Escuela Politécnica Superior

Cyberpresencia para entornos educativos usando realidad mixta

Máster Universitario en Automática y Robótica

Trabajo Fin de Máster

Autor:
Bessie Domínguez Dáger
Tutor/es:
Miguel Ángel Cazorla Quevedo
Francisco Gómez Donoso

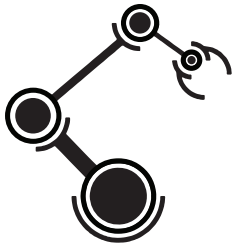
Junio 2022

Universitat d'Alacant
Universidad de Alicante

Cyberpresencia para entornos educativos usando realidad mixta

Autor
Bessie Domínguez Dáger

Tutor/es
Miguel Ángel Cazorla Quevedo
Ciencia de la Computación e Inteligencia Artificial (CCIA)
Francisco Gómez Donoso
Ciencia de la Computación e Inteligencia Artificial(CCIA)



Máster Universitario en Automática y Robótica

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Junio 2022

Preámbulo

"La posibilidad de eliminar barreras existentes en la comunicación de las personas mediadas por diferentes espacios físicos, es una idea inspiradora para el beneficio y desarrollo de la humanidad, constituyendo un paso de avance en la forma de relacionarse e interactuar. La principal razón del presente trabajo es investigar y contribuir en el campo de la telepresencia mediante el uso de realidad mixta. Su finalidad está enfocada hacia la creación de un medio de integración de entornos presenciales y remotos en el ámbito educacional, contribuyendo a disminuir las limitaciones que actualmente supone la comunicación a distancia."

Agradecimientos

En primer lugar quiero agradecer a mis tutores, Miguel Ángel Cazorla y Francisco Gómez, por su guía y apoyo incondicional en todo momento, no solo en situaciones meramente educacionales sino también personales y por proporcionarme las herramientas de hardware necesarias para el desarrollo del presente trabajo. Además, quiero agradecer a mi familia, principalmente a mi madre por su participación en todas las etapas de mi vida y sus buenos y valiosos consejos; a mis abuelos y tío por su influencia constante en mi formación, su cariño y preocupación. Doy las gracias a mis amigos, que siempre han estado disponibles tanto en las buenas como malas batallas. También, reconozco a los compañeros de clases del máster y del laboratorio Robotics & Tridimensional Vision (RoViT), que participaron en la experimentación del trabajo. Finalmente, agradezco a la beca concedida en la XII Convocatoria Banco Santander-UA y al personal de Relaciones Internacionales y Cooperación para el Desarrollo de la UA, sin los cuales no hubiese sido posible el estudio del presente máster.

Vive como si fueras
a morir mañana.
Aprende como si fueras
a vivir siempre.

Mahatma Gandhi.

Índice general

Índice de figuras

Índice de Tablas

1 Introducción

El nuevo escenario mundial impuesto por el virus SARS-CoV-2, ha provocado una situación problemática atípica en múltiples esferas. En general, ha supuesto un proceso de tránsito y adaptación de contextos presenciales a la no presencialidad. En el ámbito de la Educación Superior, la crisis sanitaria ha impuesto que el aprendizaje abandone el aula física para desenvolverse en un espacio virtual [14]. En este proceso se ha visto implicada una transformación de la información y la comunicación desarrolladas en el nuevo entorno. En la búsqueda de la continuidad de la enseñanza y el curso lectivo, han sido empleadas diversas herramientas tecnológicas que, con mayor o menor éxito, han posibilitado realizar docencia dual o en línea. Estas herramientas han consistido en distintas plataformas como Google Meet, Zoom, Webex, BigBlueButton, Elluminate Live!, Wimba, etc., las cuales permiten una conexión mediante videoconferencia [14].

El cambio brusco de todo lo presencial a lo no presencial, provocado por la pandemia, no solo ha requerido un uso generalizado de la tecnología digital, sino también un cambio en la metodología y en las interacciones comunicativas. En el contexto docente, ni el emisor (profesor) ni el receptor (alumnado) –ni tampoco el propio sistema—, estaban preparados para un cambio tan drástico [14, 15]. El aula pasó a estar mediada por una pantalla digital que influyó en la metodología y la comunicación [15]. Este tipo de comunicación implicaba la imposibilidad de poder percibir la totalidad de reacciones personales, además de la limitación de movimientos ante lo que se estaba transmitiendo, con lo que la comunicación se tornaba menos fluida. Además, ante un número elevado de estudiantes en lugares diferentes, los sistemas actuales de videoconferencia no permiten visualizar a todos los asientes de forma simultánea, perdiéndose información comunicativa no verbal.

Una vez superado el confinamiento, en un periodo excepcional por el distanciamiento social, el e-learning fue sustituido por el b-learning, con escenarios híbridos [16]. De nuevo, los cambios en la metodología y la interacción comunicativa en el aula fueron necesarios debido al uso combinado de la enseñanza presencial y no presencial. Este nuevo contexto implicó un gran desafío para el profesorado, ya que su función de emisor pasó a encontrarse en dos entornos simultáneamente: presencial (aula) y virtual (videoconferencia). Todos estos cambios impuestos en la instrucción tecnológica provocaron una ruptura de características de la educación tradicional en cuanto a unidad de tiempo, de espacio y de acción [15].

A pesar de que las situaciones descritas han generado disímiles dificultades que a la vez constituyen una oportunidad para profundizar e incorporar significativamente nuevos recursos tecnológicos disruptivos. En este sentido nos referimos a recursos que permitan delimitar los escenarios futuros en el ámbito educacional de manera que ofrezcan un potencial aprovechable, modernizando y optimizando los procesos formativos, informativos y comunicativos.

En estas condiciones, constituye no sólo una oportunidad, sino más bien una necesidad,

el hecho de configurar la forma de los escenarios formativos, informativos y comunicativos, tanto virtuales como presenciales. Bajo las premisas descritas, debemos analizar que ahora tenemos mayor disponibilidad de medios y mecanismos para contrarrestar estas circunstancias desafiantes y que, la formación telemática nos ofrece considerables posibilidades en este ámbito [15]. Nos estamos refiriendo a un cambio que implica el desarrollo de nuevos entornos educacionales basados en Inteligencia Artificial (IA) [17], Visión por Computador y Realidad Mixta (RM).

En los nuevos entornos considerados, el elemento fundamental deja de ser el contenido [18], para convertirse en los agentes participantes (emisor-receptor: profesorado-alumnado). En este contexto, la tecnología digital debe servir para potenciar los aspectos emocionales positivos involucrados en una situación de formación, información y comunicación, al mismo tiempo que neutraliza los negativos [14]. Teniendo en cuenta estos supuestos, mediante el empleo de escenarios mixtos que combinen lo presencial con lo virtual, estaremos creando innovadores espacios educativos fortalecidos para la colaboración, interacción y construcción de nuevas formas de relacionarnos con la realidad [15].

Las tecnologías de RM permiten la creación de entornos mixtos, mostrando un elevado potencial para modificar la forma en que interactuamos con el entorno, las personas y el mundo real [1]. La RM hace referencia a la combinación de la Realidad Virtual (RV) con la Realidad Aumentada (RA). La RV puede ser definida como un entorno simulado mediante tecnología informática que permite experimentar sensaciones de inmersión e interacción con el espacio recreado [19]. Sin embargo, la RA representa un sistema tecnológico más reciente que la RV que posibilita recrear escenarios con un mayor grado de sensación de realidad. Esto se debe a que la RA incorpora objetos virtuales al mundo real durante la experiencia del usuario, a la vez que funciona de forma interactiva y en tiempo real [20].

Actualmente, entre las interfaces de RM (*Mixed Reality Interfaces (MRITF)*) que más están siendo empleadas y adaptadas a diferentes escenarios se encuentran las gafas HoloLens 2 de Microsoft (Figura 1.1) [1]. Esta herramienta tecnológica presenta potentes elementos sensoriales y capacidad de cálculo integrada, de forma que permite desarrollar novedosas aplicaciones explotando los campos de la RM, la IA y la Visión por Computador [1]. Este conjunto nos ofrece una plataforma sólida, sobre la cual es posible implementar una aplicación que nos proporcione una mejor gestión de la no presencialidad, aumentando su uso y minimizando los problemas que su actual empleo conlleva.

Figura 1.1: HoloLens 2 de Microsoft [1]

Por lo antes expuesto, el presente trabajo propone el diseño e implementación de una herramienta de software en las gafas HoloLens 2 de Microsoft, basada en tecnología de RM, IA y visión por computador, que permita una mejora en los escenarios docentes universitarios, por cuanto se refiere a situaciones de formación, información y comunicación en entornos mixtos (Figura 1.2). Para ello, la herramienta constituirá un novedoso apoyo al profesorado universitario al posibilitarle integrar en un espacio común la presencialidad (alumnado en clase) con la no presencialidad (alumnado en videoconferencia).

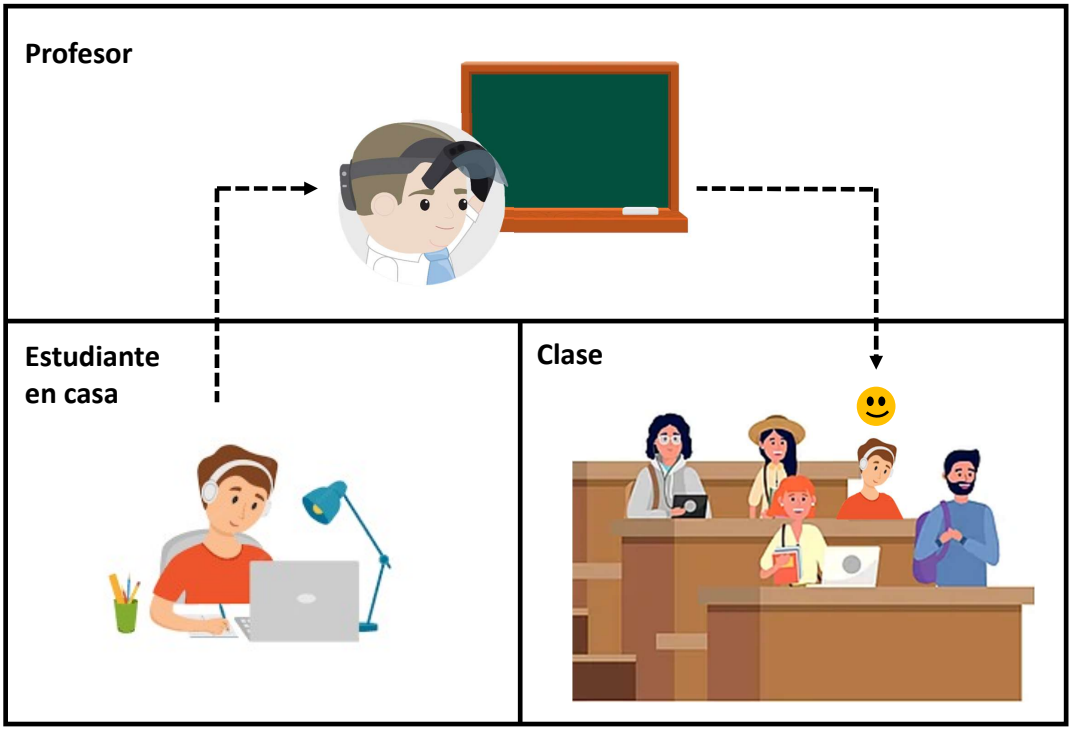


Figura 1.2: Sistema de cyberpresencia propuesto para entornos educativos empleando RM. Ahora El profesor en el aula puede visualizar al estudiante en casa conjuntamente con los estudiantes presenciales mediante el uso de HoloLens 2

El documento se estructura de la siguiente manera: el Capítulo 2 presenta el estado del arte referente a sistemas de telepresencia y al empleo de HoloLens 2 en Educación. En el Capítulo 3 se especifican los objetivos del trabajo. El Capítulo 4, presenta las herramientas y softwares utilizados para el desarrollo del sistema propuesto. Posteriormente, en el Capítulo 5 se describe cómo se emplearon las tecnologías abordadas previamente, para realizar el diseño, programación e implementación del proyecto desarrollado. En el Capítulo 6 se presentan los experimentos realizados y se analizan los resultados obtenidos. Por último, en el Capítulo 7 se exponen las conclusiones arribadas con la realización del trabajo, abordando los logros y las dificultades presentadas y proponiendo posibles trabajos futuros.

2 Marco Teórico

La investigación e implementación de tecnologías de RA y RM adquieren cada vez mayor importancia en diferentes campos como la Educación, Defensa, Automatización, Ingeniería Industrial, Arquitectura, entre otros [21]. Concretamente, en este Capítulo abordamos el estado del arte referente al uso de estas tecnologías en la Educación Superior y en el desarrollo de sistemas de telepresencia. Para ello, en la Sección 2.1 se comenta el incremento que ha indicado el empleo de RA/RM desde su primer uso en Educación Superior y se muestran ejemplos de aplicaciones desarrolladas en este sector en los últimos cinco años. Finalmente, en la Sección 2.2 se expone el sistema de telepresencia basado en RM más destacado en la actualidad y se discuten sus limitaciones.

2.1 Uso de RA/RM en Educación Superior

Según [22], el primer uso de RA en Educación Superior data del año 1995, con el diseño de una herramienta para enseñar anatomía tridimensional. Dicha herramienta fue desarrollada en la Universidad de Carolina del Norte, Estados Unidos y presentada en la primera Conferencia Internacional sobre Visión por Computador, Realidad Virtual y Robótica en Medicina (Francia, 1995). La aplicación consistía en superponer y registrar estructuras óseas generadas gráficamente, sobre partes anatómicas de un cuerpo humano real, utilizando un dispositivo Head Mounted Display (HMD). Dicho dispositivo estaba basado en un equipo de RV, formado por dos pequeñas pantallas, una para cada ojo, y un sistema dedicado al seguimiento de la cabeza. Su principal limitación estuvo dada por la baja velocidad y precisión del sistema de seguimiento.

Desde el diseño de este primer sistema, el continuo avance de las tecnologías tanto de software como de hardware, ha permitido el desarrollo de cada vez más complejas y sofisticadas aplicaciones de RA y su uso en Educación. Tanto es así, que en [22] se recoge un incremento considerable en las investigaciones realizadas en este sector, desde una publicación en 1996 a más de 400 en el 2019.

Por otro lado, el surgimiento de las tecnologías de RM han constituido un paso de avance significativo en la forma de visualizar y relacionarse con el entorno real y los elementos virtuales. La interacción con los contenidos digitales de forma dinámica y en tiempo real proporcionada por la RM, ha constituido un aporte esencial para el desarrollo de aplicaciones orientadas a mejorar las experiencias en la enseñanza y el aprendizaje [4].

Con la aparición del dispositivo Microsoft HoloLens en el año 2016, se ampliaron las posibilidades de desarrollar aplicaciones basadas en RA y RM con elevado nivel de calidad y detalle en las representaciones virtuales. Desde ese momento, han sido llevadas a cabo múltiples in-

vestigaciones para evaluar e implementar sistemas de RA y RM usando las gafas HoloLens. En la Educación Superior se han destacado aplicaciones principalmente en las ramas de la Medicina, Construcción y Diseño e Ingeniería. A continuación presentaremos algunas de las aplicaciones desarrolladas en estas ramas en los últimos cinco años (2017-2022).

2.1.1 Ejemplos de aplicaciones

La Educación de Ciencias Médicas, ha sido el campo donde más investigaciones y aplicaciones basadas en RA y RM se han desarrollado. Esto se debe a la gran ventaja que ofrecen estas tecnologías en dicho sector, al posibilitar prescindir de estructuras biológicas reales para la enseñanza y la adquisición de conocimientos médicos, permitiendo a su vez, llegar a un mayor rango de precisión y escala en las representaciones virtuales. En este contexto se han visto involucradas disímiles especiales, entre las cuales resaltan la anatomía, la cirugía y la ortopedia [21].

En el 2018, [2] desarrolló un sistema híbrido de simulación para la cirugía abierta ortopédica empleando unas gafas HoloLens. El potencial del uso de RM en el sistema fue evaluado en simulaciones de cirugías de artroplastia de cadera mostrando resultados positivos en la carga de trabajo, rendimiento, percepciones visuales y auditivas e interacciones mediante gestos y voz. Lo cual conllevó a obtener mayores velocidades en los entrenamientos de los estudiantes en la tarea realizada. La Figura 2.1 muestra imágenes tomadas del funcionamiento del sistema en la cirugía evaluada.

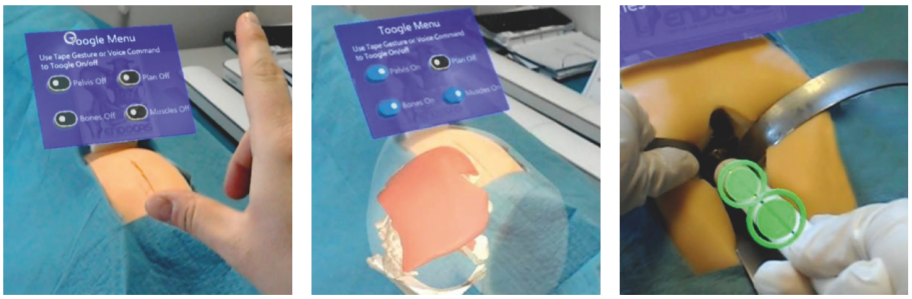


Figura 2.1: Imágenes tomadas del sistema híbrido de simulación para cirugía abierta ortopédica en una operación de artroplastia de cadera. Los contenidos virtuales y las partes físicas del simulador, fueron creados utilizando modelos anatómicos en 3D extraídos mediante tomografías computarizadas de un paciente [2]

En el 2019, la Escuela de Medicina de la Universidad Case Western Reserve, Estados Unidos, desarrolló una aplicación de anatomía holográfica llamada HoloAnatomy [3] (Figura 2.2). La aplicación está actualmente implementada en el plan de estudios de dicha universidad e incluye toda la anatomía que un estudiante de medicina debe aprender en el laboratorio de disección. Las representaciones holográficas de las bases anatómicas de cuerpos tanto femeninos como masculinos, permiten que la enseñanza se realice sin necesidad de un cadáver real. En su conjunto, se desarrolló una plataforma de enseñanza para para garantizar la facilidad de uso de la aplicación.

Figura 2.2: Ejemplo del empleo de la aplicación HoloAnatomy [3]

En [23] se evaluaron los efectos del aprendizaje en la docencia de fisiología y anatomía mediante tecnologías de RA (tabletas o teléfonos móviles) y RM (HoloLens). Para ello, se impartieron clases a un grupo de 38 estudiantes universitarios de medicina (Figura 2.3) y se realizaron pruebas antes y después de las clases para evaluar sus conocimientos. De los resultados obtenidos no se observaron diferencias significativas en las puntuaciones de las pruebas para la impartición de clases con HoloLens y con RA. Se concluyó que ambas tecnologías son eficaces en el aprendizaje, demostrando ser un apoyo para los educadores y los estudiantes de Ciencias Médicas.

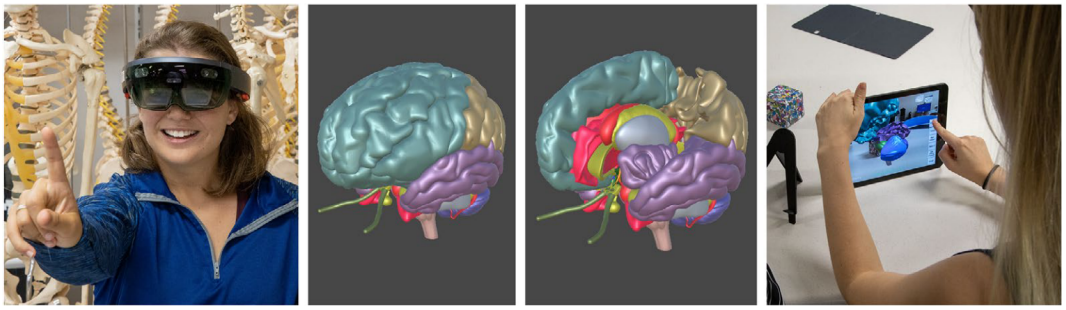


Figura 2.3: Imágenes de una clase sobre fisiología y anatomía del cerebro, usando HoloLens y tecnologías de RA [2]

En la rama de la Construcción y diseño, en [4] se creó una aplicación para la enseñanza de diseño de productos a estudiantes universitarios, empleando RM con HoloLens. La aplicación permite a los estudiantes visualizar la geometría de los objetos en 3D, así como diagramas que muestran el orden de ensamblaje de las partes que integran cualquier componente seleccionado. Los estudiantes pueden interactuar con el sistema a través de un manual de comandos o de la mirada, gestos y voz, para llevar a cabo las instrucciones. La Figura 2.4 muestra un ejemplo de interacción con el sistema. Resultados experimentales realizados mostraron que el diseño realizado influencia positivamente en la comprensión de los estudiantes en las relaciones geométricas y la creatividad.

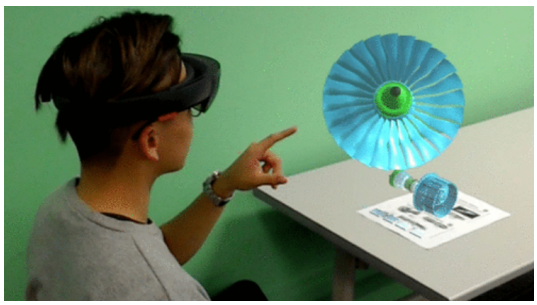


Figura 2.4: Ejemplo de interacción con la aplicación para la enseñanza de diseño de productos [4]

Además de las aplicaciones descritas, en la Tabla 2.1 se incluyen otras investigaciones realizadas empleando tecnologías de RA y RM en Educación Superior. De forma general, las herramientas mencionadas han sido diseñadas para propiciar la interacción de los estudiantes con tecnologías de visualización 3D, haciendo más viable el entendimiento de diversos conocimientos mediante el uso de representaciones principalmente holográficas. De tal manera, se elimina la necesidad de involucrar elementos reales en las tareas de aprendizaje, permitiendo interactuar con virtualizaciones que consiguen un elevado nivel de realismo y detalle. Lo cual ha demostrado mejorar la adquisición de habilidades por parte del alumnado en diferentes esferas y actividades [21].

Rama de estudio	Publicación	Año	Tecnología
Medicina	Desarrollo de un sistema de cirugía ortopédica abierta usando RM	2018	RM
	Un nuevo complemento para la disección de anatomía macroscópica: HoloAnatomy	2019	MR
	Evaluación de la utilidad de HoloLens en autopsias virtuales	2020	AR/MR
	Evaluación de los efectos de aprendizaje con RA en fisiología y anatomía	2020	AR/MR
	Simulación digital de operaciones quirúrgicas usando HoloLens	2020	MR
Construcción y diseño	Comprendiendo productos con RM: Relaciones geométricas y creatividad	2018	MR
	Propuesta de un novedoso framework de RM para diseño básico y su evaluación	2021	MR
	Entorno de RM para el aprendizaje de aplicaciones tecnológicas de medición en la construcción	2022	MR
Ingeniería	Uso de un simulador de RA en un curso presencial de Física	2017	AR
	Libros de RM: Aplicación de la RA y RV en la enseñanza de la Ingeniería de Minas	2020	VR/AR

Tabla 2.1: Aplicaciones de RA/RM en Educación Superior

2.2 Sistemas de telepresencia

De los sistemas de telepresencia desarrollados empleando tecnologías de RM como son las gafas HoloLens, el que más impacto ha tenido es **Holoportaion** [5]. Este sistema es capaz de capturar en 360° personas, objetos y movimientos dentro de una habitación, utilizando un conjunto de cámaras de profundidad. El contenido 3D es modelado, comprimido y transmitido a participantes remotos, realizando todo el proceso en tiempo. De tal forma, la plataforma permite visualizar, escuchar e interactuar entre personas en diferentes lugares con sensaciones cercanas a las reales, como si compartiesen un mismo espacio físico. La Figura 2.5 muestra algunos ejemplos de visualización y reconstrucción de modelos 3D logrados mediante Holoportation.

Figura 2.5: Ejemplos de visualización y modelos 3D en entornos recreados con Holoportation [5]

El pipeline desarrollado para el sistema Holoportation es indicado en la Figura 2.6. De la Figura se tiene que el primer paso realizado es el proceso de captura de 360° del escenario diseñado. Para ello, emplean ocho conjuntos de cámaras situadas en los límites de la habitación (Figura 2.7 izquierda). Cada conjunto (Figura 2.7 derecha) está formado por dos cámaras de Infrarrojo Cercano (NIR) y una cámara RGB. Además del empleo de un generador de patrones de luz estructurada pseudo-aleatorios (como el utilizado en Kinect V1).

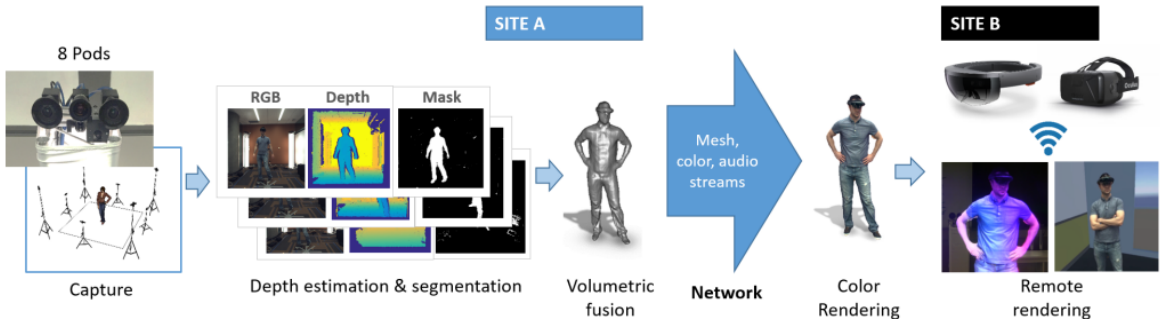


Figura 2.6: Pipeline desarrollado para el sistema Holoportation [5]

El segundo paso es la estimación de la profundidad mediante el cálculo de mallas de texturas a partir de la información de las dos cámaras NIR y los puntos proyectados por el generador de patrones. Además, se realiza la segmentación del fondo para proporcionar siluetas 2D con las regiones de interés (Region of Interest (ROI)).

Con la información obtenida se reconstruyen los modelos 3D y se fusionan conjuntamente con los datos de audio para ser transmitidos hacia el segundo entorno. Aquí, se dedican ordenadores de renderizado remoto (en el lado del receptor) para calcular las texturas a

Figura 2.7: Entorno recreado para el uso del sistema Holoportation con el posicionamiento de 8 conjuntos de cámaras estáticas (izquierda). Conjunto de cámaras utilizado en cada caso (derecha) [5]

proyectar y finalmente ser mostradas en dispositivos que pueden ser no solo de RM, sino también de RV.

Para lograr la velocidad y capacidad de cómputo requeridas en el correcto funcionamiento de esta tecnología y su ejecución en tiempo real, se emplearon varios sistemas de ordenadores con Graphics Processing Unit (GPU). En cada espacio de captura recreado, se utilizan cuatro ordenadores para el cálculo de la profundidad y la segmentación (un ordenador para dos conjuntos de cámaras). Además, los mapas de profundidad resultantes y las imágenes de color segmentadas son fusionados y transmitidos mediante el empleo de otro ordenador con GPU, requiriendo un elevado ancho de banda en la comunicación entre las salas (conexión Ethernet de 10 Gigabits).

Si bien se han realizado actualizaciones en este sistema para mejorar la reconstrucción y compresión de los modelos, su transmisión y disminuir el ancho de banda, como las propuestas en [24], actualmente Holoportation continúa siendo un sistema que requiere una elevada cantidad de hardware de alta gama para poder funcionar.

En el sector de Educación Superior, más concretamente, en el uso de entornos de RM para desarrollar sistemas de educación que integren en un espacio común la presencialidad (estudiantes en remoto) con la no presencialidad (estudiantes en el aula), Holoportation muestra las ventajas de la comunicación basada en telepresencia. Sin embargo, la necesidad del empleo de múltiples conjuntos de cámaras de profundidad y ordenadores con GPU en la sala donde se encuentra el usuario a transportar, así como toda la preparación y acondicionamiento de la sala en sí, limita considerablemente el uso de esta tecnología, siendo una solución prácticamente inviable para su uso cotidiano y por ende, para los estudiantes en casa.

En este sentido, se debe pensar en una solución que posibilite su implementación con el menor número de requerimientos de hardware necesarios, orientada hacia el desarrollo de sistemas de telepresencia en el ámbito educacional. De manera que permita aumentar su uso y la posibilidad de una nueva forma de interactuar con la realidad.

La idea que se pretende con el presente trabajo es proporcionar las bases para un sistema de telepresencia que pueda ser empleado por múltiples usuarios simultáneamente, con el menor consumo de recursos de hardware posible. Para ello, proponemos una aplicación basada en RM y HoloLens 2, que represente una opción más asequible a los estudiantes en casa mediante el empleo de una única y simple webCam.

3 Objetivos

Este trabajo tiene como objetivo general diseñar un sistema que permita una visualización donde los límites de lo presencial y lo no presencial se diluyan, dando lugar a una nueva transversalidad de lo presencial en el entorno de la Educación Superior. Para ello se definen los objetivos específicos siguientes:

- Creación de un sistema y metodología de cyberpresencia basado en hologramas
- Aplicación de dicho sistema al entorno educativo
- Evaluación cualitativa y cuantitativa del sistema propuesto en entornos reales
- Proponer, desarrollar, evaluar e integrar un sistema que reconozca las emociones de los alumnos y se las muestre al profesor

La motivación personal detrás de este trabajo incluye las siguientes razones:

- Dominar, tanto a nivel teórico como práctico, técnicas de Realidad Mixta (RM), Inteligencia Artificial (IA) y Visión por Computador.
- Desarrollar habilidades en el diseño de entornos de RM.
- Adquirir destrezas para el desarrollo de herramientas de software con un entorno de RM para las gafas Hololens 2 de Microsoft.

4 Metodología

En este Capítulo se abordan temas sobre la metodología de los experimentos, presentando las herramientas y software utilizados en la realización del trabajo. Para ello, primeramente se describe el dispositivo, software y paquetes de RM empleados. Posteriormente, se presenta el framework usado en la comunicación web implementada para el funcionamiento de la aplicación. Por último, se describen las diversas tecnologías empleadas para realizar las tareas de clasificación de emociones y segmentación de personas desarrolladas en el proyecto.

4.1 HoloLens 2 de Microsoft

Las gafas HoloLens 2 (Figura 4.1) de Microsoft son una MRITF muy potente que ofrece una experiencia inmersiva de gran comodidad para el usuario. Este dispositivo posibilita el uso de gestos para interactuar de forma natural con hologramas y su visor está diseñado especialmente al empleo de los mismos. Además, posee un sensor involucrado en el seguimiento de la mirada, permitiendo que la interacción se realice de manera aún más natural. A la vez, presenta un amplio ángulo de visión y gran resolución de imagen. Gracias a los comandos por voz y a la elevada ergonomía de las HoloLens 2, puede utilizarse durante varias horas sin inconvenientes [25].

Figura 4.1: HoloLens 2 de Microsoft [1]

El dispositivo permite incorporar las capacidades y beneficios de las soluciones basadas en RM, posibilitando además el desarrollo de capacidades basadas en Visión por Computador e IA. Esto beneficia su empleo en diversos escenarios como: Fabricación, Ingeniería y construcción, Atención sanitaria y Educación. La Tabla 4.1 muestra las principales especificaciones técnicas de las gafas HoloLens 2 [26].

Pantalla		
	Óptica	Lentes holográficas transparentes (guías de ondas)
	Resolución	2.000 motores de luz 3:2
	Densidad holográfica	>2.500 radiantes (puntos de luz por radián)
	Representación basada en los ojos	Optimización de la pantalla para la posición 3D del ojo
Sensores		
	Seguimiento de la cabeza	4 cámaras de luz visibles
	Seguimiento de los ojos	2 cámaras de infrarrojos
	Profundidad	Sensor de profundidad de tiempo de vuelo de 1 MP
	Unidad de medición inercial (IMU)	Acelerómetro, giroscopio, magnetómetro
	Cámara	Imágenes fijas de 8 MP, vídeo 1080p30
Audio y voz		
	Matriz de micrófonos	5 canales
	Altavoces	Sonido espacial integrado
Potencia		
	Duración de la batería	2-3 horas de uso activo, hasta 2 semanas en modo de espera
	Tecnología de la batería	Baterías de litio
	Comportamiento de carga	Totalmente funcional mientras carga
Software		
	Sistema operativo holográfico de Windows	

Tabla 4.1: Especificaciones técnicas de las gafas Hololens 2 de Microsoft

4.2 Unity

Unity es una herramienta de desarrollo multiplataforma enfocada a la creación de juegos y experiencias interactivas en dos- y tres-dimensiones (2D y 3D), RV y RA. Este motor de videojuegos fundado por Unity Technologies, permite ser utilizado en los sistemas operativos Microsoft Windows, Linux y Mac OS [27]. La Figura 4.2 muestra un ejemplo de visualización del entorno Unity 2020.3 LTS en Windows 10.

Las aplicaciones creadas empleando Unity presentan compatibilidad con plataformas Web, PC (Windows, Linux, OS X), dispositivos móviles, smart TV y consolas (Wii U, PlayStation 4, Xbox One, etc.). Además, soporta dispositivos de RV y RM como las gafas Oculus, PlayStation VR, Microsoft HoloLens, entre otros [27, 28]. El scripting se realiza mediante una implementación de código abierto de .NET Framework, soportando los lenguajes de programación UnityScript, C# o Boo [27, 28].

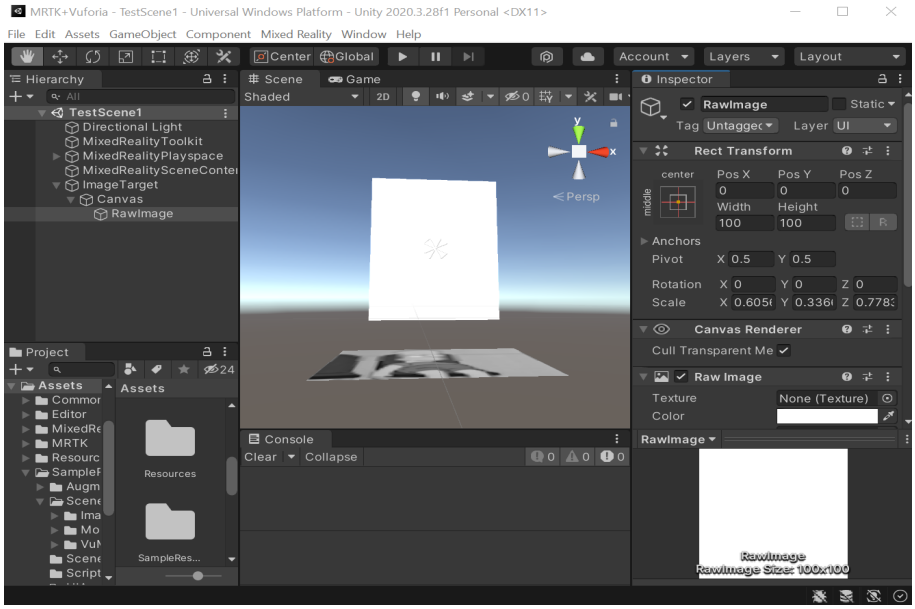


Figura 4.2: Entorno Unity 2020.3 LTS

Unity puede usarse conjuntamente con otros softwares de diseño 2D y 3D, ofreciendo gran comodidad y facilidad en el diseño, modelado y animación. Entre sus principales características resalta el soporte para mapeado de relieve y reflejos, el mapeado por paralaje y la creación de sombreadores mediante el lenguaje ShaderLab. Los sombreadores son programas informáticos que realizan cálculos gráficos a la vez que interaccionan con la unidad de procesamiento gráfico (GPU, del inglés *Graphics Processing Unit*). De esta forma permiten reducir el coste computacional y el esfuerzo del ordenador en tareas de transformaciones de vértices o coloreado de píxeles.

El motor gráfico Unity presenta además soporte para oclusión ambiental en espacio de pantalla, render a textura y efectos de post-procesamiento de pantalla completa. También incluye soporte integrado para Nvidia, permitiendo en tiempo real la creación y manipulación de mallas arbitrarias y sin piel, así como capas de colisión [27, 28]. La tecnología de animación empleada es Mecanim. Con ello, desde el propio editor de Unity es posible utilizar diversas herramientas de animación como la creación de máquinas de estados, árboles de mezcla y el retargeting automático de animaciones [27].

4.2.1 MRTK Unity

MRTK (Mixed Reality Toolkit) Unity es un kit de herramientas multiplataforma y de código abierto diseñado por Microsoft, enfocado al desarrollo de aplicaciones de RM en Unity. Este conjunto de herramientas pretende acelerar el proceso de creación de aplicaciones de RM en dispositivos como Microsoft HoloLens 1 y 2, Cascos de Windows Mixed Reality y Cascos de OpenVR (HTC Naopak/Oculus Rift). También, presenta soporte para dispositivos iOS y Android. MRTK proporciona bloques de creación básicos para las diversas plataformas a las

que ofrece soporte. Además, permite el desarrollo rápido de prototipos mediante simulación desde el propio editor [29]. Un ejemplo de aplicación de RM empleando MRTK se muestra en la Figura 4.3.

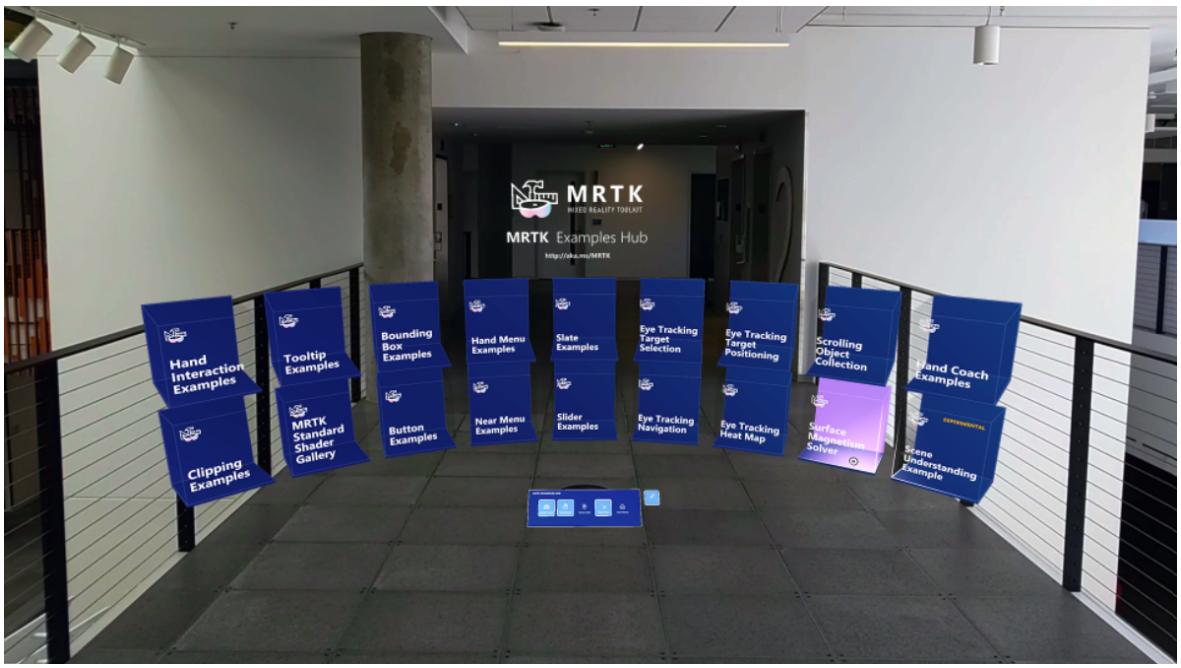


Figura 4.3: Ejemplo de RM empleando MRTK [6]

Para poder utilizar el MRTK en un proyecto de Unity, es necesario importarlo y configurarlo mediante la herramienta de características de Mixed Reality. Esta herramienta permite a los desarrolladores detectar, actualizar y agregar paquetes de características de RM. Para ello, antes de realizar la importación, posibilita la búsqueda de paquetes por nombre o categoría, consultar sus dependencias y ver los cambios propuestos en el archivo de manifiesto de sus proyectos [29].

Entre los posibles paquetes a importar, el paquete **Mixed Reality Toolkit Foundation** es imprescindible para usar MRTK en un proyecto. Este permite incluir los componentes principales necesarios para desarrollar una aplicación de RM, posibilitando al espacio de juego administrar la cámara simultáneamente con los **SDK de Windows 10**. Los SDK de Windows 10 son requeridos para la creación de vistas envolventes en Unity y su compatibilidad con Windows Mixed Reality.

4.2.2 Mixed Reality OpenXR Plugin

Mixed Reality OpenXR Plugin es un complemento en Unity, añadido mediante el MRTK, que se recomienda para el desarrollo de aplicaciones de RM en HoloLens 2. **OpenXR** es un estándar abierto y libre de costo creado por Khronos que permite a motores como Unity, tener acceso a las funciones nativas de diversas plataformas de RV, RA y RM. De esta

forma, OpenXR posibilita el desarrollo de aplicaciones en dispositivos XR, tanto holográficos como envolventes, sin necesidad de reimplementar/reescribir código o de tener en cuenta su proveedor [30, 7].

La Figura 4.4 muestra la estructura OpenXR con acceso multiplataforma y de alto rendimiento a diferentes dispositivos con soporte en la API OpenXR. Gracias a esta estructura se elimina el problema de que aplicaciones y motores de RV y RA tengan que emplear APIs propias de cada plataforma para manejar sus funcionalidades [7]. Mixed Reality OpenXR permite acceso a prestaciones de HoloLens 2 tales como: predicción de posición principal, tiempo de fotogramas, seguimiento de manos articulado, seguimiento de los ojos, asignación espacial y anclajes espaciales [30].

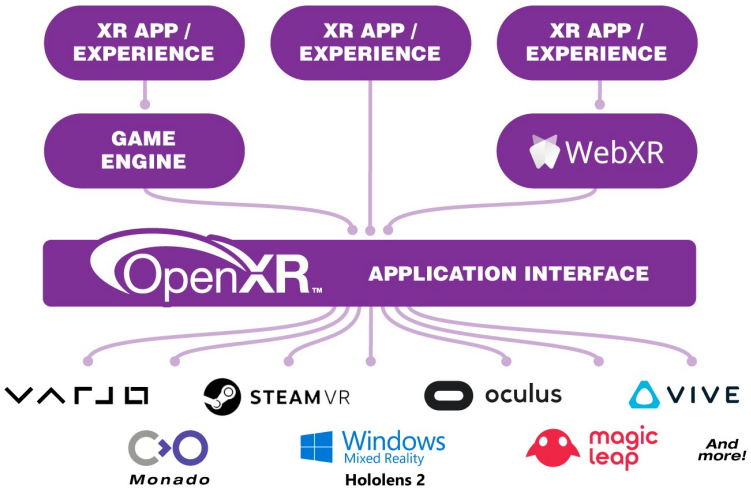


Figura 4.4: Estructura OpenXR con acceso multiplataforma y de alto rendimiento entre las aplicaciones y dispositivos XR [7]

4.3 Vuforia Engine

Vuforia Engine es una plataforma diseñada para el desarrollo de aplicaciones de RA y RM en múltiples dispositivos (teléfonos móviles, tabletas, gafas de RA/RM, etc.), que permite el reconocimiento y seguimiento robusto de imágenes 2D y objetos 3D en tiempo real. Estas imágenes u objetos se conocen como marcadores o destinos y su utilidad en Vuforia es la siguiente: una vez que la cámara del dispositivo utilizado reconoce en el mundo real un destino previamente definido en la aplicación desarrollada, esta visualiza un contenido virtual sobre la posición del destino en el entorno mixto [8]. La Figura 4.5 muestra algunos ejemplos de destinos en Vuforia.

Vuforia permite además el seguimiento sin marcadores para incorporar el contenido virtual mediante el empleo de una ubicación o posición. Posteriormente, el hardware y software utilizados en la aplicación, analizan el contenido virtual añadido como si estuviese anclado a una ubicación u objeto determinado del mundo real [8].

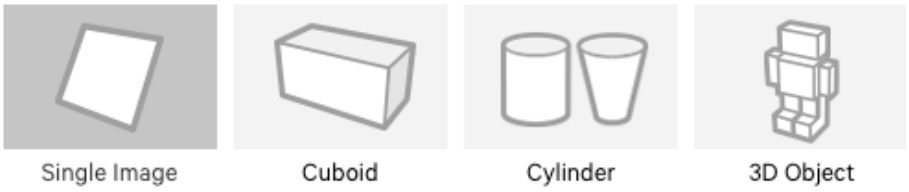


Figura 4.5: Ejemplos de marcadores o destinos en Vuforia [8]

Para desarrollar aplicaciones en HoloLens 2 empleando Vuforia Engine es posible utilizar Unity. Con la configuración adecuada (proporcionada en [31]), podemos hacer que Vuforia Engine funcione con los sistemas HoloLens asignación espacial y seguimiento posicional. Una vez que se tiene acceso a estos sistemas, Unity activa automáticamente el proceso **Device Tracking** en las aplicaciones de HoloLens. Este proceso proporciona el seguimiento de un destino incluso cuando este no está a la vista, combinando los seguimientos con y sin marcador descritos anteriormente. Con ello se consigue posicionar elementos virtuales de manera estable, independientemente de si la cámara está viendo o no el destino definido [31].

Finalmente, se debe tener en cuenta que para crear una aplicación propia con Vuforia y HoloLens es necesario registrarse en el **Vuforia Developer Portal**, el cual se puede encontrar en [32]. Mediante el portal es posible obtener una licencia de desarrollador gratuita y crear bases de datos de destinos necesarias en las aplicaciones a desarrollar [31].

En este proyecto, se ha empleado Vuforia para posicionar en tiempo real imágenes segmentadas del estudiante en una ubicación determinada mediante un patrón de imagen 2D como destino.

4.4 Flask

Flask es un framework minimalista y open-source que permite desarrollar aplicaciones web de forma rápida y sencilla mediante código en Python. Con el término minimalista no hace referencia a la creación de un proyecto necesariamente pequeño o con pocas líneas de código, sino más bien a que Flask ofrece por defecto un conjunto de funcionalidades básicas para la realización de una aplicación web inicial, pero a su vez, permite que nuevas funcionalidades puedan ser añadidas fácilmente a través de una gran variedad de extensiones disponibles. En esencia, Flask es un framework cuya idea es proporcionar una plataforma para el desarrollo de aplicaciones web de manera simple pero además extensible. Para ver todas las funcionalidades de Flask puede dirigirse a su página oficial en [33].

En este proyecto, la herramienta Flask se emplea para crear un sistema de comunicación web entre el ordenador del estudiante, las gafas HoloLens 2 y un ordenador con GPU que se encarga de procesar la información requerida por las gafas. De esta forma, Flask permite el envío y recepción de información de tipo imagen, texto, etc., entre los diferentes sistemas de hardware que integran el desarrollo del proyecto presentado.

4.5 Detección de rostros

En esta Sección se introducen tres algoritmos de detección de rostros empleados durante el desarrollo del proyecto. La detección de rostros ha sido implementada en una etapa de preprocesamiento de imágenes utilizadas en la creación de la tarea clasificación de emociones.

4.5.1 Viola and Jones

El método **Viola and Jones** (**V&J**)[34], también conocido como **Haar Cascades**, es un algoritmo de detección basado en apariencia. La aproximación emplea técnicas de aprendizaje estadístico que permiten construir un clasificador "rostro/no-rostro" mediante la búsqueda de patrones de luz y sombra que se asemejen a un rostro humano. Para la implementación de este algoritmo se ha empleado un modelo de detección de rostro frontal ofrecido por el módulo **OpenCV** [35].

4.5.2 Histogram of Oriented Gradient

El algoritmo Histograma de Gradientes Orientados (**HOG** por sus siglas del inglés **Histogram Oriented Gradient**) caracteriza la apariencia local del rostro mediante la distribución de los gradientes locales de intensidad y las direcciones de los bordes en la imagen [36]. Su implementación se realizó empleando el módulo **dlib** [37].

4.5.3 Single Shoot Detector

El detector **SSD** [38], por sus siglas del inglés **Single Shoot Detector**, está formado por una red neuronal profunda que realiza la tarea de localización y clasificación de objetos en un único paso hacia delante, garantizando eficiencia y alta precisión. Este fue implementado mediante modelos pre-entrenados basados en la estructra de red nueronal profunda Caffe ofrecidos por la biblioteca dnn, la cual está incluida en el módulo OpenCV.

4.6 Alineación de rostros

La alineación de rostros consiste en un proceso de dos pasos: 1) identificar la estructura geométrica de los rostros presentes en imágenes digitales y, 2) intentar obtener una alineación canónica de los rostros analizados, basada en transformaciones afines como la traslación, la escala y la rotación (Figura 4.6).

La implementación de dicha alineación suele ser un paso fundamental en el preprocesamiento de imágenes para diversas tareas de análisis facial, como el reconocimiento de rostros o la clasificación de emociones [39]. Con ello, los rostros utilizados en el entrenamiento de los modelos son previamente alineados, obteniéndose una forma de normalización donde todas las imágenes presentan una disposición similar. De esta manera, se consiguen resultados más estables y con mayor precisión.

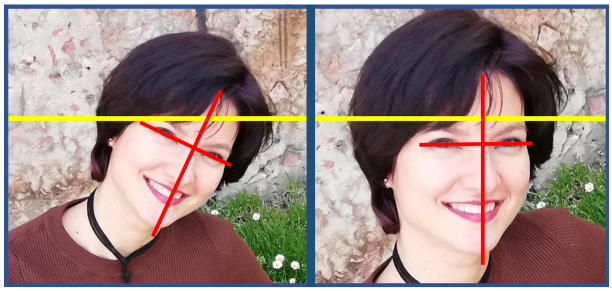


Figura 4.6: Alineación de rostros en una imagen

La metodología tradicional para realizar la alineación se enfoca en encontrar determinados puntos de referencia del rostro, como son el ángulo de los ojos, la punta de la nariz y el centro de la barbilla, mediante los cuales es posible normalizar la imagen del rostro (representación normalizada de la rotación, traslación y escala del rostro) [39]. Otros métodos más complejos buscan imponer un modelo 3D predefinido y aplicar una transformación a la imagen de entrada de manera que los puntos de referencia del rostro en dicha imagen coincidan con los puntos de referencia del modelo 3D. En la próxima Sección (4.6.1) se aborda en mayor detalle la detección de puntos de referencia del rostro.

4.6.1 Detección de puntos de referencia del rostro

La detección de puntos de referencia del rostro se basa en localizar puntos característicos de un rostro dada una región de interés (Region of Interest (ROI)) que especifica el rostro a analizar en una imagen. Para ello, primeramente es necesario detectar el rostro en la imagen y posteriormente, detectar estructuras faciales importantes en dicho rostro, mediante el empleo métodos de predicción de forma.

De forma general, los detectores de puntos de referencia del rostro, se enfocan en localizar e identificar las regiones faciales siguientes:

- Ceja derecha
- Ceja izquierda
- Ojo derecho
- Ojo izquierdo
- Nariz
- Boca
- Mandíbula

En nuestro caso, hemos empleado la detección de estos puntos de referencia, como parte del preprocesamiento de las imágenes usadas en la clasificación de emociones. Específicamente, los puntos detectados han sido utilizados en el proceso de alineación de los rostros encontrados mediante los algoritmos de detección descritos en la Sección 4.5.

Concretamente, se ha empleado el predictor de puntos característicos proporcionado por

el módulo dlib, el cual constituye una implementación del método desarrollado en [40]. Este método se basa en utilizar un conjunto de datos de entrenamiento de puntos de referencia de rostros en imágenes, donde las etiquetas especifican en coordenadas *(x, y)*, las regiones que definen cada punto característico de la estructura del rostro. A partir de esos datos, se entrena un conjunto de árboles de regresión con el objetivo de estimar las posiciones de los puntos de referencia del rostro directamente de las propias intensidades de los píxeles. De esta forma no se realiza ningún proceso de extracción de características, obteniéndose un algoritmo que funciona en tiempo real con predicciones de alta calidad [40].

Para la realización de este proyecto se ha utilizado un modelo de 68 puntos de referencia preentrenado y ofrecido por dlib. A continuación se introduce brevemente dicho modelo.

4.6.2 Modelo preentrenado de 68 puntos de referencia

El modelo preentrenado de 68 puntos de referencia dentro de la biblioteca dlib [9], se emplea para estimar con gran velocidad la ubicación de 68 coordenadas *(x, y)* correspondientes a estructuras faciales de un rostro como las indicadas en la Figura 4.7 (Izquierda).

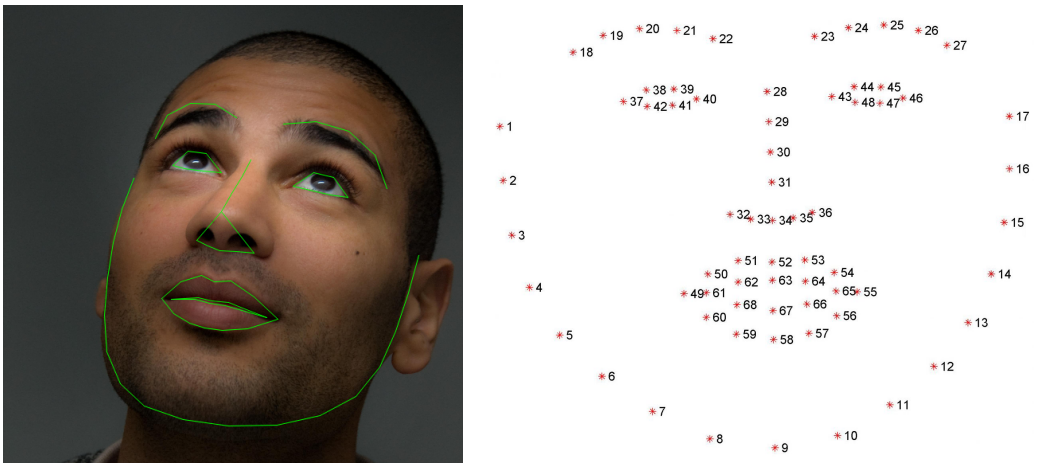


Figura 4.7: **Izquierda**: Estructuras faciales de un rostro definidas mediante el modelo pre-entrenado de 68 puntos de referencia en dlib [9]. **Derecha**: Representación de los 68 puntos de referencia del rostro del conjunto de datos iBUG 300-W [10]

La Figura 4.7 (Derecha) muestra los índices de las 68 coordenadas definidas. Estas anotaciones forman parte de la base de datos **iBUG 300-W** [10] de 68 puntos de referencia, sobre la cual fue entrenado el predictor de puntos faciales de dlib.

4.6.3 Alineación de rostros con imutils

El módulo **imutils** [41] implementado en Python, utiliza OpenCV y el modelo de 68 puntos de referencia del rostro preentrenado en dlib para realizar la alineación de rostros en imágenes digitales. Para lograrlo, el módulo permite definir al usuario el ancho y alto deseado de la ROI que contiene al rostro resultante de la alineación. Además, posibilita definir la posición

del ojo izquierdo en la imagen de salida mediante una tupla (x, y). Esta tupla contiene valores normalizados de 0 a 1 (equivalentes a 0-100 %), con los cuales se controla qué tanto del rostro finalmente alineado será visible en la imagen de salida. El rango común de valores empleados se encuentra entre el 20 y 40 %, donde valores menores corresponden a una vista ampliada del rostro y valores mayores producen una vista más alejada [11].

Posteriormente, en imutils se utiliza el modelo de 68 puntos de referencia mediante el módulo dlib y se extraen los puntos correspondientes a los ojos izquierdo y derecho. De ahí, se calculan sus medias para obtener el centro de cada ojo y con ello calcular el ángulo formado entre la línea que pasa por los dos centroides y la horizontal.

Luego, se calculan las coordenadas correspondientes a la posición deseada para el ojo derecho en la imagen de salida, además de la escala del rostro, en función de la posición definida por el usuario para el ojo izquierdo. La posición deseada del ojo derecho se obtiene basándose en la coordenada x del ojo izquierdo. Para ello, se le resta a 1 el valor correspondiente a dicha coordenada. Esto se realiza para obtener valores de coordenadas en x equidistantes de los bordes izquierdo y derecho para cada ojo, respectivamente.

La escala del rostro en la imagen resultante se determina mediante la relación dada por la distancia euclidiana entre los ojos en la imagen de entrada y la distancia entre los ojos en la imagen deseada. En el cálculo de la segunda distancia, definida como *distancia deseada*, se obtiene la diferencia entre los valores deseados en x de los ojos derecho e izquierdo y se multiplica por el ancho deseado de la imagen de salida. Esta multiplicación se realiza para escalar la distancia deseada entre los ojos, basada en el ancho deseado. Con ello, la escala del rostro se obtiene dividiendo la distancia deseada entre la distancia euclidiana [11].

Ahora, se calcula el punto medio entre los ojos y se obtiene una matriz de transformación M sobre dicho punto, empleando el ángulo y la escala determinados anteriormente. Luego, se actualiza el componente de traslación en la matriz M en función del tamaño deseado de la imagen de salida y el punto medio entre los ojos. Esta actualización está dada por las siguientes ecuaciones:

$$t_w^* = t_w + (t_x - p_c(x)), \tag{4.1}$$

$$t_h^* = t_h + (t_y - p_c(y)), \tag{4.2}$$

donde t_x es un valor de traslación en la dirección x calculado como la mitad del ancho deseado del rostro; t_y es un valor de traslación en la dirección y obtenido mediante la multiplicación de la altura deseada del rostro por el valor deseado en y del ojo izquierdo. Los valores $p_c(x)$ y $p_c(y)$ corresponden a las coordenadas $(x,\ y)$ del punto medio entre los ojos, respectivamente. Por último, t_w y t_h son las traslaciones en las direcciones x e y originalmente obtenidas en la matriz M, guardándose en t_w^* y t_h^* el resultado de su actualización.

Finalmente, para obtener el rostro alineado en la ROI de salida, se realiza una transformación afín que emplea la matriz de transformación M y el tamaño de imagen de salida deseado por el usuario. En la Figura 4.8 se muestran dos ejemplos de resultados de alineación de rostros obtenidos mediante el módulo dlib.

El método de alineación de rostros proporcionado por dlib ha sido utilizado en este proyecto en el preprocesamiento de los datos empleados por diferentes modelos de clasificación

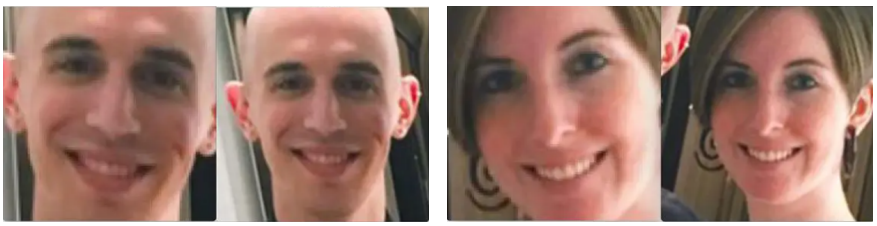


Figura 4.8: Ejemplos de alineación de rostros empleando dlib [11]

de emociones. Posteriormente, en el Capítulo de Experimentación, los resultados obtenidos serán comparados con los generados al utilizar un algoritmo de alineación presentado en el Desarrollo del trabajo (Sección 5.6.3).

4.7 Clasificación de emociones

En esta Sección se describen diversas tecnologías empleadas para desarrollar la tarea de clasificación de emociones en la aplicación desarrollada.

4.7.1 Convolutional Neural Network

Una red neuronal convolucional o **Convolutional Neural Network** (**CNN**) es un tipo de red neuronal artificial cuya arquitectura se basa en múltiples capas de filtros convolucionales de una o más dimensiones. Esta red constituye una variación de un perceptrón multicapa y su aplicación principal ha estado enfocada al análisis de imágenes. Sin embargo, también puede ser aplicada a la clasificación de series de tiempo, señales de audio (procesamiento del lenguaje natural) o datos volumétricos [42].

A modo general, una CNN se caracteriza por dos fases: la primera se encarga de la extracción de características y la segunda de la clasificación. La primera fase está compuesta por capas alternas de neuronas convolucionales y de reducción de muestreo. A medida que los datos avanzan entre las capas, disminuyen su dimensionalidad, siendo las neuronas menos afectadas por perturbaciones en los datos de entrada y activadas por características más complejas. Al final de la red se tienen neuronas de perceptrón sencillas que se encargan de realizar la fase de clasificación.

En el presente trabajo se emplea un modelo de CNN para desarrollar la tarea de clasificación de emociones en imágenes. Además, se utiliza la primera fase de dicho modelo para extraer características de las imágenes a clasificar. Estas características se extraen con el objetivo de ser utilizadas por el modelo de aprendizaje automático K-Nearest Neighbor (Sección 4.7.2) y realizar la clasificación. De esta forma es posible comparar los resultados obtenidos por las diferentes implementaciones y emplear la de mayor precisión en la aplicación desarrollada. Para la implementación del modelo CNN se han empleado en Python los frameworks Tensorflow y Keras descritos en las secciones 4.7.4 y 4.7.5, respectivamente.

4.7.2 *K*-Nearest Neighbor

El algoritmo ***K*-Nearest Neighbor** (***K*-NN**) es un método de aprendizaje supervisado no paramétrico que se aplica tanto a problemas de clasificación como de regresión. La idea en K-NN es que dado un conjunto de muestras previamente etiquetadas, un nuevo ejemplo será clasificado (tarea de clasificación) con la clase que más se repita entre los k vecinos más cercanos a el ejemplo dentro del conjunto conocido. La Figura 4.9 muestra un ejemplo del procedimiento descrito. En el caso de regresión, la salida del algoritmo será el promedio de los valores de los k vecinos más cercanos [43]. El valor de k es un número entero, comúnmente pequeño e impar.

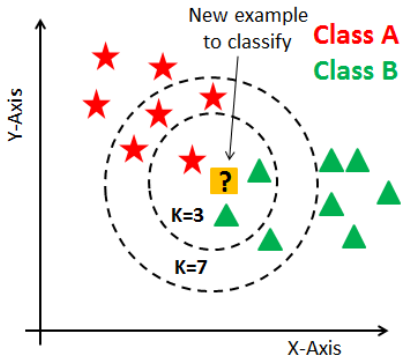


Figura 4.9: Ejemplo de funcionamiento del algoritmo para un problema de clasificación K-NN [12]. En este caso, para $k = 3$ el resultado corresponde a la clase A debido a que presenta más vecinos de esa clase, mientras que para $k = 7$ la salida corresponde a la clase B

En la etapa del algoritmo conocida como entrenamiento, solamente se almacenan los vectores de características y las etiquetas del conjunto de muestras conocidas (datos de entrenamiento). En la etapa de clasificación se emplea una métrica de distancia y se suele asignar un peso a la contribución de los vecinos, teniendo mayor peso a menor distancia del ejemplo a clasificar.

En este proyecto se emplea el algoritmo K-NN para realizar pruebas en la tarea de clasificación de emociones. La entrada al sistema será un vector de características de un rostro, obtenido mediante la arquitectura de red CNN desarrolla.

4.7.3 Principal Component Analysis

El algoritmo **Principal Component Analysis** (**PCA**) [44] es un método estadístico empleado principalmente para reducir la dimensionalidad de un conjunto de datos. Esta aproximación permite simplificar la complejidad de espacios muestrales de grandes dimensiones, pretendiendo conservar su información. La Figura 4.10 muestra un ejemplo de resultado de aplicar PCA a un conjunto de datos de 2 dimensiones.

PCA convierte un conjunto de datos cuyas características presentan una determinada correlación, en un nuevo conjunto de características sin correlación lineal a las que se les denomina

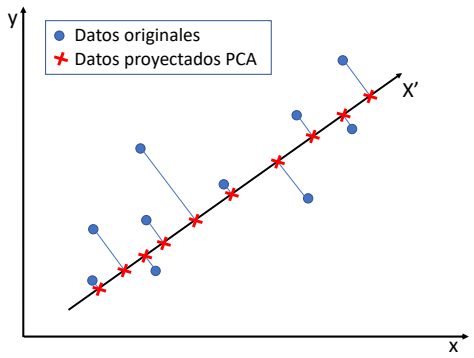


Figura 4.10: Algoritmo PCA aplicado a un conjunto de datos de 2 dimensiones

componentes principales. Los primeros componentes principales contienen la mayor parte de la varianza de los datos analizados, por lo cual, se considera que a medida que se avanza en dichos componentes, la información sobre la distribución de los datos es menos importante para su caracterización. Esto se debe a que el algoritmo PCA construye una transformación lineal que selecciona un nuevo sistema de coordenadas para el conjunto original de datos donde la varianza de mayor tamaño es representada por el primer eje principal.

Este método presenta el inconveniente de que asume que los datos analizados presentan combinaciones lineales de una base determinada. Por lo cual, en datos cuya distribución contenta una elevada correlación, el algoritmo PCA no lograría caracterizar correctamente el conjunto de datos en menos dimensiones. En el presente proyecto el método PCA se emplea para realizar una prueba y analizar los resultados obtenidos al reducir la dimensionalidad de los vectores de características producidos por el modelo de CNN y utilizados por el algoritmo K-NN.

4.7.4 TensorFlow

TensorFlow es un framework para aprendizaje automático de Google, liberado como código abierto bajo la licencia Apache 2.0 en noviembre del 2015. TensorTlow presenta una estructura flexible que permite su ejecución en múltiples CPUs y GPUs, siendo además multiplataforma con soporte en Windows, Linux, macOS y sistemas Android e iOS. La implementación de TensorFlow está hecha sobre C++ con interfaz en C++, Python, Java, entre otros lenguajes, permitiendo su uso en diversos entornos de programación [45, 46].

4.7.5 Keras

Keras es una biblioteca de código abierto escrita en Python y diseñada para el trabajo con redes neuronales. Su estructura se centra en ofrecer la posibilidad de desarrollar de forma sencilla modelos de aprendizaje profundo sin que el usuario tenga que preocuparse directamente del backend computacional empleado. Este backend de procesamiento a bajo nivel suele ser un framework como TensorFlow [47, 48].

4.7.6 Base de datos AffectNet

AffectNet es una base de datos de expresiones faciales que contiene más de 1 millón de imágenes de rostros provenientes de Internet. Las muestras fueron obtenidas mediante la consulta a tres importantes motores de búsqueda online (Google, Bing y Yahoo), utilizando 1.250 palabras claves relacionadas con diferentes emociones en seis idiomas diferentes [49]. De tal forma, AffectNet logró ser la mayor base de datos de modelos categóricos y dimensionales de emociones con imágenes no tomadas en entornos de laboratorio. La Figura 4.11 muestra algunos ejemplos de datos contenidos en AffectNet.

Figura 4.11: Ejemplos de muestras en la base de datos AffectNet

Los modelos categóricos mencionados anteriormente hacen referencia a la clasificación de una emoción elegida de una lista de categorías definidas, mientras que los dimensionales realizan la predicción de un valor en una escala emocional continua. Con ello, AffectNet permite realizar tareas de reconocimiento de emociones empleando modelos tanto discretos como continuos.

Las anotaciones en esta base de datos fueron realizadas de la siguiente manera: aproximadamente la mitad de las imágenes guardadas fueron etiquetadas manualmente por doce expertos humanos. Las restantes anotaciones se determinaron con los resultados obtenidos mediante dos redes neuronales profundas para la clasificación (modelo categórico) y la predicción (modelo dimensional) [49].

En el presente trabajo se utiliza la base de datos AffectNet y sus anotaciones categóricas para realizar el entrenamiento y testeo de modelos de clasificación de emociones desarrollados para la aplicación de RM.

4.8 Mask R-CNN

Mask R-CNN es un framework para segmentación de instancias basado en la estructura de detección de objetos Faster R-CNN [50]. La segmentación de instancias constituye un problema desafiante ya que requiere la correcta detección de todos los objetos en la imagen y la segmentación precisa de cada instancia. De esta forma, integra las tareas de detección y

localización de objetos conjuntamente con la de segmentación semántica [13].

La arquitectura Mask R-CNN amplía Faster R-CNN añadiendo una rama para la predicción de máscaras de segmentación en cada región de interés (RoI), en paralelo con la rama existente para la clasificación y la proposición de los bounding boxes (Figura 4.12). La rama de la máscara es definida como una pequeña estructura Fully Convolutional Network (FCN) [51] aplicada a cada RoI, de manera que realiza la predicción de la máscara de segmentación de píxel a píxel. Por tanto, ahora la salida de Mask R-CNN genera no solo las clases y los bounding boxes predichos mediante Faster R-CNN, sino que además produce una máscara binaria para cada RoI [13].

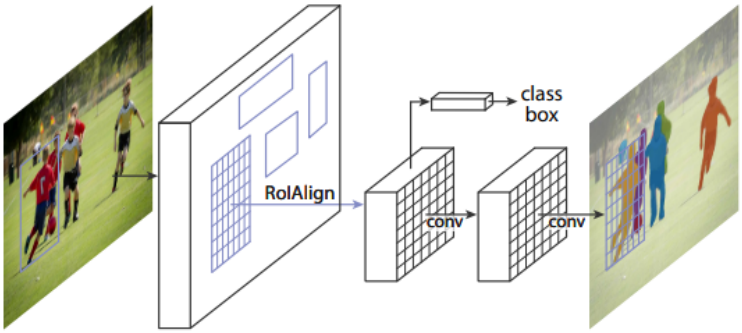


Figura 4.12: Framework Mask R-CNN para segmentación de instancias [13]

Mask R-CNN ofrece rápidos resultados gracias a la velocidad de la estructura Faster R-CNN y a que la la rama de la máscara añadida adiciona poca carga computacional. Los resultados de segmentación presentados por Mask R-CNN en el 2017 superaron todos los anteriores del estado del arte para modelos del tipo *single-model* sobre el dataset COCO. La Figura 4.13 muestra resultados de Mask R-CNN obtenidos sobre datos de test de COCO [13].

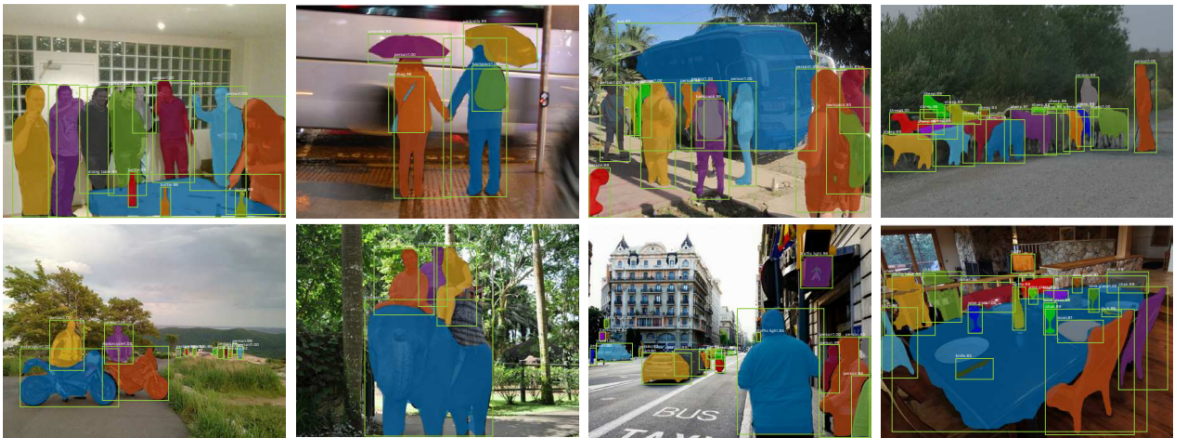


Figura 4.13: Ejemplos de resultados obtenidos con Mask R-CNN sobre datos de test del dataset COCO. Los resultados están basados en ResNet-101, obteniendo una métrica AP (Averaged over IoU thresholds) de 35.7 con un tiempo de ejecución igual a 5 fps [13]

En el proyecto se emplea Mask R-CNN para realizar la segmentación de la persona (estudiante) que se encuentra frente a la webcam y poder eliminar el fondo de la imagen. De esta forma la imagen con la persona segmentada puede ser visualizada con mayor nivel de realismo en las gafas HoloLens 2.

5 Desarrollo

En este Capítulo se describe cómo se han empleado las tecnologías abordadas en la metodología (Capítulo 4), para realizar el diseño, programación e implementación del proyecto presentado. Para ello, primero se introduce en la Sección 5.1 el sistema de cyberpresencia desarrollado. Posteriormente, en la Sección 5.2 se exponen los pasos seguidos para realizar las instalaciones y configuraciones necesarias de los entornos de software y hardware empleados. La Sección 5.3 aborda el diseño gráfico realizado de la aplicación y su implementación en HoloLens 2. Por último, en las Secciones 5.5 y 5.6 se presenta el desarrollo de las tareas de segmentación de personas y clasificación de emociones, respectivamente.

5.1 Presentación del sistema de cyberpresencia

En la Figura 5.1 se muestra un ejemplo del sistema de cyberpresencia desarrollado. En él intervienen un estudiante conectado en remoto con su webcam, el profesor en clase presencial usando unas gafas HoloLens 2 y los estudiantes presentes en el aula. En el ordenador del estudiante en remoto se captura continuamente la imagen de la webcam y se envía a un servidor web para segmentar la persona (eliminar fondo de la imagen) y clasificar su emoción.

Figura 5.1: Ejemplo del sistema de cyberpresencia desarrollado

Por otro lado, las HoloLens que está usando el profesor, realizan continuamente peticiones al mismo servidor para recibir la información tanto de segmentación como de clasificación. Cada vez que las HoloLens reciben una nueva información, esta es procesada de forma que pueda ser visualizada por el profesor en el entorno mixto creado. Tales visualizaciones (estudiante en remoto y su emoción), son representadas en una ubicación del espacio determinada por un patrón de imagen 2D.

En la Figura 5.1 se observa cómo el entorno permite al ponente integrar en un espacio común la presencialidad (alumnado en clase) con la no presencialidad (alumnado en remoto), posibilitando que se desenvuelva e interactúe con el medio de forma más cómoda y natural. Además, la aplicación le proporciona al docente información visual simple de interpretar, sobre la emoción predominante en el estudiante conectado. En este caso la emoción es representada en el nuevo escenario mediante el holograma de un emojie que varía en función de si la persona presenta emociones positivas, neutrales o negativas. De tal forma, el profesor puede analizar, entre otras características, el cómo está impactando el contenido y/o metodología aplicada durante la clase y modificar en tiempo real los aspectos que considere convenientes.

Es importante señalar que para conseguir la adecuada ejecución de la aplicación, una vez que el profesor se ha puesto las gafas, debe situar el patrón de imagen mostrado en la Figura 5.2, en el lugar donde desee que se visualice la información referente al estudiante en remoto. Después de este paso, el ponente puede ver al estudiante en casa como si estuviera en la sala e interactuar con él y con el entorno.

Figura 5.2: Patrón de imagen usado en la aplicación para definir la ubicación de la información del estudiante en remoto a proyectar

Por último, presentamos en la Figura 5.3 la arquitectura general diseñada para el funcionamiento de la aplicación, basada principalmente en las infraestructuras:

1. Dispositivo de RM HoloLens 2
2. Equipo remoto con webcam (ordenador del estudiante)
3. Sistema de servidores sobre GPU
4. Comunicación web (Hololens 2 - Servidores GPU - Equipo remoto)

Además, de emplear las tecnologías de desarrollo Unity, Visual Studio y Vuforia para la creación del entorno mixto y su funcionamiento en HoloLens 2. En el sistema de servidores sobre GPU de la Figura 5.3 se encuentra un conjunto de servicios web desarrollados mediante la plataforma Flask, para hacer posible la comunicación entre las HoloLens 2 y el equipo remoto,

a la vez que se ocupa de realizar las tareas de segmentación del estudiante y clasificación de su emoción en las imágenes procedentes del ordenador en remoto.

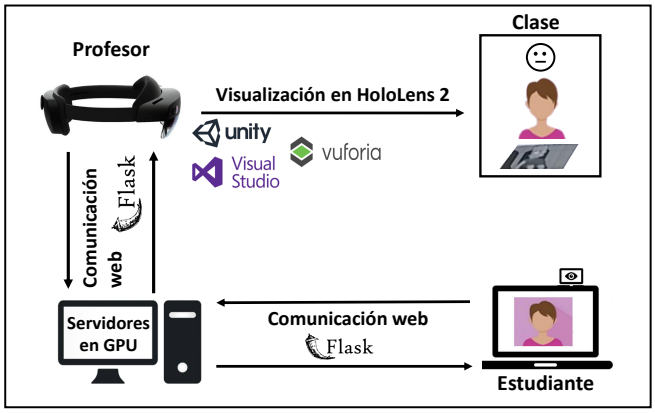


Figura 5.3: Arquitectura general del sistema de cyberpresencia

Teniendo en cuenta lo hasta aquí expuesto, la aplicación presentada sobre las gafas HoloLens 2 ofrece sólidas bases para una mejor gestión de la no presencialidad, permitiendo su integración con escenarios presenciales mediante el empleo de RM. El sistema de cyberpresencia diseñado supone una mejora en los espacios docentes universitarios, ofreciendo una nueva pero positiva metodología de formación, comunicación e interacción basada en escenarios mixtos.

5.2 Configuración del entorno de trabajo

En esta Sección se detallan los pasos seguidos para la configuración del entorno de desarrollo. Primeramente se presentan los requerimientos de instalación y posteriormente se explican las bases necesarias para crear y configurar el proyecto en Unity. Además, se aborda la importación y configuración de las herramientas de RM necesarias en la aplicación.

5.2.1 Instalación de las herramientas de desarrollo

Para desarrollar aplicaciones orientadas a las gafas HoloLens 2 de Microsoft es necesario realizar una serie de instalaciones previas. Primeramente se requiere de un equipo con sistema operativo **Windows 10** o **Windows 11**. En nuestro caso se ha empleado un sistema con Windows 10 y sobre este entorno ha sido necesario instalar las herramientas que se detallan a continuación.

Visual Studio 2022

El paquete de instalación de **Visual Studio 2022** ha sido descargado de [52] y se instalaron las cargas de trabajo siguientes:

- Desarrollo de escritorio de .NET
- Desarrollo de escritorio con C++
- Desarrollo de juegos con Unity
- Desarrollo para la Plataforma Universal de Windows (UWP)

Dentro de la carga de trabajo de UWP, se seleccionaron en la instalación los componentes:

- SDK de Windows 10 versión 10.0.20348.0
- Conectividad para dispositivos USB (necesario para implementar o depurar en HoloLens a través de USB)
- Herramientas de la Plataforma Universal de Windows para C++ (v142) (requerido al usar Unity)

Unity Hub con Unity 2020.3 LTS

La configuración de Unity actualmente recomendada por Microsoft para el desarrollo de aplicaciones para HoloLens 2 es **Unity 2020.3 LTS** con el complemento **Mixed Reality OpenXR**. La instalación de Unity se realizó mediante **Unity Hub** y para ello se siguieron los pasos:

1. Instalar Unity Hub mediante el enlace proporcionado en [53].
2. Seleccionar la pestaña "Installs" en Unity Hub y elegir "Install Editor".
3. Seleccionar "Unity 2020.3.28f1 LTS" y hacer clic en "Next".
4. Seleccionar en "Platforms" los componentes:
    - Universal Windows Platform Build Support
    - Windows Build Support (IL2CPP)
5. Seleccionar "Done".

La Figura 5.4 muestra el resultado de la correcta instalación de la versión de Unity seleccionada y sus componentes desde la ventana "Installs" de Unity Hub.

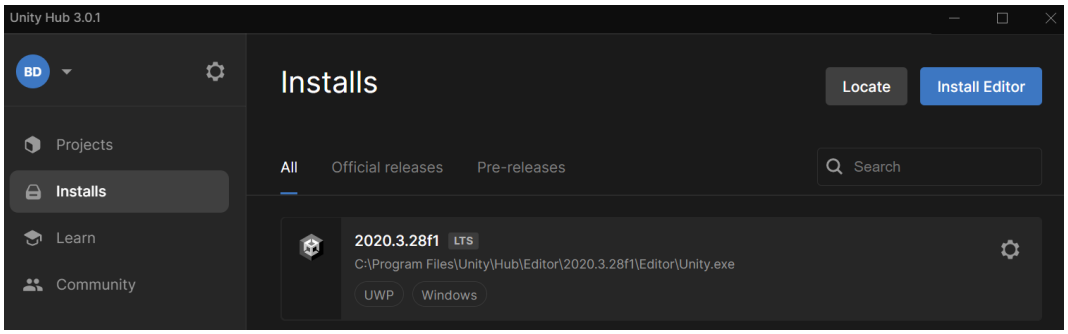


Figura 5.4: Resultado de la instalación de Unity Unity 2020.3.28f1 LTS y sus componentes desde la ventana "Installs" de Unity Hub

Herramienta de características de Mixed Reality

Para instalar la herramienta de características de Mixed Reality se descargó la versión 1.0.2111.0 del Centro Oficial de Descargas de Windows [54]. La descarga genera un archivo comprimido que en su interior contiene un ejecutable con el nombre **MixedRealityFeatureTool.exe** (Figura 5.5). Una vez descomprimido el archivo, el ejecutable es empleado para iniciar la herramienta de características de Mixed Reality.

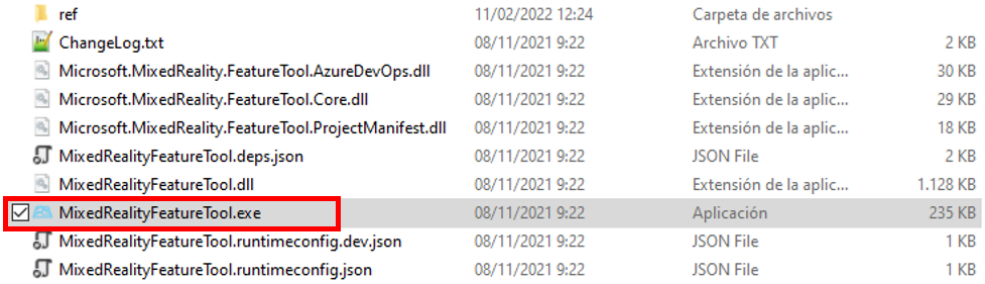


Figura 5.5: Archivo ejecutable **MixedRealityFeatureTool.exe**

5.2.2 Configuración de HoloLens 2 para el desarrollo

Para desarrollar aplicaciones que puedan ser compiladas y ejecutadas en HoloLens 2 es necesario configurar el dispositivo en modo de desarrollador. Este modo permite que la aplicación diseñada pueda ser compilada e implementada en HoloLens 2 mediante Visual Studio. Si el modo no es activado, Visual Studio no podrá conectarse con el dispositivo. Para activarlo se realizan los siguientes pasos:

1. Encender las gafas HoloLens 2 y colocárselas.
2. Acceder al menú principal.
3. Seleccionar la opción de “Configuración”.
4. En la aplicación de Configuración seleccionar “Actualizaciones & Seguridad”.
5. Seleccionar el elemento de menú “Para desarrolladores”.
6. Habilitar “Usar las funciones de desarrollador” (permite implementar las aplicaciones de Visual Studio).
7. Habilitar “Detección de dispositivos” (permite ejecutar Windows Holographic).

Además, al desplazarse hacia abajo en el menú “Para desarrolladores”, es posible habilitar la opción “Portal de dispositivos”. Esto permite acceder al Portal de dispositivos Windows en HoloLens desde un explorador web, lo cual resulta muy conveniente para consultar desde el equipo desarrollador diversos aspectos del dispositivo de RM. Con equipo desarrollador se hace referencia al sistema con Windows 10 instalado sobre el cual se desarrolla la aplicación antes de ser implementada en HoloLens 2.

5.2.3 Configuración de Unity para Windows Mixed Reality

En esta Sección primeramente describiremos como crear un nuevo proyecto de Unity. Para ello, desde la ventana "Projects" de Unity Hub se selecciona la opción "New Project". Posteriormente, se selecciona la plantilla **3D Core** y se define el nombre del proyecto a desarrollar como se indica en la Figura 5.6. En este caso el nombre del proyecto ha sido MRTK+Vuforia. Además, se define la ubicación donde se desea guardar el proyecto mediante el recuadro "Location", haciendo clic en el icono de carpeta. De ahí se busca o crea el directorio deseado. Por último, se selecciona "Create project" y el nuevo proyecto creado se abre en Unity.

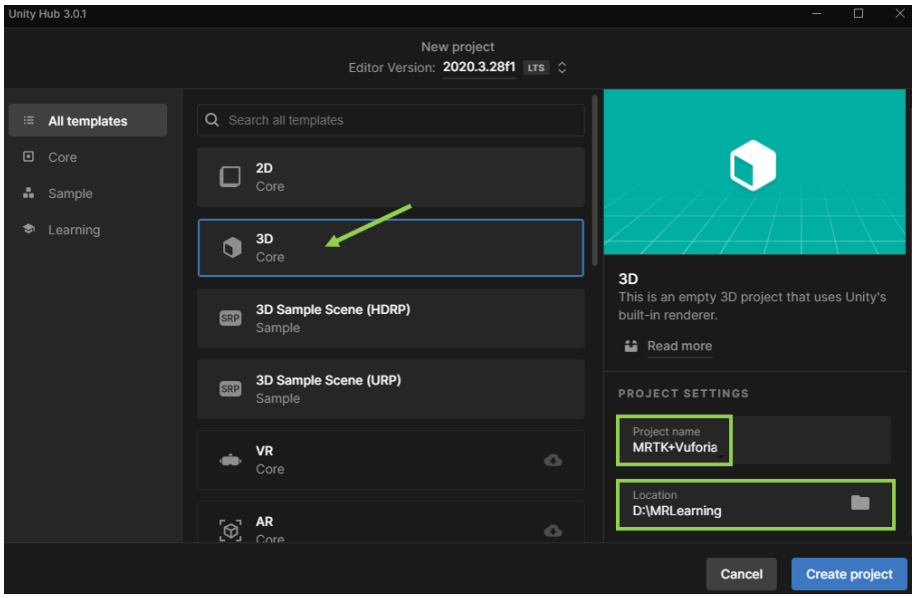


Figura 5.6: Creación de un nuevo proyecto en Unity mediante Unity Hub

Ahora, explicaremos como configurar el proyecto creado para desarrollar la aplicación con Windows Mixed Reality para HoloLens 2. El primer paso es configurar el proyecto de forma que se exporte como una aplicación de la Plataforma Universal de Windows. Esto se realiza seleccionando en Unity la opción **Build Settings** del menú "File". En la ventana que aparece se selecciona la Plataforma Universal de Windows con la configuración mostrada en la Figura 5.7. Finalmente, se hace clic en el botón "Switch Platform" para guardar los cambios de la plataforma configurada.

5.2.4 Importación de MRTK y OpenXR

Una vez que se tiene instalada la herramienta de características de Mixed Reality como se ha explicado en la Sección 5.2.1, se abre el archivo ejecutable y aparece una ventana donde se presiona "Start" para iniciar la herramienta. Luego se visualiza una nueva ventana para seleccionar el proyecto de Unity al cual se desean añadir las características de RM. De ahí, la herramienta comprueba que se trata de una carpeta de proyecto de Unity válida. Si la validación es correcta se activa la opción "Discover Features".

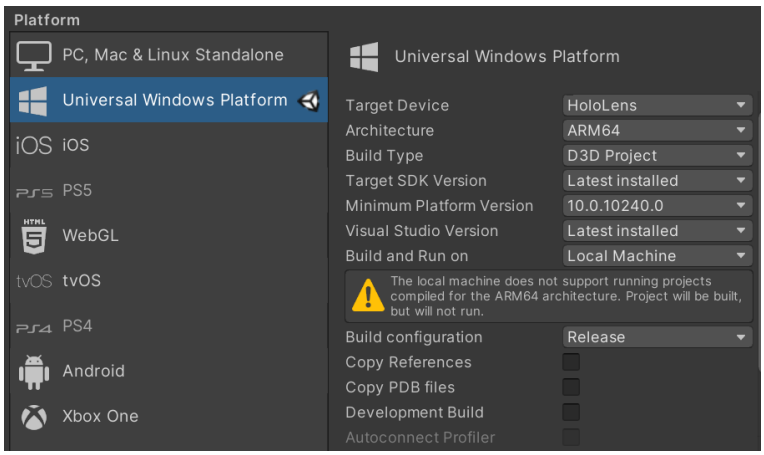


Figura 5.7: Configuración de compilación para aplicaciones de HoloLens 2

Ahora, la importación de los paquetes de RM requeridos en el desarrollo de aplicaciones para HoloLens 2, se realiza mediante la opción "Discover Features" activada. Aquí se seleccionan los paquetes **Mixed Reality Toolkit Foundation** dentro del menú "Mixed Reality Toolkit (0 of 10)" y **Mixed Reality OpenXR Plugin** dentro del menú "Platform Support (0 of 5)", como se muestra en la Figura 5.8. En ambos casos se importa la versión más reciente de cada paquete.

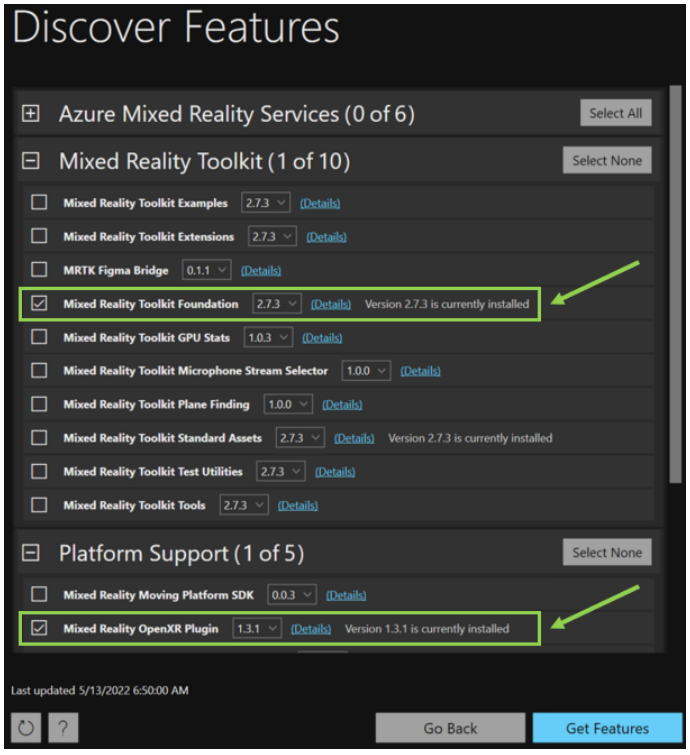


Figura 5.8: Importación de paquetes de RM con la herramienta de características de Mixed Reality

Después de definir los paquetes a importar se selecciona la opción “Get Features”. Ahora aparece la página **Import Features** y en ella se selecciona “Validate” para validar los paquetes seleccionados. Si las importaciones son correctas se muestra un cuadro de diálogo que indica *No validation issues were detected* y se presiona el botón “OK”. Posteriormente, se selecciona “Import” y aparece la página **Review and Approve** donde seleccionamos “Approve” para terminar el proceso de importación desde la herramienta de características. Una vez realizado este procedimiento volvemos al editor de Unity y hacemos clic en un área en blanco del editor para comenzar a importar los paquetes.

5.2.5 Configuración de OpenXR

Cuando se terminan las importaciones de la Sección anterior, aparece una advertencia para habilitar los back-end reiniciando el editor y seleccionamos “Yes”. Unity se reinicia y aparece una ventana de configuración del proyecto MRTK a partir de la cual se configura OpenXR para emplearlo con HoloLens 2. A esta ventana también se puede acceder mediante la barra de menús **Mixed Reality » Toolkit » Utilities » Configure Project for MRTK**.

En la ventana **MRTK Project Configuration** se selecciona la opción “Unity OpenXR Plugin” para habilitar la administración de complementos XR al proyecto. Ahora, aparece la ventana **Welcome to MRTK!** y seleccionamos “Show XR Plug-In Management Settings” para abrir la ventana de configuración del proyecto: **Project Settings**.

Dentro de la ventana de configuración, se selecciona en la página “XR Plug-in Management”, la Plataforma universal de Windows mediante la pestaña con el logotipo de Windows. En esta página se debe seleccionar la opción “Initialize XR on Startup” y en **Plugin Providers**, la opción “Open XR”. Debajo de “Open XR” se selecciona “Microsoft HoloLens feature group”. La configuración descrita se indica en la Figura 5.9.

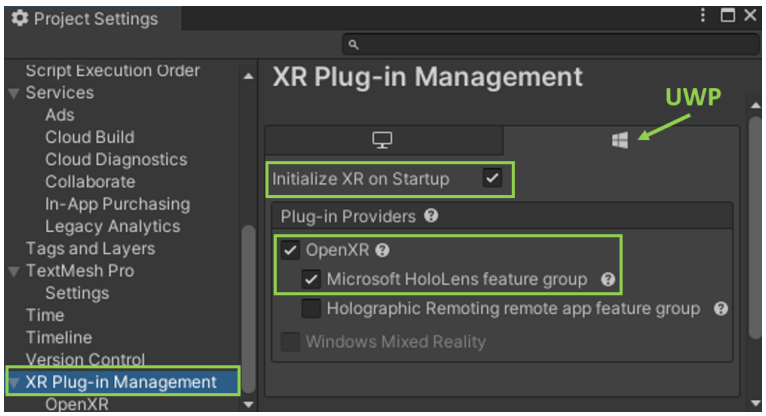


Figura 5.9: Configuración de XR Plug-in Management para HoloLens 2

Al definir esta configuración sale un símbolo de advertencia junto al recuadro de “OpenXR”. Esto significa que existen configuraciones incompatibles que deben resolverse. Presionando sobre el símbolo aparece una ventana de validación del proyecto, en ella se selecciona “Fix All” para corregir los problemas de incompatibilidad presentados. Después de esto sigue

apareciendo un problema relativo a un perfil de interacción. Para resolverlo, hacemos clic en Edit y vamos a la configuración del complemento **OpenXR** (notar que antes estábamos trabajando en la configuración del complemento **XR Plug-in Management**).

En la nueva ventana abierta, sale por defecto el perfil **Microsoft Hand Interaction Profile** en **Interaction Profiles**. En caso de que no estuviera la opción por defecto, se requiere que sea seleccionada. Ahora, en la Sección **OpenXR Feature Groups** se seleccionan los elementos: **Microsoft HoloLens**, **Hand Tracking** y **Motion Controller Model**.

Por último, quedaría configurar los modos de renderizado y de envío de profundidad (**Render Mode** y **Depth Submission Mode**). La opción de renderizado se establece como **Single Pass Instanced**. Esto se debe a que en Unity, la representación de la escena para aplicaciones de RM, se realiza por defecto dos veces, una para cada ojo. Lo cual conlleva a que se duplica la carga computacional. De ahí que para optimizar el proceso, se seleccione la representación de instancia de paso único, de manera que se realice una llamada única a la representación de instancias. Esta ruta se considera la más eficaz en Unity en términos de salvar tiempo de CPU y GPU.

En el modo de profundidad se selecciona "Depth 16 Bit". Este formato tiende a mejorar el rendimiento de los gráficos en el proyecto, suponiendo mayor estabilización de los hologramas. Por el contrario, no se recomienda escoger un formato de profundidad mayor (24 bits), debido a que incrementa el coste commputacional. La selección de toda la configuración descrita se muestra en la Figura 5.10.

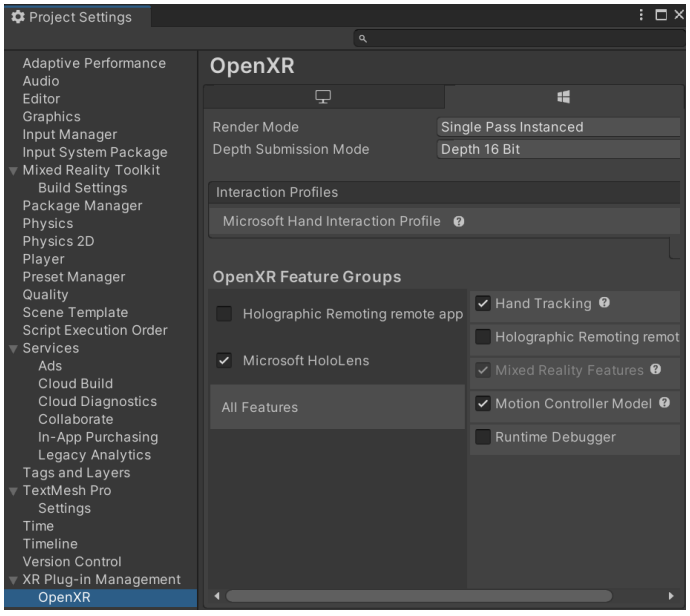


Figura 5.10: Configuración de OpenXR para HoloLens 2

Finalmente, después de cerrar la ventana, en la barra de menús se selecciona **Mixed Reality » Project » Appply recommended project setting for HoloLens 2** para mejorar el rendimiento de la aplicación.

5.2.6 Configuración de MRTK

Una vez que se tiene OpenXR configurado para el proyecto con HoloLens 2, pasamos a configurar MRTK. Para ello, primeramente se crea la escena sobre la cual vamos a diseñar las representaciones. La creación de la escena se realiza accediendo en la barra de menús a **File » New Scene**. Aquí se abre una ventana donde se selecciona el tipo de escena que se desea crear, en este caso **Basic (Built-in)** y hacemos clic en "Create" (Figura 5.11).

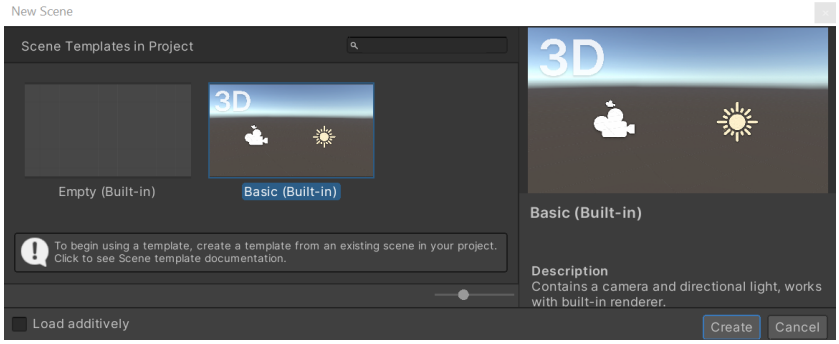


Figura 5.11: Creación de una escena en Unity

Después de creada la nueva escena, le agregamos las capacidades de MRTK seleccionando en la barra de menús: **Mixed Reality » Toolkit » Add to Scene and Configure...** y aparece la configuración de MRTK en el inspector de Unity. Después de haber agregado MRTK a la escena del proyecto, se añaden dos nuevos objetos en la ventana de jerarquía de la escena: **MixedRealityToolkit** y **MixedRealityPlayspace**. El primero de estos objetos contiene el kit de herramientas de RM y el segundo asegura la correcta administración en la escena de las gafas, controladores y sistemas que sean necesarios emplear.

Aquí el objeto **Main Camera** pasa a ser un elemento secundario del objeto **MixedRealityPlayspace**. Esto se debe a que el dispositivo de RM a utilizar se convierte en el centro del mundo holográfico. De esta forma se permite al espacio de juego administrar la cámara conjuntamente con los SDK de Windows 10.

Por último, queda seleccionar el perfil de configuración de MRTK. Para ello, en el inspector con las propiedades del objeto **MixedRealityToolkit** se selecciona el perfil **DefaultHoloLens2ConfigurationProfile** (Figura 5.12). MRTK proporciona este perfil predeterminado especialmente optimizado para el desarrollo de aplicaciones en HoloLens 2.

Figura 5.12: Perfil de MRTK optimizado para HoloLens 2

Para guardar la escena creada seleccionamos en la barra de menús **File » Save As...**. Luego, en el directorio del proyecto **Assets » Scenes**, se define un nombre a la escena y se guarda.

5.2.7 Importación de Vuforia Engine

El último paquete utilizado en el proyecto que falta por importar en Unity es Vuforia Engine. Existen tres maneras de añadir Vuforia Engine a un proyecto de Unity. Las tres variantes son:

1. Añadir el paquete mediante un script del editor.
2. Añadir la dependencia en el gestor de paquetes o en el manifiesto.
3. Descargar el paquete y añadirlo manualmente.

En nuestro caso ha resultado más conveniente emplear la tercera variante y explicamos a continuación cómo realizarla. Primeramente se ha descargado la versión 10.5.5 (última versión en el momento de la descarga) del paquete Vuforia Engine SDK como un fichero comprimido .zip. La descarga se ha hecho mediante el repositorio **git-repo.developer.vuforia.com**, donde la carpeta con el paquete tiene el nombre: **com.ptc.vuforia.engine**.

Una vez descargado, se descomprime el archivo y se mueve a la carpeta **Packages** del proyecto de Unity. Ahora volvemos a Unity y accedemos al gestor de paquetes mediante la barra de menús: **Window » Package Manager**. En la ventana del gestor pulsamos el ícono "+" y seleccionamos "Add package from disk". Buscamos el directorio del paquete y abrimos el fichero **package.json**. Si se ha importado Vuforia Engine correctamente en nuestro proyecto aparecerá en la lista de paquetes como se muestra en la Figura 5.13.

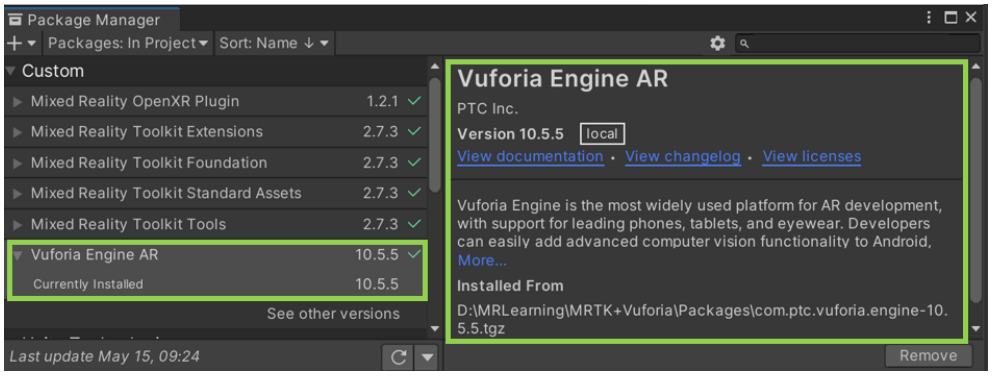


Figura 5.13: Resultado de importar el paquete Vuforia Engine desde el gestor de paquetes de Unity

5.2.8 Configuración de Vuforia Engine

Ahora que ha sido importado el paquete de Vuforia Engine, procedemos a su configuración. Para ello, accedemos a **File » Build Settings** y en la ventana de configuración hacemos clic en el botón "Player Settings". Ahora, en las opciones de **Player** desplegamos

el menú "Publishing Settings" y en el apartado "Capabilities" seleccionamos los siguientes elementos: **InternetClient**, **Webcam**, **Microphone** y **SpatialPerception**. Además, en el apartado "Supported Device Families" es necesario verificar que está seleccionada la opción **Holographic**. Estas configuraciones son mostradas en la Figura 5.14.

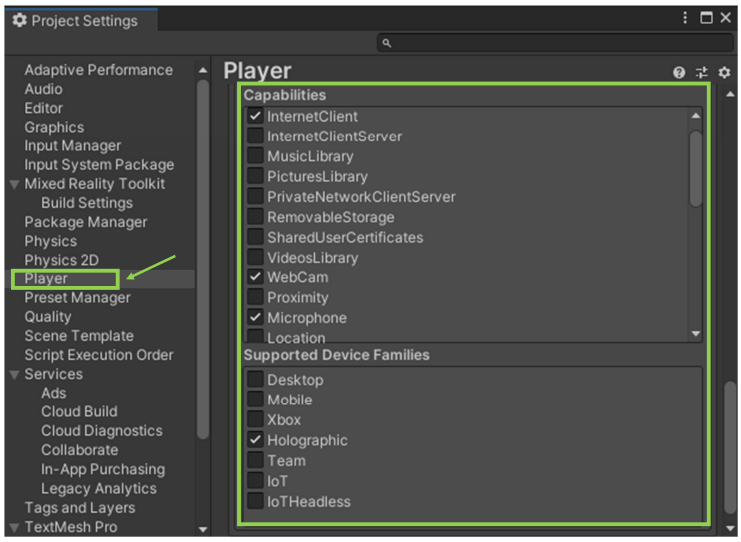


Figura 5.14: Configuración de "Publishing Settings" para el empleo de Vuforia Engine en el proyecto

En el menú desplegable "Resolution and Presentation" es necesario deshabilitar "Run in Background" para evitar que Vuforia Engine siga ejecutándose cuando la aplicación esté en segundo plano. Esto permite además que Vuforia retome el acceso a la cámara una vez que la aplicación haya vuelto a ejecutarse en primer plano. Por último, en la lista desplegable "Default Orientation" debe verificarse que está seleccionado: "Landscape Left". La Figura 5.15 muestra la configuración descrita.

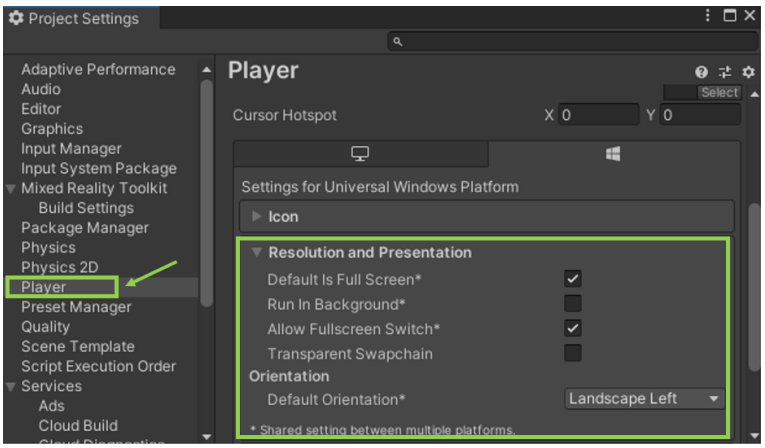


Figura 5.15: Configuración de "Resolution and Presentation" para el empleo de Vuforia Engine en el proyecto

5.3 Diseño gráfico y compilación de la aplicación

Una vez terminadas todas las instalaciones, importaciones y configuraciones necesarias para la creación del proyecto en Unity, se presenta el diseño gráfico realizado de la aplicación para HoloLens 2. Sobre la escena creada en la Sección anterior, el diseño se ha basado en tres elementos principales:

1. **Patrón** a reconocer para representar al estudiante,
2. **Plano** sobre el cual proyectar la imagen del estudiante segmentada.
3. **Holograma** representativo de la emoción del estudiante.

En la Figura 5.16 se indican los elementos mencionados sobre la escena creada. A continuación se explicarán detalladamente los procedimientos seguidos para desarrollar cada uno de ellos.

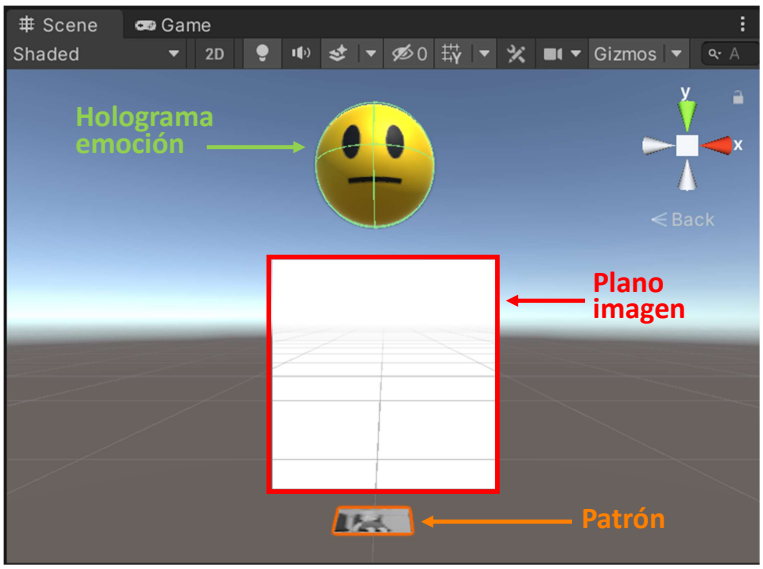


Figura 5.16: Diseño de la escena con los elementos principales

5.3.1 Empleo de Vuforia Engine

En esta Sección se describe cómo se ha empleado Vuforia Engine para detectar dónde situar al estudiante en el entorno de RM creado, mediante el empleo de un patrón de imagen como destino. El primer paso es añadir el componente **Vuforia Behaviour** al objeto "Main Camara". Para ello, en el panel de jerarquía de Unity, se expande el objeto "MixedRealityPlaySpace" y se selecciona "Main Camera". Ahora, con el objeto "Main Camera" seleccionado, se accede al panel de inspección y se desciende en él hasta encontrar la opción"Add Component". Al seleccionar dicha opción aparece una ventana de búsqueda donde se escribe el nombre del componente deseado, en este caso "Vuforia Behaviour". Cuando se observa el nombre del componente se selecciona para añadirlo (Figura 5.17).

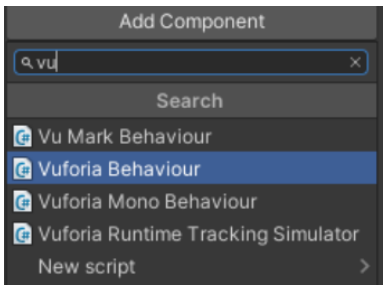


Figura 5.17: Añadiendo componente Vuforia Behaviour

Ahora añadimos el objeto "Image Target" de Vuforia haciendo clic derecho en el panel de jerarquía y seleccionando la opción **Vuforia Engine » Image Target**. Este objeto será nuestro patrón a reconocer. Para poder definir el destino de imagen deseado como patrón, es necesario primero obtener una licencia de Vuforia y crear una base de datos propia. Esto se requiere en todas las aplicaciones en que se hace uso de Vuforia Engine.

Licencia de desarrollador

Para crear y utilizar la licencia de Vuforia en el proyecto, seleccionamos nuevamente el objeto "Main Camera" y accedemos al componente Vuforia Behaviour en el inspector. Aquí se selecciona "Open Vuforia Engine" y posteriormente, se hace clic sobre el botón "Add License". Automáticamente se nos abrirá Vuforia Developer Portal y aquí es necesario registrarse. La creación de una nueva cuenta es libre de costo siempre que la aplicación a desarrollar no tenga fines comerciales.

Después de haber creado la cuenta e identificarse con ella en el Vuforia Developer Portal, se tiene la interfaz de la Figura 5.18. Aquí hacemos clic en "Get Basic", definimos un nombre de licencia y seleccionamos "Confirm". En este caso, el nombre de la licencia creada ha sido **HoloLensPatternRecognition**. Ahora en la ventana de la Figura 5.18 aparecerá la licencia obtenida. Hacemos clic en el nombre de la misma y copiamos la clave que se muestra. Volvemos a Unity y pegamos la clave en la opción "App License Key".

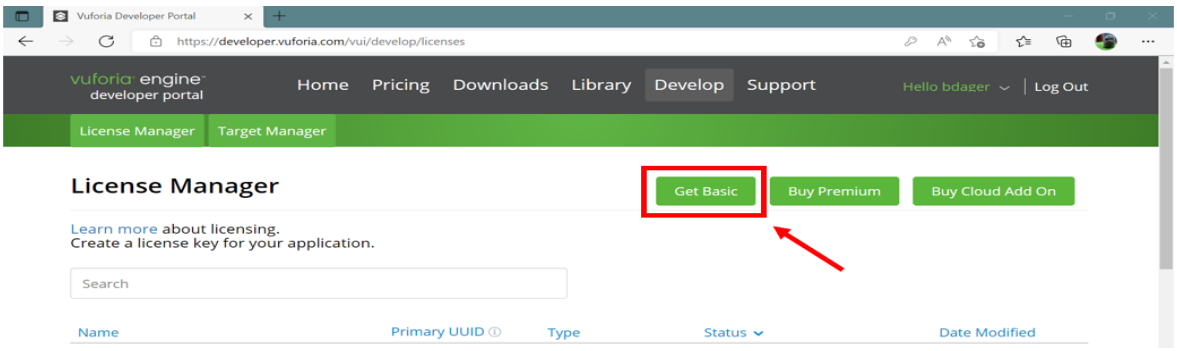


Figura 5.18: Vuforia Developer Portal

Base de datos de destinos

Una vez que ya se tiene la licencia requerida, pasamos a crear la base de datos con la imagen que se desea emplear como patrón. Para ello, en el panel de jerarquía en Unity se selecciona el objeto "Image Target" y en el inspector se hace clic en el botón "Add Target" dentro del "Image Target Behaviour Component". Nuevamente somos redirigidos al Vuforia Developer Portal, pero en este caso, bajo la pestaña **Target Manager**. Aquí seleccionamos la opción "Add Database" y se muestra una ventana donde se define el nombre y tipo de la base de datos. En este caso se requiere seleccionar como tipo **Device**. Hacemos clic en "Create" y aparece en el portal el nombre de la base de datos creada.

Ahora, para añadir una imagen a la base de datos se hace clic sobre el nombre de la misma y se abre una pestaña donde se selecciona la opción "Add Target". Luego aparece una ventana donde se define el tipo de patrón, la imagen a cargar y el ancho de la misma. Una vez que se ha subido el patrón, Vuforia lo califica en base a la confianza con la cual puede identificarlo (Figura 5.19). En nuestro caso le ha sido asignado un valor de 4 (el máximo es 5), lo cual es más que suficiente para el proyecto desarrollado ya que el patrón se encontrará en una posición fija durante la clase, por lo cual será reconocido una vez al iniciar la aplicación.

Target Name	Type	Rating	Status	Date Modified
bart1	Single Image	★★★★☆	Active	Feb 23, 2022 10:36

Figura 5.19: Patrón subido a la base de datos creada para trabajar con Vuforia

Finalmente descargamos la base de datos seleccionando como plataforma de desarrollo **Unity Editor** y se genera el archivo FirstDB.unitypackage. Ahora, es posible importarlo en Unity mediante **Assets » Import packages » Custom Package...**. Despúes de importar la base de datos, se selecciona el objeto "Target Image" y en el panel de inspección, se accede al componente "Image Target Behaviour". Dentro de las opciones del componente se hace clic en el menú desplegable correspondiente a "Database". Ahora en el menú aparece el nombre de la base de datos recién añadida y la seleccionamos. Después de este paso, en el menú "Image Target" se selecciona el patrón que incluimos en la base de datos (bart1). Con ello hemos terminado de configurar el patrón a reconocer por Vuforia mediante un destino de imagen 2D (Figura 5.20).

5.3.2 Representación del estudiante

Una vez que ya tenemos el patrón de Vuforia añadido y configurado, creamos un plano perpendicular a dicho patrón, sobre el cual vamos a representar al estudiante conectado desde casa. Para ello, desde el panel de jerarquía hacemos clic derecho sobre "Image Target" y añadimos un objeto tipo **UI » Canvas**. Aquí utilizamos este tipo de objetos ya que permite el trabajo y la manipulación de elementos 2D en entornos 3D. Posteriormente, dentro del Canvas añadimos un objeto **RawImage** que hará la función del plano perpendicular. RawImage muestra una imagen no-interactiva al usuario que permite ser cambiada desde script, lo cual nos permitirá la actualización de la representación mediante código.

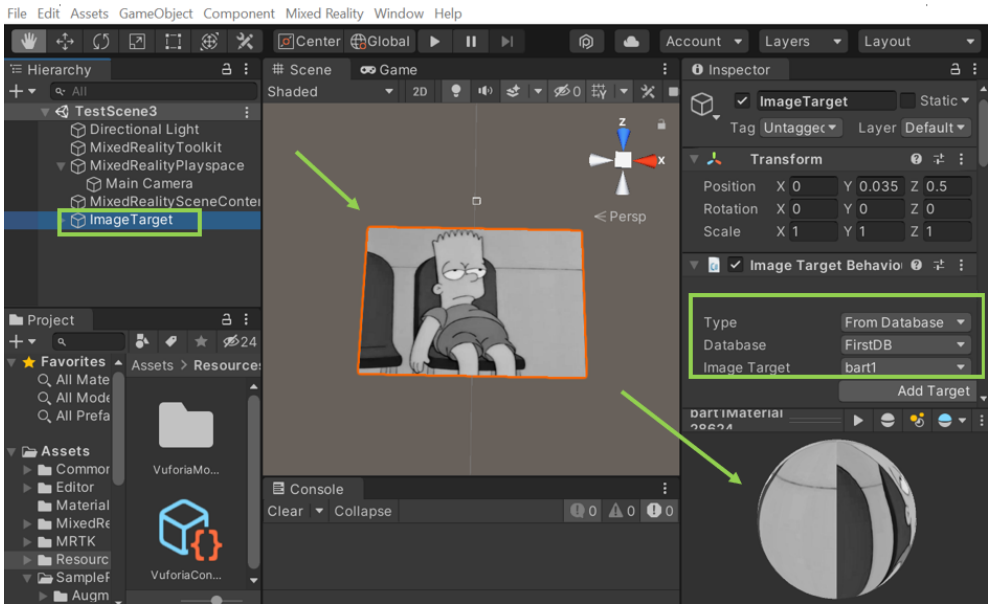


Figura 5.20: Patrón añadido y configurado en el proyecto de Unity para ser reconocido mediante Vuforia Engine

Hasta ahora lo que tenemos es que, al ser detectado el patrón por las HoloLens, se mostraría un plano en blanco. Para proyectar la imagen segmentada del estudiante, se crea un script en C# (**getImg.cs** añadido al objeto "Canvas"), que se encarga de pedir la imagen al servidor principal (Sección 5.4) y una vez que la ha recibido, crea una textura con ella y la carga en el objeto "RawImage". El proceso de petición de la imagen se realiza de forma paralela a la ejecución del ciclo principal. Sin embargo, Unity no permite el trabajo con texturas de forma paralela (hilos de ejecución), por lo cual la textura es creada y cargada con la nueva imagen dentro del ciclo de ejecución principal (función **Update** dentro del código en C#).

5.3.3 Representación de emociones

Para la representación de la emoción predominante en el estudiante, se crea un holograma con un emojie indicativo de la emoción clasificada. Este emojie se sitúa sobre la imagen de la persona proyectada y al igual que en la Sección anterior (5.3.2), solo debe aparecer después de que se haya reconocido el patrón de Vuforia. Para ello, se crea un nuevo objeto tipo **Sphere** que hereda del objeto "Image Target". Esto se realiza haciendo clic derecho sobre "Image Target" en el inspector y seleccionando **3D Object » Sphere**.

Una vez que tenemos creado el objeto sobre el cual representar la emoción, es necesario crear los materiales que contendrán los emojies. En la tarea de clasificación de emociones (Sección 5.6) se han definido tres estados generales: Negativo, Positivo y Neutro. Por tal razón se definen tres materiales para cada uno de los estados.

Creación de materiales

Para desarrollar esta tarea, primeramente se crea en la carpeta **Assets** del proyecto, un nuevo directorio con el nombre **Materials** donde se van a guardar los materiales. Además, se crea otro directorio, **Textures**, para almacenar las texturas que vamos a emplear en la definición de los materiales. En este directorio se han creado y guardado tres imágenes, una para cada material. Las imágenes son mostradas en la Figura 5.21 (Izquierda).

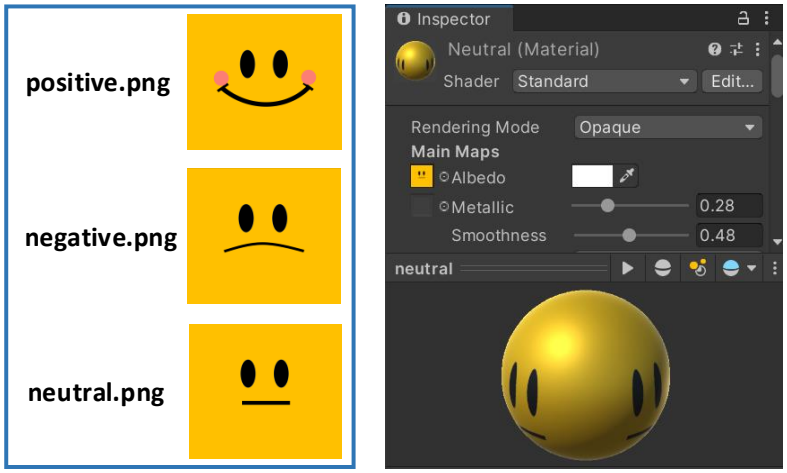


Figura 5.21: Izquierda: Texturas creadas para cada emoción. **Derecha:** Material neutral creado

Ahora, en el panel del proyecto en Unity, accedemos a la carpeta Materials dentro de Assets y hacemos clic derecho para seleccionar **Create » Material** y crear el primer material. Le definimos un nombre y en el inspector le añadimos la imagen de la emoción arrastrándola hasta el cuadro en la izquierda de la propiedad "Albedo". Luego cambiamos los parámetros "Metallic" y "Smoothness" hasta lograr el efecto visual deseado. La Figura 5.21 (Derecha) muestra el resultado desde el inspector de Unity. El mismo proceso se repite para crear los dos restantes materiales.

Una vez que ya tenemos los materiales necesarios, en las propiedades del objeto "Sphere" en el inspector, desplegamos el menú "Materials" y seleccionamos uno de los materiales creados. Esto añade el material a la esfera y nos permite visualizar el resultado en Unity. De esta forma es posible saber si es necesario realizar algún cambio para obtener la visualización deseada.

Actualización del material

Ahora el objetivo es lograr que el emojie de la emoción se actualice cada vez que el estudiante cambie su estado. Para ello, en el script de C# introducido en la Sección anterior (5.3.2), se definen tres materiales como variables públicas. Volvemos a Unity y al seleccionar el objeto "Canvas", es posible observar en el inspector los nuevos materiales bajo las propiedades del script **getImg.cs**. Aquí damos valor a cada material del script con uno de los materiales definidos previamente, para poder utilizarlos mediante código (Figura 5.22).

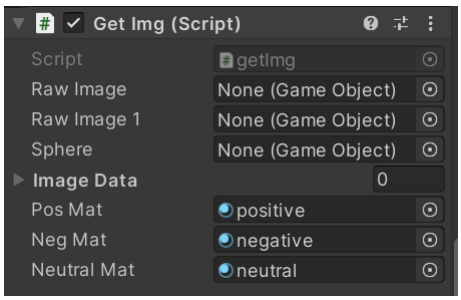


Figura 5.22: Definiendo materiales del script

Cuando ya se tienen definidos los valores de los materiales del script, en el código se utilizan para cambiar el material actual de la esfera proyectada. Al igual que en la representación del estudiante, aquí se requiere el empleo de un servidor web (Section 5.4) que se encarga de dar información referente a la emoción clasificada. Teniendo en cuenta la información adquirida (mediante un hilo de ejecución en paralelo), se actualiza el material en el ciclo principal del programa. De esta forma obtenemos la visualización deseada.

5.3.4 Compilación de la aplicación

Una vez creado el proyecto con los elementos deseados, es necesario compilarlo tanto en Unity como en Visual Studio para que pueda ser ejecutado en HoloLens 2. Para ello, primero se selecciona en la barra de menús: **File»Build Settings...**. Ahora, en la ventana de configuración de compilación, añadimos en la Sección **Scenes In Build**, la escena actual para compilarla. Esto lo hacemos presionando en el botón "Add Open Scenes" como se indica en la Figura 5.23. Luego, se hace clic "Build" y se selecciona la carpeta donde se desea almacenar la compilación.

Una vez seleccionada la carpeta, Unity comienza la compilación y muestra mediante una barra de progreso el estado de la misma. Cuando se termina este proceso, el Explorador de archivos de Windows abre automáticamente la carpeta de compilación. Luego, accedemos a la carpeta y seleccionamos el archivo **.sln** generado haciendo doble clic. Esto abrirá el fichero en Visual Studio.

Ahora, se procede a configurar el entorno de Visual Studio para HoloLens 2. Primero se selecciona la configuración **Release** y la arquitectura **ARM64**. Estas características coinciden con la configuración definida en el proyecto de Unity en la Sección 5.2.3. Además, se selecciona como destino de implementación **Device** ya que vamos a realizar la compilación mediante USB. La configuración descrita se muestra en la Figura 5.24.

Por último nos resta realizar la compilación en las HoloLens, pero antes es necesario emparejarlas con el equipo desarrollador (Sección 5.3.5). Una vez realizado el emparejamiento, conectamos las HoloLens al equipo mediante USB y en Visual Studio seleccionamos **Depurar » Iniciar**. Esto nos permite implementar la aplicación en HoloLens e iniciarla automáticamente sin el depurador de Visual Studio adjunto. Si se desea implementar la aplicación sin que se inicie de forma automática se debe seleccionar **Compilar » Implementar**.

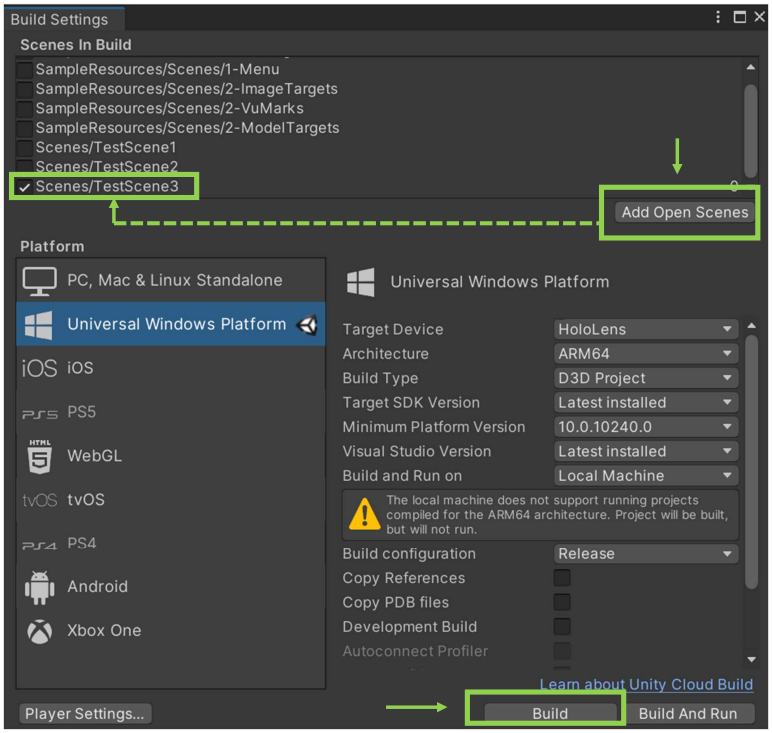


Figura 5.23: Ventana Build Settings. El botón "Add Open Scenes" añade la escena actual y el botón "Build" inicia la compilación

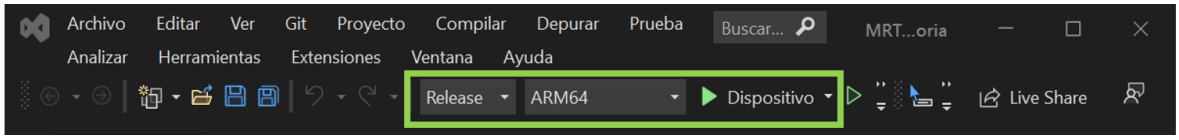


Figura 5.24: Configuración de compilación en Visual Studio

5.3.5 Emparejamiento de HoloLens 2

Como se explicó en la Sección 5.2.2, para realizar una compilación en HoloLens 2 es necesario que esté habilitado el modo de desarrollador. Además de esto, se requiere emparejar el dispositivo con el equipo desarrollador antes de hacer la primera implementación de una aplicación. Esto es necesario una sola vez por cada equipo desarrollador que se emplee, posteriormente la configuración queda guardada.

Para realizar el emparejamiento, cuando se va a realizar la primera implementación mediante Visual Studio, es necesario escribir un PIN proporcionado por HoloLens para el emparejamiento. Para obtener el PIN se accede en HoloLens a "Menú principal » Configuración » Actualizaciones & Seguridad » Para desarrolladores" y se selecciona la opción **Emparejar**. Después de esto el PIN es visualizado en HoloLens.

Una vez que se tiene el PIN, se escribe en el cuadro de diálogo de Visual Studio y se selecciona "Listo" en HoloLens. A partir de aquí, el equipo estará emparejado con Holo-

Lens y será posible implementar aplicaciones de forma automática sin tener que repetir este procedimiento.

5.3.6 Ejecución de la aplicación en HoloLens 2

Cuando el proyecto termina de compilarse, aparece en HoloLens una ventana con el nombre de la aplicación recién implementada y es posible ejecutarla. Además, se podrá acceder a la misma mediante el menú principal de HoloLens. Al ejecutar la aplicación, por defecto sale el generador de perfiles de diagnóstico. Este elemento indica la velocidad con que se están mostrando los fotogramas en la aplicación. Mientras se está en el proceso de desarrollo del proyecto, esta opción resulta conveniente para analizar el rendimiento de la aplicación. Para activar/desactivar la visualización del generador se emplea el comando de voz "Toggle Profiler".

5.4 Sistema de comunicación web

El sistema de comunicación web desarrollado está enfocado en permitir un medio de intercambio de información entre las HoloLens, el ordenador con webcam del estudiante y un sistema de procesamiento potente que se encarga del procesamiento previo de las imágenes. El sistema está basado en una arquitectura cliente-servidor implementada mediante **Flask**. La Figura 5.25 muestra la arquitectura propuesta. A continuación se describen cada uno de los componentes que la integran.

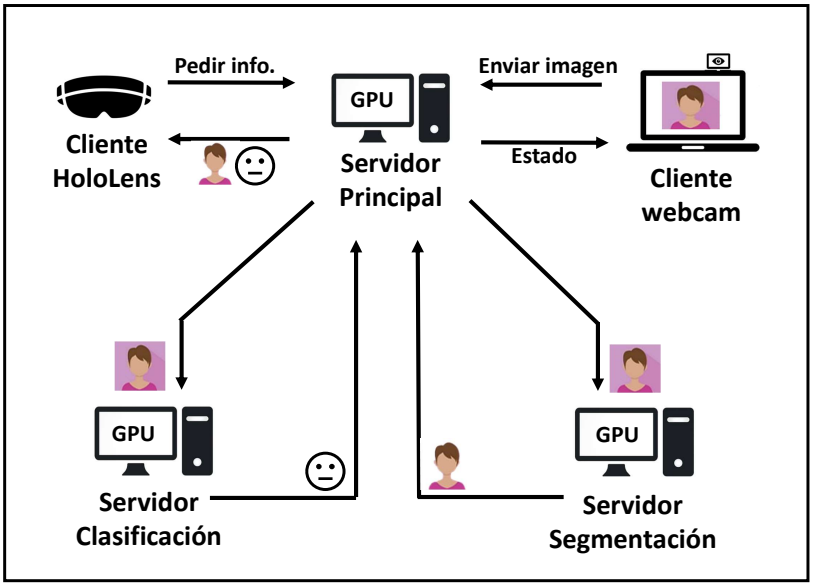


Figura 5.25: Arquitectura del sistema de comunicación web

Cliente webcam: Este servicio web se encuentra en el ordenador del estudiante capturando ininterrumpidamente imágenes de la webcam y enviándolas al servidor principal. Para su envío la imagen es codificada como una cadena de bytes (Binary string). Cada vez que publica una imagen espera una respuesta con un estado que indica si ha sido correctamente recibida o si se ha producido algún error.

Servidor Principal: Servidor a través del cual se enlazan los otros cuatro componentes del sistema. En el servidor principal se está recibiendo continuamente la imagen procedente de la webcam del estudiante. Una vez que las HoloLens realizan una petición, este servidor envía la última imagen recibida a los servidores de segmentación y clasificación y espera por sus respuestas. Las llamadas a los dos servidores se realizan de forma paralela mediante dos hilos de ejecución. Cuando las dos respuestas (imagen segmentada y emoción) han sido recibidas, se codifica la información como un archivo JSON y se devuelve a las HoloLens. Los tres servidores creados han sido desarrollados sobre un sistema con GPU.

Servidor Segmentación: Este servidor se encarga de ejecutar la tarea de segmentación de la persona en la imagen capturada por la webcam. Primero decodifica la imagen y realiza el procesamiento necesario para la segmentación (Sección 5.5). Luego, codifica la imagen segmentada sin fondo, como un archivo JSON y la devuelva al servidor principal.

Servidor Clasificación: En este servidor se desarrolla la tarea de clasificación de emociones. Para ello, primeramente decodifica la imagen recibida del servidor principal y realiza la clasificación sobre el rostro presente en ella (Sección 5.6). El resultado es indicado con un valor numérico correspondiente a cada emoción definida (0: Neutral, 1: Negativo, 2: Positivo, -1: Rostro no detectado). Posteriormente, este resultado es codificado mediante un archivo JSON y devuelto al servidor principal.

Cliente HoloLens: El cliente HoloLens se ocupa de pedir la información de imagen del estudiante segmentada, conjuntamente con su emoción, al servidor principal. Una vez que recibe la información solicitada, la decodifica para que pueda ser utilizada en la aplicación de RM. Este servicio ha sido desarrollado en el lenguaje de programación C# para poder ser utilizado en la aplicación creada para HoloLens sobre Unity. El resto de los servicios han sido desarrollados en Python.

5.5 Segmentación de personas

La segmentación de personas se efectúa sobre cada imagen capturada por la webcam del estudiante con el objetivo de eliminar el fondo de la misma y garantizar una representación más realista en el entorno mixto creado. Dicha segmentación se ha desarrollado mediante el algoritmo **Mask R-CNN**. En este caso se ha implementado una segmentación de instancias de la clase persona y se obtiene la máscara de la instancia que presenta la mayor probabilidad para la clase analizada.

La implementación del algoritmo Mask R-CNN empleado está basada en [55], con pesos preentrenados en el dataset: MS COCO. El lenguaje de programación empleado ha sido Python y entre los módulos utilizados se encuentran: Keras, Tensorflow, scikit-image, pycocotools

y h5py. La secuencia de pasos desarrollados para realizar la segmentación de la persona se indica a continuación:

1. Importar la librería Mask R-CNN proporcionada en [55].
2. Importar el dataset COCO como librería desde el directorio *"samples/coco/"* (dateset coco empleado en el entrenamiento de la red).
3. Descargar los pesos preentrenados en COCO.
4. Definir el tamaño del batch a 1 ya que la inferencia se realiza en una imagen a la vez.
5. Crear el modelo de Mask R-CNN con el modo inferencia como argumento.
6. Cargar los pesos pre-entrenados en MS-COCO al modelo creado en el paso anterior.
7. Ejecutar la segmentación de todas las instancias de las diferentes clases definidas en COCO para las que el modelo fue entrenado previamente.
8. Obtener la instancia de la clase persona que presenta el mayor valor de probabilidad obtenido en el paso anterior. Para ello, conocemos que la clase persona corresponde al identificador 1. Teniendo esto en cuenta, iteramos sobre todas las inferencias resultantes y verificamos si existe alguna de la clase persona (*id=1*). En caso positivo comparamos el valor de probabilidad obtenido para dicha inferencia con la mayor probabilidad alcanzada previamente (inicialmente asignamos 0 al valor de mayor probabilidad). Si se cumplen las condiciones anteriores actualizamos el valor mayor de probabilidad obtenido y guardamos el índice de la inferencia correspondiente.
9. Crear una máscara con el fondo de la imagen (diferencia entre la imagen completa y el resultado de la segmentación de la persona).
10. Convertir la imagen a BGRA e insertar un cuarto canal (alpha) con la máscara del fondo para añadirlo como región transparente de la imagen final en formato .png.

5.6 Clasificación de emociones

En esta Sección se presenta el desarrollo de la tarea de clasificación de emociones realizada en la aplicación. Para ello, se han creado varios modelos de clasificación basados en CNN y K-NN. Además, se presenta un modelo de K-NN en el cual previamente se reduce la dimensión de los datos analizados mediante el algoritmo PCA. El objetivo de emplear varios modelos es comparar los resultados producidos por cada uno y utilizar finalmente el que mejor se ajuste a la aplicación en cuestión.

A modo general, el pipeline de clasificación implementado se define mediante dos procesos: 1) preprocesamiento de los datos a clasificar, 2) clasificación de emoción mediante el uso de uno de los modelos implementados. La Figura 5.26 muestra la estructura desarrollada con los procesos principales y sus subprocesos, además de los modelos empleados.

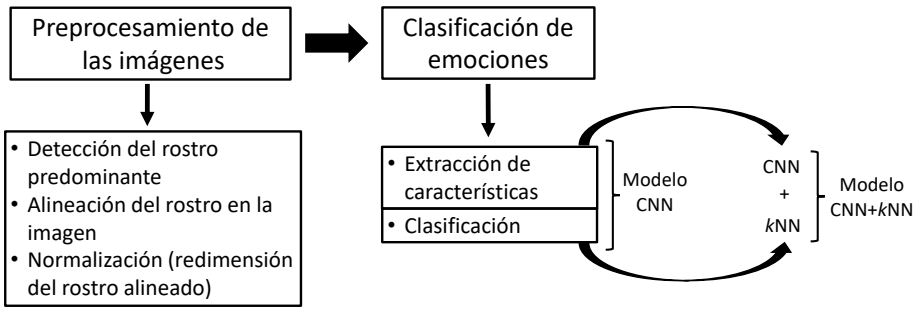


Figura 5.26: Procesos en la clasificación de emociones

Los modelos utilizados han sido definidos para clasificar emociones con una generalización de las 8 emociones más conocidas. Esta generalización se ha planteado de la forma:

1. **Neutral:** neutral, sorpresa
2. **Negative Emotion:** tristeza, miedo, disgusto, enfado, desprecio
3. **Positive Emotion:** alegría

A continuación, se explican los modelos empleados, así como el preprocesamiento de las imágenes realizado. En todos las implementaciones, el lenguaje de programación usado ha sido Python.

5.6.1 Modelo CNN

La estructura del modelo CNN, en adición con un conjunto de pesos preentrenados para la clasificación de emociones, fueron obtenidos de [56]. La implementación del modelo se realizó mediante el módulo Keras con la estructura secuencial mostrada en la Figura 5.27. De forma resumida el modelo presenta:

- 4 capas convolucionales (CNN)
- 3 capas de MaxPooling
- 3 capas de Dropout
- 1 capa Flatten
- 2 capas densas (fully connected)

Las capas convolucionales y la primera densa utilizadas presentan función de activación tipo **relu**. La última capa densa emplea la función de activación **softmax** debido a que se está trabajando con un problema de clasificación multiclase. La salida del modelo es de tamaño tres, ya que cada elemento corresponde a un valor de probabilidad determinado para cada uno de los tres conjuntos de emociones definidos en el papeline de clasificación (tres categorías correspondientes a Neutral, Negative Emotion y Positive Emotion).

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 3075
=================================================================
Total params: 2,341,507
Trainable params: 2,341,507
Non-trainable params: 0
```

Figura 5.27: Estructura del modelo CNN

5.6.2 Modelo CNN+*K*NN

El modelo CNN+*K*NN desarrollado se ha creado con la idea de comparar los resultados de clasificación de un algoritmo de Machine Learning (ML) tradicional frente a una red convolucional profunda como es el modelo CNN y escoger el que produzca los resultados más adecuados para la aplicación desarrollada. Para lograr esto, se implementó el algoritmo k-NN para realizar el proceso de clasificación mediante el módulo **scikit-learn** y conjuntamente, se empleó parte del modelo de CNN de la Sección 5.6.1 para realizar el proceso de extracción de características como se planteó en la Figura 5.26.

Concretamente, una vez que se tiene creado y compilado el modelo CNN con los pesos preentrenados para la clasificación de emociones, hemos creado un segundo modelo CNN (Figura 5.28) a partir del primero, eliminando las capas densas y obteniendo como salida un vector de características de tamaño igual a 2048. Estas características pasan a ser la entrada del algoritmo K-NN para realizar la clasificación.

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 46, 46, 32)        320
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 44, 44, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 20, 20, 128)       73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 10, 10, 128)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
=================================================================
Total params: 240,256
Trainable params: 240,256
Non-trainable params: 0
```

Figura 5.28: Estructura del segundo modelo CNN empleado para la extracción de características

Como el vector de características a emplear es de un tamaño considerable, hemos definido también una variante del modelo donde luego de obtener las características, realizamos una reducción de su dimensión empleando el algoritmo PCA. A esta variante del modelo se le ha denominado CNN+KNN+PCA.

5.6.3 Preprocesamiento de las imágenes

La estructura del modelo CNN fue entrenada para realizar la clasificación de emociones sobre imágenes cuyo contenido está ajustado a un rostro alineado y con dimensiones de 48x48 píxeles. Por lo cual, el resultado de este método puede ser beneficiado al alinear y redimensionar el rostro antes de realizar la clasificación. En esta Sección se describe el procedimiento desarrollado para implementar dicho preprocesamiento.

De forma concreta, al finalizar el preprocesamiento del conjunto de datos a clasificar se deben tener todos los rostros de manera que: 1) estén centrados en la imagen, 2) se encuentren alineados, encontrándose los ojos ubicados en una línea horizontal (rostro girado con los ojos ubicados sobre las mismas coordenadas en el eje y), 3) hayan sido normalizados presentando la misma escala (rostros redimensionados). La Figura 5.29 muestra los pasos seguidos para lograr estos objetivos y realizar el preprocesamiento.

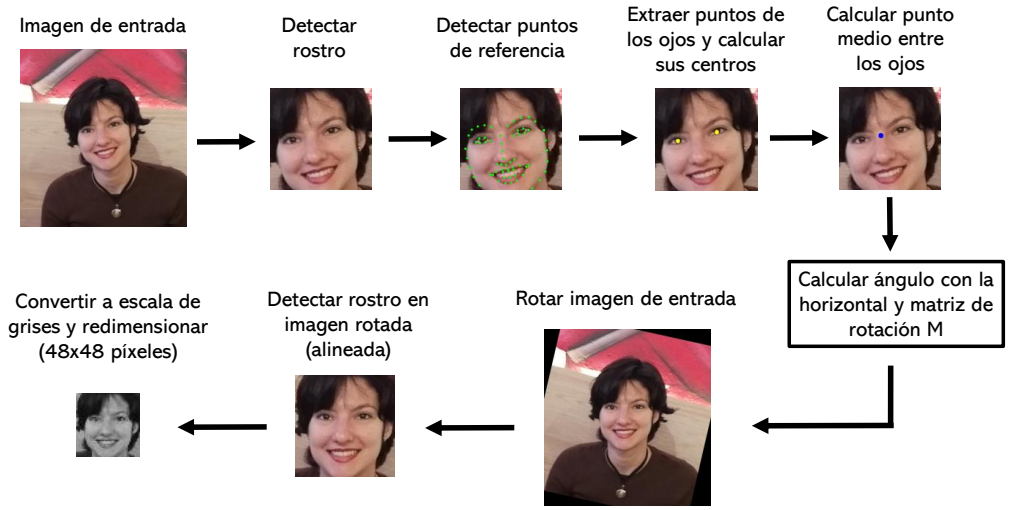


Figura 5.29: Pasos seguidos en el preprocesamiento de las imágenes

Como se observa en la Figura 5.29, para realizar el preprocesamiento el primer paso es detectar el rostro predominante en la imagen de entrada. Aquí se han utilizado los algoritmos de detección V&J, HOG y SSD, presentados en la Metodología (Sección 4.5). La salida de los tres algoritmos sobre una misma imagen de entrada puede verse en la Figura 5.30. De la Figura se observa la diferencia en la ROI generada por cada método. En el Capítulo 6 se analizará la variación de los resultados de clasificación en función de cada uno de los tres detectores empleados.

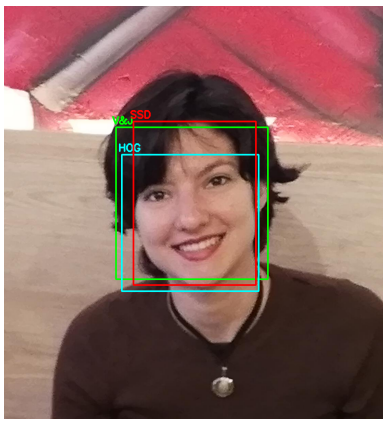


Figura 5.30: Resultados de detección de rostro mediante los algoritmos V&J (ROI verde), HOG (ROI cyan) y SSD (ROI rojo)

Luego de haber aplicado uno de los algoritmos de detección de rostros, si ha sido detectado más de un rostro en la imagen, se procede a calcular el área de cada una de las ROI obtenidas. Posteriormente, nos quedamos con la ROI a la cual corresponde el área mayor. Con esto se intenta garantizar que se analice solamente el rostro del estudiante frente a la webcam, asumiendo el mismo como el de mayor tamaño presente en la imagen a procesar (rostro

predominante).

Una vez que se tiene el rostro de mayor tamaño, se realiza el proceso de alineación. Para lograr esto, primeramente se extraen los puntos de referencia del rostro empleando el modelo de 68 puntos preentrenado en dlib (introducido en la Sección 4.6.2). Cuando se tienen las coordenadas *(x, y)* correspondientes a cada uno de los puntos de referencia en la imagen (Figura 5.31a), se procede a la extracción de las que representan los puntos característicos de los ojos. En el modelo utilizado, 12 de los puntos corresponden a los ojos, seis para el ojo izquierdo (puntos del 37 al 42) y seis para el ojo derecho (puntos del 43 al 48). Conociendo los índices de los puntos que se desean analizar, se obtienen las coordenadas de los ojos en la imagen y se realiza una media de los valores en cada ojo para obtener sus centros (Figura 5.31b).

(a) **(b)** **(c)**

Figura 5.31: Imágenes de izquierda a derecha: a) puntos de referencia en el rostro obtenidos con el modelo de 68 puntos de referencia; b) centro de cada ojo; c) punto medio entre los ojos

Teniendo el centro de cada ojo, se procede a calcular el ángulo formado entre la línea que pasa por los dos puntos centrales de los ojos y la horizontal. Este ángulo se calcula como:

$$\alpha = arctan(\frac{y_2 - y_1}{x_2 - x_1}), \tag{5.1}$$

donde x_1 e y_1 corresponden a las coordenadas *(x, y)* del centro del ojo izquierdo, x_2 e y_2 representan las coordenadas del centro del ojo derecho y α es el ángulo que se desea calcular en grados.

Además, se calcula el punto medio entre los ojos (Figura 5.31c) como el promedio de los puntos centrales obtenidos previamente. En esencia, este punto medio estará ubicado en la parte superior de la nariz y es el punto sobre el cual vamos a realizar el giro necesario en el rostro para alinearlo.

Ahora, con el punto sobre el cual realizar el giro y el ángulo requerido, se calcula la matriz de rotación M, con un factor de escalado isotrópico igual a 1 para mantener la escala de la imagen original. Posteriormente, se alinea el rostro en la imagen original (imagen de entrada), realizando una transformación afín de rotación que emplea la matriz M e interpolación bicúbica (en este paso se realiza la transformación afín sobre la imagen original, por lo cual se rota la imagen completa).

Aquí, la interpolación bicúbica fue escogida ya que ofrece menor pérdida de información al realizar la rotación de la imagen comparado con otros métodos como: interpolación del

vecino más cercano o bilineal. Sin embargo, la interpolación bicúbica, al tener en cuenta los 16 píxeles más cercanos, supone un mayor coste computacional. A pesar de esto, se ha seleccionado porque mayor resolución de la imagen representa mayor probabilidad de extraer características correctamente para la clasificación.

Finalmente, para obtener solamente la ROI correspondiente al rostro alineado, se vuelve a realizar la detección de rostros pero ahora sobre la imagen de entrada rotada (alineada). Con ello se genera una ROI cuyo contenido es el rostro predominante alineado y centrado (ver Figura 5.32). Por último, para finalizar el preprocesamiento de la imagen, la ROI obtenida se convierte a escalas de grises y se redimensiona a 48x48 píxeles. Al proceso de alineación de rostros presentado en esta Sección le hemos denominado Self-Calculation of Face Alignment (S-CFA).

Figura 5.32: Ejemplos de rostros antes y después de la alineación en la imagen de entrada

6 Experimentación

En este Capítulo se presentan experimentos realizados durante el desarrollo de la aplicación de RM y después de su implementación en HoloLens 2. Además, se analizan y discuten los resultados obtenidos. Para ello, en la Sección 6.1 se abordan las pruebas efectuadas con el objetivo de seleccionar el modelo de clasificación de emociones con mejores resultados. En la Sección 6.2 primero se exponen los experimentos realizados sobre la implementación de la aplicación y los resultados producidos. Luego, se analizan las limitaciones encontradas en dicha implementación. Por último, se presentan y comentan los resultados de una encuesta realizada teniendo en cuenta aspectos claves en el uso de la aplicación.

6.1 Modelos de clasificación de emociones

En este apartado se describen las pruebas realizadas con los modelos de clasificación de emociones: CNN, CNN+kNN y CNN+kNN+PCA, definidos en la Sección 5.6. En los experimentos se han empleado además distintos detectores de rostros (V&J, HOG, SSD), así como diferentes métodos de alineación (S-CFA, implementación de imutils), en el preprocesamiento de las imágenes. La finalidad de las pruebas aquí presentadas ha sido:

1. Evaluar los resultados obtenidos con las diferentes combinaciones de detectores de rostros, métodos de alineación y modelos de clasificación de emociones.
2. Seleccionar la configuración que presente la mejor opción de clasificación a usar en la aplicación de RM con HoloLens 2.

Para la realización de los experimentos, los modelos han sido entrenados y testeados sobre la base de datos AffectNet. A excepción del entrenamiento del modelo CNN donde se utilizaron pesos preentrenados obtenidos de [56]. De modo general, para efectuar las pruebas se siguieron los siguientes pasos:

1. Cargar los datos de entrenamiento y test de la base de datos AffectNet.
2. Redefinir los conjuntos de etiquetas (labels) tanto de los datos de entrenamiento como de test cargados, en correspondencia con los tres conjuntos de emociones a clasificar por los modelos presentados. A este proceso le hemos denominado **Agrupamiento**.
3. Detectar, alinear y normalizar los rostros de las imágenes cargadas (preprocesamiento).
4. Eliminar muestras donde no se detectó ningún rostro.
5. Importar los pesos pre-entrenados del modelo CNN.
6. Crear y compilar los modelos utilizados.

7. Entrenar y testear los modelos con los conjuntos de muestras resultantes del paso 4.
8. Calcular métricas de error sobre los resultados obtenidos.

Teniendo en cuenta los pasos seguidos, a continuación, se describen los experimentos realizados en el preprocesamiento de las imágenes y en la clasificación de emociones. Además de discutir los resultados obtenidos.

6.1.1 Preprocesamiento de los datos

Inicialmente, para el entrenamiento de los modelos se escogieron 20000 muestras del conjunto de datos de entrenamiento ofrecido por la base de datos empleada. El conjunto de test se definió con 3000 muestras de los datos de validación de AffectNet. Esta base de datos no presenta aún datos de test y sugiere utilizar el conjunto de validación como test.

AffectNet presenta 11 categorías, etiquetadas del 0 al 10 de la forma: **0- Neutral**, **1- Happy**, **2- Sad**, **3- Surprise**, **4- Fear**, **5- Disgust**, **6- Anger**, **7- Contempt**, **8- None**, **9- Urcentain**, **10- No-Face**. En el paso de agrupamiento las etiquetas del 0 al 7 fueron redefinidas dentro de las categorías: **0- Neutral**, **1- Negative Emotion** y **2- Positive Emotion** de la forma presentada en la Sección 5.6. Además, las muestras correspondientes a las etiquetas 8, 9 y 10 fueron eliminadas del conjunto de datos a utilizar. Esto se debió a que dichas etiquetas no presentaron información de interés en el pipeline de clasificación desarrollado (None: Ninguna de las ocho emociones, Uncertain: emoción no clasificada debido a elevada incertidumbre en su categoría, No-Face: Ningún rostro detectado en la imagen).

Posteriormente, se realizó la detección de rostros sobre las datos obtenidos del agrupamiento y se eliminaron las imágenes en las que no se produjo detección. La Tabla 6.1 indica la cantidad de muestras resultantes de cada proceso, siendo empleados los algoritmos V&J, HOG y SSD en la etapa de detección.

Detector de rostros	Número de muestras					
	Inicialmente		Agrupamiento		Detección	
	Train	Test	Train	Test	Train	Test
V&J	20000	3000	13889	2196	10959	1690
HOG	20000	3000	13889	2196	13225	2078
SSD	20000	3000	13889	2196	10553	1595

Tabla 6.1: Número de muestras inicialmente escogidas del dataset AffectNet para realizar los experimentos, seguido del número de muestras resultantes de la agrupación de emociones (3 conjuntos) y la eliminación de las clases **None, Uncertain** y **No-Face** del dataset. Por último, número de muestras después de aplicar la detección de rostros, eliminando las muestras donde el algoritmo no obtuvo detección.

Para completar el preprocesamiento, los rostros detectados fueron alineados utilizando los métodos: S-CFA y el implementado en imutils, para finalmente ser normalizados. La Tabla 6.2 muestra los tiempos de ejecución producidos en el prepocesamiento. Estos tiempos han sido determinados como la suma de los tiempos de ejecución presentados por los conjuntos de detector de rostros y método de alineación. Los experimentos fueron realizados en un

ordenador personal con 8 GB de RAM y procesador Intel Core i5-6200U CPU @ 2.30GHz×4.

Detector de rostros	Método de alineación	Tiempo preprocesamiento	
		Train	Test
V&J	S-CFA	1754.93 s	257.86 s
	imutils	1169.39 s	183.00 s
HOG	S-CFA	41913.66 s	5956.98 s
	imutils	14669.96 s	2191.32 s
SSD	S-CFA	1323.68 s	209.22 s
	imutils	1053.96 s	174.46 s

Tabla 6.2: Tiempos de preprocesamiento obtenidos para diferentes combinaciones de detector de rostros y método de alineación

Con los resultados generados es posible llegar a varias conclusiones comentadas a continuación. Primeramente, de la Tabla 6.1 se observa la considerable reducción de los conjuntos de entrenamiento y test después de realizar el paso de agrupamiento. Con respecto a la detección de rostros, de la misma Tabla se aprecia que el algoritmo HOG presentó los mejores resultados, logrando detectar rostros en un número mayor de muestras que los algoritmos V&J y SSD. Sin embargo, en la Tabla 6.2 se tiene que el tiempo de ejecución requerido por el método HOG fue significativamente mayor que el de los restantes (más de 10 veces).

Por el contrario, el algoritmo SSD presentó los menores tiempos de ejecución, siendo el método más rápido pero a su vez el menos preciso (menor número de detecciones de rostros). El detector V&J mostró resultados intermedios con respecto a los otros dos métodos de detección. Con esto se hace referencia a que produjo mayor número de ROI que SSD pero bastante menos que HOG. De igual manera, presentó mayores tiempos de ejecución que SSD pero considerablemente menores que HOG.

Con respecto a los métodos de alineación utilizados, en la Tabla 6.2 se observa que el tiempo de preprocesamiento requerido al utilizar la aproximación S-CFA es mayor que el tiempo de ejecución presentado por el método de imutils. Esto se debe a que en el primero de los algoritmos la detección de rostros se realiza dos veces, lo cual enlentece el proceso ya que la etapa de detección es la que presenta mayor coste computacional y por ende, mayor consumo de tiempo. Lo explicado es realmente notable en el caso donde el algoritmo de detección es lento, como se tiene en el método HOG.

Teniendo en cuenta los criterios analizados, a modo de resumen se tiene:

- V&J: presenta número de detecciones y tiempos de ejecución intermedios
- HOG: más lento pero más preciso (mayor número de detecciones)
- SSD: más rápido, menos preciso (menor número de detecciones)
- S-CFA: mayores tiempos de ejecución que la implementación de alineación de rostros en imutils

Los tiempos de ejecución requeridos constituyen un factor importante en la selección de la configuración final: detector + método de alineación + clasificador, ya que se desea que

la aplicación de RM funcione en tiempo real. Pero también, debe tenerse en cuenta la exactitud (accuracy) alcanzada por los modelos de clasificación. A continuación se analizan los resultados obtenidos en la etapa de clasificación para poder determinar la configuración más adecuada para la aplicación desarrollada.

6.1.2 Resultados de clasificación

Teniendo los rostros detectados, alineados y normalizados (en ambos conjuntos de entrenamiento y test), se realizó el entrenamiento de los modelos y posteriormente, su testeo. La Tabla 6.3 muestra los resultados de clasificación obtenidos por las diferentes configuraciones implementadas en base a su accuracy. En la Tabla, el número seguido del algoritmo *k*-NN hace referencia al número de vecinos empleado.

Detector de rostros	Método de alineación	Accuracy					
		CNN	kNN3	kNN5	kNN7	kNN9	kNN5PCA
V&J	S-CFA	58.52 %	51.42 %	53.02 %	52.96 %	54.14 %	25.86 %
	imutils	51.74 %	51.45 %	54.13 %	56.26 %	55.55 %	21.40 %
HOG	S-CFA	56.26 %	50.19 %	49.95 %	50.10 %	49.85 %	33.95 %
	imutils	51.59 %	53.95 %	53.85 %	55.73 %	56.98 %	23.58 %
SSD	S-CFA	43.07 %	40.94 %	40.50 %	39.31 %	37.81 %	25.81 %
	imutils	47.64 %	37.95 %	37.89 %	37.83 %	37.47 %	24.03 %

Tabla 6.3: Resultados de accuracy obtenidos con los modelos de clasificación de emociones para diferentes combinaciones de detectores de rostro y métodos de alineación

El resultado de accuracy reportado ha sido calculado como el por ciento de acierto alcanzado por los modelos de clasificación para los conjuntos de rostros obtenidos por cada uno de los algoritmos de detección. En esta Sección se evalúan los modelos de clasificación de emociones y no los de detección de rostros, por lo cual, los rostros no detectados no han sido contados como error para determinar el accuracy.

De la Tabla 6.3 se observan los siguientes resultados con respecto al accuracy alcanzado por los modelos de clasificación:

Resultados CNN: El mejor resultado obtenido fue producido por el modelo CNN con la combinación V&J + S-CFA, presentando el mayor valor de accuracy (58.52 %). Aunque el valor continúa pareciendo un porcentaje bajo, se debe tener en cuenta que el mejor resultado de clasificación obtenido sobre AffectNet para 8 emociones es de 62 %. Lo cual puede verse en [57]. Si bien aquí no han sido clasificadas las 8 emociones por separadas, si se han tenido en cuenta en la agrupación realizada, por lo que podemos tener cierta referencia de comparación.

Resultados CNN+kNN: En el modelo CNN+*k*-NN, al variar el número de vecinos se obtuvieron resultados similares, lo cual indica que los resultados obtenidos son fiables y que el modelo no presenta un sobre-entrenamiento. Esto se entiende porque cuando el algoritmo *k*-NN presenta sobre-entrenamiento, su resultado es dependiente del valor de *k* utilizado. La configuración HOG + imutils + *k*NN9 presentó el segundo mejor

accuracy de clasificación.

Resultados PCA: El modelo CNN+k-NN+PCA arrojó los peores porcentajes, presentando valores bastante bajos. Se entiende que estos porcentajes de clasificación fueron obtenidos debido a que la salida de la red convolucional profunda empleada para la extracción de características, devuelve datos con elevada correlación (vector de características con gran correlación). Por lo cual, al reducir la dimensión de las muestras (de 2048 a 100) podemos estar eliminando características importantes para la clasificación.

Además, de la Tabla 6.3 se aprecia que el detector de rostros con peores resultados en la clasificación fue SSD. De aquí se asume que el bajo accuracy presentado se debe a que las ROI obtenidas mediante SSD presentan una forma rectangular muy marcada. Esto conlleva a una gran deformación del rostro al redimensionar la ROI a 48x48 píxeles (forma cuadrada). En esta deformación se pueden perder o modificar características importantes para la correcta clasificación de emociones.

Análisis de confusión

Ahora, se analiza la confusión presente en los cuatro mejores resultados de accuracy obtenidos. Dichos resultados han sido resaltados en la Tabla 6.3 y sus matrices de confusión son mostradas en la Figura 6.1. De la Figura se aprecia que en todos los casos se alcanza un alto nivel de correcta clasificación para la categoría *Positive Emotion*. Sin embargo, a excepción de la configuración V&J + S-CFA + CNN, las restantes presentan confusión en las categorías *Neutral* y *Negative Emotion*.

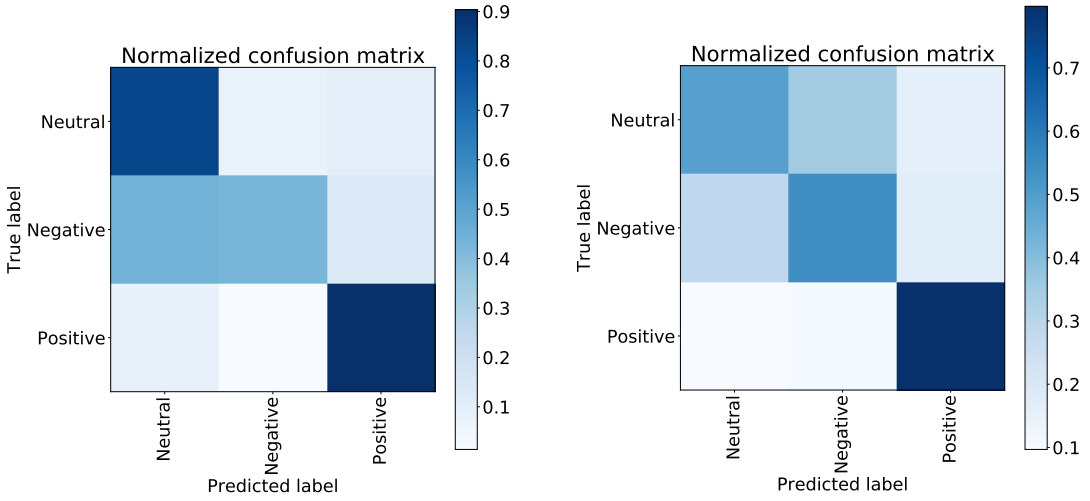
Otro detalle observado de la Figura 6.1 es que si bien el detector HOG obtiene mayor número de detecciones que el V&J (Sección 6.1.1), genera peores resultados en la matriz de confusión. De igual manera se aprecia que las combinaciones que emplean la alineación de rostros mediante el módulo imutils, presentan mayor confusión que las que utilizan el método S-CFA.

Finalmente, de la Figura 6.1 se concluye que la configuración V&J + S-CFA + CNN logró los mejores resultados de clasificación, ya que a diferencia de las restantes combinaciones, indicó confusión únicamente de la clase *Negative Emotion* con *Neutral* y no en viceversa. Además, presenta los mayores niveles de acierto en la clasificación (mayor que 0.8) para las categorías: *Neutral* y *Positive Emotion*.

Tiempos de ejecución

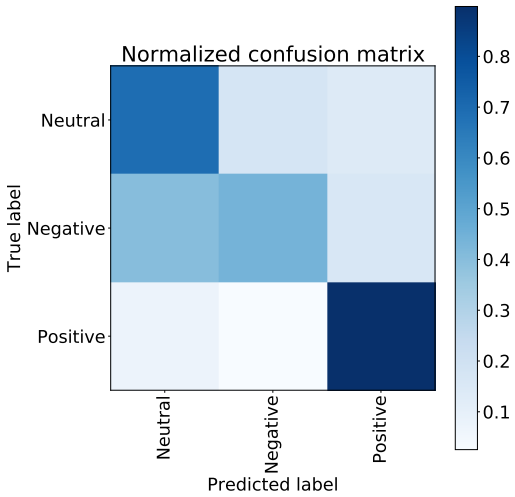
Por último, analizaremos los tiempos de ejecución generados en la clasificación para concluir con la selección de la configuración más adecuada. Para ello, en la Tabla 6.4 se indican los tiempos de ejecución presentados por los modelos de clasificación. En los modelos que emplean k-NN fueron sumados los tiempos de ejecución producidos en las etapas de extracción de características y de clasificación.

De las pruebas realizadas se observó que los resultados de clasificación fueron obtenidos en menor tiempo de ejecución por el modelo CNN con respecto al CNN+k-NN. Esto se debe

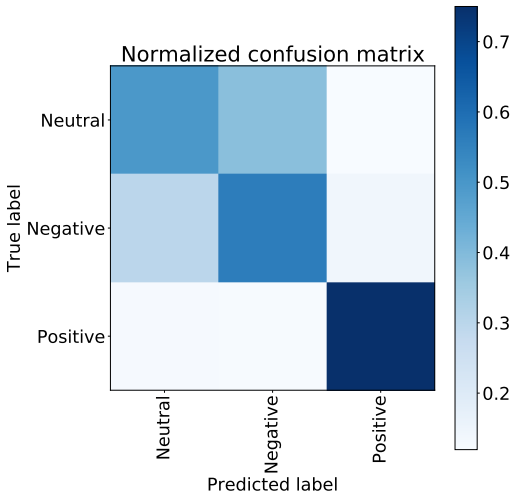


(a) V&J + S-CFA + CNN

(b) V&J + imutils + CNN+kNN_7



(c) HOG + S-CFA + CNN



(d) HOG + imutils + CNN+kNN_9

Figura 6.1: Matrices de confusión normalizadas de las cuatro combinaciones de detectores, métodos de alineación y modelos de alineación, que presentaron los mayores valores de accuracy

a que la implementación de k-NN mediante el módulo scikit-learn no está optimizada para generar bajos tiempos de ejecución.

En el caso del modelo CNN+k-NN+PCA se observa, que logró disminuir considerablemente los tiempos de ejecución al reducir la dimensión del vector de características. Sin embargo, como se comentó anteriormente esto conllevó a valores de accuracy muy bajos.

Con respecto a la influencia de las combinaciones de detectores y métodos de alineación en los tiempos de ejecución de la clasificación, la diferencia presentada se debe principalmente a la variación en la cantidad de datos resultantes del preprocesamiento. Con esto hacemos

Detector de rostros	Método de alineación	Tiempo de ejecución					
		CNN	kNN3	kNN5	kNN7	kNN9	kNN5PCA
V&J	S-CFA	3.54 s	98.46 s	103.95 s	101.80 s	101.01 s	3.22 s
	imutils	5.73 s	97.39 s	98.65 s	99.01 s	99.39 s	4.70 s
HOG	S-CFA	5.02 s	120.85 s	120.91 s	122.33 s	122.87 s	3.61 s
	imutils	5.41 s	123.02 s	122.75 s	121.19 s	122.89 s	3.28 s
SSD	S-CFA	4.09 s	82.81 s	80.40 s	79.94 s	79.97 s	2.71 s
	imutils	6.40 s	94.55 s	95.01 s	95.20 s	97.66 s	5.54 s

Tabla 6.4: Tiempos de ejecución obtenidos con los modelos de clasificación de emociones para diferentes combinaciones de detectores de rostro y métodos de alineación

referencia al número de ROI producidas por cada algoritmo de detección. Por lo que menores tiempos se deben a que son analizadas menores cantidades de muestras. Teniendo esto en cuenta, se puede decir que los modelos de clasificación no presentan gran variación en sus tiempos de ejecución en función de la combinación de algoritmos empleada en el preprocesamiento, sino de la cantidad de muestras analizadas.

En general se observa que los modelos de clasificación funcionan bastante rápido, aunque es considerablemente más lenta la implementación de los modelos con k-NN. Pero lo que realmente supone mayores tiempos de ejecución es la etapa de preprocesamiento con la detección y alineación del rostro (Sección 6.1.1).

A modo de resumen se tiene:

- HOG presentó mejor detección de rostros pero peor clasificación de emociones que al emplear el detector V&J.
- La alineación de rostros con imutils presentó menores tiempos de ejecución que S-CFA en el preprocesamiento pero no mejores resultados en la clasificación que con S-CFA.
- El mayor valor de accuracy fue alcanzado por la combinación V&J +S-CFA + CNN con 58.52 %.
- Dicha combinación alcanzó los valores más elevados de acierto en la clasificación de las categorías *Neutral* y *Positive Emotion*.
- El modelo CNN+kNN presentó mayores tiempos de ejecución debido a la implementación no optimizada de k-NN en scikit-learn.
- El modelo que emplea PCA obtuvo bajo porcentaje debido a la alta correlación de los vectores de características.

Por todo lo antes expuesto, se ha concluido que la configuración "V&J + S-CFA + CNN" logró los mejores resultados tanto de accuracy como en la matriz de confusión, presentando además bajos tiempos de ejecución. Por lo cual, después de los experimentos realizados, se ha seleccionado dicha combinación como la más adecuada para ejecutar la tarea de clasificación de emociones, en la aplicación de RM desarrollada para HoloLens 2.

6.2 Validación de la aplicación en HoloLens 2

En esta Sección se presenta un conjunto de experimentos realizados para validar y analizar el funcionamiento de la aplicación en HoloLens 2. Para desarrollar las pruebas, fueron empleados dos laboratorios del grupo de investigación Robotics & Tridimensional Vision (RoViT), de la Universidad de Alicante (UA). Estos espacios se usaron a modo de simular la comunicación remota entre el profesor y el estudiante.

En el primer laboratorio se estableció el conjunto de servidores necesarios sobre un sistema con GPU y, en el segundo laboratorio se realizaron las pruebas con HoloLens 2. Para simplificar a nivel de experimentación, en el servidor principal se integró el servidor webcam. De esta forma, para las pruebas se determinó que en el laboratorio 1 se encontraba el "estudiante en remoto" y en el laboratorio 2 el "profesor" (Figura 6.2).

Figura 6.2: Vistas de los laboratorios empleados en los experimentos

Un vídeo demostrativo del funcionamiento de la aplicación en el entorno creado se puede encontrar en `https://youtu.be/K54IApErf50`. En el sistema mostrado, se creó una videoconferencia mediante Google Meet, para que el estudiante recibiera la clase desde su ordenador (en laboratorio 1). Paralelamente, el profesor podía visualizar al estudiante en tiempo real (en el laboratorio 2) mediante el uso de la aplicación en HoloLens 2.

A continuación, se presentan resultados obtenidos tras un conjunto de pruebas realizadas para analizar las tareas de clasificación de emociones y segmentación de personas durante la ejecución de la aplicación. Los experimentos fueron desarrollados con diferentes personas de forma que permitiera estudiar la robustez de la implementación. Las muestras estuvieron caracterizadas por un total de ocho personas, de ellas, tres profesores y cinco estudiantes. De igual modo todos los participantes probaron el funcionamiento de la aplicación tanto del lado del estudiante como del lado del profesor para luego manifestar sus puntos de vista al respecto. Todos los resultados mostrados fueron capturados desde la vista de la aplicación en HoloLens 2.

6.2.1 Resultados correctos de clasificación y segmentación

Las Figuras 6.3 y 6.4 muestran ejemplos de resultados de clasificación alcanzados para la categoría *Neutral*. En la Figura 6.3 se aprecia la correcta clasificación en todas las muestras, incluso para el caso donde la persona tiene las manos alrededor de la cara (imagen abajo-derecha). Además de la correcta clasificación obtenida, también se observan adecuados resultados en la segmentación de las personas proyectadas.

Figura 6.3: Ejemplos de correctos resultados de clasificación de la categoría *Neutral* y segmentación de la persona proyectada en la aplicación implementada en HoloLens 2

En las dos imágenes de la Figura 6.4, se aprecia que aunque se tienen elementos que presentan cierta oclusión con la persona proyectada (teléfono en imagen de la izquierda y tasa en la imagen de la derecha), se sigue realizando la adecuada segmentación. En la imagen de la derecha se analiza que el algoritmo de detección de rostros no ha podido detectar ninguna cara debido a la oclusión con la tasa. Por lo cual, tampoco se ha podido ejecutar el método de clasificación de emociones. Sin embargo, en la aplicación se muestra el último estado clasificado. Con ello se entiende que en el instante anterior a la toma de la tasa, la emoción de la persona fue clasificada como *Neutral*. En la aplicación el mismo emojie se continúa mostrando hasta que se vuelva a detectar el rostro y se pueda clasificar la emoción.

Figura 6.4: Ejemplos de clasificación de la categoría *Neutral* en la aplicación implementada en HoloLens 2, En este caso se tienen elementos que presentan oclusión parcial con la persona proyectada, pero se continúa observando la correcta segmentación

Figura 6.5: Ejemplos de correctos resultados de clasificación de la categoría *Negative Emotion* y segmentación de la persona proyectada en la aplicación implementada en HoloLens 2

En la Figura 6.5 se muestran correctos ejemplos de de clasificación de la categoría *Negative Emotion*. De la Figura se puede apreciar la obtención de acertadas clasificaciones incluso con elementos como gafas (imagen abajo-izquierda) y manos situadas alrededor del rostro (imagen abajo-derecha). Aquí también se observan adecuados resultados en la segmentación de la persona proyectada.

La Figura 6.6 refleja acertados resultados de clasificación de la categoría *Positive Emotion*. De la Figura se aprecia la correcta clasificación de la emoción ante cierto nivel de movimiento lateral de la cabeza de la persona proyectada (imagen abajo-derecha). Nuevamente, se tiene una adecuada segmentación del estudiante conectado en remoto desde el laboratorio 1.

Figura 6.6: Ejemplos de correctos resultados de clasificación de la categoría *Positive Emotion* en la aplicación implementada en HoloLens 2. Se observa además, adecuada segmentación de la persona proyectada (estudiante frente a la webcam en el laboratorio 1)

En la Figura 6.7 se muestra el resultado de acercarse y posteriormente alejarse el estudiante de la webcam. Aquí se analiza que el tamaño de la representación de la persona varía en función de la cercanía con la webcam debido a que varía la proporción del estudiante en la imagen capturada. De igual modo se siguen obteniendo resultados adecuados de segmentación y clasificación. Además, en esta prueba el estudiante ha cerrado los ojos y se aprecia como incluso en este caso se ha logrado la correcta clasificación de la emoción.

Figura 6.7: Resultado de acercarse y alejarse el estudiante de la webcam

6.2.2 Resultados en fotogramas con gestos

Las Figuras 6.8 y 6.9 muestran resultados obtenidos con las personas proyectadas realizando determinados gestos con las manos. Este experimento se ha realizado con la finalidad de observar el comportamiento del algoritmo de segmentación ante eventos como los mostrados, ya que gestos como indicar algo de la pizarra, levantar la mano o situarla cerca de la cara, pueden ser cotidianos en el desenvolvimiento de una clase. Los casos presentados han sido considerados con una segmentación, que aunque no perfecta en todas las muestras, suficientemente adecuada para transmitir el gesto deseado.

Figura 6.8: Resultados de segmentación de una persona realizando gestos con la mano frente a la webcam, vistos desde la aplicación en HoloLens 2

Figura 6.9: Resultados de segmentación de tres personas diferentes, realizando gestos con la mano frente a la webcam, vistos desde la aplicación en HoloLens 2

6.2.3 Limitaciones

De todos los experimentos realizados donde se tuvieron estados neutrales y positivos, se puede decir que de forma general los resultados obtenidos para esas clases fueron acertados. Sin embargo, se presentaron algunos errores en la incorrecta clasificación de emociones negativas en neutrales. Esto tiene sentido ya que se corresponde a los resultados obtenidos en la matriz de confusión generada por la combinación "V&J + S-CFA + CNN" seleccionada en la Sección anterior (6.1.2). En la Figura 6.10 se muestran dos ejemplos de clasificación errónea de la categoría *Negative Emotion* como *Neutral*.

Por otro lado, en algunos casos se mostraron determinados errores en la segmentación como los mostrados en la Figura 6.11. Estos errores se debieron principalmente a dos factores:

1. Situaciones donde el fondo se caracterizó por colores con gran parecido a los presentes en la piel o ropa del estudiante, sobre todo en pequeñas zonas de fondo contenidas dentro de cierta pose de la persona (primera imagen de la Figura 6.11).
2. Determinados gestos, como movimientos del brazo en alto, ante los cuales el algoritmo de segmentación no acierta en todos los casos (segunda imagen de la Figura 6.11).


Figura 6.10: Errores en la clasificación de emociones. Aquí los fallos se presentaron principalmente en la errónea clasificación de emociones negativas en neutrales


Figura 6.11: Errores en la segmentación

6.3 Encuesta realizada

Luego del desarrollo de los experimentos, se realizó una encuesta online donde participaron siete de las personas que probaron la aplicación. Todas las personas de la encuesta experimentaron el funcionamiento de la aplicación tanto desde el punto de vista del estudiante como desde el del profesor. La encuesta, diseñada mediante Google Form, se creó con el objetivo de tener retroalimentación de las opiniones de los participantes y por tanto, una valoración cualitativa de las experiencias desarrolladas. Dicha encuesta se confeccionó teniendo en cuenta indicadores de simplicidad, rendimiento, realismo, aporte de la clasificación de emociones y aplicabilidad. Con ello, se definieron las cinco preguntas siguientes:

1. ¿Cuán atractivo y/o sencillo de usar te pareció el entorno diseñado?
2. ¿Cómo calificarías el rendimiento de la interfaz?

3. ¿Qué nivel de realismo consideras presente en la aplicación?
4. ¿Valoras como un aporte positivo la clasificación de emociones?
5. ¿Cuán aplicable consideras la propuesta presentada?

Todas las preguntas tienen un rango de calificación de 1 a 5 (números enteros), donde 1 se consideró como el mayor nivel de insatisfacción y 5 como la mejor valoración. La Tabla 6.5 refleja los resultados obtenidos después de efectuar la encuesta. Aquí, para las preguntas realizadas, se indica el número de personas que seleccionaron cada nivel de calificación. Además, se muestra el por ciento que representa dicha cantidad de personas con respecto al total de participantes.

	Calificación				
No. Pregunta	1	2	3	4	5
1	0 (0 %)	0 (0 %)	0 (0 %)	1 (14,3 %)	6 (85.7 %)
2	0 (0 %)	0 (0 %)	2 (28,6 %)	2 (28,6 %)	3 (42,9 %)
3	0 (0 %)	0 (0 %)	1 (14,3 %)	5 (71,4 %)	1 (14,3 %)
4	0 (0 %)	1 (14,3 %)	0 (0 %)	1 (14,3 %)	5 (71,4 %)
5	0 (0 %)	0 (0 %)	1 (14,3 %)	3 (42,9 %)	3 (42,9 %)

Tabla 6.5: Número de personas por calificación obtenida en cada una de las cinco preguntas de la encuesta realizada. El por ciento corresponde a la cantidad de personas que seleccionaron una calificación con respecto al total de participantes

Con los resultados obtenidos en la Tabla 6.5 se calculó el promedio de las calificaciones generadas por los participantes para cada pregunta. Este promedio es mostrado en la gráfica de la Figura 6.12. Además de la encuesta se realizaron entrevistas a los participantes para tener una mayor retroalimentación de sus opiniones. A continuación serán discutidos los resultados de la encuesta y se comentarán algunos de los criterios emitidos en las entrevistas.

De la Figura 6.12 se observa que el promedio de calificaciones obtenido para todas las preguntas fue elevado, obteniéndose valores a partir de 4 puntos. La calificación más alta la obtuvo la primera pregunta, significando que la aplicación logró mostrar un entorno simple y atractivo para los diferentes usuarios. De la Tabla 6.5 se observa que seis de los siete encuestados valoraron esta pregunta con la máxima calificación y solamente uno con 4 puntos. Lo cual denotó un elevado nivel de empatía con el indicador analizado. Este punto es muy importante debido a que una aplicación con elevado nivel de complejidad en su diseño puede conllevar a la desmotivación por parte de los profesores en su uso. Una interfaz no intuitiva o engorrosa puede recrear un entorno no agradable, además de requerir una elevada curva de aprendizaje para su manipulación.

El segundo indicador con la valoración más alta fue el empleo de la clasificación de emociones. De manera general, los encuestados consideraron como un valor añadido el uso de dicha tarea ya que les brindó información adicional del estado del estudiante conectado. Sin embargo, como se muestra en la Tabla 6.5 una persona no coincidió con este criterio. En las entrevistas realizadas los usuarios manifestaron su agrado hacia la clasificación de emociones,

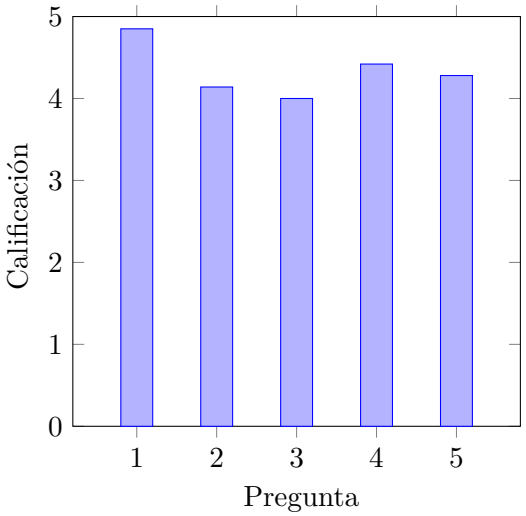


Figura 6.12: Promedio de calificaciones obtenidas en la encuesta de 5 preguntas, realizada a 7 personas que realizaron pruebas con la aplicación. Las personas experimentaron realizando la función tanto de estudiante como de profesor. En el rango de valores de calificación, el 5 es considerado como la mejor evaluación y el 1 como la peor

considerando que permite al profesorado tener cierta idea de cómo están reaccionando los alumnos ante el contenido y/o método de impartirlo, de forma que pueda modificarlos en tiempo real en función del feedback de emociones recibido.

La aplicabilidad de la propuesta también presentó alta puntuación. Esto indicó que el sistema mostrado no solo fue valorado como interesante a nivel de experimentación sino que además fue considerado su potencial futuro como aplicación a implementar en entornos educativos. Aquí es de destacar la facilidad que ofrece la aplicación para interactuar con el escenario mixto, permitiendo al profesor desenvolverse en el entorno con naturalidad, a la vez que visualiza al estudiante online. Al mismo tiempo esto proporciona una interacción menos fría entre el profesor y los alumnos (tanto online como presenciales), contribuyendo positivamente a la impartición y adquisición de los conocimientos.

Ahora presentaremos concretamente tres de las opiniones de los usuarios entrevistados:

Usuario 1: Considero que el sistema probado constituye una tecnología aplicable, presentando gran dinámica de cara a personas que no puedan estar presentes en clases. Creo que el uso del emoticono es un aporte novedoso para identificar la emoción de la persona en clase.

Usuario 2: Representa una tecnología muy interesante sobre todo a nivel de aplicarlo en clases online y el mero hecho de que pueda verse al alumno con el emojie representativo de la emoción puede hacer variar la clase en función de adaptarla a los alumnos. Por lo cual lo considero muy realizable en un futuro.

Usuario 3: Es un sistema muy fácil de usar, bastante cómodo para lo que es la universidad y para garantizar la asistencia a clases de personas que por algún motivo no pueden asistir. Presenta el problema del coste del dispositivo y la duración de la batería, pero

son problemas secundarios que se podrían solucionar a largo plazo. Creo que el sistema funciona muy bien, constituyendo un aporte adicional para el alumnado y los profesores ya que al trabajar a través de Zoom u otra plataforma online de videoconferencia, el profesor no puede ver a los alumnos ni saber si se están quedando con la información que se les está dando. También el análisis de emociones es importante ya que de cierto modo transmite como están tratando los alumnos con el contenido que se les está enseñando. Por lo cual lo considero como una buena herramienta para utilizar en un futuro en las instituciones universitarias, pero además en reuniones como por ejemplo en empresas cuando sea necesario hacer algo más interactivo.

De estas entrevistas también se aprecian opiniones positivas acerca de los indicadores destacados en los resultados anteriores: 1) atractivo/simplicidad de la aplicación, 2) empleo de la clasificación de emociones y, 3) aplicabilidad de la propuesta. Sin embargo, se encontró como problema el coste del dispositivo y la duración de la batería. Es de reconocer que el dispositivo presenta un precio de mercado no bajo (alrededor de 3 mil euros) y se espera que en el transcurso de un tiempo determinado su coste vaya disminuyendo de forma que pueda ser más asequible por las universidades.

Con respecto a la batería del dispositivo, su duración es de entre 2 y 3 horas en uso activo. La misma puede ser completamente recargada en menos de 65 minutos cuando el dispositivo está en reposo mediante un cargador de pared. Pero este tiempo no sería suficiente para permitir la docencia de forma continuada en un período de tiempo mayor. Como solución a este inconveniente, HoloLens 2 acepta utilizar simples baterías externas (siempre que proporcionen como mínimo 5V/1.5A), permitiendo la total funcionalidad del dispositivo mientras se carga. Esto permite que con la adición de una batería externa en un bolsillo de una camisa o un pantalón, conectada mediante un cable USB a las HoloLens, pueda utilizarse el dispositivo continuamente durante el tiempo requerido.

Por otro lado, de la Tabla 6.5 y Figura 6.12 se observa que los indicadores relativos al rendimiento y nivel de realismo fueron menos puntuados. Por lo cual pueden ser considerados indicadores a mejorar y a continuación se realizan algunos comentarios.

La pregunta con menor calificación fue la referente al nivel de realismo conseguido en la aplicación. Es cierto que el holograma de la persona no es completo, apareciendo solo la parte del cuerpo que es capturada por la webcam. La parte inferior del cuerpo que debería estar situada por debajo de la mesa, no es visualizada, afectando el contexto real observado. Si bien esto es una limitante, es una consecuencia de emplear los requisitos de hardware mínimos por parte del estudiante, como es el empleo de una simple y única webcam. Además de no requerir procesos previos de capturas del cuerpo completo antes de comenzar la clase para guardar esos datos y luego emplearlos en alguna forma de visualización. Todo esto garantiza menores tiempos para comenzar a emplear el sistema. A la vez que aumenta su uso por parte del alumnado al no imponer grandes requerimientos de hardware y por ende, de coste.

El nivel de realismo pudiese mejorarse con el empleo de cámaras de captación 3D, como una LiDAR o Kinect, o con el uso de más de una webcam. Pero la utilización de tecnologías más complejas de captación de imágenes, incumpliría con las ideas del sistema planteadas en el párrafo anterior. Además, teniendo en cuenta que los alumnos se encuentran a una determinada distancia del profesor en el aula, tampoco se garantizaría la ganancia de consi-

derables detalles al emplear visualización 3D. Una opción más viable pudiese ser interesante evaluar sería utilizar algún tipo de diseño predefinido para la parte inferior del cuerpo en la visualización.

Por otro lado, en determinadas ocasiones el nivel de realismo se ve afectado también por situaciones de incorrecta segmentación de la persona, como los analizados en la Sección 6.2.3. Esto es un inconveniente que limita la forma en que el docente percibe la persona proyectada en el momento en el que sucede la inadecuada de segmentación.

Con respecto al rendimiento de la aplicación, si bien hemos logrado que funcione en tiempo real, las HoloLens presentan requerimientos elevados en la velocidad de visualización de los fotogramas (60 fps). Por lo cual en determinados fotogramas puede aparecer cierta demora en la actualización de los hologramas con respecto al estado estado real del estudiante. Esto no significa que la aplicación se paralice o ralenticen sus representaciones, ya que si no se tiene nueva información se continúa mostrando la última recibida, pero puede que esta información tenga cierto atraso con respecto al estado actual de la persona conectada.

La demora anteriormente descrita se debe al tiempo de ejecución requerido por los algoritmos de segmentación de personas y clasificación de emociones implementados, además de los tiempos de comunicación. Una posible mejora pudiera ser usar un sistema de cómputo aún más potente en los servidores creados, o bien optimizar la implementación de las tareas desarrolladas en ellos mediante un lenguaje de programación que permita menores tiempos de ejecución como es C++.

Como resumen de las valoraciones obtenidas de la aplicación, es posible decir que los indicadores de atractivo/sencillez del entorno de RM, el uso de la clasificación de emociones y la aplicabilidad del sistema propuesto, lograron altos índices de aceptación y calificación. Por otro lado, aspectos como el nivel de realismo y el rendimiento, son factores en los que aún se deben trabajar para lograr el desempeño óptimo de la aplicación.

7 Conclusiones

Al término del presente trabajo es posible concluir que los objetivos planteados al inicio del mismo (Capítulo 3) fueron logrados. Con respecto al objetivo general, se logró desarrollar un sistema de cyberpresencia para entornos educativos, que mediante el empleo de tecnologías de RM, permite integrar en un espacio común la presencialidad con la no presencialidad. De tal forma, el proyecto presentado ha conseguido la comunicación remota entre dos espacios físicos diferentes en tiempo real, presentando un novedoso apoyo al profesorado universitario y a la Educación Superior.

Teniendo en cuenta los objetivos específicos propuestos, se logró diseñar, implementar y evaluar un sistema de reconocimiento de emociones, alcanzándose un 58.52 % de accuracy sobre un conjunto de test de la base de datos AffectNet. La configuración empleada obtuvo elevados resultados de clasificación en las categorías *Positive Emotion* y *Neutral*, sin embargo, presentó confusión en la clase *Negative Emotion* con la *Neutral*.

Al mismo tiempo, mediante el empleo de las gafas HoloLens 2, se logró realizar una metodología de cyberpresencia basada en el uso de hologramas para representar tanto al estudiante en remoto como su emoción. Los experimentos realizados con profesores y estudiantes, mostraron la correcta segmentación de los alumnos con el algoritmo utilizado, en gran parte de los casos analizados. Sin embargo, se encontraron limitaciones en situaciones donde el fondo se caracterizó por colores similares a los presentes en el estudiante; y ante determinados gestos, como movimientos del brazo en alto.

La evaluación cualitativa del sistema (encuesta), indicó un elevado nivel de satisfacción con la sencillez del entorno de RM desarrollado, así como del uso de la clasificación de emociones y la aplicabilidad del sistema propuesto. En cambio, el nivel de realismo y el rendimiento, son factores en los que se debe seguir trabajando para lograr el desempeño óptimo de la aplicación.

Por último, decir que con la realización de este proyecto se cumplió con la motivación personal considerada al inicio del trabajo.

8 Trabajos Futuros

Como trabajos futuros se propone desarrollar una arquitectura de comunicación web, que sobre las bases del sistema presentado, permita la interacción con múltiples clientes webcam de forma dinámica. El objetivo sería hacer posible la conexión de varios estudiantes al mismo tiempo sin que el servidor principal tenga de forma predefinida la cantidad de estudiantes que se van a conectar. Como segunda propuesta se tiene obtener información de la vista de las gafas HoloLens 2 en tiempo de ejecución de la aplicación, para realizar la clasificación de las emociones de todos los estudiante en la clase (presenciales y no presenciales). Conjuntamente se propone mejorar los resultados del algoritmo de clasificación de emociones empleado en la clase **Negative Emotion** . Por último, se propone proyectar la representación del estudiante con método que no requiera el empleo de un patrón.

Bibliografía

[1] D. Ungureanu, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger, *et al.*, "Hololens 2 research mode as a tool for computer vision research," *arXiv preprint arXiv:2008.11239*, 2020.

[2] S. Condino, G. Turini, P. D. Parchi, R. M. Viglialoro, N. Piolanti, M. Gesi, M. Ferrari, and V. Ferrari, "How to build a patient-specific hybrid simulator for orthopaedic open surgery: benefits and limits of mixed-reality using the microsoft hololens," *Journal of healthcare engineering*, vol. 2018, 2018.

[3] S. Wish-Baratz, A. P. Gubatina, R. Enterline, and M. A. Griswold, "A new supplement to gross anatomy dissection: Holoanatomy," *Medical Education*, vol. 53, no. 5, pp. 522–523, 2019.

[4] Y. Tang, K. Au, and Y. Leung, "Comprehending products with mixed reality: Geometric relationships and creativity," *International Journal of Engineering Business Management*, vol. 10, p. 1847979018809599, 2018.

[5] S. Orts-Escolano, C. Rhemann, S. Fanello, W. Chang, A. Kowdle, Y. Degtyarev, D. Kim, P. L. Davidson, S. Khamis, M. Dou, *et al.*, "Holoportation: Virtual 3d teleportation in real-time," in *Proceedings of the 29th annual symposium on user interface software and technology*, pp. 741–754, 2016.

[6] "Centro de ejemplos de mrtk - mixed reality toolkit | microsoft docs." `https://docs.microsoft.com/es-es/windows/mixed-reality/mrtk-unity/features/example-scenes/example-hub?view=mrtkunity-2021-05`, último acceso en 15-05-22.

[7] "Openxr overview - the khronos group inc." `https://www.khronos.org/api/index_2017/openxr`, último acceso en 23-03-22.

[8] "Vuforia - unity manual." `https://docs.unity3d.com/es/2018.4/Manual/vuforia-sdk-overview.html`, último acceso en 06-03-22.

[9] "dlib c++ library: Real-time face pose estimation." `http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html`, último acceso en 20-05-22.

[10] "i-bug - resources - facial point annotations." `https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/`, último acceso en 20-05-22.

[11] "Face alignment with opencv and python - pyimagesearch," 2021. `https://pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/`, último acceso en 15-05-22.

[12] "Knn classification tutorial using sklearn python | datacamp." `https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn`, último acceso en 01-05-22.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[14] R. Roig-Vila, M. E. Urrea Solano, G. Merma-Molina, *et al.*, "La comunicación en el aula universitaria en el contexto del covid-19 a partir de la videoconferencia con google meet," *Revista Iberoamericana de Educación a Distancia (RIED)*, vol. 24, no. 1, pp. 197–220, 2021.

[15] J. Cabero-Almenara and C. Llorente-Cejudo, "Covid-19: transformación radical de la digitalización en las instituciones universitarias," *Campus virtuales*, vol. 9, no. 2, pp. 25–34, 2020.

[16] J. Imants and M. M. Van der Wal, "A model of teacher agency in professional development and school reform," *Journal of Curriculum Studies*, vol. 52, no. 1, pp. 1–14, 2020.

[17] V. González-Calatayud, P. Prendes-Espinosa, and R. Roig-Vila, "Artificial intelligence for student assessment: A systematic review," *Applied Sciences*, vol. 11, no. 12, p. 5467, 2021.

[18] R. Roig-Vila and V. Moreno-Isac, "El pensamiento computacional en educación. análisis bibliométrico y temático.," *Revista De Educación a Distancia (RED)*, vol. 20, no. 63, 2020.

[19] D. S. Özgen, Y. Afacan, and E. Sürer, "Usability of virtual reality for basic design education: a comparative study with paper-based design," *International Journal of Technology and Design Education*, vol. 31, no. 2, pp. 357–377, 2021.

[20] P. Cipresso, I. A. C. Giglioli, M. A. Raya, and G. Riva, "The past, present, and future of virtual and augmented reality research: A network and cluster analysis of the literature," *Frontiers in Psychology*, vol. 9, 2018.

[21] S. Park, S. Bokijonov, and Y. Choi, "Review of microsoft hololens applications over the past five years," *Applied Sciences*, vol. 11, no. 16, p. 7259, 2021.

[22] J. Garzón, "An overview of twenty-five years of augmented reality in education," *Multimodal Technologies and Interaction*, vol. 5, no. 7, 2021.

[23] C. Moro, C. Phelps, P. Redmond, and Z. Stromberga, "Hololens and mobile augmented reality in medical and health science education: A randomised controlled trial," *British Journal of Educational Technology*, vol. 52, no. 2, pp. 680–694, 2021.

[24] H. Sanyal and R. Agrawal, "Study of holoportation: Using network errors for improving accuracy and efficiency," in *Proceedings of International Conference on Sustainable Expert Systems*, pp. 107–118, Springer, 2021.

[25] "Realidad mixta en las aulas con las gafas hololens 2." `https://www.educaciontrespuntocero.com/noticias/realidad-mixta-gafas-hololens-2/`, último acceso en 01-03-22.

[26] "Hololens 2: información general, características y especificaciones | microsoft hololens." `https://www.microsoft.com/es-es/hololens/hardware`, último acceso en 01-03-22.

[27] "Unity (motor de videojuego)." `https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)`, último acceso en 10-03-22.

[28] N. Nurym, R. Sambetova, M. Azybaev, and N. Kerimbayev, "Virtual reality and using the unity 3d platform for android games," in *2020 IEEE 10th International Conference on Intelligent Systems (IS)*, pp. 539–544, IEEE, 2020.

[29] "Introducción al kit de herramientas de mixed reality: configuración del proyecto y uso de interacción manual." `https://docs.microsoft.com/es-es/learn/modules/learn-mrtk-tutorials/`, último acceso en 10-03-22.

[30] "Openxr." `https://docs.microsoft.com/es-es/windows/mixed-reality/develop/native/openxr`, último acceso en 10-03-22.

[31] "Uso de vuforia con unity - mixed reality | microsoft docs." `https://docs.microsoft.com/es-es/windows/mixed-reality/develop/unity/vuforia-development-overview`, último acceso en 06-03-22.

[32] "Vuforia developer portal." `https://docs.unity3d.com/es/2018.4/Manual/vuforia-sdk-overview.html`, último acceso en 06-03-22.

[33] "Welcome to flask – flask documentation (2.1.x)." `https://flask.palletsprojects.com/en/2.1.x/`, último acceso en 10-03-22.

[34] S. Agrawal and P. Khatri, "Facial expression detection techniques: based on viola and jones algorithm and principal component analysis," in *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, pp. 108–112, IEEE, 2015.

[35] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[36] C. V. Alcántara-Montiel, J. C. Pedraza-Ortega, J. M. Ramos-Arreguín, E. Gorrostieta-Hurtado, S. Tovar-Arriaga, and J. E. Vargas-Soto, "Detección efectiva de rostros en imágenes utilizando descriptores basados en hog," *Research in Computing Science*, vol. 148, pp. 371–385, 2019.

[37] D. King, "dlib c++ library," 2017. urlhttps://github.com/davisking/dlib, último acceso 15-04-2022.

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[39] X. Zhu, Z. Lei, X. Liu, H. Shi, and S. Z. Li, "Face alignment across large poses: A 3d solution," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 146–155, 2016.

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

[41] "Github - pyimagesearch/imutils," 2021. `https://github.com/jrosebr1/imutils`, último acceso en 20-05-22.

[42] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018.

[43] Y. Mezquita, R. S. Alonso, R. Casado-Vara, J. Prieto, and J. M. Corchado, "A review of k-nn algorithm based on classical and quantum machine learning," in *International Symposium on Distributed Computing and Artificial Intelligence*, pp. 189–198, Springer, 2020.

[44] M. P. Uddin, M. A. Mamun, and M. A. Hossain, "Pca-based feature reduction for hyperspectral remote sensing image classification," *IETE Technical Review*, vol. 38, no. 4, pp. 377–396, 2021.

[45] "Tensorflow." `https://www.tensorflow.org/?hl=es-419`, último acceso en 11-05-22.

[46] "Tensorflow - wikipedia, la enciclopedia libre." `https://es.wikipedia.org/wiki/TensorFlow#cite_note-YoutubeClip-1`, último acceso en 11-05-22.

[47] "Keras: the python deep learning api." `https://keras.io/`, último acceso en 11-05-22.

[48] "Keras - wikipedia, la enciclopedia libre." `https://es.wikipedia.org/wiki/Keras`, último acceso en 11-05-22.

[49] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "Affectnet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Transactions on Affective Computing*, vol. 10, no. 1, pp. 18–31, 2017.

[50] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[51] W. Sun and R. Wang, "Fully convolutional networks for semantic segmentation of very high resolution remotely sensed images combined with dsm," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 3, pp. 474–478, 2018.

[52] "Descargar visual studio tools: instalación gratuita para windows, mac, linux." `https://visualstudio.microsoft.com/es/downloads/`, último acceso en 14-02-22.

[53] "Download - unity." `https://unity3d.com/get-unity/download`, último acceso en 14-02-22.

[54] "Download mixed reality feature tool from official microsoft download center." `https://www.microsoft.com/en-us/download/details.aspx?id=102778`, último acceso en 15-02-22.

[55] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow." `https://github.com/matterport/Mask_RCNN`, 2017.

[56] F. Gomez-Donoso, F. Escalona, J. Fernandez-Herrero, R. Roig-Vila, and M. Cazorla, "Training emotion detection for autistic persons with ai and ar," in *INTED2022 Proceedings*, 16th International Technology, Education and Development Conference, pp. 953–958, IATED, 7-8 March, 2022 2022.

[57] "Affecnet benchmark (facial expression recognition) | paper with code." `https://paperswithcode.com/sota/facial-expression-recognition-on-affectnet`, último acceso en 28-04-22.

[58] A. Vidal-Balea, O. Blanco-Novoa, I. Picallo-Guembe, M. Celaya-Echarri, P. Fraga-Lamas, P. Lopez-Iturri, L. Azpilicueta, F. Falcone, and T. M. Fernández-Caramés, "Analysis, design and practical validation of an augmented reality teaching system based on microsoft hololens 2 and edge computing," *Engineering Proceedings*, vol. 2, no. 1, p. 52, 2020.

[59] M. P. Prendes Espinosa, F. J. Montiel Ruiz, and V. González Calatayud, "Uso de tic por parte del profesorado de enseñanza secundaria analizado a partir del modelo de ecologías de aprendizaje: estudio de caso en la región de murcia," 2021.

[60] "Openxr plugin." `https://docs.unity3d.com/Packages/com.unity.xr.openxr@0.1/manual/index.html`, último acceso en 10-03-22.

[61] "Affecnet - mohammad h. mahoor, phd." `http://mohammadmahoor.com/affectnet/`, último acceso en 27-04-22.

[62] J. S. Ruthberg, G. Tingle, L. Tan, L. Ulrey, S. Simonson-Shick, R. Enterline, H. Eastman, J. Mlakar, R. Gotschall, E. Henninger, *et al.*, "Mixed reality as a time-efficient alternative to cadaveric dissection," *Medical Teacher*, vol. 42, no. 8, pp. 896–901, 2020.

Lista de Acrónimos y Abreviaturas

***k*-NN**	k-Nearest Neighboors.
CNN	Convolutional Neural Network.
fps	Frames per Second.
GPU	Graphics Processing Unit.
HMD	Head Mounted Display.
HOG	Histogram of Oriented Gradient.
IA	Inteligencia Artificial.
ML	Machine Learning.
MRITF	Mixed Reality Interfaces.
NIR	Near Infra-Red.
PCA	Principal Component Analysis.
RA	Realidad Aumentada.
RGB	Red Green Blue.
RM	Realidad Mixta.
ROI	Region of Interest.
RoViT	Robotics & Tridimensional Vision.
RV	Realidad Virtual.
S-CFA	Self-Calculation of Face Alignment.
SSD	Single Shoot Detector.
UA	Universidad de Alicante.
USB	Universal Serial Bus.
V&J	Viola and Jones.