware* y adentrarnos en un campo de frontera.

Creemos que debido a dichas dificultades la importancia de nuestro TFG

recae en la investigación de dichas tecnologías y su aplicación. Así mismo este proyecto nos ha servido para aprender a trabajar en equipo, distribuir tareas y carga de trabajo. Pensamos que ha sido muy enriquecedor a nivel personal el poder trabajar con tantas tecnologías diferentes y además hacer uso de un componente *hardware* que se corresponde más con la titulación que hemos cursado.

Mediante la colaboración con la unidad de telemedicina hemos podido experimentar el contacto con un entorno real de trabajo y conocer los pormenores de esta unidad en nuestro país.

8.2. Conclusions

From the study on available AR smart glasses, there is a wide range of utilisations in the medicine field. However, those applicatioins are restriced by a large number of companies with their own prototypes and scarce final devices rendering difficult implementing all offered opportunities by AR.

Nowadays, available devices are built with low technical specifications compared to an increasingly progress in almost every component stemming from smartphones development. An usage of more up to date parts, could fulfil the minimum criteria of for this project. Some essential aspects as the camera quality or the battery duration must be improved. Therefore, greater investments are to be made in those directions. This situation can be partially solved with the newer models like Epson Moverio BT-300 or the Hololens device from Microsoft.

At a software level, AR frameworks are excessively relying on hardware video sources, reducing our options to work in this context.

Working with open source code has lead to finding a scattered and little participating developer community. In general, both projects and contributions were old, without regular continuity from 2011 onwards.

For this project we were able to explore several technologies both software and hardware wise as well as step into a cutting edge field.

We strongly believe that due to those circumstances our Final Year Project mainly consists in a research on those technologies and their applications. From this project we have learnt how to work in a team, distributing tasks and workloads. We think this oppportunity to study such variety of different technologies has been fulfilling, without forgetting about the use of a hardware device befitting our degree.

Through the partnership with the telemedicine department, we have been granted a feeling of a real work environment and learn in detail about this unit in our country.

8.3. Lineas de trabajo futuro

Una vez finalizado el desarrollo surgen varias ideas que podrían ampliar las funcionalidades implementadas.

- Aplicación AR gafas: desarrollo de una aplicación con AR para las gafas inteligentes que muestren al cirujano las mismas indicaciones que a los alumnos en el aula de docencia. Esta opción vendría a solucionar el problema real del que parte este TFG y se podría utilizar también fuera de las instalaciones del hospital para prestar asistencia en las zonas donde estén desplazadas las tropas.

- Portabilidad aplicación docencia: modificación de la aplicación de docencia del hospital para su uso en otros sistemas operativos móviles (Android, iOS...) o de escritorio como Mac OS o Linux pudiendo proveer la ayuda de una forma más ágil al estar disponible en múltiples sistemas.

- Integración del audio: tal como se habla anteriormente, dejamos el código preparado para probar a enviar y recibir audio, interesante a futuro.

- Integración con Vuforia: Esta línea de trabajo depende de que Vuforia implemente el poder cambiar una fuente de video y que sea capaz de recepcionar y reconocer *live streaming*.

Parte V

Apéndices

Apéndice A

Aportaciones

A.1. Atribuciones

Este TFG no podría haberse llevado a término sin el trabajo en equipo de todos y cada uno de los alumnos integrantes. Esto se ha aplicado en cada una de las fases del proyecto empezando por la búsqueda de temática y director.

En un trabajo de frontera como este, donde el camino no estaba bien definido en ningún instante, se ha tratado de repartir las tareas que iban surgiendo para avanzar con el desarrollo de forma equitativa entre los tres miembros.

Aún así, no se ha establecido una división estricta de los objetivos a cumplir, lo que ha permitido un importante trabajo en equipo resultando en una estrecha colaboración, aporte de ideas, y revisión del trabajo realizado reduciendo así la propensión a errores.

Debido a una afinidad, posiblemente por primera asignación, se puede construir los siguientes conjuntos de atribuciones y responsabilidades en función de los estudiantes.

A.2. Aporte de Nicolás Bueno Mora

Aportaciones a nivel técnico:

- A lo largo del desarrollo del proyecto, la parte de más peso dentro del mismo ha sido la parte técnica.

- Debido a que cursé de Erasmus una asignatura sobre Android y mis compañeros no tenían conocimientos previos, decidí encargarme yo de llevar la parte de Android, tanto para generar aplicaciones, como para las pruebas y la compilación de librerías adicionales cuando fuera necesario.

- He ampliado mis conocimientos de Android Studio, profundizado los de Eclipse para las aplicaciones, así cómo las particularidades para la integración del NDK en ambos entornos.

- Con respecto al tema de los tres servidores que se han probado, Wowza, FFserver e Icecast, he sido participe en el estudio de los mismos y parte fundamental en la configuración y pruebas de los mismos.

- Durante el estudio y configuración de los servidores, he tenido que investigar a nivel de los protocolos de comunicación, los codecs y los contenedores utilizados.

- Para establecer una comunicación entre el servidor y el cliente receptor me ha sido imperativo el documentarme sobre la recepción del vídeo en Unity y ARToolKit. En referencia a éste último, para poder habilitar esta funcionalidad, he recompilado el módulo.

- Esta tarea la he tenido que repetir para obtener las librerías de ARToolKit y poder ejecutar el código de dibujado en 3D.

- Al igual que con las aplicaciones de Android, para el emisor finalmente elegido he estudiado y probado hasta encontrar la configuración que permita enviar el vídeo.

  Aportaciones a nivel personal y humanista:

- He realizado el conjunto de escritos en inglés, para poder poner en práctica el aumento de conocimientos adquiridos en dicha estancia. ¡Gracias Leeds!

A.3. Aporte de Ignacio Cerdá Sánchez

He participado en todas las etapas del proyecto pero sobre todo en la implantación del servidor intermedio, el desarrollo de la aplicación cliente receptora y la memoria. A continuación, comento las tareas que he realizado durante el proyecto:

- Investigación y adquisición de los conocimientos mínimos sobre realidad aumentada y su funcionamiento, con la finalidad de conocer una de las tecnologías con la íbamos a trabajar durante el proyecto.

- Búsqueda y estudio de los primeros *scripts* para la inclusión de *streaming* de vídeo/audio y dibujo a mano alzada en Unity.

- Desarrollo de diferentes proyectos de realidad aumentada básicos tanto con el uso del *framework* Vuforia como ARToolKit.

- Visita junto con mi compañero Ricardo a las instalaciones (incluidos los quirófanos) de Telemedicina en el Hospital Central de la Defensa Gómez Ulla para tomar nota de los requisitos y sugerencias del personal de cara al desarrollo del proyecto.
- Exploración sobre el uso del servidor de código abierto Icecast para reemplazar el fallido uso de FFserver.
- Búsqueda y prueba de diferentes versiones de Unity según las necesidades del proyecto en cada momento.
- Aprendizaje autodidacta sobre el uso de LaTeX, plantillas y software asociado para la realización de la memoria.
- Creación de esta memoria y escritura en la mayoría los capítulos de la misma. Revisión de todo el documento.
- Realización y edición de la mayoría de las imágenes incluidas en la memoria.
- Puesta en marcha del repositorio de control de versión basado en Git a través del proveedor Bitbucket y búsqueda de un cliente para su uso bajo Microsoft Windows.
- Proveedor del material necesario para la creación de una red inalámbrica local para el proyecto, que evitase los problemas derivados del uso de la red Eduroam (Eduroam, 2017) de la universidad.

A.4. Aporte de Ricardo Eugui Fernández

Mis aportaciones al proyecto:
A nivel personal:

- Contactar con Luis Vázquez Martínez para gestionar una reunión con Fernando Setién y ver si había posibilidades de colaboración con el ejercito.
- Convencer a Fernando Setién para que el servicio de telemedicina nos sufragase el coste de las gafas de realidad aumentada.
- Cuando surgieron problemas y gestiones que hacer fui el encargado de llevarlas a cabo, tanto con Fernando Setién como con Luis Vázquez.
- Alentar, motivar, aconsejar y escuchar a mis compañeros, aprovechándome de mi mayor edad y experiencia.

A nivel técnico:

- Aprendizaje de Android Studio, Unity y ARToolKit tanto a través de tutoriales como de ejemplos para la realización de aplicaciones.
- Búsqueda de un repositorio basado en Git que sea capaz de tener proyectos privados para evitar el robo del código como nos pasó en TP el año que la cursamos. El elegido fue BitBucket.
- Instalación y configuración del cliente para gestionar BitBucket.
- Investigación de diferentes *frameworks* como alternativa a Vuforia, tanto a nivel técnico, como a nivel económico.
- La idea de utilizar software similar a Skype para la comunicación entre el emisor y receptor, y la búsqueda de posibles aplicaciones de código libre que pudiesen sernos útiles. Concretamente TOX, una aplicación que comunica a dos dispositivos, se rechazó porque aunque era capaz de conectar y enviar vídeo y audio a través de Windows, no estaba desarrollado para Android, que era una de las ideas que teníamos para la aplicación del cliente emisor.
- Creación de aplicación que recibe un video con *live streaming* y es capaz de pintar diferentes lineas sobre el, permitiendo que el cliente receptor sea capaz de ver tanto el *streaming* como las lineas impresas correctamente. Añadí la funcionalidad para poder cambiar el grosor de las lineas en colaboración con mi buen compañero Nicolás.
- Encontré un libro totalmente recomendable llamado Realidad Aumentada. Un Enfoque Práctico con ARToolKit y Blender (González Morcillo et al., 2011) donde pudimos aprender mucho sobre la realidad aumentada, practicar con ejemplos y poder usar como guía uno de los ejemplos para el poder pintar con realidad aumentada sobre una superficie definida por un marcador.

Apéndice B

Acrónimos

En este apartado se enumeran los acrónimos utilizados a lo largo de esta memoria.

AR Augmented Reality, Realidad Aumentada.

ARToolKit Augmented Reality Toolkit, Kit de Herramientas de realidad Aumentada.

Crowdfounding Crowdfounding, Recolección Popular de Fondos.

DICOM Digital Imaging and Communication in Medicine, Estándar para el intercambio de pruebas médicas.

ECTED Escenarios Cientificos y Tecnológicos de la Defensa.

ECG Electrocardiogram o EKG en alemán, Elektrokardiogramm, Representación gráfica de la actividad eléctrica del corazón.

GLUT the OpenGL Utility Toolkit, Kit de Herramientas OpenGL.

GPL General Public License, Licencia Pública General de GNU.

HMD Helmet Mounted Display, Sistema de visualización montado en casco.

HTTP Hypertext Transfer Protocol, Protocolo de transferencia de hipertexto.

IP Internet Protocol.

KARMA Knowledge-based Augmented Reality for Maintenance Assistance, Realidad Aumentada basada en el Conocimiento para Asistencia de Mantenimiento.

NDK Native Development Kit, Kit de desarrollo de código nativo para la progrmación en C en Android.

OMS Organización Mundial de la Salud.

qHD Quarter of Full HD, Un cuarto de Full HD.

RTSP Real Time Streaming Protocol, Protocolo de transmisión en tiempo real.

RTMP Real Time Messaging Protocol, Protocolo de transmisión de mensajes tiempo real.

SIP Session Initiation Protocol, Protocolo de inicio de sesiones.

SDK Software Development Kit, Kit de desarrollo de software.

SSL Secure Sockets Layer, Capa de puertos seguros.

TFG Trabajo Fin de Grado.

TLS/SSL Transport Layer Security, Seguridad de la capa de transporte.

VOD Video On Demand, Video Bajo Demanda.

VGA Video Graphics Array, Adaptador Gráfico de Video.

VLC VideoLAN Client, Reproductor multimedia.

VR Virtual Reality, Realidad Virtual.

WWW World Wide Web, Red Informática Mundial.

XML Extensible Markup Language, Lenguaje de Marcas Extensible.

Bibliografía

No lo intentes. Hazlo, o no lo hagas,
pero no lo intentes.

Yoda a Luke Skywalker. STAR WARS,
episodio V, El Imperio Contrataca

DE ANDALUCÍA, J. Proyecto palante (andalucía). Disponible en `http://www.juntadeandalucia.es/servicioandaluzdesalud/principal/documentosacc.asp?pagina=pr_palante_1`.

ARTOOLKIT. Configuring video capture in artoolkit. Disponible en `http://artoolkit.org/documentation/doku.php?id=2_Configuration:config_video_capture`.

AUDET, S. Java interface to opencv and more. Disponible en `https://github.com/bytedeco/javacv/`.

BYTEDECO. javacv/samples/recordactivity.java. Disponible en `https://github.com/bytedeco/javacv/blob/master/samples/RecordActivity.java`.

CORP., A. P. Telnet / ssh simple client. Disponible en `https://play.google.com/store/apps/details?id=apc.android.tool.telnet&hl=es`.

EDUROAM. Eduroam. Disponible en `https://www.eduroam.es/`.

EPIRAT. Icecast protocol specification. Disponible en `https://gist.github.com/ePirat/adc3b8ba00d85b7e3870`.

FFMPEG. Ffmpeg and h.264 encoding guide. Disponible en `https://trac.ffmpeg.org/wiki/Encode/H.264`.

FFMPEG. July 10th, 2016, ffserver program being dropped. Disponible en `http://ffmpeg.org/index.html#ffserv`.

FFMPEG. About ffmpeg. Disponible en `http://ffmpeg.org/about.html`.

FYHERTZ, S. libstreaming. Disponible en https://github.com/fyhertz/libstreaming.

GONZÁLEZ MORCILLO, C., VALLEJO FERNÁNDEZ, D., ALBUSAC JIMÉNEZ, J. A. y CASTRO SÁNCHEZ, J. J. Realidad aumentada. un enfoque práctico con artoolkit y blender. Disponible en http://www.librorealidadaumentada.com.

GOOGLE. Google - android wear. Disponible en https://www.android.com/intl/es_es/wear/.

GOOGLE. android.net.sip. Disponible en https://developer.android.com/reference/android/net/sip/package-summary.html.

GOOGLE. Supported media formats. Disponible en https://developer.android.com/guide/topics/media/media-formats.html.

HILLÁN GARCÍA, L., SETIÉN DODERO, F. y DEL REAL COLOMO, A. El sistema de telemedicina militar en España: una aproximación histórica. Disponible en http://scielo.isciii.es/scielo.php?pid=S1887-85712014000200010&script=sci_arttext.

HIPERTEXTUAL. Pokémon go y la realidad aumentada, ¿el futuro de los videojuegos? Disponible en https://hipertextual.com/2016/07/pokemon-go-futuro-videojuegos-exito.

HUA, L. X. simplest ffmpeg android streamer. Disponible en https://github.com/leixiaohua1020.

ICO. Plan e-tico (cataluña). Disponible en goo.gl/rf2FoH.

ISO/IEC 14496-14:2003, O. I. D. N. Iso/iec 14496-14:2003. Disponible en http://www.iso.org/iso/catalogue_detail.htm?csnumber=38538,Year=2014.

LAMB, P. Artoolkit video configuration. Disponible en http://artoolkit.sourceforge.net/apidoc/video/#VideoWin32DirectShow271.

MICROSOFT. Microsoft - hololens. Disponible en https://www.microsoft.com/microsoft-hololens/en-us.

NACHFUELLBAR. Video is unsupported on android. Disponible en https://github.com/uTox/uTox/issues/134.

NIEDERMAYER, M. About ffmpeg. Disponible en https://osdn.net/projects/android-x86/scm/git/external-ffmpeg/commits/ac957bc60cec69bd5724fe6f41b6fb979ac6612c.

Osakidetza. Inicitativa telepoc (pais vasco). Disponible en http://www.osakidetza.euskadi.eus/contenidos/informacion/jornada_cronicidad/eu_jarduna/adjuntos/08.pdf.

Palevich, J. Android-terminal-emulator. Disponible en https://github.com/jackpal/Android-Terminal-Emulator.

Roman10. How to build ffmpeg with ndk r9. Disponible en http://www.roman10.net/2013/08/18/how-to-build-ffmpeg-with-ndk-r9/.

RTVE. Telemedicina hostpital de guadarrama con hospital ramón y cajál (madrid). Disponible en http://www.rtve.es/alacarta/audios/espana-vuelta-y-vuelta/espana-vuelta-vuelta-10-05-16/3603659/.

Technologies, U. Www.movie. Disponible en https://docs.unity3d.com/ScriptReference/WWW-movie.html.

uTox. utox the lightest and fluffiest tox client. Disponible en https://github.com/uTox/uTox.

VideoLAN. VLC. Disponible en https://www.videolan.org/vlc/index.es.html.

Wikipedia (3GP). Entrada: "3GP". Disponible en https://en.wikipedia.org/wiki/3GP_and_3G2 (último acceso, December, 2016).

Wikipedia (AAC). Entrada: "AAC". Disponible en https://en.wikipedia.org/wiki/Advanced_Audio_Coding (último acceso, February, 2017).

Wikipedia (Google Glass). Entrada: "Google Glass". Disponible en https://es.wikipedia.org/wiki/Google_Glass (último acceso, January, 2017).

Wikipedia (H.263). Entrada: "H.263". Disponible en https://en.wikipedia.org/wiki/H.263 (último acceso, January, 2017).

Wikipedia (H.264). Entrada: "H.264". Disponible en https://es.wikipedia.org/wiki/H.264/MPEG-4_AVC (último acceso, December, 2016).

Wikipedia (MP3). Entrada: "MP3". Disponible en https://en.wikipedia.org/wiki/MP3 (último acceso, January, 2017).

Wikipedia (MP4). Entrada: "MP4". Disponible en https://en.wikipedia.org/wiki/MPEG-4 (último acceso, December, 2016).

WIKIPEDIA (Ogg). Entrada: "Ogg". Disponible en https://en.wikipedia.org/wiki/Ogg (último acceso, December, 2016).

WIKIPEDIA (QuickTime File Format). Entrada: "QuickTime File Format". Disponible en https://en.wikipedia.org/wiki/QuickTime_File_Format (último acceso, December, 2016).

WIKIPEDIA (Real Time Messaging Protocol). Entrada: "Real Time Messaging Protocol". Disponible en https://en.wikipedia.org/wiki/Real-Time_Messaging_Protocol (último acceso, January, 2017).

WIKIPEDIA (Real Time Streaming Protocol). Entrada: "Real Time Streaming Protocol". Disponible en https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol (último acceso, January, 2017).

WIKIPEDIA (Session Initiation Protocol). Entrada: "Session Initiation Protocol". Disponible en https://en.wikipedia.org/wiki/Session_Initiation_Protocol (último acceso, January, 2017).

WIKIPEDIA (Streaming Guide). Entrada: "Streaming Guide". Disponible en https://trac.ffmpeg.org/wiki/StreamingGuide (último acceso, January, 2017).

WIKIPEDIA (Streaming WebM to Icecast with FFmpeg). Entrada: "Streaming WebM to Icecast with FFmpeg". Disponible en https://wiki.xiph.org/Icecast_Server/Streaming_WebM_to_Icecast_with_FFmpeg (último acceso, January, 2017).

WIKIPEDIA (Vorbis). Entrada: "Vorbis". Disponible en https://en.wikipedia.org/wiki/Vorbis (último acceso, February, 2017).

WIKIPEDIA (VP8). Entrada: "VP8". Disponible en https://en.wikipedia.org/wiki/VP8 (último acceso, January, 2017).

WIKIPEDIA (WebM). Entrada: "WebM". Disponible en https://en.wikipedia.org/wiki/WebM (último acceso, January, 2017).

WIKIPEDIA. Telemedicine history. Disponible en https://en.wikipedia.org/wiki/Telemedicine#History.

WOWZA. How to use wowza gocoder video broadcasting android app. Disponible en https://www.wowza.com/forums/content.php?595-How-to-use-Wowza-GoCoder-video-broadcasting-Android-app#troubleshooting.

WOWZA. Wowza streaming engine. Disponible en https://www.wowza.com/products/streaming-engine.

Xataka. ¿cómo quedaría ese muebles ikea en tu casa? compruébalo con su aplicación móvil. Disponible en `https://goo.gl/JB2mGv`.

Xiph. Icecast server/streaming webm to icecast with ffmpeg. Disponible en `https://wiki.xiph.org/Icecast_Server/Streaming_WebM_to_Icecast_with_FFmpeg`.

Dime y lo olvido, enséñame y lo recuerdo, involúcrame y lo aprendo.
Benjamin Franklin

Para viajar lejos no hay mejor nave que un libro.
Emily Dickinson

Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.
Albert Einstein

En cuestiones de cultura y de saber, sólo se pierde lo que se guarda; sólo se gana lo que se da.
Antonio Machado