UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

ENTORNO DE TRABAJO PARA EL DESARROLLO DE APLICACIONES DE REALIDAD MIXTA

TESIS

QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA:

DALILA SCARLETT HERNÁNDEZ PUMARINO

TUTOR PRINCIPAL:

DR. ALFONSO GASTÉLUM STROZZI ICAT-UNAM

MIEMBROS DEL COMITÉ TUTOR:

DR. PATRICE JEAN DELMAS	UNIVERSITY OF AUCKLAND/NEW ZELAND
DR. JORGE ALBERTO MARQUEZ FLORES	ICAT-UNAM
DR. MIGUEL ANGEL PADILLA CASTAÑEDA	ICAT-UNAM
DR. FERNANDO ARÁMBULA COSÍO	ICAT-UNAM

CIUDAD UNIVERSITARIA, CD. MX., ENERO 2019 😊

AGRADECIMIENTOS

A mis padres, Jose Luis Hernandez y Rosario Pumarino, y a mis hermanos Andrea y Luis Hernandez, por su cariño y apoyo, ante todo.

A mi prometido, Argenis Bastida que me impulso para entrar en este programa, para ir a estudiar a Madrid, España y para hacer todas las cosas que he hecho hasta ahora. Por su amor, ánimo, ayuda, desvelos y paciencia.

A mis compañeros y amigos que hicieron de este viaje uno más ameno.

A todos mis profesores de la maestría por compartir sus conocimientos y por su invaluable ayuda y orientación a lo largo del camino.

A todo el personal del Posgrado en Ciencia e Ingeniería de la Computación, especialmente a Lourdes Gonzalez, Amalia Arriaga y Cecilia Mandujano por su valiosa labor y guía en todos los procesos.

Al IIMAS e ICAT de la UNAM por los espacios brindados para trabajar.

A Héctor Hernández de la dirección de Posgrado y a Socorro Camacho de la DGECI por toda su ayuda y guía en el proceso de intercambio.

A todo el personal de la Universidad Autónoma de Madrid por brindarme una estadía agradable y darme la oportunidad de estudiar en otro país.

Al CONACYT por el apoyo económico durante este proceso.

A mi gato, por el apoyo moral.

A todas y cada una de las personas que hicieron esto posible, infinitas gracias.

1 TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

2 RESUMEN / ABSTRACT

La realidad mixta se refiere a la fusión del mundo físico y objetos virtuales con el fin de producir nuevos ambientes donde lo real y digital pueda coexistir e interactuar en tiempo real (Carrer & Gabriel, 2009). Según el continuo de virtualidad (Milgramt & Kishinott, 1994) la realidad mixta se encuentra en cualquier lugar entre el Ambiente real y el Ambiente virtual, juntando diferentes aspectos correspondientes a la realidad, la realidad aumentada y la realidad virtual.

Una gran cantidad de dispositivos y software pueden ser usados como herramientas para crear realidad mixta, como lentes de realidad virtual, guantes hápticos, pantallas, computadoras, cámaras y teléfonos móviles. Gracias a las características presentes en estos, es posible crear ambientes donde la realidad y virtualidad coexistan.

Sin embargo, existen muy pocas herramientas disponibles para la creación de realidad mixta (haciendo referencia específicamente a la adquisición del mundo real a través de cámaras para su posterior combinación con objetos virtuales), además de que dichas herramientas están diseñadas para funcionar con un hardware especifico únicamente, haciendo su importación a otros trabajos imposible, como es el caso del MixedRealityToolkit de Microsoft creado para los Hololens ("Microsoft HoloLens", 2018).

Por otra parte un entorno de trabajo en el área de computación se puede definir como un conjunto de código reutilizable que provee una solución a un problema específico, donde el desarrollador (en este caso usuario) lo utiliza y modifica posteriormente para crear una aplicación independiente y con un fin específico (Johnson & Foote, 1988).

El objetivo del presente trabajo es la elaboración de un entorno de trabajo para la creación de aplicaciones de realidad mixta, basándose en la integración de diversas herramientas enfocadas a la realidad virtual y aumentada, visión estéreo y procesamiento de imágenes. Enfocándose en la adquisición, procesamiento y despliegue de imágenes provenientes de cámaras para su posterior combinación con objetos virtuales.

Se utilizo un casco para realidad virtual, el casco de Lenovo Explorer (Lenovo Explorer, 2018) compatible con Windows 10 de Microsoft como base, además de dispositivos móviles en conjunto con una serie de cámaras externas. Se utilizaron y desarrollaron las herramientas necesarias para el control y combinación de los dispositivos involucrados y su correcto funcionamiento.

El resultado final a manera general incluye un paquete de código con diversas funciones que pueden ser utilizadas dentro de Unity (motor de videojuegos) para acceder a las imágenes de cualquier cámara o cámaras conectadas al dispositivo objetivo. Estas

herramientas transforman el flujo de video en texturas reconocibles por Unity para que puedan ser manipuladas fácilmente dentro del entorno virtual. Además, se proporciona una conexión con OpenCV, que interviene las imágenes de así desearlo y permite al usuario realizar cualquier tipo de procesamiento que dicho software tenga la facultad de efectuar. El flujo de video de la cámara es entonces transformado a un formato reconocible por OpenCV donde se pueden hacer diversas operaciones sobre las imágenes y posteriormente el resultado final es traducido al formato de textura de Unity.

Mixed reality is known as the merge of the physical real world and virtual objects with the objective of producing new environments where real and digital can coexist and interact in harmony on real time (Carrer & Gabriel, 2009). According to the virtual continuum of Milgram (Milgramt & Kishinott, 1994), mixed reality is placed between the real world and the virtual world, sharing different aspects of reality, augmented reality and virtual reality.

A huge number of devices and software can be used like tools with the purpose of creating mixed reality, head mounted devices for virtual reality, haptic gloves, all kind of screens, computers, cameras and smartphones. Thanks to the specific characteristics contained on them it is possible to create environments where reality and virtual generated content coexist.

However, there are a few tools available for the creation of mixed reality (specifically for the acquisition of the real world through cameras in order to combine this information with virtual objects). The existing tools are design to only function in combination with specific hardware. That is the case of the MixedRealityToolkit of Microsoft, made to work only with the Hololens ("Microsoft HoloLens", 2018).

On the other hand, a framework in the computer field could be defined as a set of reusable code that provides a solution to a specific problem. The developer (in this case the user) can use them and modify them later to create an independent application. (Johnson & Foote, 1988).

The main objective of the present work is the creation of a framework of mixed reality, based on the integration of different tools from virtual reality and augmented reality, stereo vision and image processing. Focused on acquisition, processing and display of images through cameras.

For this purpose, it was used a headset made specifically for virtual reality called Lenovo Explorer, compatible with Windows 10, Microsoft as the base and mobile devices. Also, there will be included a set of external cameras connected directly to the computer. In

addition, there were used and developed the necessary tools for the correct control and combination of the involved devices and their proper functioning.

In general, the final result contains a set of different functions that can be used inside Unity (video game engine) to access the images provided by any camera or cameras connected to the target device. These tools transform the video streaming to Unity recognizable textures, so they can be manipulated easily inside the virtual environment. There's also included a connection with OpenCV, allowing the user to make any kind of processing. The video streaming is transformed to a recognizable OpenCV format where different operations over the images can take place, the result is then translated to a Unity texture.

3 ORGANIZACIÓN DE LA TESIS

En la primera parte a modo de introducción se describen algunos conceptos y nociones básicos para tener una comprensión más acertada y general de las tecnologías descritas y utilizadas para este trabajo. Posteriormente se describen las distintas tecnologías relacionadas con la realidad mixta y sus antecedentes. Se hace una referencia a los dispositivos con que se puede trabajar y se comentan sobre algunas de las muchas aplicaciones de las tecnologías mencionadas. Por último, se hace una breve introducción al sistema de realidad mixta propuesto en el trabajo.

Posteriormente se enlistan los distintos componentes que formaran parte del sistema propuesto. Se hablará de las características principales de cada uno, su funcionamiento y relación con los otros.

En la siguiente parte se plantearán las etapas de elaboración del entorno de trabajo, así como las funciones que este permitirá realizar, a modo de documentación del mismo. Para finalizar se mostrarán algunos ejemplos de las funciones del sistema procurando resaltar las ventajas de la creación de un entorno de trabajo para realidad mixta. Por último, se hará una recapitulación del trabajo y se planteará el trabajo futuro en esta área y puntos de mejora del sistema.

3.1 OBJETIVOS

El objetivo central del presente trabajo consiste en desarrollar e implementar un entorno de trabajo enfocado en la Realidad Mixta utilizando herramientas de Realidad Aumentada y Virtual. Para esto será necesario realizar la integración y creación de diversas herramientas, que en conjunto proporcionen una serie de funciones que permitan el desarrollo de una aplicación de realidad mixta, centrándose el trabajo presente principalmente en el aspecto visual. El resultado final esperado es el de un complemento o plugin para Unity 3D que contenga una serie de scripts para la realización de realidad mixta.

3.2 PLANTEAMIENTO DEL PROBLEMA

Para resolver el problema de realidad mixta se realizará una mezcla entre realidad virtual y realidad aumentada. El sistema presente pretende permitir tomar la realidad a través de cámaras, para presentarla por medio de lentes especializados al usuario (casco de realidad mixta) y hacer uso de otros dispositivos de video para complementar la información presentada. Los puntos expuestos a continuación hacen referencia a las distintas problemáticas que deberán ser atacadas para el cumplimiento de los objetivos.

- Adquisición de imagen/video y despliegue. Que hace referencia al acceso de cámaras, así como a los dispositivos de despliegue del casco de realidad virtual, para poder tomar imágenes de la realidad y presentarlas al usuario por medio de los lentes. En este sistema se pretende permitir la inclusión de varios dispositivos de entrada de video funcionando en un mismo proyecto como pueden ser una cámara web, cámara infrarroja, cámara térmica, etc., para aumentar la información que se puede presentar al usuario. El sistema, por sus características de entorno de trabajo, no pretende especializarse en un solo tipo de entrada, sino que se plantea la creación de funciones generales para que el desarrollador pueda incluir cualquier cámara que le sea útil para cubrir sus necesidades específicas. Los videos externos podrán ser desplegados posteriormente de diversas formas en la aplicación. Una vez más se hace hincapié en la naturaleza de este trabajo que fungirá como una herramienta más que como un aplicativo especifico.
- Aumentar la realidad. Que consiste en los procesos necesarios para añadir objetos virtuales en combinación con la información adquirida del mundo real.
- Seguimiento del movimiento del usuario. Que es necesario para lograr una correcta inmersión y despliegue de información. Más que desarrollar herramientas para esta parte se utilizaran aquellas con las que cuenta Unity.
- Interacción del usuario con la realidad mixta. Que es el conjunto de todas las partes anteriormente expuestas con el fin de generar una experiencia interactiva en tiempo real, además de incluir el uso de otras herramientas como son audífonos, micrófonos o controles para proporcionar un medio de comunicación entre el usuario y el entorno virtual más allá de la mera visualización de la escena. Una vez más para esta parte solo serán utilizadas las herramientas ya existentes en el motor de juegos utilizado como base para la aplicación.
- Caracterización del entorno de trabajo y documentación. Como finalización del trabajo se deberá hacer una breve documentación de las funciones que comprenden el trabajo, así como su correcta utilización para permitir la creación de aplicaciones a partir de este.

4 INTRODUCCIÓN

Todos los seres humanos habitamos en una realidad, en la mayoría de los casos esta realidad se compone por el mundo físico. Durante miles de años el ser humano se ha adaptado y familiarizado a dicha realidad, de tal forma que somos capaces de predecirla hasta cierto punto, por ejemplo; sabemos con cuanta fuerza hay que tomar los objetos, podemos predecir donde caerá una pelota o simplemente reconocer objetos sin importar el ángulo en que los veamos o si existe alguna oclusión *(Sutherland, 196*5). Sin embargo, lo que define nuestra realidad es lo que nuestros sentidos perciben de ella y la manera en que esta información es procesada en el cerebro. Es a partir de esto que podemos crear realidades "alternas" únicamente modificando los estímulos que serán proporcionados al usuario y este podrá posteriormente interpretarlos por medio de sus sentidos.

Esta idea no es nueva, la realidad virtual es un concepto que lleva varias décadas siendo estudiado, el primer acercamiento se hizo por medio de estímulos visuales, pues cualquier pantalla se podría convertir en una ventana por la cual podemos ver el mundo virtual *(Sutherland, 1965)*.

4.1 LOS SENTIDOS Y LA TECNOLOGÍA

El sentido de la vista ha sido uno de los más estudiados en este campo. Quizá por la facilidad con que puede ser engañado, quizá por ser uno de los sentidos en los que más nos apoyamos los seres humanos que tenemos la fortuna de conservarlo sino intacto, funcional, para interactuar con nuestros alrededores. O quizá por nuestro amplio conocimiento sobre su funcionamiento y por tanto nuestra capacidad de reproducirlo digitalmente. De cualquier forma, siempre es importante hacer una pequeña revisión sobre la anatomía general para poder proporcionar posteriormente una experiencia inmersiva adecuada. El primer paso es entender la percepción visual humana pues, al fin y al cabo, las tecnologías de visualización tanto de captura como despliegue se inspiran en ello.

4.1.1 El sentido de la vista

El ojo humano tiene un promedio de dos centímetros de diámetro (longitud axial, medida desde el centro de la pupila hacia el fondo del ojo). Aunque existe la creencia popular de que el ojo es el único órgano del cuerpo humano que no crece, varios estudios afirman lo contrario. Durante la infancia los ojos pueden medir entre 15.5 y 18 milímetros mientras que en la edad adulta el diámetro axial aumenta; entre 23 y 24.5 milímetros, hasta alcanzar la madurez alrededor de los 15 años *(Scammon, 1925).* Sin embargo, su constitución original se conserva. Por lo general se divide en tres capas principales: la capa exterior conformada por la córnea y esclera, seguido por la capa media (coroides) y por último la retina.

La cornea es el tejido traslucido que cubre la parte de enfrente del ojo, la esclera es una capa obscura (vista desde fuera es blanca) que recubre el resto. La coroides es la capa intermedia del globo ocular, esta parte está cubierta por vasos sanguíneos y contiene una gran cantidad de pigmento para reducir la entrada de luz por las partes posteriores del ojo. Los vasos sanguíneos por su parte se encargan de nutrir el ojo. En la coroides es donde empieza el proceso de la visión, se divide entre el cuerpo ciliar y el diafragma del iris el cual es el encargado de contraerse o expandirse para controlar la cantidad de luz que pasa por la pupila la cual puede variar de un mínimo de dos milímetros y hasta un máximo de 8 milímetros aproximadamente. El lente del ojo (parte frontal) es el encargado de permitir el paso de la luz y es flexible, se curva o aplana, para poder enfocar correctamente *(Adler, 2004).*

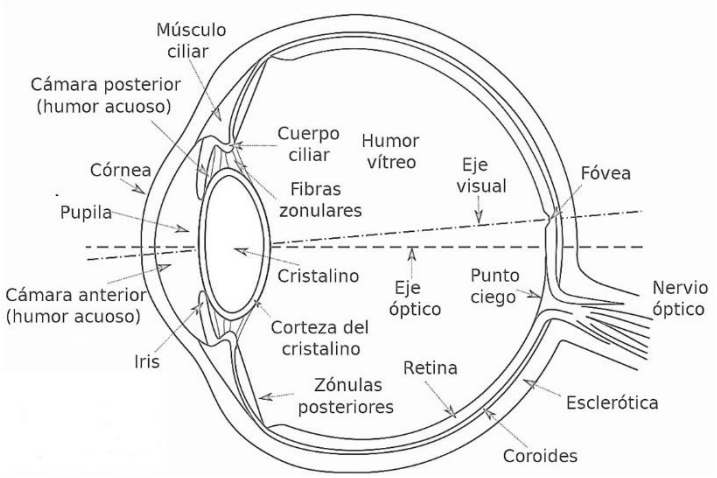

FIGURA 1. Modelo simplificado del ojo humano IMAGEN: (Rodriguez,2018).

En la parte interior del globo ocular se encuentra la retina, también conocida como capa neuronal que es donde se forman las imágenes a partir de la luz recibida (la cual ha tenido que pasar por el lente ocular y la pupila). La retina contiene receptores de luz a lo largo de su superficie, estos receptores son células conocidas como conos (aproximadamente entre seis y siete millones por ojo) y bastones (entre setenta y cinco y ciento cincuenta millones por ojo). Los conos se encuentran en la parte central de la retina conocida como fóvea y son sensibles al color, cada uno de estos receptores tiene su propia terminación nerviosa, convirtiéndolos en los protagonistas de la visión fina. Se pueden dividir en conos rojos, verdes y azules basados en su capacidad de respuesta a cierta parte del espectro electromagnético, por lo general un ser humano promedio tiene mayor número de conos rojos (64%) que verdes (32%) y azules (2%). Los conos azules, sin embargo, a pesar de ser los que se encuentran en menor medida, son los más sensibles. Por su parte los bastones están relacionados más con el brillo y se asocian a una visión más general, se localizan a lo largo de la retina y varios de estos receptores utilizan una única terminación nerviosa, sin embargo, son sensibles aun cuando los niveles de iluminación son bajos. La distribución de

los receptores también es importante, se concentran en mayor cantidad alrededor de la fóvea exceptuando la parte donde está el nervio óptico que es el punto ciego del ojo. Los conos se encuentran con mayor densidad en el centro mientras que los bastones incrementan en la periferia.

La distancia entre el lente y la retina determina la distancia focal y esta varia de entre catorce y diecisiete milímetros. La distancia focal como su nombre lo indica es la distancia de los objetos que podemos tener enfocados, este enfoque disminuye aproximadamente después de los tres metros, donde el lente ha alcanzado su forma plana máxima.

La percepción visual se genera gracias a una combinación de todo lo anterior, cuando la luz atraviesa el lente y excita los receptores de luz de la retina. Estos receptores transforman la energía recibida en impulsos eléctricos que son transferidos al cerebro mediante el nervio óptico para ser decodificados en el cerebro.

4.1.2 Visión estereoscópica

Retomemos el concepto de la cámara pinhole. La idea es bastante sencilla, sin embargo, podría decirse que es la base de cualquier cámara actual, e incluso de cualquier sistema de visión basado en luz. Digamos que tenemos una "película", retícula o pantalla que bloquea el paso de luz (a forma de caja cerrada), sobre este material se realiza un pequeño orificio que permita pasar únicamente cierta cantidad de energía, a través de este agujero algunos rayos de luz ya sean emitidos o reflejados por los objetos lograran llegar a la parte trasera de la caja. Esto creará una imagen que se verá reflejada al revés en la pantalla. Esta imagen es una proyección dentro de la caja del mundo exterior, difícilmente tendrá las mismas dimensiones que los objetos reales reflejados, y como solo captara los rayos de luz que vienen de un único plano en Z, el resultado obtenido será una copia de dos dimensiones de

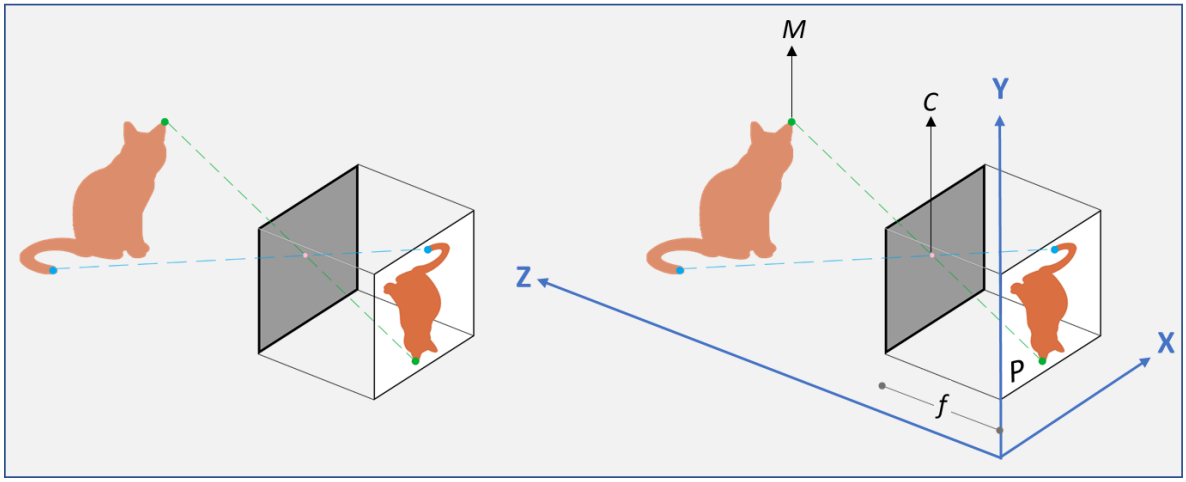

FIGURA 2 Modelo simplificado de la cámara tipo pinhole.

un objeto u escena originalmente tridimensional. A esto se le llama proyección en perspectiva. En la figura 2 podemos observar un ejemplo del concepto de cámara pinhole.

Los ojos funcionan de manera similar. La esclera funge como la caja que bloquea el paso de luz, la pupila funge como el pequeño orificio por el que los rayos pasan y por último la retina donde se encuentran las células receptoras captaran la proyección inversa que llega al ojo. ¿Por qué no vemos al revés? Bueno evolutivamente es poco conveniente, así que una vez recibida la señal visual, el cerebro la modifica.

Por supuesto el proceso en general no podría parecer más sencillo, sin embargo, tiene algunas reglas que seguir, para entenderlo mejor existen los modelos simplificados. Tomemos por ejemplo las siguientes variables del esquema de la figura 2.

- C es el centro óptico (es el punto por donde pasan los rayos de luz).
- P es el plano de la imagen donde la proyección de dos dimensiones es creada.
- f es la distancia focal que equivale a la distancia que existe entre el punto C y P.
- M es un punto cualquiera en la escena de tres dimensiones.
- z es la distancia en el eje Z entre M y C.

Una imagen será formada en el plano P a través de la proyección de perspectiva. Un punto M con una profundidad z será proyectado al plano P que intercepta el rayo formado por CM. Generando de esa manera la imagen invertida que se menciona con anterioridad.

El modelo más simplificado surge de la suposición de que la imagen proyectada no sufrió ningún tipo de distorsión antes de llegar al plano P, que el punto C es el centro u origen del sistema y que el plano P es perpendicular al eje Z.

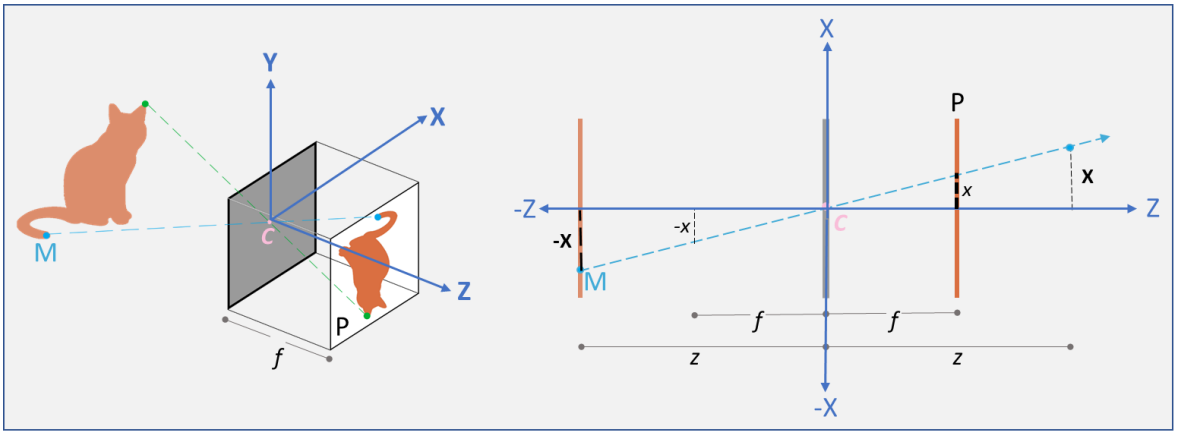

FIGURA 3 Modelo geométrico de la cámara de pinhole.

Los puntos (X, Y, Z) se consideran puntos del espacio mientras que los puntos (x, y) se consideran puntos de la imagen. Por similitud del área triangular que se forma entre el eje Z y la proyección del punto M que pasa por C obtenemos que:

$$\frac{x}{f} = \frac{X}{Z} \quad despejando: \quad x = \frac{fX}{Z} \qquad\qquad \frac{y}{f} = \frac{Y}{Z} \quad despejando: \quad y = \frac{fY}{Z}$$

Que puede ser simplificado de la siguiente forma (asumiendo que tanto f como Z son mayores a cero):

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix}$$

Esta última expresión describe la relación entre las coordenadas (X, Y, Z) del espacio tridimensional con las coordenadas (x, y) en el plano de la imagen o proyección para un punto determinado M.

El modelo simplificado de proyección para la cámara de pinhole también se puede expresar en coordenadas homogéneas (Malek, Belhocine, Zenati-Henda, & Benbelakacem, 2011) realizando la representación de la posición de un punto en el espacio utilizando $(n + 1)$ dimensiones. Supongamos que A es la representación de un punto tridimensional (vector de cuatro dimensiones) y B la representación del punto en la imagen (vector de tres dimensiones), ambos se relacionan de la siguiente manera: $A = TB$. Donde T es una matriz de $(3 * 4)$ también conocida como matriz de proyección y describe las transformaciones necesarias de los puntos tridimensionales de la escena al mundo 2D de la imagen proyectada.

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & \\ & f & & \\ & & 1 & \end{bmatrix}\begin{bmatrix} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

La principal problemática con este tipo de arreglo es que la imagen resultante es únicamente una proyección de dos dimensiones de un objeto tridimensional. Digamos que el objeto tridimensional es una caja, sabemos que dicha caja mide un metro de cada lado, al medir la proyección nos damos cuenta de que la caja mide diez centímetros por lado, así que ahora sabemos la proporción del mundo real a la proyección. Sin embargo, esto solo es cierto para los objetos que se encuentren a esa misma distancia que la caja, si un objeto se encontrara más lejano podría reflejarse más pequeño a pesar de no serlo, por la perspectiva. Para conocer Z (profundidad) además tendríamos que conocer la distancia focal f. Todo esto suponiendo que el sistema real representa fielmente el modelo de la cámara de pinhole. En la realidad esto no ocurre, los lentes de las cámaras agregan cierto nivel de distorsión a la imagen final, además las características como la distancia focal pueden variar gracias a la influencia de agentes externos como puede ser el calor que

deforma en mayor o menor medida los materiales con los que está hecha la cámara. El resultado es la perdida de profundidad, de una dimensión entera. Para resolver este problema es que existe la visión estereoscópica.

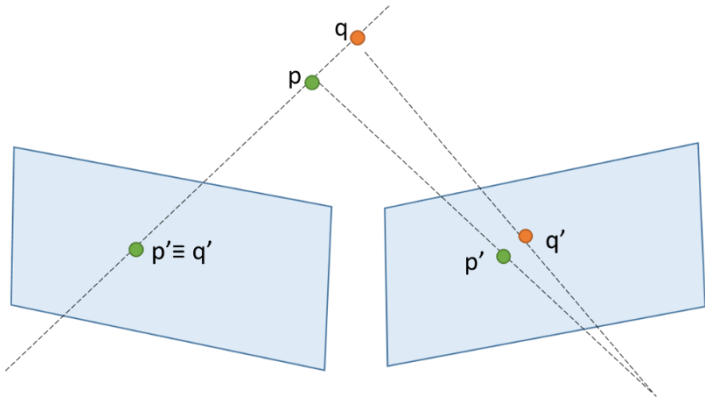

FIGURA 4 Representación simplificada de estereovisión.

Imaginemos que tenemos un plano en el espacio, sobre este plano se proyectaran dos puntos del espacio tridimensional. Sin embargo, ambos puntos se encuentran sobre la misma línea de visión para el plano, por lo tanto, en el plano únicamente se podrá ver un punto, el más cercano, pues los demás serán ocluidos por este. Si añadimos otro plano a la ecuación y vemos la proyección resultante podremos ver los puntos ocluidos en el primero.

Teniendo dos o más puntos de vista (perspectivas) de un punto podemos inferir la profundidad a la que se encuentra con mayor precisión dicho objeto, pues la dimensión que se pierde al hacer la proyección en dos dimensiones se puede calcular utilizando la información proporcionada por otra perspectiva, utilizando una serie de relaciones geométricas entre los puntos tridimensionales y sus proyecciones. Dichas relaciones son obtenidas bajo la asunción de que ambas proyecciones pueden ser aproximadas a través del modelo de la cámara de pinhole.

La visión estereoscópica en si es una capacidad que posee el ser humano; consta de dos partes principales, la facultad física, es decir cuenta con dos ojos que están al frente alineados de cierta forma (dos planos o dos puntos de vista) y psicológica, el cerebro se encarga de reconstruir las dos imágenes provenientes de cada ojo dando como resultado una sola imagen en tercera dimensión. Cada ojo, utilizando su retina produce dos imágenes del mismo objeto, dichas imágenes son muy similares entre sí, pero no son iguales, ya que cada imagen se forma con la perspectiva respectiva del ojo que ve el objeto. La separación entre los receptores (en este caso los ojos) puede variar dependiendo del objetivo del sistema de visión, en los humanos ronda los sesenta y cinco milímetros aproximadamente y se conoce como “distancia Inter pupilar”.

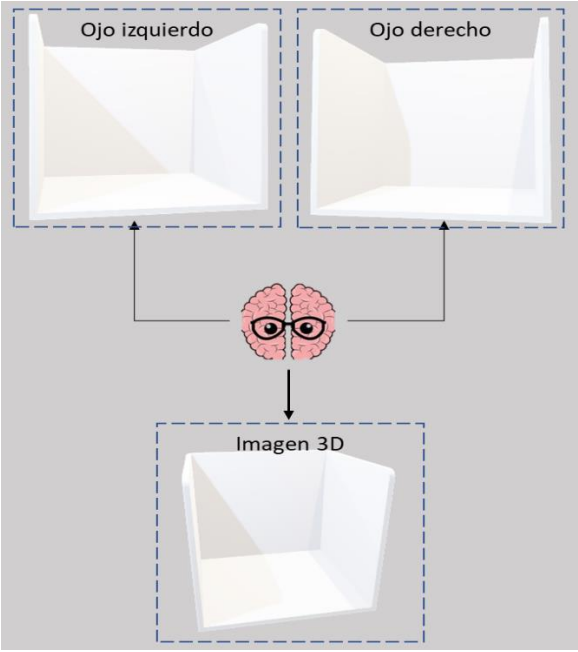

FIGURA 5 Esquema simplificado de la visión estéreo del ser humano.

Utilizando ambas imágenes es posible reconstruir el factor faltante que es la profundidad del objeto. Ambos ojos se encuentran posicionados aproximadamente a la misma altura y profundidad y varían en su posición horizontal. Esto no es por azar, la razón es en realidad bastante simple, si ambas perspectivas comparten posición en dos de las tres dimensiones calcular la faltante (la diferencia entre una y otra) debería ser más fácil. De igual forma, la disimilitud que puedan existir entre altura y profundidad entre ambos ojos son compensadas por el cerebro. Con esta información podremos entonces determinar la profundidad de los objetos en el espacio.

Para hacer los cálculos de esto a partir de dos imágenes planas, el primer paso es calcular los puntos tridimensionales correspondientes, es decir saber que punto de la imagen A corresponde con que punto de la imagen B que son el mismo en el espacio tridimensional. Posteriormente se puede utilizar un modelo derivado de la cámara de pinhole para reconstruir los planos de perspectiva y posteriormente calcular la diferencia de un punto entre una imagen y otra tomando en cuenta las distancias y posiciones a las que se encontraban ambos planos de proyección.

Actualmente existen muchos trabajos enfocados únicamente en esta área ya que se trata de un problema clásico en visión por computadora. Existen en el mercado dispositivos especializados en medir profundidad como son el Kinect de Microsoft, algunos otros dispositivos basados en arreglos de cámaras con el mismo fin y otros que combinan ambos para lograr un mejor resultado (Zhang, Wang, & Chan, 2013), así como procesos estudiados de calibración para asegurar los mejores resultados y garantizar fiabilidad cuando se toman medidas del espacio utilizando estos métodos (Vasileiou & Psarakis, 2015).

4.1.3 Espectro de luz y color

Isaac Newton descubrió hace ya mucho tiempo que cuando un rayo de luz blanca pasaba a través de un prisma se descomponía en un espectro continuo de colores que iban del violeta al rojo. Esto forma parte de un conjunto de ondas electromagnéticas o radiación que emite o absorbe dicha energía. El espectro electromagnético puede ser expresado en términos de ondas sinusoidales que se propagan con cierta frecuencia. Otra aproximación es el de entenderlo como partículas que viajan a la velocidad de la luz siguiendo un patrón de onda. Ambas aproximaciones son correctas pues la luz es emitida y absorbida por los objetos y presenta características de ambas formas, ondas y partículas. A mayor frecuencia o menor longitud de onda el fenómeno electromagnético es más energético. Estas ondas se extienden más allá de la luz visible e incluyen otras como son las ondas de radio, microondas, infrarrojo (baja energía) hasta llegar a la luz visible, ultravioleta o rayos-X (energía alta).

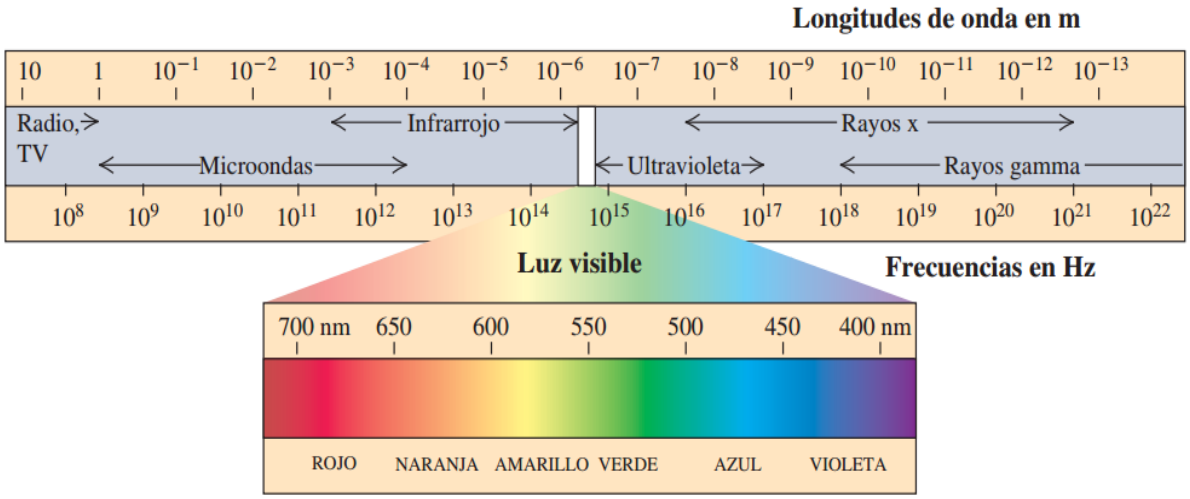

FIGURA 6 Espectro electromagnético IMAGEN: (Fonrouge, 2018).

La luz es un tipo de radiación electromagnética, específicamente es el único tipo de radiación visible para el ojo humano. Existen algunos animales que cuentan con un tipo de célula receptora extra lo que les permite observar otra parte del espectro, por ejemplo, algunos pájaros pueden ver luz ultravioleta, mientras que existen otros animales cuyos ojos están adaptados para ver ciertas ondas del espectro de luz visible y no otras como es el caso de los perros. Para los humanos la luz visible está comprendida en el rango entre 380 y 740

nanómetros, como se menciona anteriormente siendo más sensibles a unos que a otros gracias a la forma en que los conos se encuentran distribuidos dentro del ojo.

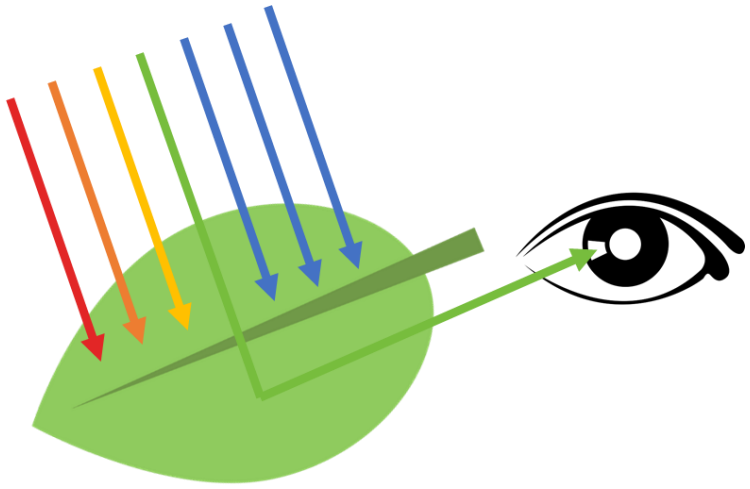

FIGURA 7 Proceso de absorción y reflexión de luz que da lugar al color característico de un objeto.

Los colores que un humano percibe están determinados por la naturaleza de la interacción entre la luz y el objeto en cuestión. Un fotón puede mostrar diferentes comportamientos, los más comunes son absorción y reflexión. Las moléculas de los objetos tienen un espectro de energía y si los fotones que llegan a ellos armonizan con esta energía, entonces la luz en esa longitud de onda será absorbida, por el contrario, aquellos fotones que no se igualen serán reflejados. Por ejemplo, un objeto que se ve azul está absorbiendo todas las longitudes de onda menos aquellas en el rango de 425 a 475 nanómetros. Un objeto blanco es aquel que refleja de cierta manera balanceada todas las longitudes de onda, un objeto negro es aquel que refleja poco o nada de luz. El color verde por ejemplo suele ser sinónimo de vegetación culturalmente hablando, sin embargo, al ser este el color que percibimos de las plantas significa que de todo el espectro el verde (entre 500 a 570 nanómetros) es el que está en menor armonía con las moléculas de las hojas. Por ello cuando se desea añadir luz artificial para el cultivo de una planta, por lo general se utilizan lamparas que emitan poca luz verde, ya que esta solo será reflejada por la planta, sin afectar a su desarrollo.

De igual forma la fuente de luz es importante, si la fuente de luz únicamente emite luz roja, los objetos que intercepten dicha luz solo podrán absorber o reflejar dicho color. Es así como la iluminación de una escena es importante tomarla en cuenta, pues un mismo objeto puede y lucirá diferente bajo distintos rangos del espectro electromagnético. Actualmente existen lamparas cuyo objetivo es emitir espectros controlados de luz. Cuando se tiene una situación experimental es importante tomar en cuenta las condiciones de iluminación bajo las que se va a trabajar, saber si la luz estará controlada o por el contrario se trata de luz ambiental que varía constantemente. Por poner un ejemplo, en el caso de realizar procesamiento de imagen sobre heridas para determinar si están infectadas o no, es necesario tener en cuenta el tipo de luz bajo la que se tomen las imágenes. La fuente de luz equivocada puede hacer parecer a una herida leve como grabe, la coloración de la piel es fundamental para determinar el estado actual de la herida.

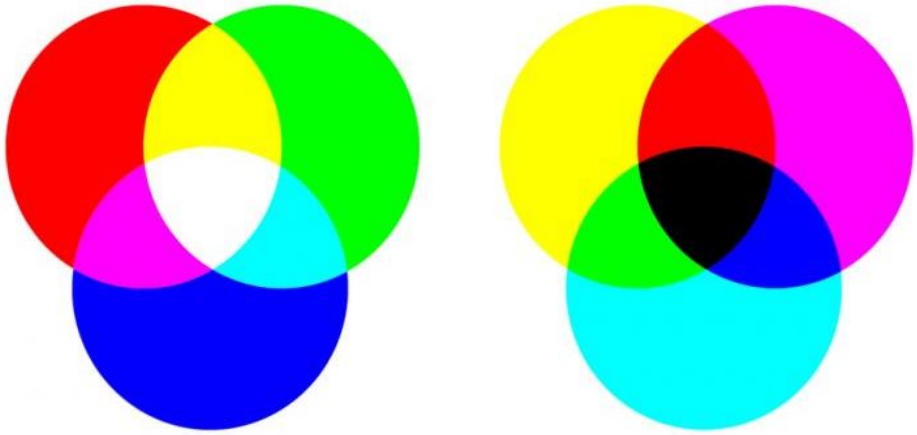

FIGURA 8 Modelos de color RGB (luz, izquierda) y CMYK (pigmento, derecha).

La luz puede ser monocromática o cromática. En el primer caso se trata de luz sin color o con color nulo pues su único atributo es la intensidad. En procesamiento de imágenes a este término se le conoce como nivel de gris, que va del negro, pasando por una serie de grises hasta llegar al blanco. La luz cromática por su parte es la que se extiende por el espectro.

Como se menciona anteriormente, el color reflejado por los objetos depende de su composición física y la forma en que interactúa con la luz que llega. La luz del espectro visible se diferencia en algunos colores básico como el violeta, azul, verde, amarillo, naranja y rojo entre otros que forman las combinaciones de ellos. Comúnmente en computación, el color se representa a través de un modelo matemático que permite representar un color numéricamente. El uso del color está justificado por dos razones principales: en primer lugar, es un descriptor poderoso de los objetos y puede simplificar la identificación y extracción de objetos, comúnmente cuando buscamos algo una de las primeras cosas que describimos es el color. Los humanos por nuestra parte podemos distinguir cientos en

comparación con solo un par de tonos de gris. Gracias a la manera en que funcionan las células de nuestra retina que podríamos decir burdamente detectan únicamente tres colores (rojo, verde y azul), los colores resultantes son combinaciones variables de estos colores llamados primarios. Estos colores son mejor conocidos como colores primarios de luz que pueden mezclarse para producir otros colores como el magenta (mezcla de rojo y azul), el cian (verde y azul) y el amarillo (rojo y verde). La mezcla de todos los colores produce luz blanca.

Sin embargo, los colores primarios de luz no son suficientes, pues si somos observadores notaremos que, al mezclar pintura roja, azul y verde lo que obtendremos será algo más parecido al negro que al blanco. Es así como es importante distinguir entre los colores primarios de luz y de pigmento. Cuando se habla de pigmento el concepto original de color cambia puesto que estos pigmentos no son luz en sí misma, si no reflejos de luz. Los colores primarios del pigmento son el magenta, cian y amarillo (igual que las combinaciones de sus homólogos de luz que son el rojo, verde y azul). Un color primario de pigmento es aquel que absorbe un color primario de luz y refleja los otros dos. Por ejemplo, el amarillo absorbe la luz azul y refleja la luz roja y verde. La combinación de los colores primarios de pigmento da como resultado el color negro. Esto en la realidad no es así, la mezcla de todos los pigmentos da un color cercano al negro con reflejos cafés, pues los materiales con los que se crean los pigmentos no absorben toda la luz por completo.

Generalmente se utilizan ciertas características para distinguir los colores, estás características son el tono que es un atributo asociado a la longitud de onda de luz, representa el color dominante reflejado y percibido por el observador. La saturación que se

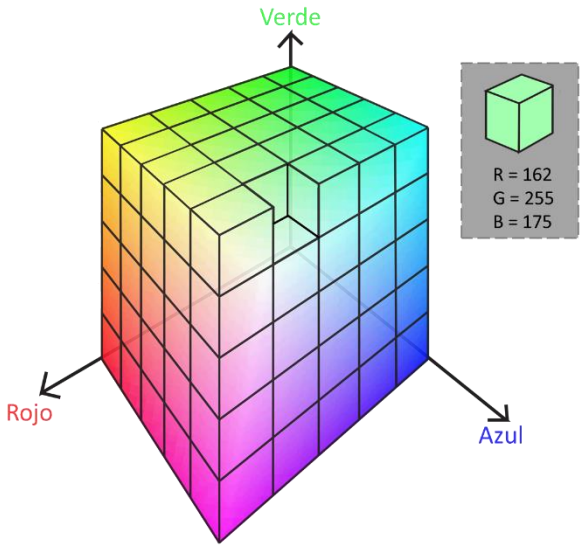

FIGURA 9 Modelo de color RGB simplificado en un cubo con tres coordenadas.

refiere a la pureza del color, los colores puros se dice que están completamente saturados, entre más o menor luz blanca contengan estarán menor o mayormente suturados. A la mezcla de estas dos características se le conoce como cromaticidad. El conjunto de la cromaticidad y el brillo dan como resultado un color. El brillo es un descriptor subjetivo que es prácticamente imposible de medir pero que percibimos a través de nuestro sentido de la vista, comprende nuestra noción acromática de intensidad.

Las tres formas de entender el color descritas anteriormente dan lugar a los modelos de color. El propósito de estos es estandarizar los colores y no son más que sistemas coordenados donde cada color puede ser representado por un punto en el espacio. Los más comunes son RGB (colores de luz, rojo verde y azul por sus siglas en ingles), CMYK (colores de pigmento, cian amarillo y magenta por sus siglas en inglés, la K hace referencia al color negro que se añade posteriormente como un color distinto al modelo ya que como se comenta anteriormente en la practica la añadidura de los tres colores no produce negro puro). Y por último el modelo HSI haciendo referencia a las características que se utilizan para diferenciar el color (Tono o Hue en inglés, saturación y brillo o Intensidad) y que es el más cercano a la forma en que los humanos perciben el color. Los modelos de color están orientados comúnmente al hardware (ya sea la representación de estos a través de pantallas o la impresión) o a las aplicaciones que requieren manipular el color como las animaciones digitales.

EL modelo RGB es uno de los más utilizados pues es la base de la mayoría de los monitores comerciales, así como de las cámaras en color. Este modelo se basa en un sistema de coordenadas cartesianas en tres dimensiones, representado comúnmente como un cubo. Tres esquinas del cubo corresponden a los tres colores primarios de luz y las otras tres a las intersecciones de estos colores que son los colores primarios de pigmentación. Las dos esquinas restantes corresponden al negro y al blanco (esquinas contrarias), la línea imaginaria entre estas dos esquinas corresponde a intensidades de gris que se representan numéricamente por coordenadas iguales en todos los ejes. Los colores en este modelo son representados por puntos dentro del cubo que tienen tres coordenadas. Por convención, se asume que todos los colores han sido normalizados de tal forma que el cubo resultante tiene como longitud 1 en todos sus lados. Las imágenes representadas en este modelo de color tienen tres componentes, una para cada uno de los colores primarios, y en el valor de cada pixel se almacena una de las coordenadas dentro del cubo. El número de bits que se utilice para representar cada pixel se llama profundidad de pixel y determina la cantidad de colores que pueden ser representados, por ejemplo, el número de colores total de una imagen de 24 bits tendrá $(2^8)^3$ = 16,777,216 posibles representaciones de colores. Una imagen a color con este modelo puede ser obtenida utilizando sensores sensibles a la luz roja, verde y azul. Cada conjunto de arreglos obtendrá una imagen monocromática cuya intensidad corresponderá a la cantidad del color primario que capto, es decir una coordenada del cubo. Juntando las tres imágenes se obtendrá como resultado una imagen

a color. Es importante utilizar en los sistemas el hardware y software adecuado que nos permita elegir el mejor modelo para trabajar y la profundidad de pixel que sea necesaria para poder reproducir fielmente la información visual necesaria. De igual manera cualquier modelo puede ser convertido en otro por medio de operaciones matemáticas simples.

4.1.4 Cámaras y comparativa con el ojo humano.

Las cámaras fotográficas actuales (digitales y análogas) funcionan mediante un principio de captación de luz muy similar a la del ojo humano. Las cámaras fotográficas se componen de un conjunto óptico en su parte delantera que simulan una caja negra similar a la del modelo pinhole. Un pequeño orificio capta los rayos de luz del exterior y los proyecta hacia la parte trasera de la cámara, en la mayoría de las cámaras este orificio es variable para permitir pasar mayor o menor cantidad de luz dependiendo de la escena, al igual que la pupila, en el caso de las cámaras para celulares la reducción de luz se hace digitalmente y no mecánicamente como en el caso de los obturadores. Así mismo la distancia focal de las cámaras se puede variar comúnmente para lograr enfocar los objetos de interés, alejando o acercando el lente, que simula a su vez la forma en que el lente en nuestro ojo se deforma con el mismo fin. En el caso de las cámaras análogas, una imagen nítida proveniente del lente recae sobre la película fotográfica la cual contiene compuestos químicos fotosensibles que reaccionan a la luz, produciendo la imagen y dejando el fotograma plasmado, la cual se hará latente en el proceso posterior llamado revelado.

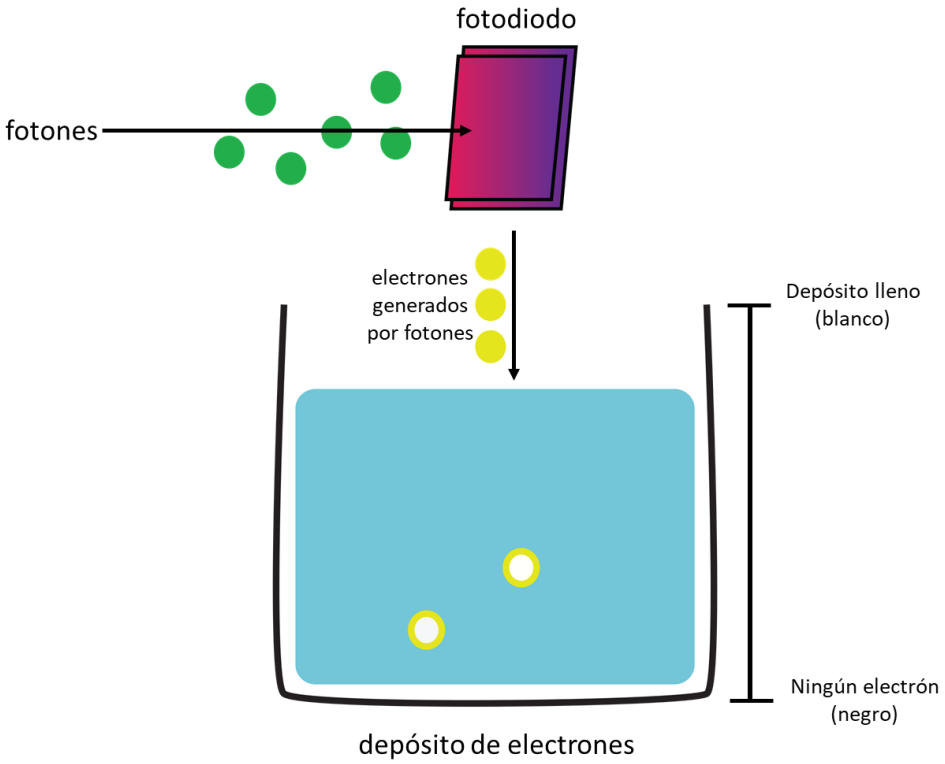

FIGURA 10 Funcionamiento de un fotodiodo (sensor).

En las cámaras digitales, la parte de la película es suplida por un sensor capaz de detectar la energía radiada de cierta banda del espectro electromagnético, compuesto por muchas células fotosensibles. Cada que se toma una fotografía, cada célula fotosensible recibe una cantidad de luz, utilizando un fotodiodo convierte la luz en voltaje, el cual es digitalizado y almacenado. Los fotodiodos producen un voltaje proporcional a la luz recibida, colocando delante de este un filtro se puede obligar al fotodiodo a ser más sensible a cierto tipo de luz. Los sensores pueden ser dispuestos de diferentes formas, por ejemplo, puede ser un solo sensor o una línea de sensores (ambos necesitan que el sensor se mueva en el espacio para poder captar una escena), el arreglo de sensores más utilizado en las cámaras digitales suele ser cuadrado, compuesto por N x M sensores, su ventaja es que no necesita ningún tipo de movimiento y adquiere toda la imagen al mismo tiempo (en paralelo).

Cuando los fotones alcanzan el fotodiodo, estos se convierten en electrones, los cuales se acumulan en un pequeño deposito o condensador. La dejar de llegar fotones cada celda del sensor tendrá un determinado nivel de electrones en él, esto determinará el voltaje final emitido.

4.1.5 Formación de la imagen

Una imagen puede ser definida como una función bidimensional con coordenadas x e y correspondientes al espacio ocupado en el arreglo de sensores f(x,y) = i. Donde i hace referencia al valor de intensidad captado por ese sensor. Recordemos que el mundo físico es análogo y continuo y para obtener las imágenes a través de una cámara utilizando sensores es necesario digitalizarlas, esta digitalización siempre viene acompañada de perdida de información y se lleva a cabo de dos maneras: Se hace un muestreo de la imagen, es decir ciertos rayos son captados por ciertos sensores, de toda la luz emitida por el espacio continuo solo será conservada una parte en los sensores y ocupara un lugar espacial finito, por ultimo ocurre una cuantización del nivel de gris, que es cuando la amplitud de onda es convertida voltaje y se le asigna un valor finito para identificarlo. La calidad de la imagen dependerá de la cantidad de sensores que se tengan y la cantidad de valores de intensidad que puedan ser almacenados.

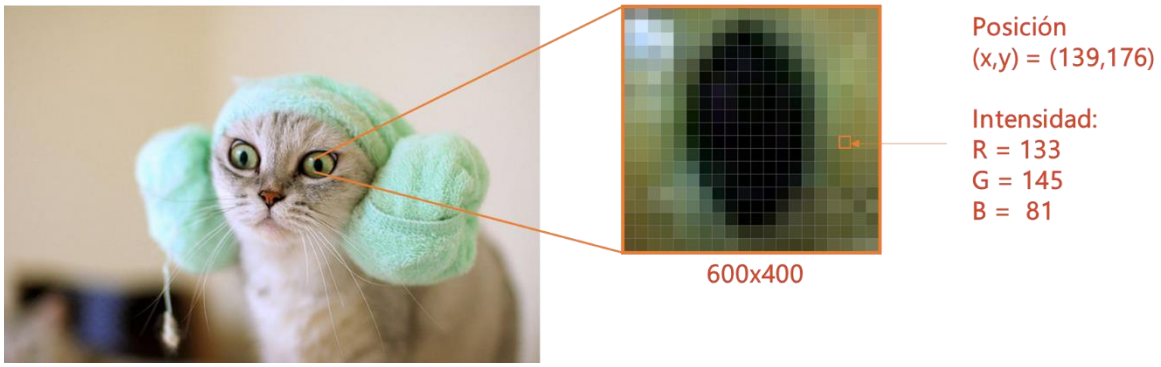

FIGURA 11 Partes básicas de una imagen digital (modelo de color RGB).

Por ejemplo, si se tienen disponibles únicamente dos valores de intensidad (correspondientes al blanco y negro), al llegar la información en forma de voltaje esta deberá ser clasificada como uno u otro nivel utilizando algún criterio. El resultado de muestrear y cuantizar una imagen respectivamente es una matriz de números con M renglones y N columnas correspondiente al número de sensores. Dentro de cada celda será almacenada un valor correspondiente a la intensidad. Para las imágenes a color utilizando el modelo RGB se obtendrán tres imágenes digitales en total, las intensidades almacenadas en cada una serán aquellas referentes a cada uno de los colores primarios rojo, verde y azul. Al recuperar el valor de i de cada imagen en la misma posición (x,y) se obtendrá una coordenada de tres dimensiones que hace referencia a ese modelo, haciendo posible el posterior despliegue de dicha información de forma entendible para el ser humano a través de un display que emite luz.

4.1.6 Displays

Hablamos al inicio de este trabajo sobre engañar los sentidos, ahora que ya entendemos el funcionamiento de la vista, la forma en que podemos captar y almacenar imágenes y cómo es que vemos en tercera dimensión es importante tratar el tema de despliegue. Al fin y al cabo, los estímulos visuales que serán creados para engañar la vista del usuario serán entregados por lo general a través de una pantalla con el fin de emitir luz compuesta en ciertos patrones para representar una escena en específico. De forma abstracta podríamos definir una pantalla como un arreglo de celdas (pixeles) que tienen la capacidad de emitir luz. Por lo general cada uno de estos pixeles tiene la capacidad de emitir alguno de los tres colores de luz principales, la solución más común es agrupar tres emisores de luz en una misma celda (emisor rojo, azul y verde) con el fin de poder crear colores complejos provenientes de la mezcla de estos en distintas intensidades.

Actualmente las pantallas que se utilizan en la mayoría de los dispositivos móviles, de cómputo y de entretenimiento en el hogar, son pantallas del tipo LCD (pantalla de cristal líquido), la cual es una pantalla plana que se basa en el uso de una sustancia líquida atrapada entre dos placas de vidrio, las cuales al ser aplicada una corriente eléctrica a una zona o punto específico estas se vuelven opaca y ocluyen con la luz trasera. Esta capa se encuentra entre dos electrodos, que dependiendo de la carga eléctrica que se

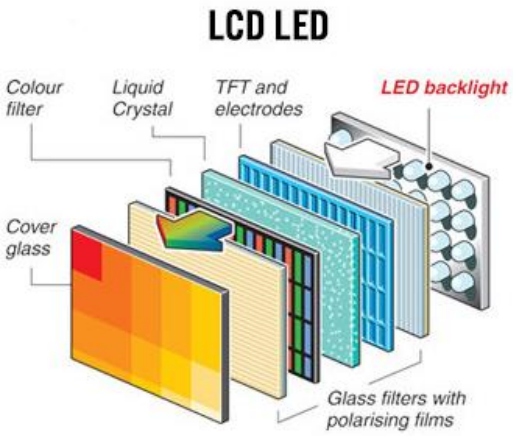

FIGURA 12 Partes que componen una pantalla LCD IMAGEN: (Rodriguez,2018).

le emita, es capaz de cambiar la orientación de sus moléculas de cristal.

Para la creación del color, se pone una capa que filtra el color, es decir, se convierte la luz blanca en otro color, en resumen, no es más que un cristal pintado. Sin embargo, para la creación de colores complejos se utilizan varios de estos, por cada pixel, se ponen 3 de estos subpíxeles en el espectro RBG (rojo, azul y verde), que al estar tan juntos uno al otro, el ojo humano es incapaz de distinguirlos.

Las pantallas LCD utilizan una iluminación que procede de la parte trasera, por medio de un tipo específico de lámparas fluorescentes de cátodo frio, las cuales, al no contar con una resistencia física como un foco convencional, no generan tanto calor. Por último, la capa del panel de cristal es la última capa de cara al usuario, que es la que se puede tocar y protege todo el conjunto del panel.

Sin embargo, la representación final obtenida es en dos dimensiones, para lograr el efecto de tercera dimensión y que las imágenes cuenten con un efecto de profundidad es necesario involucrar otro tipo de tecnología. En primera instancia las imágenes presentadas a través del display deben tener algún tipo de tratamiento especial para lograr el efecto deseado, en conjunto con dispositivos especiales que tomen dicha información de las imágenes y la desplieguen correctamente. Existen dos tipos de tecnologías distintas para proyectar el efecto de tercera dimensión:

a) Estereoscópicos: cuyas imágenes deben proceder de una cámara fotográfica o de video estereoscópica o tener un tratamiento de postproducción especial para simular tal efecto, logrando tener dos imágenes similares de la misma escena, pero con dos perspectivas distintas (comúnmente presentadas en la pantalla al mismo tiempo). Las dos imágenes se traslapan enteras sobre la pantalla y mientras el usuario debe utilizar algún dispositivo, por lo general lentes especialmente diseñados para que cada ojo reciba únicamente una de las perspectivas, siendo el cerebro el responsable de dar la sensación de profundidad. En esta rama existen dos tipos de tecnologías principales para lograr tal efecto clasificados a partir de los lentes necesarios para interpretar las imágenes y el tipo de despliegue:

   Por un lado, está la tecnología activa, funciona mediante obturación. Para el despliegue se requiere que la pantalla muestre las perspectivas en serie, es decir muestra la perspectiva uno, después la perspectiva dos y luego la perspectiva uno de nuevo, de tal forma que solo una de las imágenes es visible en un momento de tiempo determinado. Los lentes por su parte están sincronizados con la proyección, de tal forma que cuando esta una de las dos proyecciones presente en la pantalla, los lentes detienen la entrada de luz a uno de los ojos. Por ejemplo, si en pantalla esta la perspectiva uno que pertenece al ojo derecho, la luz que pueda entrar al ojo

izquierdo será bloqueada temporalmente por el lente y así sucesivamente. Esto requiere que la presentación de las imágenes y la obturación de los lentes este perfectamente sincronizado y se realice lo suficientemente rápido para que el cerebro no sea capaz de detectar los cambios entre una imagen y otra y las interprete como una misma. Regularmente la calidad es buena ya que permite disfrutar por cada ojo la máxima resolución de la pantalla, sin embargo, estos dispositivos requieren electrónica especializada para lograr el flasheo en la pantalla y los lentes deben ser alimentados por una fuente de energía externa para poder discernir el momento en que se opaca uno de sus lados. Estos lentes a pesar de ser activos no muestran las imágenes, solo se encargan de obstruir el paso de luz de forma sincronizada.

Por otro lado, siendo estos los más comunes existe la tecnología pasiva o polarizada. Tienden a ser los más utilizados puesto que no es necesario ningún tipo de hardware especializado. En la pantalla se emiten ambas imágenes o proyecciones al mismo tiempo, una perspectiva se distingue de otra por sus colores o dirección en que la luz es emitida. Los lentes simplemente funcionan con un filtro que no requiere de alimentación energética externa. Los filtros pueden ser de color o con un principio de polarización, los de color como su nombre lo indica ocluyen cierta energía del espectro electromagnético (reflejándola). Por ejemplo, una de las perspectivas puede ser de color rojo, mientras la otra se presenta en color azul, ambas imágenes son proyectadas en la pantalla al mismo tiempo y los lentes tienen un filtro azul y uno rojo, uno en cada ojo, de tal forma que a cada ojo solo llegara la luz del color contrario al filtro del lente. Los lentes con un filtro de polarización siguen el mismo principio, pero en lugar de utilizar filtros de color, filtran las ondas de luz que viajan en cierta dirección. Este tipo de tecnología suele ser el más utilizado por su bajo costo sin embargo tiene la desventaja de cada imagen será de la mitad de resolución máxima de la pantalla.

b) Auto Estereoscópicos: de igual forma las imágenes deben proceder del mismo tipo de cámaras o procesado que las anteriores, salvo que, a nivel usuario, éste solamente debe colocarse en una posición determinada, ya que la misma pantalla emite diferentes imágenes a cada ojo por medio de barreras de paralelaje. Este tipo de tecnología es mucho menos utilizado pues son las pantallas las que deben ser modificadas y es mucho más fácil para los usuarios por lo general adquirir un par de lentes que cambiar completamente su televisor.

Por supuesto todas estas tecnologías de despliegue para simular el efecto de tercera dimensión son fijas. Es decir, el usuario debe permanecer quieto para notar el efecto. Las escenas suelen ser fijas también, se trata de videos o fotografías, para añadir a la ecuación

movilidad y libertad de explorar la escena es necesario incursar en otro tipo de tecnología como lo son los lentes de realidad virtual.

4.1.7 Lentes de realidad virtual

En 1838 Charles Wheatstone demostró que el cerebro procesa la diferencia entre dos imágenes independientes de cada ojo como una sola en tres dimensiones (Wheatstone, 1838). Viendo dos imágenes (fotografías o pinturas en la época) colocadas una al lado de la otra a través de un estereoscopio le brindaba al usuario la sensación de profundidad a partir de elementos de dos dimensiones. Posteriormente se creó una patente para dicho aparato ganando popularidad en 1839 utilizado para turismo virtual.

FIGURA 13 Estereoscopio e imágenes estereoscópicas IMAGEN: (Wikipedia, 2018).

El principio bajo el que trabajaba el estereoscopio es muy simple, en él se colocaban dos imágenes de la misma escena ligeramente movidas (con dos perspectivas diferentes), el aparato consistía en una caja que obstruía la entrada de luz externa y permitía poner ambas imágenes a cierta distancia de los ojos del espectador para evitar la fatiga visual y facilitarle el enfoque en diferentes partes de la imagen. Estos aparatos contaban con dos lentes, uno para cada ojo, de tal forma que la imagen mostrada parezca más lejana de lo que realmente es y una entrada de luz que iluminará la imagen estereoscópica.

Los principios de este diseño se mantienen prácticamente intactos hoy en día, un ejemplo de esto es el Google Cardboard (Google, 2016), que es un dispositivo montado en la cabeza para realidad virtual de bajo presupuesto. La única diferencia con el estereoscopio, además de los materiales utilizados, es que las imágenes fijas son sustituidas por imágenes generadas por computadora, mostradas a través del display de un teléfono móvil que a su vez cuenta con su propia fuente de iluminación.

Entre 1950 y 1960 Morton Heilig presento lo que quizá serían los primeros lentes de realidad virtual, no incluían ningún tipo de sensor para el seguimiento de la cabeza, pero presentaban imágenes y sonido estéreo. Posteriormente en 1961 apareció el precursor actual de los lentes de hoy en día, creado por Philco Corporation, contaba con una pantalla

para cada ojo y un dispositivo de rastreo magnético para la cabeza, en aquel entonces el concepto de realidad virtual aún era desconocido y el principal objetivo de este dispositivo era mostrar situaciones peligrosas para los militares. Los movimientos de la cabeza movían una cámara remota dejando que el usuario explorara naturalmente el ambiente en el que era inmerso.

FIGURA 14 Google Cardboard (estereoscopio moderno) IMAGEN: (Google VR, 2018).

En 1965 Ivan Sutherland (Sutherland, 1965), introdujo el concepto del ultimo display que podía simular una realidad hasta el punto de que una persona no pueda distinguir entre el mundo físico y el mundo virtual. Para lograr esto insistió en tres partes principales: en primer lugar, la necesidad de un mundo virtual que sería presentado a través de un dispositivo para la cabeza como son los lentes anteriormente mencionados, que deberían incluir sonido y retroalimentación táctil (háptica) cuando menos, y preferiblemente estímulos olfativos. En segundo lugar, el hardware necesario para crear el mundo virtual y presentarlo en tiempo real y por último hacía hincapié en la capacidad de los usuarios de poder interactuar con los objetos del mundo virtual de una forma intuitiva como se hace en el mundo físico.

Sin embargo, no fue hasta 1987 que el concepto de realidad virtual comenzó a sonar por primera vez para describir todos los avances en este campo. En la década de los noventas varias compañías de videojuegos introdujeron sus propios lentes de realidad virtual siendo SEGA y NINTENDO los principales precursores. Sin embargo, no fue hasta los últimos 15 años que hemos presenciado un avance más significativo y rápido en el desarrollo de la tecnología concerniente a la realidad virtual. En parte gracias a la tecnología de los teléfonos celulares que incluye el desarrollo de pantallas muy pequeñas con grandes resoluciones y pequeños dispositivos capaces de procesar grandes cantidades de información en poco tiempo.

Los llamados lentes de realidad virtual tienen en su interior auténticos paneles LCD que proyectan para cada ojo una imagen de perspectiva para producir el efecto de tercera dimensión. No requieren ningún otro aparato de video, ya que las imágenes son desplegadas en los mismos lentes directamente hacia el usuario. La fuente del contenido sin embargo es producida por otro dispositivo, aun así, parece que la tendencia va hacia crear lentes que requieran en la menor medida posible, el uso de cables para transferir el contenido, e incluso apostando por ser en sí mismos capaces de descargar y reproducir las diferentes aplicaciones como hacen los celulares hoy en día.

FIGURA 15 Virtual Boy (Casco de realidad virtual más control) de Nintendo IMAGEN: (Wikipedia, 2018)

Este tipo de dispositivos pueden incluir sensores de movimiento como son los acelerómetros y giroscopios con el fin de sincronizar el movimiento del usuario con la visualización de la escena tridimensional. Además, pueden añadir entornos de sonido, y otro tipo de sensores como micrófonos para brindar una experiencia más envolvente.

4.2 ¿POR QUÉ SURGEN ESTAS TECNOLOGÍAS?

Últimamente ha habido mucha difusión sobre la realidad virtual, y posteriormente de la realidad aumentada y la realidad mixta. Constantemente en internet nos podemos encontrar con artículos y noticias dedicadas a discutir estas nuevas tecnologías. Especialmente ahora con el surgimiento de teléfonos móviles que incluyen algún tipo de característica especializada para esta área, y con el surgimiento constante de dispositivos en el mercado como son los Oculus (Oculus,2018) y los Lentes de Realidad Mixta de Microsoft (Windows Mixed Reality Headsets, Microsoft 2018). Sin embargo, aún hay quien asegura que estas tecnologías con el tiempo sufrirán el mismo destino que las televisiones 3D, que tuvieron un fuerte auge algunos años, antes de desaparecer de las mentes de los compradores como una característica indispensable.

El mayor problema con la tecnología 3D para casa fue quizá que no existía suficiente contenido que lograra persuadir al consumidor a gastar cantidades significativas de dinero en ello. Sin embargo, desde mi punto de vista, esto no ocurre con la realidad virtual, pues actualmente los dispositivos como son los cascos, sino baratos, son más accesibles y portátiles (a diferencia de un televisor) y ponen a manos del consumidor grandes cantidades de aplicaciones de todo tipo, especialmente videojuegos, una de las industrias con más crecimiento en el área del entretenimiento. Esto aunado a la principal diferencia entre los

televisores tridimensionales y los casos de realidad virtual, que es involucrar la interacción del usuario, pueden significar una mayor aceptación.

La humanidad parece tener una pequeña obsesión con la tecnología, prueba de ello es el constante crecimiento y aceptación de la misma a lo largo de los últimos veinte años. Es un hecho, que hoy en día, cada aspecto de los negocios, procesos de enseñanza y nuestras vidas personales incluyen en mayor o menor medida diferentes avances tecnológicos, por poner un ejemplo muy sencillo, el correo electrónico. Vivimos en una sociedad que exige día con día acceso en tiempo real a la información y servicios que utilizamos en nuestra vida diaria. Imaginemos como serian nuestras vidas sin la posibilidad de intercambiar mensajes de manera instantánea con las personas que nos rodean, como era en los tiempos en que las cartas físicas y el correo tradicional vieron su auge.

El avance significativo que trae consigo el poder mostrar a una persona un mundo o lugar completamente distintito del real parece ser un marcador importante en la industria, desde la medicina, educación, entretenimiento entre muchos otros. Conforme el mundo avanza nos enfrentamos a crecientes y cambiantes necesidades en distintas áreas, por ejemplo, actualmente las practicas medicas que deben cubrir los estudiantes antes de interactuar con pacientes reales son mucho más elevadas que hace algunos años, y esto es gracias a que la tecnología ya les permite aprender a través de simuladores hiper realistas.

En general, la realidad virtual, aumentada y mixta surgen en respuesta a las necesidades crecientes de una sociedad que cambia constante y rápidamente.

4.3 MUCHAS REALIDADES (VIRTUAL, AUMENTADA Y MIXTA)

El concepto de realidad virtual se ha extendido a lo largo de los años y ha comenzado a integrar otras ideas y formas de excitar los sentidos. Así mismo han surgido otras prácticas similares y con ellas nuevas definiciones para tratar de diferenciar entre la realidad virtual y las nuevas tecnologías.

La realidad virtual de manera muy general se puede definir como cualquiera en la que el usuario se sienta efectivamente inmerso en un mundo virtual responsivo(Brooks, 1999). No obstante, dicha definición abre el camino para preguntarnos ¿Realmente es necesaria la completa inmersión del usuario para hablar de realidad virtual?

4.3.1 Inmersión en el mundo virtual

La respuesta a esta pregunta es no, en 1993 Robertson G., habla de la realidad virtual no inmersiva. Aunque la inmersión aparece como algo bueno y deseable, queda estipulado que los entornos no inmersivos presentan tres grandes ventajas; la primera que no existe la posibilidad de perder la inmersión, la segunda que el retraso o desface en los sistemas

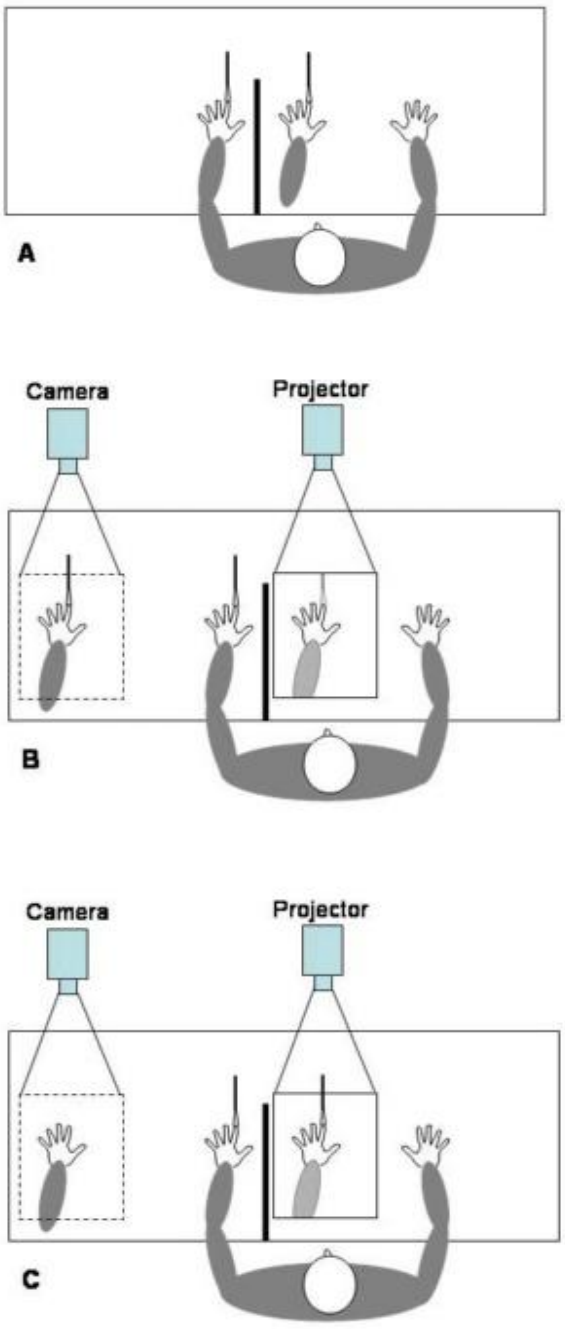

FIGURA 16 Diferentes configuraciones para el experimento de la mano de goma. A) La mano de goma y el estímulo se muestran físicamente al sujeto. B) Realidad virtual: La mano de goma y el estímulo son presentados como proyecciones. C) Realidad mixta: La mano de goma es proyectada al sujeto mientras el estímulo se aplica a la proyección. (IJsselsteijn, de Kort, Haans, Kort, & Dickinson, 2006)

puede ser fácilmente atacado, y por último la mayor de las ventajas de un sistema de realidad virtual no inmersivo radica en que los dispositivos a utilizar pueden ser de uso cotidiano y no requieren ninguna configuración específica, como los dispositivos que se montan en la cabeza y hacen despliegue estéreo por ejemplo (Robertson, Card, Mackinlay, & Parc, 1993).

En un estudio conocido de manera más popular como "la ilusión de la mano de goma" (Botvinick, 1998), se propone un experimento muy simple relacionado con la propiocepción (la percepción que tenemos de nuestro propio cuerpo, incluyendo posición, alcance, sensaciones, etc.), que intenta descifrar la relación que existe entre la visión, el tacto y la percepción de una extremidad, en este caso la mano. El experimento consiste en colocar a una persona sentada frente a una mesa, el brazo real del sujeto de estudio se retira de su campo visual y en su lugar se coloca una mano de goma, de ahí el nombre por el que se le conoce. Una vez esto hecho se realiza un procedimiento que pretende engañar al cerebro del individuo, se toca su mano real y la mano de goma al mismo tiempo en el mismo sitio. Este estudio reporta que después de varias repeticiones las personas comienzan a atribuirle dichas sensaciones a la mano de goma, formando esta parte de su propiocepción actual en lugar de su mano real, llegando incluso a reaccionar negativamente si se le hace daño a la mano de goma.

Recapitulando, podemos extraer dos puntos importantes de este escenario; en primer lugar, consideremos que el elemento visual es fundamental pues si a los sujetos de estudio se les vendan los ojos, su propiocepción no

cambiara sin importar la cantidad de repeticiones táctiles que se realicen. En segundo lugar, las personas no reportan un cambio en su percepción de sí mismos hasta después de varias repeticiones, lo que implica que existe un proceso para que el sujeto se sienta inmerso.

Partiendo de este estudio, en 2005 se realizó uno muy similar que integra la realidad virtual a este experimento (IJsselsteijn, de Kort, Haans, Kort, & Dickinson, 2006). Se plantea el mismo escenario, con la diferencia que en esta ocasión se intercambian los objetos físicos (la mano de goma y el objeto con que se toca esta) con objetos generados por computador (siendo estos últimos, proyecciones 2D sobre la mesa en que está sentado el sujeto) y se realizan tres combinaciones; objetos reales, objetos virtuales y objetos reales y virtuales mezclados. Este experimento llega a la conclusión de que incluso cuando se sustituyen por completo los objetos reales por virtuales se obtiene el mismo resultado. Esto debido a que la percepción de nuestro cuerpo es maleable y el cerebro se adapta fácilmente como mecanismo de sobrevivencia. Y una vez más se hace hincapié en la gran importancia de la percepción visual para ayudar al cerebro a decidir dónde están nuestras extremidades y construir a partir de ello nuestra percepción corporal (Welch, 1972). Welch en un estudio realizado en 1972 afirma que la respuesta inmersiva no solo depende de la percepción visual sino de la discrepancia que pueda existir entre los estímulos recibidos y lo que perciben nuestros sentidos. Además, hace hincapié en la importancia de un proceso por el cual el cerebro del sujeto sea preparado de alguna forma para lograr mayor inmersión y en el factor temporal, pues pasado cierto tiempo sin retroalimentación visual, la propiocepción de los sujetos experimentales regresa a ser la normal, es decir vuelven a sentir sus extremidades donde realmente están, teniendo que pasar por un nuevo proceso de adaptación para lograr la inmersión nuevamente.

En cuanto a la inmersión, parece que algunos aspectos como el visual y el auditivo han sido resueltos en los últimos años, tal es el caso de las imágenes y audio 3D. Sin embargo, un aspecto muy importante como es el háptico aún tiene mucho camino por delante. El ambiente virtual inmersivo ideal podría definirse como aquel en el que el usuario puede interactuar con este del mismo modo que interactúa con el mundo real. Para lograr esto último es importante tener en cuenta los siguientes conceptos:

Por una parte, está la imagen corporal, que se refiere a la percepción que uno mismo tiene de su cuerpo y del tamaño y forma del mismo. Por otra parte, el esquema corporal que nos indica la posición de nuestras extremidades. Por ejemplo, en algunas ocasiones podemos realizar tareas sin utilizar nuestra visión, tal es el caso de las personas que escriben en ordenadores sin mirar el teclado, salvo en algunas ocasiones determinadas. O cuando tomamos un objeto sin verlo y sin prestar gran atención. Todo esto es gracias a la sensación que tenemos de nuestro propio cuerpo y la manera en que se relaciona con el exterior.

Cuando se habla de inmersión, esta parte parece fundamental estudiarla. ¿Qué pasaría si durante una simulación virtual, la persona sintiera objetos que no puede ver, o al contrario viera objetos que no puede sentir o con los que no puede interactuar?

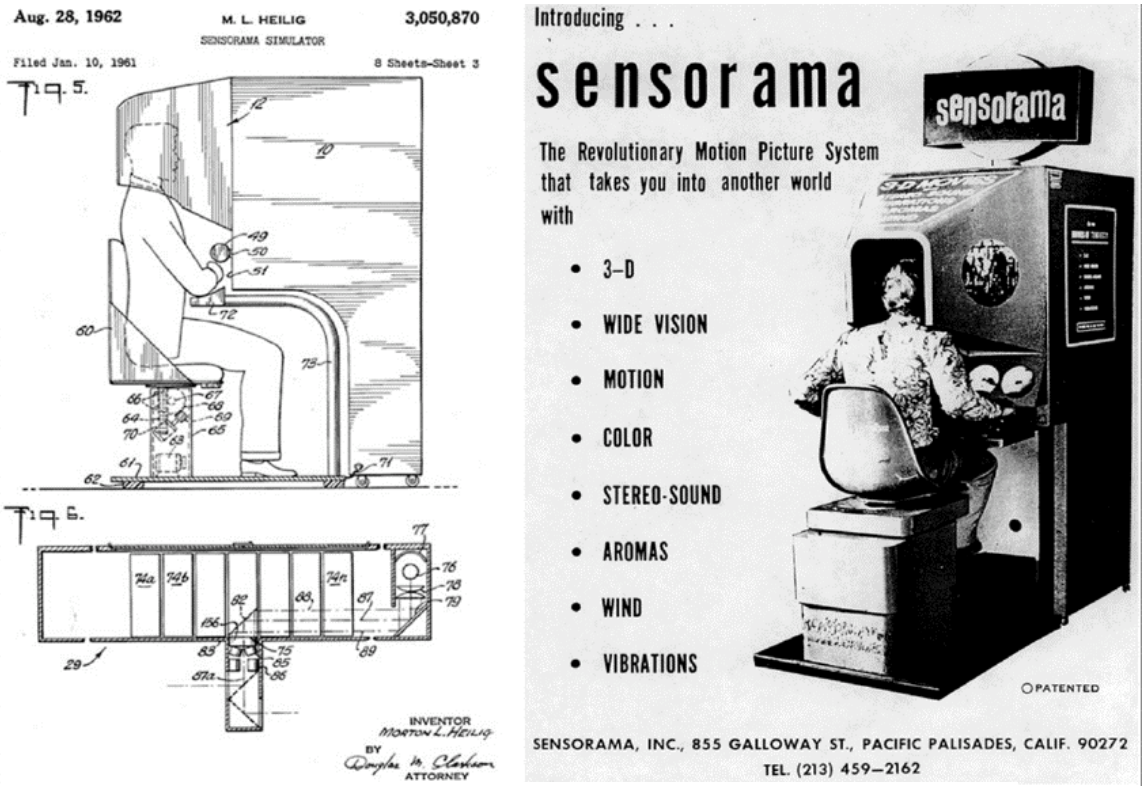

FIGURA 17 Sensorama Simulator de 1962 (Dzierako-Bukal I., Lebicda J., 2018)

Lo cierto es que la realidad virtual provee un método prometedor para estudiar este tipo de fenómenos, los cuales a su vez pueden cambiar la forma en que experimentamos e interactuamos con la realidad virtual.

En 2014, se realizó un estudio que ahonda en los conceptos de propiedad y agencia en relación con la realidad virtual (Padilla, Frisoli, Pabon, & Bergamasco, 2014). La propiedad se relaciona con la sensación del cuerpo, es decir sentir el cuerpo como de uno mismo, identificarlo como fuente de sensaciones. La agencia por otra parte es saberse el autor de los movimientos corporales y la relación de estos con los eventos del exterior, por ejemplo, si se avienta una pelota se espera que esta reaccione de acuerdo con las leyes de la física y se aleje de nosotros. Esta intención motora y la retroalimentación sensorial juegan un papel importante en la inmersión en un mundo virtual, pues si las condiciones físicas de la simulación no parecen plausibles al usuario, la inmersión se perderá. En conclusión, para lograr correctamente dentro de un ambiente virtual la ilusión de la mano de goma es necesaria la integración correcta de información multisensorial (haciendo hincapié en una combinación visual y táctil) y la existencia de un ambiente responsivo que demuestre al usuario la relación entre sus acciones y los eventos externos.

Aún falta mucho camino por recorrer en esta área, no solo por ser nueva, sino por la enrome complejidad que representa estudiar fenómenos como este, relacionados con el sentido del tacto y la propiocepción. Sin embargo, es importante mantenerlo en la mira, pues es el último salto para lograr la inmersión total en los ambientes virtuales.

4.3.2 Realidad virtual

Lo más adecuado parece ser definir la realidad virtual en términos de su funcionalidad y no de los dispositivos que se utilizan para generarla ni de su nivel de inmersión. Usualmente se le relaciona con lentes de realidad virtual por ser de las primeras aproximaciones más populares disponibles al público, sin embargo, la realidad virtual hoy en día se logra utilizando diferentes tecnologías. La realidad virtual es una simulación o interface que presenta un mundo o ambiente hasta cierto punto realista que involucra simulación en tiempo real e interacciones a través de diversos canales sensoriales (Burdea & Coiffet, 2003). Es decir, debe ser interactiva y hasta cierto grado inmersiva. En 1962, Morton Heilig, presento uno de los primeros sistemas de realidad virtual, llamado Sensorama Simulator (Steindecker, Stelzer, & Saske, 2014) el cual era una estación que ofrecía un estímulo visual en tercera dimensión, movimiento, sonido estéreo, aromas, efectos de viento y vibradores, para simular un recorrido en motocicleta.

Hoy en día, se siguen investigando formas de mejorar las experiencias de realidad virtual que se ofrecen a los usuarios, principalmente para la simulación de otros estímulos no visuales como el tacto, el oído, el gusto o el olfato, pues, aunque se siguen buscando maneras de mejorar aspectos como la fidelidad, la resolución o el tiempo de respuesta, la parte visual podríamos decir que está resuelta.

Por supuesto el uso de la realidad virtual y sus vertientes ha despertado el interés dentro del campo de la medicina e investigación, pues permite a los usuarios hacer frente a diversas situaciones bajo ambientes estandarizados que gracias a la inmersión son capaces de evocar estados emocionales y cognitivos (Bouchard, Stéphane; Wiederhold, 2014). Gracias a esto se han realizado diferentes estudios al respecto, no solo de las ventajas, también de las desventajas.

4.3.3 Cybersickness

La principal de ellas es el llamado “cybersickness” o traducido como ciber mareo, que consiste comúnmente en problemas oculares, náuseas y desorientación (Quintana, Bouchard, Serrano, & Cárdenas-López, 2014). Estos síntomas se tiene la teoría de que son resultado de un conflicto entre el sistema del oído interno y los otros sentidos como la vista y la propiocepción. Es decir, cuando dichos canales reciben diferente información, la más común cuando las señales del sistema visual y de movimiento son incompatibles (Bos, Bles, & Groen, 2008), por ejemplo cuando el usuario gira la cabeza y el ambiente de realidad

virtual permanece estático. De un estudio donde participaron 200 personas, el 80% de los participantes reportaron en mayor o menor medida algún síntoma de los expuestos anteriormente, mientras el 5% experimento síntomas tan severos que tuvieron que parar de inmediato (Sharples, Cobb, Moody, & Wilson, 2008). Aunque claro no todo son malas noticias y los síntomas suelen disminuir con el tiempo y práctica, es decir entre más veces una persona utilice el dispositivo y más tiempo lo tenga, los síntomas tienden a disminuir y en muchos casos a desaparecer (Stanney, Mourant, & Kennedy, 1998).

Retomando el concepto de la estereovisión, recordemos que la sensación de profundidad es generada a partir de dos imágenes diferentes provenientes de cada ojo. Los humanos podemos tolerar ciertas cantidades pequeñas de desenfoque o disparidad de las imágenes que interfieren con nuestra habilidad de observar nítidamente los objetos. La profundidad de foco describe la variación de la distancia entre el objeto y el sistema óptico que pueden ser tolerados antes de incurrir en una falta de nitidez de la imagen. En las últimas décadas este concepto ha sido cada vez más estudiado pues se relaciona directamente con los problemas de salud y seguridad provocados por las nuevas tecnologías de despliegue como son la realidad virtual (Lambooij & İ, 2009). Los términos más comunes son la fatiga visual y la disconformidad visual, ambos relacionados con el despliegue de ambientes tridimensionales, presentando entre otros síntomas, dolor de la zona ocular, el cuello y hombros.

4.3.4 Realidad aumentada

Lamentablemente el ciber mareo no es el único de los problemas que presenta la realidad virtual en su forma más pura, un punto muy importante es que una vez inmersos en ella, perdemos la capacidad de interactuar con el mundo real (Azuma, 1997; Billinghurst & Kato, 1999).

Por esta razón el siguiente problema a resolver fue la realidad aumentada, la cual como su nombre lo indica, consiste en aumentar la realidad.

La realidad aumentada es una variante de la realidad virtual, que permite al usuario seguir observando su realidad mientras se superponen a ella objetos virtuales o información adicional. Se espera que complemente o aporte algo a la realidad en lugar de reemplazarla. Idealmente se desea que dicho aporte coexista con la realidad y compartan un mismo espacio. Para que un sistema se considere como realidad aumentada debe cumplir con tres características principales; debe combinar lo real y virtual, ser interactivo en tiempo real y los estímulos visuales virtuales deben ser percibidos en tres dimensiones (Azuma, 1997). Esta última característica, puede o no ser cierta, pues hay trabajos donde se distingue entre dos formas de desplegar la información: en el espacio de la pantalla y el espacio del mundo, es decir la información en el espacio de la pantalla no se mezcla con el mundo real, solo es

posicionada estratégicamente, mientras que la información desplegada en el espacio del mundo requiere efectivamente aparentar tener una posición en tres dimensiones (Moser et al., 2015). Esta última forma de crear realidad aumentada requiere de lograr una relación precisa entre el mundo real y la información virtual para que funcione correctamente (Livingston & Ai, 2008). Esto último le da una ventaja importante sobre la información únicamente desplegada en el espacio de la pantalla: el usuario no se ve en la necesidad de cambiar el foco de atención visual constantemente, y las distracciones que podrían provocar los elementos virtuales se comparan a las distracciones del día a día que no afectan nuestra seguridad (Sabelman & Lam, 2015).

¿Por qué combinar lo real y virtual de esta forma puede ser útil? La realidad aumentada permite expandir nuestra percepción y adquirir estímulos que de otra forma nuestros sentidos no podrían percibir. Esto incluye mejorar aspectos como nuestra capacidad de medición, por ejemplo, ya que, aunque los seres humanos podemos hasta cierto punto hacer aproximaciones de distancias, en muchos casos sería más útil tener la medida real y no un dato que puede o no ser cierto. Así la información adicional nos puede incluso ayudar a realizar tareas más fácilmente en el mundo real.

FIGURA 18 Aplicación de Realidad Aumentada, Pokémon Go por Niantic, Inc. IMAGEN: (New Jersey Institute of Technology, 2018)

A diferencia de la realidad virtual, la realidad aumentada requiere, además de un medio para desplegar (pantalla), un medio de adquisición de la realidad (una cámara, por ejemplo) y los procesos necesarios para aumentar dicha realidad de forma adecuada y coherente. En el campo de lo visual, la realidad aumentada se puede clasificar en dos desde el punto de vista del uso que se les da a los dispositivos para crearla. Una de estas formas, a su vez la más común, es lo que se conoce como "video see-through", que hace referencia a los sistemas en los que el usuario debe ver el mundo real a través de una proyección 2D, por ejemplo, sosteniendo su dispositivo celular y observando a través de la pantalla. La segunda forma es utilizando un display que se pueda montar en la cabeza, desplegando a modo estero las imágenes para que sean percibidas en 3D (Moser et al., 2015).

Las principales ventajas de esta configuración recaen en la capacidad de crear un sistema donde las manos del usuario estén libres, y pueda mantener una perspectiva del mundo real tal como lo hace en lo cotidiano (Moser et al., 2015). Además, disminuye el riesgo de sufrir accidentes provocados comúnmente porque el campo visual es reducido y el usuario se ve obligado a fijar su atención en una proyección 2D (Livingston et al., 2002).

La seguridad es un tema importante en esta área, pues se debe tomar en cuenta que el campo de visión de las personas será más reducido de lo normal o desaparecer totalmente (como en el caso de la realidad virtual) y que el mundo será visto a través de una proyección 2D, lo cual puede provocar accidentes, sin embargo, se pueden realizar algunos "trucos" para hacer esta tecnología más segura, por ejemplo añadiendo sensores que permitan mostrar advertencias a los usuarios para que estos se mantengan alejados del peligro (Sabelman & Lam, 2015).

4.3.5 Seguridad del usuario

La realidad aumentada redefine los límites entre la realidad física y el mundo virtual. Mas que presentar contenido virtual que diverge del mundo real, las aplicaciones de realidad aumentada suelen integran contenido virtual dentro del mundo físico recolectando información de este y del punto de vista del usuario. Esta tecnología es la base para aplicaciones potenciales de realidad mixta que logren combinar experiencias reales y virtuales como una misma. Actualmente estas herramientas, específicamente aquellas que funcionan sobre dispositivos montados en la cabeza o cascos ya son una realidad gracias a la intervención e inversión de compañías como Microsoft en el tema. Hoy en día, las aplicaciones típicas de realidad aumentada funcionan sobre dispositivos convencionales como son los teléfonos inteligentes dentro de sus propios contextos aislados. Sin embargo, la tendencia indica que estos aplicativos se mueven en busca de dispositivos más inmersivos que la pantalla de un teléfono. Actualmente los dispositivos como son los lentes desarrollados por Microsoft de realidad mixta (Windows Mixed Reality Headsets, Microsoft 2018) buscan proveer experiencias más completas combinando distintas herramientas y proveyendo al usuario la capacidad de interactuar con más de una aplicación a la vez.

Mientras que estas aplicaciones prometen tener un gran potencial, traen consigo un número importante de desafíos para garantizar la seguridad y privacidad del usuario. Algunos esfuerzos (Jana et al., 2013) se han enfocado únicamente en el tema de privacidad ya que los aplicativos de esta naturaleza por lo general tienen acceso sin ningún tipo de restricción a la información recolectada por los sensores, permitiendo la posible recolección de datos sensibles como son video o audio privados, sin embargo poco trabajo se ha desarrollado en este campo, dejando desatendidos los posibles riesgos presentes en aplicaciones sin regulaciones legales que podrían ser potencialmente maliciosas. Por ejemplo, una aplicación corrupta cuya finalidad sea proporcionar a un cirujano información extra en tiempo real durante una operación es altamente peligrosa, una imagen maliciosa interfiriendo puedo significar la muerte del paciente.

Sin embargo, a pesar de que asegurar la seguridad en ambientes tan particulares como este que están en constante crecimiento puede ser difícil, existen algunos trabajos enfocados en

ello. Específicamente en ambientes inmersivos (Lebeck, Kohno, & Roesner, 2016) donde un posible malfuncionamiento puede significar un mayor problema.

4.3.6 Realidad mixta

Por último, tenemos la realidad mixta, que surge en respuesta a una problemática de la realidad aumentada: la falta de interacción entre realidad y virtualidad. La realidad mixta la podríamos definir como aquella en la que se crea un espacio donde los usuarios pueden estar física y virtualmente presentes (Milgram & Colquhoun, 1999; Poussard et al., 2014) e interactuar con ambas partes de manera indistinta (Tamura, Yamamoto, & Katayama, 2001). Se podría decir que la realidad mixta pretende lograr una mezcla entre la realidad virtual y aumentada, tomando aspectos de cada una e integrándolos de la mejor manera posible.

Para ahondar más a fondo entre las diferencias que existen entre las tres tecnologías planteadas anteriormente (realidad virtual, aumentada y mixta), se hace referencia al concepto de "virtuality continuum", introducido por Milgram en 1994. Este, traducido al español, continuo de virtualidad se refiere a la mezcla de objetos reales y virtuales presentados a través de algún medio, comúnmente un display, como se ilustra en la figura 1 (Milgramt & Kishinott, 1994).

Donde el ambiente real se muestra en contra posición de un ambiente completamente virtual. El caso de la izquierda define ambientes creados solamente de objetos reales, como por ejemplo un recorrido en formato de video de algún lugar, una fotografía sin retoque o la mera observación directa del mundo físico. El caso de la derecha se refiere a ambientes únicamente virtuales como es el caso de una simulación por computadora.

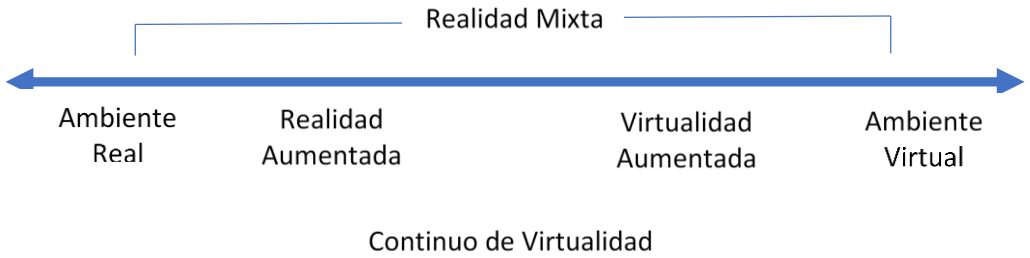

FIGURA 19 Representación del continuo de virtualidad de Milgram.

Sin importar de cuál de las "realidades" se hable, todas tienen algo en común; requieren de un medio para desplegar la información, el desarrollo de la parte virtual; lo que implica una forma de procesar información (Poussard et al., 2014), y en el caso de la realidad

aumentada y mixta un medio de adquisición de la realidad y un medio de interacción. Tan solo para lograr resolver la parte visual se necesitaría como mínimo una cámara, una pantalla y un procesador (sin contar el medio de interacción), lo que se puede traducir como un conjunto de hardware complejo, y muchas veces grande, pesado o difícil de manipular (Wang, 2011).

Sin embargo, en la actualidad contamos con una maravilla de la tecnología que concentra la mayoría sino es que todo el hardware necesario para la creación de realidad virtual, aumentada o mixta. Los dispositivos móviles, han demostrado ser un excelente medio para

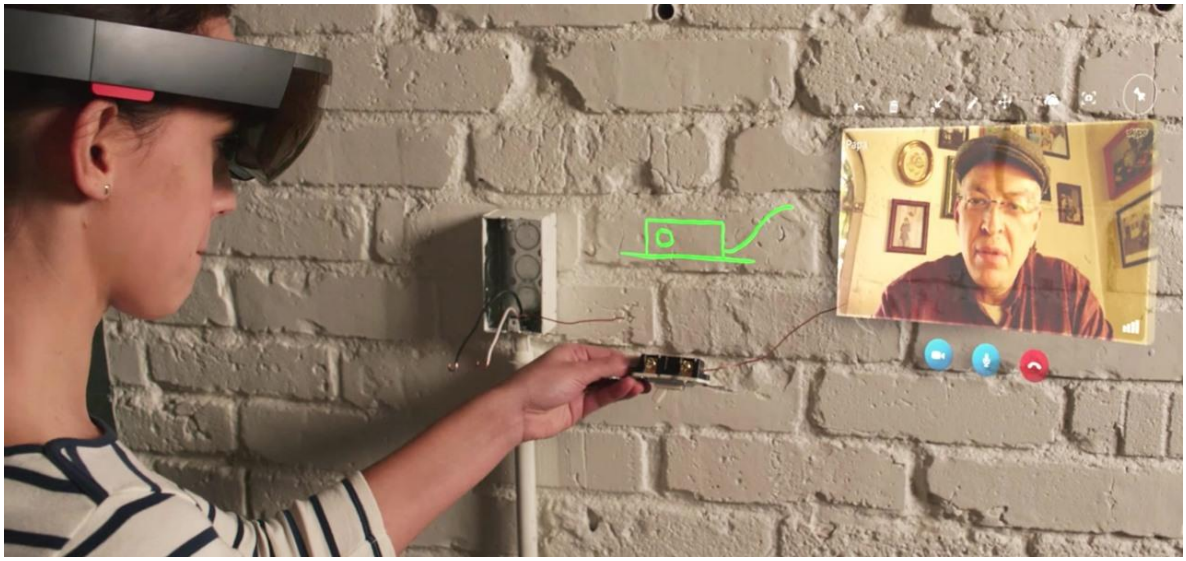

FIGURA 20 Demostración de la aplicación Skype (aplicación de videollamadas) embebida por medio de realidad mixta a través de cascos montados en la cabeza por Microsoft. (Dachis, 2018)

la creación de este tipo de ambientes, gracias al rápido crecimiento de sus capacidades, que ha tenido lugar en los últimos años (Chatzopoulos, Bermejo, Huang, & Hui, 2017). Las ventajas que representan son diversas, por ejemplo; una gran cantidad de personas poseen un teléfono inteligente, en muchos casos son más accesibles que configuraciones especializadas para realidad virtual o aumentada, como en el caso de sobreponer un ultrasonido a una mujer embarazada para realizar una operación (Bajura, Fuchs, & Ohbuchi, 1992) o la creación de un CAVE, que consiste en proyectar imágenes en un espacio comúnmente cerrado (Cruz-Neira, Sandin, & DeFanti, 1993).

Desde el punto de vista de sus capacidades de procesamiento y componentes físicos ("hardware") estos tienen un buen poder de procesamiento, acompañados de baterías que permiten su uso por tiempos prolongados sin necesidad de conectarlos, así mismo estos contienen un numero de componentes integrados los cuales en el presente trabajo se propone utilizar en el desarrollo de los métodos y algoritmos para presentar al usuario una realidad mixta, entre estos componentes destacan: cámara(s), micrófono(s), pantalla, giroscopio, acelerómetro, y GPS.

Estos dispositivos presentan una nueva oportunidad para la creación y desarrollo de la realidad mixta, permitiéndonos añadir estímulos para otros sentidos además de la vista, como son el oído a través del sonido estereoscópico (Cullen, Collins, Hogue, & Kapralos, 2016) y el tacto como los guantes hápticos (Dipietro, Sabatini, & Dario, 2008; Sturman & Zeltzer, 1994) introduciendo componentes externas controladas por el dispositivo móvil. Los cuales son de suma importancia para lograr el efecto de inmersión deseado, pero representan retos más grandes en cuanto a implementación, a diferencia de los estímulos visuales.

4.3.7 Calibración de cámaras

El modelo de la cámara de pinhole nos permite mapear los puntos de una escena 3D con sus equivalentes en la proyección 2D obtenida por la cámara. Sin embargo, como se discutió anteriormente, para poder utilizar este modelo es necesario que la imagen proyectada no haya sufrido ningún tipo de distorsión antes de llegar al plano de proyección.

En la práctica esto no ocurre, las cámaras reales no siguen de forma estricta el modelo de la cámara de pinhole, las imágenes obtenidas suelen tener algún tipo de alteración, provocada la gran mayoría de las veces por aspectos físicos difíciles de controlar. Por ejemplo, el lente de la cámara añade una deformación en sí mismo pues no es tomado en cuenta como parte del modelo de pinhole. A su vez los componentes de la cámara pueden no estar perfectamente alineados en paralelo con el sensor que funge como plano de proyección. El calor u otros elementos del exterior pueden deformar los distintos elementos de una cámara constantemente.

Sin embargo, si se lograran tener en cuenta todas estas alteraciones, se podría utilizar el modelo de pinhole sin ningún problema. La calibración de una cámara funge como solución para este problema. Es por eso que la calibración suele ser fundamental en visión por computadora.

El objetivo es poder obtener parámetros extrínsecos (la rotación y traslación, importantes en visión estéreo para entender la relación entre dos cámaras) e intrínsecos (como la distancia focal y el centro óptico) de la cámara en cuestión. Calibrar una cámara es igual a determinar los parámetros necesarios para mapear un punto tridimensional correctamente a su equivalente 2D. Los parámetros externos están relacionados con la posición de la cámara en el mundo real, mientras que los internos se relacionan con las propiedades físicas de la misma. Ambos necesarios para el modelo de la cámara de pinhole.

Cuando se requiere hacer reconstrucción 3D, visión estereoscópica, medición de profundidad o cualquier tipo de medición utilizando una cámara, es fundamental calibrar las cámaras que se utilicen para poder garantizar que los datos recabados son fieles a la

realidad. Dependiendo de la aplicación será la frecuencia con que dichas cámaras deberán ser calibradas.

Existen muchos métodos para calibrar una cámara, pero en general se utilizan varios puntos de referencia en el mundo real, de los que se conoce con antelación su posición y distancia entre sí. Posteriormente una o varias imágenes son tomadas con la cámara para poder comparar la variación que existe entre lo esperado (puntos conocidos) y el resultado obtenido y así obtener la matriz de transformación de la cámara.

En general los dos algoritmos más utilizados para calibración de cámaras son el de Zhang Z. y Heikkila y Silven (Zhang Z., 2000; Heikkila J. & Silven O., 1997), para la obtención de la matriz de transformación según el modelo de la cámara de pinhole y la distorsión del lente respectivamente. Estos están implementados en herramientas enfocadas al procesamiento de imágenes como es el caso de OpenCV y MATLAB.

4.3.8 ¿Cómo mezclar lo real y lo virtual correctamente?

Cuando se busca lograr la convivencia del mundo real y objetos virtuales, es necesario contar con diversos métodos que permitan mezclar ambas partes con la mayor armonía posible. Por ejemplo, cuando se trata del mundo real, la posición de los objetos respecto a otros obedece las leyes de la física, como que un objeto no puede ocupar el mismo espacio que otro, o la gravedad que los obliga a permanecer sobre una superficie. Los objetos virtuales no se rigen por ninguna de estas reglas, surgiendo de aquí el reto principal cuando se habla de realidad aumentada o mixta.

Dado que los objetos virtuales pueden ser manipulados fácilmente dentro del mundo virtual, el desafío principal aparece en como indicarle al ordenador donde y como colocarlos. Recordemos que las cámaras nos proporcionan información visual del mundo real a modo de imágenes, las cuales para la computadora son solo un conjunto de pixeles que a su vez se podrían traducir como un conjunto de números con una posición determinada.

Una de las formas más sencilla para juntar la realidad con lo virtual es el de superposición, donde se toma la imagen del mundo real y encima se coloca un objeto virtual que ocluye al mundo real (similar a colocar un objeto frente a una fotografía). Sin embargo, este método no proporciona la sensación de inmersión al usuario.

Con el crecimiento de la tecnología y el interés en los campos como son la realidad aumentada y la realidad mixta, han ido surgiendo nuevos métodos para poder determinar las características del mundo real a través de procesamiento de imágenes que permitan convivir en armonía ambas partes.

A continuación, se discutirán algunos de los métodos más comunes para lograr este cometido.

4.3.8.1 Marcadores

Los marcadores son un tipo de código visual fácil de reconocer por medio de procesamiento de imágenes. Pueden ser en realidad cualquier imagen o código. El algoritmo busca en cada cuadro de la imagen (enviado desde una cámara) puntos que coincidan con un marcador previamente cargado en la aplicación. Una vez encontrado se desencadena alguna acción que por lo general consiste en "aparecer" un objeto virtual sobre este. Los algoritmos más avanzados toman en cuenta el modelo de la cámara de pinhole para determinar la posición del marcador, así como su traslación y rotación, algunos otros toman en cuenta también la perspectiva. Los marcadores no son solo visuales, pueden ser de cualquier tipo adaptados a la clase de sensor que se desea utilizar para detectarlos. Sin embargo, los más comunes son imágenes para ser detectadas por una cámara.

El más común de sus usos es el rastreo óptico que es un tipo de tecnología para localización en tres dimensiones basado en monitorear un objeto o espacio previamente medido por una o más cámaras. La posición puede ser calculada usando un único marcador, para medir la orientación o seguir varios objetos es necesario colocar múltiples marcadores. Una de las configuraciones típicas está basada en la utilización de cámaras infrarrojas activas, las cuales despiden cierta cantidad de luz en esta longitud de onda (no es dañina para el ojo humano) y captan los reflejos de esta. Los objetos que requieren ser rastreados son marcados con algún material reflejante, que en la cámara infrarroja puede verse como un punto saturado (blanco). Los reflejos son captados por la cámara la cual conoce previamente la forma y medidas de los marcadores, por medio de un proceso de calibración, se puede conocer entonces la posición del marcador respecto a la cámara (Rehder, Nikolic, Schneider, & Siegwart, 2017; Vasileiou & Psarakis, 2015).

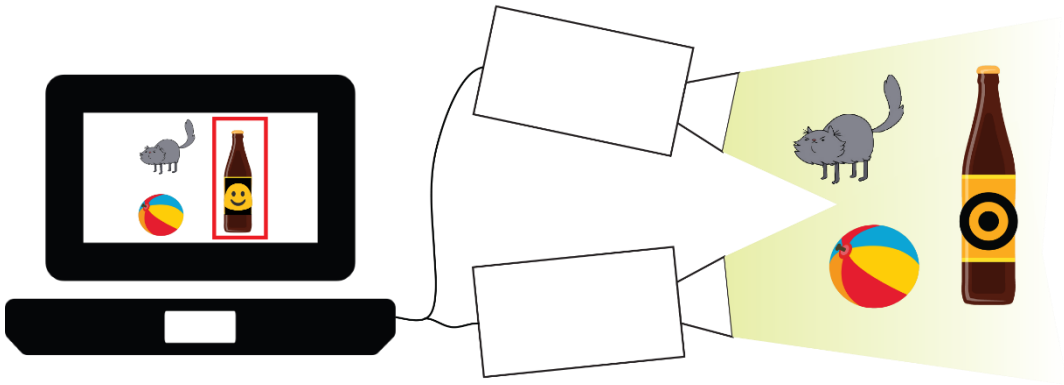

FIGURA 21 Sistema clásico de rastreo 3D por marcador.

Este método es de los más utilizados por su facilidad de implementar y rapidez. Sin embargo, las desventajas principales de los marcadores se deben a la naturaleza de estos. En primer lugar, deben existir físicamente en el mundo real, así que son sujetos de desgaste, deterioro, oclusiones y deben ser colocados estratégicamente si se desea que den buenos resultados.

4.3.8.2 Métodos sin marcadores

Dadas todas las desventajas de utilizar marcadores, se han buscado diversos métodos para prescindir de ellos. Esto requiere un mayor procesamiento de la imagen, lo cual puede volverse muy complejo dependiendo del objetivo. La mayoría de estos métodos tiene que ver con visión por computadora, donde se intenta analizar y entender la escena presente en la imagen y así poder colocar sobre ella de la manera más adecuada los componentes virtuales. Los métodos son tan amplios como el tema de visión por computadora, y por lo general se especializan en un área, por ejemplo, determinar las fuentes de luz de la escena para que el objeto virtual proyecte una sombra acorde (Jung, Choi & Hong, 2014). Algunos otros se centran en la búsqueda de planos utilizando visión estéreo y obteniendo medidas de profundidad, y otros como es el caso de los filtros en redes sociales como Instagram y Snapchat se encargan del reconocimiento de objetos o cosas específicas, en este caso el reconocimiento facial.

Existen en el mercado varios aplicativos enfocados exclusivamente en la detección de objetos para la mezcla de lo real y lo virtual, tal es el caso de Vuforia, que compila distintos métodos tanto de marcadores como de visión por computadora. Es multiplataforma y se puede utilizar en conjunto con Unity para crear aplicaciones de realidad aumentada.

La utilización del método adecuado dependerá de las necesidades de la aplicación, los marcadores suelen proveer mayor precisión que los métodos que no los utilizan, sin embargo, con el paso del tiempo los nuevos algoritmos sin marcadores seguramente los superaran si no es que ya lo han hecho.

4.4 APLICACIONES

Esta parte del trabajo presente se centrará en las distintas aplicaciones de la Realidad Aumentada y posteriormente de la Realidad Mixta. Este tipo de tecnología brinda un gran número de posibilidades nuevas para los usuarios y desarrolladores, añadiendo elementos a la clásica configuración de una pantalla y cámaras. En cualquier ámbito donde sea posible la utilización de dispositivos electrónicos buscando una interacción entre las personas y lo virtual es posible integrar esta tecnología, por ejemplo, en la educación, la arquitectura, el entretenimiento, la medicina, entrenamiento o rehabilitación entre otras.

Dichas aplicaciones pueden ser tan elaboradas como se desee. Pueden ir desde simples implementaciones como mostrar objetos virtuales al usuario hasta interacciones complejas entre el mundo real y virtual.

4.4.1 Entretenimiento.

Uno de los mercados que explota la realidad aumentada y mixta es el del entretenimiento. Los videojuegos son una muestra de ello. El sueño de cualquier jugador es vivir la experiencia del juego en carne propia, es decir participar activamente en él, descubrir los diferentes lugares y acciones que fuera del juego sería imposible realizar. Esto ha llevado a enfocar los diferentes avances en estas áreas al entretenimiento, brindando experiencias más realistas e inmersivas para las personas.

Una de las desventajas de jugar videojuegos es que la navegación del ambiente y la comunicación con otros jugadores debe hacerse por medio de dispositivos muchas veces difíciles de usar para algunos usuarios, a diferencia del mundo real donde dichas interacciones ocurren de forma natural (como sostener una conversación con alguien frente a frente). La interfaz del mundo real es intuitiva y bien conocida por cualquier persona de cualquier edad o nivel de experiencia. Las personas se relacionan social y físicamente sin mayores complicaciones, mientras que para lograrlo utilizando algún medio electrónico debe pasarse por una etapa de aprendizaje que permita entender y utilizar el dispositivo adecuadamente. Sin embargo, los videojuegos tienen la ventaja de permitir mostrar contenido sofisticado como son animaciones y modelos tridimensionales que son muchas veces de cosas imposibles en el mundo real, además de permitir a usuarios de distintas partes del mundo interactuar entre ellos desde la comodidad de su hogar. Actualmente los juegos de realidad aumentada buscan mantener las ventajas de los videojuegos y eliminar la desventaja principal que es la de entorpecer las interacciones intuitivas de los jugadores

FIGURA 22 Una de las estatuas de la casa embrujada de Disney, sin y con la proyección de realidad aumentada. Un escenario tridimensional interactivo que cambia con la interacción de los usuarios. (Mine, Van Baar, Grundhofer, Rose, & Yang, 2012)

entre sí y con el mundo real. Los objetivos pueden variar, pero en general se espera que el usuario sea capaz de permanecer dentro del mundo real y a través de un dispositivo permitirle acceder a las partes virtuales que conformaran el juego. Un ejemplo de esto es este pequeño juego que permite formar equipos que deberán buscar y encontrar ciertos objetos virtuales, cada persona explora el mundo físico donde se encuentra para lograr completar el objetivo que es la recolección de todos los objetos antes que el otro equipo (Mulloni, Wagner, & Schmalstieg, 2008). Este ejemplo sencillo pero eficiente es una muestra del objetivo de utilizar la realidad aumentada en el campo del entretenimiento, en este caso se conservan las capacidades de las personas de interactuar entre sí y con el mundo real, manteniendo la interacción con los objetos virtuales del juego y la mecánica de este. Las posibilidades son muchas.

Otro ejemplo de utilizar realidad aumentada es el de incluirla en los parques de Disney, estos parques están hechos de tal forma que los usuarios sientan que entran en un mundo fantástico y mágico, y la tecnología ha demostrado tener un papel importante (Mine, Van Baar, Grundhofer, Rose, & Yang, 2012). Por medio de un sistema de cámaras y proyectores, se toman los elementos reales como son esculturas, edificios o escenarios y se proyectan sobre ellos imágenes o luces que brindan la sensación de estar en movimiento. Por ejemplo en la casa del terror se tienen algunas esculturas que son incapaces de moverse por sí mismas, posteriormente se proyectan sobre ellas imágenes para hacerlas parecer en movimiento, la ventaja de hacer esto con realidad aumentada a diferencia de hacerlo con piezas movibles por ejemplo es que además de obtener el resultado deseado se cuenta con la posibilidad de cambiar fácilmente la proyección en caso de ser necesario, puesto que se trata solo de una imagen o video y no de toda una instalación lo que debe ser cambiado.

Otro de los ejemplos es la caja de arena, este escenario es un poco más complejo, pues se trata de un claro ejemplo de realidad mixta. Sobre una caja de arena se colocan un par de cámaras o algún dispositivo que sea capaz de medir profundidad y un proyector. Las cámaras determinan la altura de la arena (la cual puede ser modificada por los usuarios) y el proyector cambia el color de una zona de acuerdo con su altura. Digamos que toda la arena se encuentra aplanada, toda lucirá del mismo color, en el momento en que alguien acumule arena en un punto haciendo una montaña, esta será coloreada de un tono diferente.

Un ejemplo más claro de realidad mixta es este juego (Sing & Xie, 2016) llamado Garden. El juego consiste en un casco que va montado en la cabeza, a través del cual se ve el mundo real, el escenario donde está el usuario es escaneado y sobre él se crea una cuadricula virtual con la cual el usuario puede interactuar, creando y colocando objetos virtuales. El juego responde a movimientos de las manos del usuario. Gracias a la configuración del hardware el usuario tiene las manos y piernas libres todo el tiempo, a través del casco

observa el mundo real en el que se encuentra evitándole chocar con los objetos y paredes, y al mismo tiempo puede interactuar con su jardín virtual.

4.4.2 Diseño e Industria

En el campo de la industria muchas veces es muy costoso crear prototipos de algún objeto, sobre todo porque la creación de estos requiere el gasto de energía y materiales reales. La realidad aumentada y mixta pueden formar parte importante sino esencial del proceso de creación de algún producto. Por ejemplo, es muy común en arquitectura que antes de iniciar con el proceso de finalización de un lugar, por ejemplo, una vez que se tiene la construcción física se debe planear entonces el tipo de piso, el color de las paredes, la posición de lamparas y muebles que necesitan instalaciones especiales de electricidad, agua o gas. Antes de incurrir en ningún gasto para instalar todos estos elementos lo que lleva tiempo y dinero se hace una muestra digital de como quedara el producto final y la construcción no se inicia hasta que esta se aprueba. Incluir la tecnología puede ser importante, pues estas muestras digitales suelen ser fotografías que no cubren todos los aspectos de la construcción, sino solo muestra fotografías de áreas muy generales. Imaginemos la diferencia e impacto que tendría poder mostrar al cliente, por medio de un casco de realidad virtual como se verá el lugar una vez terminado, y permitirle caminar libremente por el área viendo y viviendo la experiencia completa. Esto le permitiría tomar mejores decisiones sobre el rumbo del diseño y tomar atención a todos y cada uno de los detalles de este.

Este concepto ha ganado popularidad con el tiempo, pues puede ahorrar a una empresa mucho dinero ya que la retroalimentación y cambios al producto pueden ser hechos en una etapa anterior al desarrollo y prototipado (Purdy & Choi, 2014). Tener un sistema que permita crear los objetos para ser analizados, pero no solo eso sino permitir la visualización de los mismos de una manera más intuitiva e incluso añadir otros tipos de interacción como la háptica con los objetos.

Un ejemplo más sobre esto, es este sistema que permite practicar las habilidades para esculpir (Yamamoto, Kawagoe, Otsuki, Shibata, & Kimura, 2017). Al usuario se le da un trozo de madera u otro material, entonces el usuario con los cascos de realidad virtual los ve y se le dan herramientas hápticas que simulan las herramientas reales utilizadas para esculpir, el usuario así puede practicar sus habilidades, viendo los resultados de sus acciones en tiempo real sin desperdiciar materiales o herramientas reales.

Por último, el uso de aplicaciones de realidad mixta puede ser utilizado para garantizar la seguridad de las personas que trabajan en ambientes peligrosos. El entrenamiento en estos ambientes suele ser en un salón de clases, donde se enseña únicamente por medio de teoría los procedimientos necesarios para operar maquinaria, por ejemplo. Sin embargo, de la

teoría a la práctica existe un mundo de distancia, en primer lugar, operar maquinaria puede ser peligroso para aquel que no cuenta con la experiencia necesaria en el campo, además de los costos que traerá consigo el caso de un accidente o de una falla por falta de experiencia del operador. Aun que las empresas en su mayoría buscan personas capacitadas previamente por lo general en la industria se requiere algún tipo de entrenamiento, pues las maquinas suelen ser diferentes en cada caso, ya que son hechas a la medida y destinadas con objetivos específicos. La creación de aplicaciones de realidad mixta puede servir de avance para estas capacitaciones necesarias. Se le permite al usuario interactuar con la maquina virtualmente, en un ambiente controlado donde los errores no significan gran problema. Un ejemplo de esto es este sistema de realidad mixta que permite entrenar a los nuevos empleados en el correcto uso y protocolos a seguir dentro de una refinería (Träskbäack & Haller, 2004).

4.4.3 Medicina y Rehabilitación

A principios de los noventa comenzaron a existir diversas propuestas y lentamente la realidad aumentada fue siendo aceptada en este campo dando paso a diversos prototipos y aplicaciones enfocadas a áreas específicas de la medicina. Algunos de los más comunes están relacionados con el entrenamiento de estudiantes en diversas áreas para evitar que estos practiquen con seres vivos, haciendo el proceso de aprendizaje menos estresante para ellos. Sin embargo, el desarrollo de sistemas de realidad aumentada o mixta para el uso en el campo de la medicina ha presentado diversos retos a atacar como es lograr la correcta percepción de los objetos virtuales embebidos en la realidad, la integración de esta tecnología en trabajos médicos más complejos y sobre todo un choque cultural. Cambiar la forma en que funciona la educación tradicional y los procedimientos de entrenamiento es quizá la parte más complicada, pues existe una reticencia por parte de los expertos a abrazar estas nuevas formas. A pesar de esto muchos desarrolladores aun enfocan sus esfuerzos en demostrar la viabilidad del uso de sistemas virtuales tanto en los salones de clase como las salas de operaciones.

Uno de los más interesantes quizá sea este dispositivo que proyecta sobre la persona imágenes de rayos X, evitando que el usuario deba ser expuesto a radiación en diferentes ocasiones. El sistema cuenta con una cámara especial y un espejo sobre el que se refleja la proyección superpuesta de los rayos X. Estos están alineados correctamente con la extremidad del usuario, permitiendo al médico observar en tiempo real al usuario y su radiografía, facilitando así el proceso de operar a un individuo con precisión (Navab, Blum, Wang, Okur, & Wendler, 2012).

Las ventajas de tener un sistema capaz de mostrar al médico información relevante para una intervención pueden parecer obvias. Esto podría reducir en gran nivel la dificultad para llevar a cabo un procedimiento tan complejo como es una operación, donde literalmente la vida del individuo depende de las capacidades y experiencia de quien lo interviene. Por tanto, reduciendo los riesgos que se corren con este tipo de experiencias.

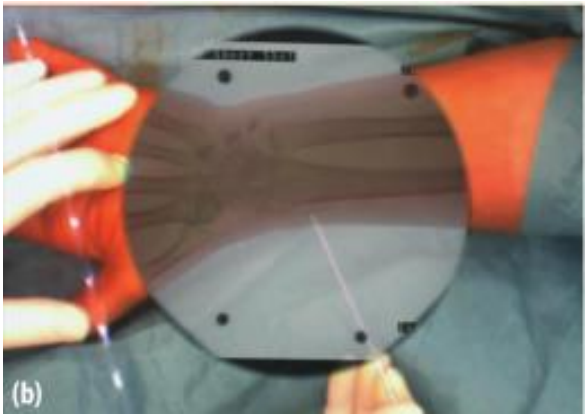

FIGURA 23 Imagen del CamC en acción superponiendo rayos X al brazo de un paciente. (Navab, Blum, Wang, Okur, & Wendler, 2012)

Sin embargo, la realidad aumentada y mixta se pueden utilizar para otras variantes como son la rehabilitación de los pacientes. Actualmente es muy común que una persona deba pasar por un proceso metódico con el fin de lograr su completa recuperación después de un suceso traumático como puede ser una operación o algún suceso de gran impacto psicológico. Comúnmente las personas son tratadas por un profesional que debe evaluar el estado del paciente y determinar si debe suspender, cambiar o continuar con el proceso, esta evaluación suele ser subjetiva pues depende en gran medida de la retroalimentación que da el paciente, así como del nivel de experiencia del terapeuta. Otra desventaja de los métodos tradicionales de recuperación es que estos deben realizarse en lugares fuera de casa, ya sea por la necesidad de utilizar aparatos especiales o por la incapacidad del terapeuta de moverse hasta donde está el paciente.

Existen distintas formas en que se puede aplicar la realidad aumentada y mixta a estos métodos. Por ejemplo uno de los más comunes es atacando problemas psicológicos como son las fobias de un paciente (Carlin, Hoffman, & Weghorst, 1997). La fobia a las arañas es una de las más comunes, y puede llegar a casos extremos afectando la vida de la persona que la padece. Las respuestas de ansiedad elevadas pueden impactar de forma negativa las relaciones con el mundo real de un sujeto en particular. Por su parte el paciente reconoce que su miedo es excesivo e irracional sin embargo la incapacidad que siente de superarlo puede añadir un factor extra al estrés de por si provocado por el objeto de su fobia. Sin embargo, el proceso de rehabilitación comúnmente incluye exponer a la persona a dicho objeto, aun que se haga de forma paulatina, poner al sujeto en contacto directo con el causante de su ansiedad puede provocar el efecto contrario al deseado, aumentando los

niveles de estrés y ansiedad, logrando que la persona haga relaciones aún más negativas con ese evento en particular.

La introducción de un proceso de realidad aumentada o mixta puede ser de gran ayuda en el tratamiento de problemas psicológicos. En primer lugar, permite desarrollar un ambiente controlado en el que la persona será sumergida, a diferencia del enfoque realista, los gráficos generados por computadora pueden proveer de herramientas diversas como son crear modelos tridimensionales caricaturizados del objeto causante de ansiedad. Así mismo se puede medir en tiempo real los niveles de estrés del paciente, pudiendo

FIGURA 24 Lucas la araña, personaje ficticio para la posible rehabilitación de la aracnofobia. IMAGEN: (Wikipedia, 2018)

suspender inmediatamente de ser necesario el tratamiento. Se obtiene mayor control sobre lo que el usuario ve y como responde ante dicho estimulo y permite la creación de escenarios controlados. Por ejemplo, para un usuario que les teme a las arañas se puede colocar una araña virtual en la habitación con él, detrás de un cristal virtual y permitir al usuario acercase a su tiempo, así mismo se puede crear una araña menos realista que en una etapa inicial de la rehabilitación sirva como incentivo para que el usuario continúe a su ritmo con la misma.

Aunque no utiliza realidad virtual como tal actualmente se considera utilizar a este personaje (Figura 23) para el tratamiento de la aracnofobia. Este personaje es generado por computadora y actualmente es protagonista de diversos videos, el fin es el de ayudar a las personas que temen a las arañas a superar sus miedos por medio de un personaje caricaturesco y tierno.

Por supuesto la recuperación psicológica no es la única que puede beneficiarse de la tecnología, otros procesos que involucran rehabilitación física del paciente han mostrado excelentes resultados al integrar en su metodología herramientas de realidad mixta o aumentada. Estas herramientas pueden por una parte proporcionar un atractivo extra de motivación para la persona que debe realizar ciertas actividades metódicas por largos

periodos de tiempo, en forma de juego, por ejemplo. Al mismo tiempo puede proporcionar retroalimentación en tiempo real al usuario sobre su desempeño y por último puede ser utilizado como herramienta en el proceso mismo, permitiendo al usuario interactuar con objetos reales y generados por computadora al mismo tiempo. Este juego es un claro ejemplo de esto (Bur et al., 2010), con el fin de rehabilitar a pacientes que sufrieron de un ataque cardiaco y se encuentran imposibilitados de alguna de sus extremidades superiores, se crearon varios juegos virtuales, que en conjunto con objetos reales piden al paciente realizar ciertas actividades.

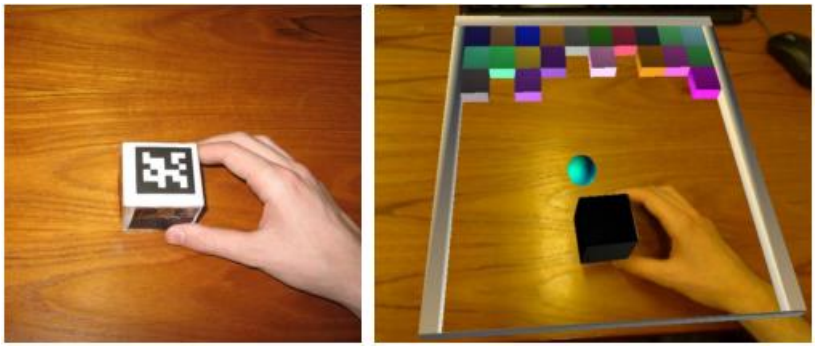

FIGURA 25 Una escena real con el jugador sosteniendo un cubo y su contraparte virtual inmerso en un juego. (Bur et al., 2010)

El sistema funciona con marcadores que son colocados en los objetos a manipular por la persona para ser complementados con un ambiente virtual que constituye un pequeño videojuego. Un ejemplo más de los muchos que existen es este juego para niños (Munroe, Meng, Yanco, & Begum, 2016) con parálisis cerebral, el objetivo es que los pacientes puedan realizar actividades relacionadas con su rehabilitación en el propio hogar, siendo supervisados por un adulto que ve los resultados reflejados en el sistema. El niño o niña puede mover objetos en patrones previamente definidos por un terapeuta embebidos posteriormente a modo de juego en la aplicación. Este es un claro ejemplo de reducción de costos, pues una vez adquirido el sistema este tipo de procesos que deben realizarse por largos periodos de tiempo, se pueden hacer en casa, un ambiente controlado donde el infante se siente seguro y evitando a los tutores la necesidad de desplazarse físicamente para lograrlo, lo que en muchas ocasiones puede provocar el abandono del tratamiento.

4.5 ¿CÓMO LLEGAMOS A MUCHOS USUARIOS?

Las aplicaciones son muchas en distintos campos de estudio, como hemos visto en el bloque anterior, sin embargo, la mayoría de estas aplicaciones son muy específicas y requieren de configuraciones especificas tanto de hardware como software. Esto entorpece el desarrollo de nuevas tecnologías en el área, pues cada nuevo desarrollo debe empezar de cero a pesar de que podría fácilmente alimentarse de otros aplicativos desarrollados con anterioridad.

A finales de los ochenta y principios de los noventas durante el auge de los videojuegos, cada pieza que alcanzaba el mercado era hecha completamente de forma independiente. Cada decisión sobre las características, diseño e implementación era desarrollada individual y manualmente. El contenido gráfico y sonoro era incluido directamente en el código como grandes piezas de números binarios. Sin embargo, los desarrolladores se dieron cuenta de que grandes partes del código de un juego eran reutilizables y con las modificaciones necesarias podían permitir a expertos de otras áreas como diseñadores y músicos incluir sus obras dentro del producto final sin la necesidad de saber cómo escribir código. A esto se le empezó a conocer como motor de juego, y por muchos años fueron librerías que se movían y modificaban dentro de una empresa.

Poco a poco se fueron creando herramientas que permitan a los desarrolladores hacer aplicaciones cada vez más complejas en menos tiempo procurando hacer de la curva de aprendizaje para el desarrollo una más corta y fácil de entender. Lo que permitió dar acceso a personas externas a esta tecnología y hacerla disponible para cualquier usuario.

Así mismo la aparición de los dispositivos móviles como una herramienta más común y mundialmente usada ha permitido llevar a muchos usuarios distintos tipos de tecnología. Como se discute anteriormente, el auge de la realidad virtual, aumentada y mixta sea visto beneficiada por empresas que invierten en dispositivos más accesibles para los usuarios comunes.

4.5.1 Motores de juegos

Un motor de juego es un paquete de software creado con la intención de que un desarrollador pueda crear un videojuego especifico utilizando las herramientas incluidas en el y añadiendo código propio que permita funciones específicas para dicho fin. En otras palabras, suele ser una librería de software, muchas veces acompañado de una interfaz gráfica.

Por supuesto el uso de estos motores es opcional, pues se puede crear un juego de la nada con puro código, pero un videojuego que no utilizo ningún motor de juegos para su creación de igual forma contendrá el mismo código que aquel que lo utiliza. Sin embargo, la utilización de estos permite la creación de contenido más rápida y eficiente en lugar de dejar al desarrollador incursionar por si mismo en todos los aspectos necesarios para obtener un producto final funcional.

Existen actualmente muchos motores diferentes que se dedican específicamente a ciertas tareas, como pueden ser desde simuladores físicos hasta motores específicos para hacer render de escenas. Por lo general la combinación de uno o más motores permite crear contenido más sofisticado de lo que un equipo de desarrollo podría hacer por sí mismo.

4.6 ¿POR QUÉ UN ENTORNO DE TRABAJO?

En términos generales un entorno de trabajo es una estructura ya sea real o conceptual que tiene la intención de servir como soporte o guía en la creación de algo con objetivos específicos (Riehle, 2000).

Específicamente en el área de computación, un entorno de trabajo es comúnmente un conjunto de estructuras ya sea en forma de documento o código que permita la construcción de un programa o aplicación y especifique la forma en que unas partes conviven con otras dentro del programa. Algunos entornos de trabajo incluyen código o programas en sí mismos reutilizables en otros proyectos proporcionando diversas herramientas para un fin específico.

En el área de realidad mixta, a pesar de la popularidad ganada en los últimos años y los diferentes esfuerzos de investigación y avance, aun parece existir un gran hueco en cuanto a la creación de aplicaciones, pues cada una debe ser desarrollada independientemente, comúnmente utilizando elementos de realidad virtual y realidad aumentada muchas veces difíciles de combinar entre sí.

La creación de un entorno de trabajo, a diferencia del desarrollo de una aplicación específica, permite su fácil incorporación a otros trabajos y aplicaciones, además de facilitar el contarte mejoramiento de dicho software (María & Haro, 2008).

Específicamente los entornos de trabajo orientados a objetos, facilita el proceso de diseño, aprendizaje y creación a comparación con los modelos tradicionales de reutilización de software (Riehle, 2000). Esto promete mayor productividad y tiempos más cortos de realización para el desarrollo de aplicaciones.

Sin embargo, esta configuración da pie a algunos problemas, no para quien utiliza el entorno de trabajo, sino para el desarrollador de este. En primer lugar, es necesario un diseño correcto de este que permita a otros usuarios añadir contenido, así como la fácil utilización de este. Es común que el proceso de creación de un aplicativo deba ser suspendido por un framework que no es cien por ciento funcional. La creación de un entorno de trabajo requiere clases complejas que definan correctamente el comportamiento de objetos y sus diversas instancias y sean lo suficientemente flexibles para funcionar en distintos contextos y con distintos objetivos. Así mismo dichas clases deben estar correctamente aisladas unas de otras, de tal forma que el usuario final pueda acceder a las distintas partes de las herramientas proporcionadas a discreción.

4.7 DESCRIPCIÓN GENERAL DEL SISTEMA

El trabajo presente pretende atacar varias problemáticas las cuales de desglosan a continuación.

- Adquisición de información visual. Creación de un sistema que permita integrar una o tantas cámaras como sean necesarias a un proyecto. Esto se logrará a partir de código aislado en capsulas, donde cada una tendrá una función distinta, desde desplegar las cámaras conectadas y disponibles hasta asignar a un objeto dicha cámara para que pueda ser visualizada.
- El despliegue visual de información. Que contiene métodos de despliegue en un dispositivo de realidad virtual especializado con estereovisión para crear la sensación de profundidad. Esto incluye el despliegue de superficies virtuales, así como imágenes del mundo real adquiridas a través de las cámaras anteriormente mencionadas.
- Intercepción de las imágenes. Con el fin de proporcionar un puente por el cual se puedan incluir filtros y otras herramientas de procesamiento de imágenes en el trabajo fácilmente.

4.7.1 Contribución y relevancia.

- La creación de un entorno de trabajo, a diferencia del desarrollo de una aplicación específica, permite su fácil incorporación a otros trabajos y aplicaciones, además de facilitar el constante mejoramiento de dicho software (María & Haro, 2008).
- Creación de entornos experimentales que permitan la mejor compresión de cómo se relacionan los entornos virtuales y la forma en que los humanos interactúan con ellos (Stanney et al., 1998).
- Creación de entornos que faciliten el entrenamiento de alguna tarea en el mundo real (Kawagoe, Otsuki, Shibata, & Kimura, 2016).
- Facilitar la creación de aplicaciones cuyo objetivo sea la rehabilitación de pacientes, así como la extracción de datos sobre los mismos (Tzovaras, Moustakas, Nikolakis, & Strintzis, 2009).
- Creación de entornos de realidad mixta para el entretenimiento y la industria (Arisandi et al., 2012; Träskbäack & Haller, 2004).

5 HARDWARE Y SOFTWARE PARA EL ENTORNO DE TRABAJO DE REALIDAD MIXTA

La integración del sistema de Realidad Mixta se lleva a cabo, como se menciona en el objetivo, interconectando distintos elementos disponibles actualmente. Para lograrlo se recurrió a distintos dispositivos y herramientas que se encuentran disponibles en el mercado.

A continuación, se provee con una descripción más detalla de dichos componentes, el funcionamiento, así como las especificaciones de cada uno de forma independiente.

5.1 HERRAMIENTAS

Se dice por ahí que aquel que crea todo su código desde cero más que ser muy inteligente es un tonto. Actualmente existen cientos de herramientas y programas disponibles en la web, la mayoría de ellos incluso contando con una versión gratuita. Siguiendo la misma filosofía que impulsa al desarrollo de un entorno de trabajo en lugar de una aplicación especifica (fácil incorporación, modificación y reutilización en otros trabajos), para crearlo se combinaran diversas herramientas ya existentes en el mercado. Las principales tanto en hardware como de software se especifican a continuación.

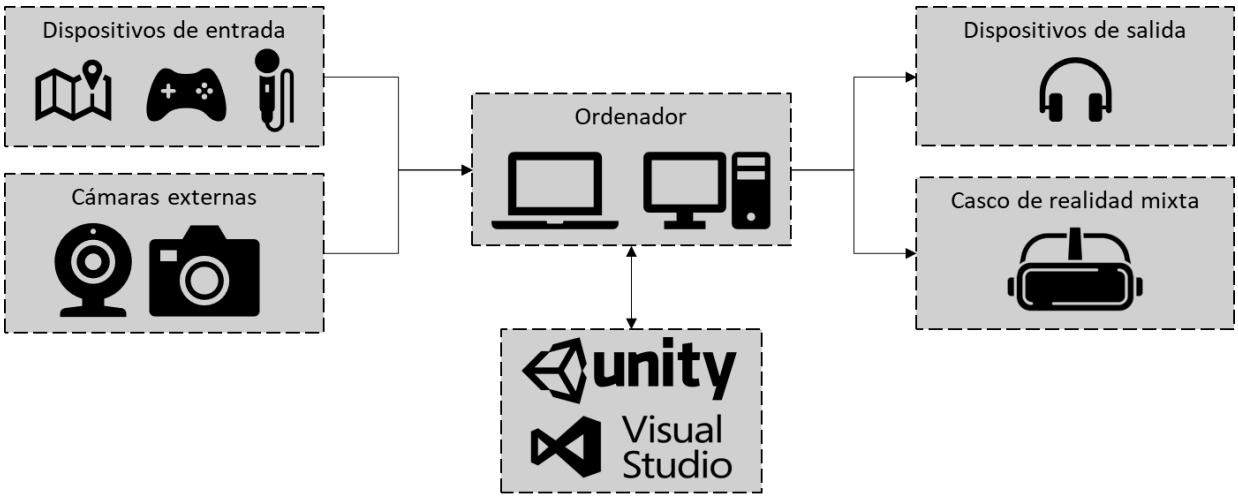

FIGURA 26 Diagrama general de las herramientas a utilizar.

5.1.1 Casco de realidad Mixta para Windows por Lenovo (Explorer)

Son unos lentes de inmersión de realidad virtual, compatible con la Realidad Virtual Mixta de Microsoft, ya sea a través de más de 20,000 aplicaciones de la tienda Microsoft, en la creación de entornos de productividad a través de Microsoft Office Suite o para el desarrollo de aplicaciones de realidad virtual, aumentada o mixta para la plataforma universal de Windows. Tiene un peso de solo 380 gramos y cuenta con sujeciones superiores ajustables, así como una parte acolchada en las gafas para ofrecer comodidad al usuario y evitar la entrada de luz exterior a las pantallas internas.

FIGURA 27 Casco de Realidad Mixta Lenovo Explorer y controles de movimiento. (Lenovo Explorer, 2018)

El equipo cuenta para su función con dos pantallas internas de tipo LCD con resolución 2880x1440 de 2.89 pulgadas con una tasa de refresco de 90Hz, estas pantallas generan un campo de visión de hasta 110 grados. Por la parte externa, el dispositivo monta 2 cámaras de seguimiento de movimiento inverso (inside-out tracking), así como sensores de proximidad, giroscopio, acelerómetro y magnetómetro para registrar giros del usuario, así como posición con respecto al horizonte de la cabeza. Se enlaza al equipo de cómputo por medio de cables HDMI, USB 3.0 y Jack de Audio.

La comunicación e interacción con los entornos de realidad mixta se realizan a través de los mandos inalámbricos Lenovo Explorer, los cuales además de contar con sensores de movimiento, llevan además una palanca análoga direccional. Además de estos mandos específicos, es compatible con controles estándar de Xbox, teclado y ratón, e incluso comandos de voz del asistente virtual de Microsoft, Cortana.

El dispositivo es compatible con sistemas operativos Windows 10, especialmente si estos cuentan con la certificación Windows Mixed Reality, y WMR ULTRA (distintivo otorgado a los equipos de cómputo con capacidades superiores de procesamiento de entornos 3D), un equipo típico compatible cumplirá los siguientes requerimientos:

- CPU Intel® Core™ i7-7700HQ a 2,80 GHz

- Intel HD Graphics 630 / NVIDIA GeForce GTX 1050 Ti
- 15,3" / 1920x1080 / 16:9 / 144 ppi
- 16 GB de RAM

Es posible comprobar mediante la aplicación “Windows Mixed Reality PC Check” si el equipo de cómputo es compatible.

Todos los datos fueron obtenidos de la página del fabricante (Lenovo, 2018).

5.1.2 Cámaras externas

Para la creación del entorno de trabajo, así como para las pruebas posteriores se utilizaron algunas cámaras externas. Estas cámaras se enlistan a continuación junto con algunas de sus características básicas. Posteriormente se hace una mención a las cámaras infrarrojas y térmicas que pueden ser igualmente utilizadas dentro de la aplicación y por ultimo a las cámaras de dispositivos móviles más relevantes del mercado para este tipo de aplicaciones, recordando que cualquiera puede ser utilizada como cámara externa por medio de algún aplicativo ajeno de conexión.

5.1.3 Cámara Web Logitech c270 HD

FIGURA 28 Cámara web Logitech c270 HD. (Logitech, 2018)

La cámara Web marca Logitech, diseñada especialmente para videoconferencias móviles en alta definición, tiene una resolución máxima en vídeo de 720p a 30 fotogramas por segundo, el enfoque es foco fijo y es un lente estándar, el campo visual se sitúa en hasta 60 grados ofreciendo así un amplio rango de visión. La distancia focal es de 4.0mm, la resolución óptica real es de 1280x960 (1.2MP).

Cuenta con corrección de luz automática por software para entornos con mayor o menor iluminación, para el audio, el dispositivo cuenta con un micrófono monoaural incorporado a un lado del lente, con reductor de ruido incorporado.

La cámara tiene una montura tipo clip universal, se conecta al equipo de cómputo por medio de un único cable USB 2.0 en modo USB Video Device Class (UVC), y es compatible con sistemas operativos Windows 7, 8, 10 o posterior, macOS 10.10 o superior, Chrome OS e incluso sistemas móviles como Android 5.0 o mayor.

5.1.4 Cámara Intel RealSense Depth Camera D435

La cámara de profundidad Intel RealSense utiliza visión estéreo para calcular profundidad. La D435 es un equipo alimentado por USB 3.0 Tipo C y consiste en un par de sensores de profundidad, sensor RGB y proyector de infrarrojos.

FIGURA 29 Cámara Intel RealSense de profundidad D435. (Intel® RealSense™)

El equipo está preparado con un potente procesador de visión Intel® RealSense™ Visión D4 que está construido con una tecnología de 28 nanómetros y admite hasta 5 líneas de interfaz de cámara MIPI. Así mismo cuenta con un nuevo y avanzado algoritmo de profundidad estéreo integrado para una percepción precisa de la profundidad y un largo alcance, en conjunto con los sensores de imagen que permiten capturar la disparidad entre imágenes con una resolución de hasta 1280 x 720, un proyector infrarrojo activo para iluminar objetos y mejorar los datos de profundidad completan el arsenal de tecnología. Esta cámara es compatible con el SDK Intel RealSense 2.0 el cual es de código abierto y multiplataforma.

En cuanto a especificaciones técnicas, la cámara alcanza una profundidad del campo de visión de 86 x 57 x 94 (+/- 3 grados), con una resolución de salida de hasta 1280x720 con una tasa de 90 cuadros por segundo, la distancia mínima de profundidad (Z) se sitúa en 0.2m y el máximo en hasta 10 metros. En cuanto al sensor RGB alcanza una resolución de 1920 x 1080 a 30 fps.

5.1.5 Cámaras infrarrojas/Térmicas

Este tipo de cámaras se basan en el principio de que todos los objetos emiten una energía infrarroja, conocida también como señal calórica. La cámara convierte estos datos infrarrojos en una imagen electrónica que muestra la temperatura aparente de la superficie del objeto medido.

Una cámara infrarroja tiene un sistema óptico que enfoca la energía infrarroja en un detector que trabaja en conjunto con el sensor, que contiene miles de píxeles organizados en una cuadrícula. Las cámaras web comunes por lo generan son capaces de captar esta radiación en cierta medida, y se les agregan filtros especiales al lente para evitar que dicha energía contamine la imagen a color.

Cada pixel de este conjunto reacciona a la energía infrarroja concentrada en él y al estímulo, produce una señal electrónica, es cuando el procesador de la cámara obtiene la señal de

cada pixel y realiza un cálculo matemático para crear un mapa de color de la temperatura aparente de cada objeto. En función de la temperatura se le asigna un color distinto (utilizando técnicas de pseudo color muy comunes de procesamiento de imágenes, que consiste en asignar cada valor de pixel un color ya sea determinado por una lista de colores previamente designados o una escala), esta matriz resultante se envía a la memoria y a la pantalla de la cámara como una imagen de la temperatura de los objetos en una escena.

5.1.6 Cámaras de dispositivos móviles

Actualmente cualquier smartphone cuenta con al menos una cámara digital dedicada a captar las instantáneas del usuario, almacenarlas en la memoria del dispositivo y ser utilizadas de la manera que se desee. Los equipos de gama de entrada por lo regular cuentan con una cámara trasera (principal) y una frontal, las cuales tienen usos distintos, desde fotografías o videos, hasta fotos frontales o selfis y videoconferencias móviles.

Las configuraciones actuales de hardware en cámaras en las gamas de entrada se sitúan en cámaras principales fijas de 8Mpx con aperturas desde los f/2.2 y frontales de 5Mpx f/2.2, ambas con la capacidad de toque para enfocar (a través de la pantalla del equipo) y video a partir de 720p a 30fps.

Algunas de las últimas propuestas en hardware fotográfico (uno de los reclamos principales en la industria del móvil), cuyos resultados han llegado a rivalizar ya no solo con las tradicionales cámaras compactas, sino con equipos del tipo Réflex, aumentando capacidades de distancias focales variables, obturadores móviles, incluso doble, triple, y hasta cuádruple cámaras en las partes traseras y frontales de los equipos.

Estos avances los vemos en casos como el recién lanzado Samsung Galaxy A9 con 4 cámaras en la parte trasera de ese equipo:

- Cámara principal con 24 megapíxeles y una óptica con apertura f/1.7, la cual fungirá para casi cualquier situación, muy lumínica, y con estabilización óptica.
- Cámara de Zoom: de 10Mpx apertura de f/2.4 y un zoom óptico de 2X.
- Cámara de Gran Angular: con un campo angular de hasta 120 grados, 8Mpx y apertura f/2.4

- Cámara de Profundidad: es la más interesante porque es con la que no cuenta ningún otro teléfono móvil hasta ahora. Tiene un sensor de 5 Megapíxeles y una óptica con valor de apertura f/2.2. Samsung la llama 'cámara de profundidad' porque está ideada para ser utilizada en modo retrato y poner a nuestra disposición un margen de maniobra amplio cuando decidamos recurrir al desenfoque del fondo.

FIGURA 30 Cámaras del smartphone Samsung A9. (Samsung, 2018)

Samsung Galaxy S9+

El cual cuenta con una cámara principal dual trasera, la principal de 12 megapíxeles con tecnología Dual Pixel, la cual es una innovación recientemente incorporada a las cámaras de los teléfonos móviles, y que consiste en construir sensores CMOS que tienen dos fotodiodos por cada pixel en lugar de uno, con lo cual este sensor está capturando dos fotografías de forma simultánea cada vez que se toca el disparador. Además, esta cámara principal está estabilizada ópticamente, y la innovación principal en exclusiva de este equipo, la apertura variable que va desde f/1.5 a 2.4, esta apertura variable permitirá ajustar la cantidad de luz que entra en el sensor dependiendo de las condiciones externas. Por ejemplo, la apertura f/2.4 será la utilizada en las escenas diurnas y con mucha luz, mientras que cuando haya baja luz se podrá delegar en una apertura f/1.5.

La secundaria, se trata de una de 12Mpx con apertura f/2.4, el conjunto es capaz de grabar vídeo a 4K 30fps y cámara super lenta de 960fps a 720p.

FIGURA 31 Cámara del smartphone Samsung S9 plus. (Samsung, 2018)

Lenovo Phab 2 Pro

Este dispositivo de la marca China, ofreció como novedad en su lanzamiento ser el primer smartphone preparado para el proyecto tango de Google en el mundo, lo cual le hizo contener hardware específico para enriquecer la experiencia de AR.

Las características más relevantes con las que cuentas este dispositivo es que cuenta con cuatro cámaras: Una cámara de color RGB de 16 MP, una cámara infrarroja activa de profundidad, una cámara con sensor de movimiento tipo ojo de pez y en la parte frontal una cámara de color de 8MP. Además, la pantalla cuenta con la capacidad de desplegar 16 millones de colores y tiene una resolución de 1440x2560 pixeles con una densidad de 459 ppi. Su batería tiene una duración de aproximadamente 15 horas y en la parte de sonido cuenta con tecnología Dolby Atmos® y Dolby Audio™ Capture 5.1 para realizar grabaciones y reproducir audio 3D. El sensor

FIGURA 32 Cámara del smartphone de Lenovo Phab 2 Pro. (Lenovo, 2017)

de profundidad además se ayuda de infrarrojos para ayudar a trazar el entorno real, así como un sensor de movimiento que se especializa en localizar y mantener puntos fijos localizados en las paredes, esquinas y techos ayudando a mantener así funciones especiales de Tango (realidad aumentada) en funcionamiento.

5.1.7 Router

Un router es una pieza de hardware utilizado en redes, literalmente es un "enrutador" o "encaminador" que nos sirve para interconectar redes de computadoras o dispositivos móviles (en el caso de que éste tenga características inalámbricas) y que actualmente implementan puertas de acceso a internet.

Partiendo de aquí, en el entorno hogar/pequeña oficina la necesidad de un router surge cuando es necesario conectar dos o más dispositivos entre si, es decir, si se tiene un solo dispositivo, lo normal sería conectarlo directamente al modem mediante su cable Ethernet, pero si se tienen más equipos a los cuales dotar no sólo de acceso a internet, sino de posibilidades de comunicarse entre sí mismos, es donde nace la necesidad del uso de este tipo de hardware para que esta red local también pueda conectarse a la red del proveedor de acceso a internet y este conecte a internet compartiendo el ancho de banda entre los distintos dispositivos de la red. De esta manera el router se convierte en el intermediario entre la red local y privada e internet.

Específicamente para este trabajo, se utilizó un router común que es proporcionado por el proveedor de internet.

Arris TG862

Se trata de un equipo integral que incluye las puertas de enlace de telefonía residencial, combina dos puertos FXS de VoIP de clase portadora, un enrutador gigabit de 4 puertos y un punto de acceso inalámbrico 802.11n de 2.4GHz con opciones de respaldo de batería en un solo dispositivo integrado.

Este equipo residencial, al contar con la norma inalámbrica IEEE 802.11n o Wi-Fi N permite una velocidad máxima teórica de 600 Mbps. Sin embargo, cumple los estándares b, g y n los cuales permiten velocidades de acuerdo con la siguiente tabla:

Estándar	Acceso inalámbrico	Velocidad
IEEE 802.11b	802.11B, Wi-Fi B	11 Mbps
IEEE 802.11g	802.11G, Wi-Fi G	54 Mbps
IEEE 802.11n	802.11N, Wi-Fi N	600 Mbps

5.1.8 Motor de juegos Unity 3D

Unity es una plataforma desarrollada por Unity Technologies creada en 2005 como un motor de juegos exclusivo para el sistema operativo de Apple Inc. Actualmente la plataforma funciona en la mayoría de los sistemas operativos disponibles y permite la creación de contenido para hasta 27 plataformas diferentes, entre algunas de las más importantes están Android y iOS para dispositivos móviles, Linux, Windows y MacOS para dispositivos de escritorio, entre otras.

Las herramientas disponibles permiten crear contenido enfocado tanto para juegos tridimensionales como de dos dimensiones, así como simulaciones sencillas para todas las plataformas disponibles.

Unity ofrece su API principal para creación de código en C#, ya sea que dicho código sea creado desde cero en la aplicación o agregado como elemento externo en forma de complemento (plugin) compatible. Además, contiene una interfaz gráfica que por medio de mecanismos de botones o arrastrar y soltar permite crear contenido de juego sin programar. Sin embargo, las funciones más complejas deben ser programadas por el usuario.

Se eligió este software en particular de entre todos los motores de videojuegos por su compatibilidad con distintos elementos de Microsoft, como son los lentes de realidad mixta que se utilizan en el trabajo presente, la integración con Visual Studio y su capacidad de generar aplicaciones para Windows Universal Platform.

5.1.8.1 Assets

Dentro de Unity un asset es cualquier objeto que pueda ser usado dentro de un proyecto. Puede provenir de un archivo externo creado fuera de la plataforma, como son sonidos, modelos tridimensionales, imágenes, videos y cualquier otro compatible con la plataforma. Algunos de estos también pueden ser creados directamente de la plataforma como formas básicas en tres dimensiones, materiales y código entre otros.

5.1.8.2 Prefabs

Es común tener objetos del juego (assets) en la escena virtual de los cuales es deseable conservar sus propiedades para poderlos utilizarlos más adelante en el mismo o diferentes proyectos. Unity permite crear este tipo especializado de asset que permite almacenar un objeto junto con todas las modificaciones realizadas por el usuario. Los prefabs son similares a una plantilla a partir de la cual se crean copias o nuevas instancias del objeto mencionado. Cualquier modificación hecha sobre un prefab se verá reflejado en todas las instancias de este.

5.1.8.3 Escenas

Las escenas son aquellas que contienen un entorno completo, sus objetos y las interacciones entre ellos. Por ejemplo, en un juego una escena puede contener una serie de objetos tridimensionales que en conjunto formen un ambiente virtual, además contendrá las luces, cámaras, personajes, comportamientos programados y cualquier otro objeto necesario para el juego. Un mismo proyecto puede contener una variedad de escenas diferentes con propósitos específicos, como diferentes niveles de un juego o menús.

5.1.8.4 Paquetes

Los paquetes contienen elementos que pueden ser utilizados en distintos proyectos. Podría decirse que son archivos comprimidos que abarcan cualquier tipo de asset aceptado por Unity. Un paquete puede contener desde herramientas individuales, hasta videojuegos completos. El propósito de los paquetes es poder compartir y migrar con mayor facilidad elementos que podrían reutilizarse.

Una de las partes más importantes de este motor de videojuegos es que funciona a través de objetos individuales. Es decir, cada objeto pertenece a su propia clase y está aislado de los demás siempre y cuando no se especifique lo contrario. Esto quiere decir que un asset puede ser reutilizado indefinidamente. Por ejemplo, si se crea un material para un objeto de color verde, este material puede ser comprimido en un paquete e incluido posteriormente en otro proyecto, permitiendo utilizarlo en otros objetos. Esta idea permanece para todos los assets, incluyendo código.

5.1.8.5 Scripts

La programación es parte esencial de esta plataforma, a través de pedazos de código aislados es que se definen los distintos comportamientos de objetos en el mundo virtual. Incluso el más simple de los proyectos requiere de scripts para responder a las posibles acciones del usuario o modificar el comportamiento de los objetos.

Cada script está aislado en su propia clase, lo que significa que son independientes uno de otro. La diferencia entre estos y la programación convencional es que dependen del objeto al que son asignados. Por ejemplo, digamos que se tiene un script cuya función es rotar un objeto sobre un eje en específico, dicho script es asignado a un cubo y una esfera, entonces ambos objetos giraran, utilizando un único pedazo de código, en otras palabras, no es necesario programar el comportamiento del cubo y esfera por separado.

Así mismo los scripts permiten crear pequeñas interfaces de usuario visualizables dentro de Unity que permiten modificar los valores de las distintas variables que conforman el código

sin tener que modificarlo directamente. Siguiendo el ejemplo anterior, con el mismo script se podría hacer que el cubo gire más rápido que la esfera utilizando las opciones graficas disponibles para modificar la velocidad.

5.1.9 Visual Studio de Microsoft

Este es un ambiente integrado de desarrollo o IDE (Integrated development environment) perteneciente a la compañía Microsoft. Se utiliza para desarrollar diferentes programas de computadora, al igual que páginas web y aplicaciones para dispositivos móviles. Incluye un editor de código, así como un depurador, entre otras herramientas como es el diseñador web o diseñador de bases de datos. Soporta alrededor de 36 lenguajes de programación y permite integrar herramientas para prácticamente cualquier lenguaje. Unos de los lenguajes incluidos por defecto son C#, C++, C y JavaScript. Todos ellos forman parte del motor de juegos Unity también.

Esta herramienta es esencial para desarrollar un plugin para Unity 3D que permita el acceso al casco de realidad mixta de Microsoft, así como facilitar el acceso a otros tipos de hardware externo como son las cámaras que se discuten en el apartado anterior y combinarlas.

Por su parte Unity 3D permite crear proyectos de Visual Studio desde su plataforma. El motor de juego mantiene actualizada constantemente la solución y permite acceder a funciones más complejas, así como modificar algunas otras que sería imposible desde la interfaz gráfica de Unity.

5.1.10 OpenCV

OpenCV (Open Source Computer Vision) es una librería que contiene diferentes funciones, principalmente orientadas al procesamiento de imágenes y a la visión por computadora a través de C++. Las librerías son multiplataforma, actualmente existen versiones para Windows, MacOS y Linux, además de Android y iOS para dispositivos móviles. Así mismo soporta algunos entornos de trabajo enfocados en aprendizaje profundo. La primera versión de OpenCV fue publicada en el año 2000. OpenCV2 que incluye cambios mayores a la interface de C++, nuevas funciones y cambios importantes en el rendimiento de las funciones existentes, fue lanzada el año 2009. Las librerías están escritas en C++, lo que permite su integración a distintas plataformas que soporten plugins en formato .dll. Por otra parte, existen versiones para otros lenguajes como son Python y Java.

Esta herramienta puede ser de gran ayuda en conjunto con Unity, sus propiedades multiplataforma siguen la filosofía de Unity3D de llegar a distintos dispositivos y sistemas operativos. La integración de la misma dentro del entorno de trabajo de Unity con C# puede brindar a la aplicación desarrollada todas las opciones que OpenCV ofrece.

5.2 INTEGRACIÓN DEL HARDWARE/SOFTWARE

Como se puede observar en la figura 17 la integración del hardware es bastante simple. Los cascos de realidad mixta traen consigo dos conexiones, una USB tipo C para datos y otra HDMI para la transmisión de información visual entre el ordenador y el dispositivo.

5.2.1 Audio

Al casco pueden ir integrados un par de audífonos con una conexión del tipo Jack estéreo de 1/8. Este tipo de conexiones de audio es quizá la más común y popular en el mercado de todas, se puede encontrar en prácticamente todos los audífonos comerciales o de dispositivos móviles. Es capaz de transportar dos canales de audio independientes (por ejemplo, uno para el oído derecho y otro para el izquierdo). La clavija contiene dos aros distintos, la parte superior arriba del primer aro se conoce como "tip" (o punta en inglés), la siguiente sección que se encuentra entre los dos aros se conoce como "ring" (anillo en inglés) y la restante se conoce como "sleeve" (mango en inglés). La punta y el anillo son las que llevan las señales de audio derecho e izquierdo, la parte restante funciona únicamente como punto de tierra. Los dos aros que se pueden observar son materiales aislantes para separar las dos señales. En realidad, no es necesario ningún otro tipo de conexión para audio, algunos expertos podrán discutir sobre la calidad de audio, pero la realidad es que un par de audífonos comerciales comunes con este tipo de conexión son suficientes para reproducir audio estéreo de buena calidad. El puerto incluido en los cascos es de doble sentido, lo que significa que si se conectan unos audífonos con micrófono integrado o un micrófono cuya salida este configurada para este tipo de conexión se puede hacer funcional como entrada de audio.

FIGURA 33 Conexión de audio tipo Jack Estéreo de 1/8. IMAGEN: (Logitech H110, 2018)

Esto está pensado de tal forma que se puedan utilizar comando de voz con las aplicaciones de realidad mixta, sin embargo, hacerlo por medio de un micrófono de baja calidad puede afectar el rendimiento de la aplicación en esta área. En general una configuración más adecuada seria conectar el micrófono directamente a la computadora para lograr un mejor rendimiento y disminuir la latencia que pueda provocar conectar el dispositivo al casco y el casco al ordenador, sin embargo, este tipo de configuración entorpecería el movimiento del usuario. Para esto se pueden utilizar preferentemente audífonos y micrófonos conectados a través de Bluetooth ya sea directamente al ordenador o al casco que cuenta con esta opción también.

5.2.2 Cámaras

Las cámaras externas por su parte si deben ser conectadas al ordenador directamente. La realidad es que se pueden conectar tantas cámaras como sea necesario, el framework está pensado para utilizar más de una, y tantas como se desee. Por supuesto esto puede significar un conflicto posterior, ya que la tendencia es eliminar tantos puertos como sea posible del hardware. Aunque esta tendencia aun no parece alcanzar a los ordenadores de escritorio, lo podemos observar tanto en dispositivos telefónicos como computadoras portátiles. Cada cámara por lo general requerirá su propio puerto USB, y aun que se pueden colocar extensiones, esto puede alentar el tiempo de respuesta.

Una solución preliminar a este problema es utilizar conexiones sin cable para las cámaras externas. Existen varias aplicaciones que permiten conectar la cámara del dispositivo móvil al ordenador sin utilizar un cable atreves de la señal de Wi-fi o bluetooth. Como de costumbre el problema de este tipo de conexiones a pesar de ser la más elegante y menos estorbosa, también suele ser más lenta que la conexión convencional por cable.

Además de los dispositivos que, en una configuración básica por cable, pensando que ya se tienen conectado el casco de realidad mixta y una cámara externa, ya se utilizan dos puertos USB y uno HDMI, es posible integrar otros dispositivos de entrada como lo son controles para moverse dentro de la escena virtual. De manera ideal se esperaría que la persona pudiera moverse libremente por la escena sin ningún problema e interactuar con ella con naturalidad como lo hace en el mundo físico. Sin embargo, esto aún es posible. Por lo general en este tipo de configuraciones se incluyen otro tipo de sensores para mejorar la parte de interacción, algunos de los que vienen integrados en el casco son giroscopio y acelerómetro para la posición de la cabeza.

5.2.3 Controls Inside-out / Outside-in tracking

Los controladores de movimiento integrados con el casco se conectan por medio de bluetooth al dispositivo. Utilizan un tipo de tecnología conocido como Inside-out tracking que es un método de seguimiento posicional muy utilizado en realidad virtual, específicamente para seguir el movimiento de la cabeza.

5.2.4 Outside-in tracking

Para entenderlo mejor quizá sea necesario explicar su contraparte como lo es el Outside-in tracking que utiliza cámaras u otro sensor colocado en una posición fija y orientado hacia el objeto del que se desea hacer seguimiento que puede moverse dentro de un área designada dependiendo del rango de alcance del sensor que lo detecta, el objeto de interés es seguido u observado desde fuera.

Un ejemplo de esta configuración es poner una cámara dentro del casco o fijada en alguna parte de la habitación, y poner algún tipo de marcador fácil de reconocer en el control. De esta forma por medio de software se puede calcular el lugar en que está posicionado el control. Por supuesto esto se puede lograr sin marcadores como es el caso del Kinect perteneciente a Microsoft.

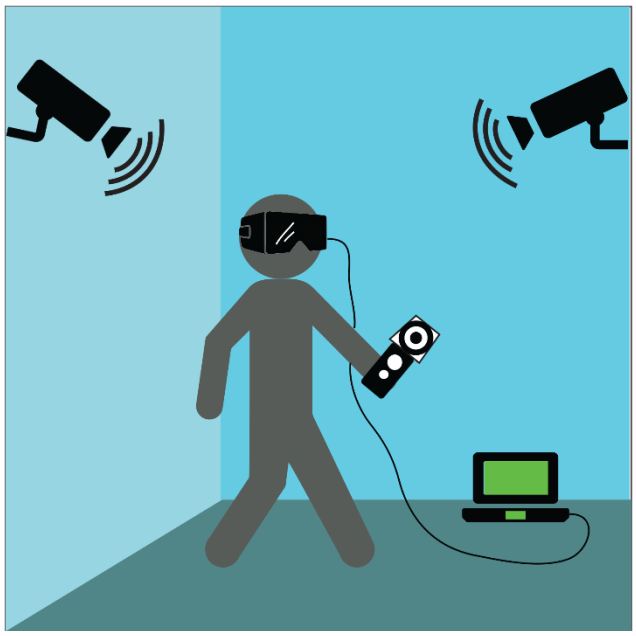

FIGURA 34 Outside-in tracking.

5.2.5 Ventajas y desventajas del outside-in tracking

Las soluciones de este tipo permanecen fijas y actualmente son las más precisas además de que permiten agregar más de un marcador ya se apara incrementar la exactitud o para seguir varios objetos a la vez utilizando un único sensor.

Otra de las ventajas de este tipo de arreglo es que tienen menor latencia para responder, disminuyendo la posibilidad de experimentar algunos de los efectos negativos de la inmersión que se mencionan en la introducción como es el cybersickness. Este es en la actualidad la forma más poderosa de seguimiento, sin embargo, las configuraciones de hardware necesarias suelen ser poco accesibles económicamente.

Una de las principales desventajas de estos sistemas es la oclusión, que se puede definir como cosas que se interponen en el camino. El objeto que se desea censar debe estar a la vista del sensor todo el tiempo para garantizar su correcto funcionamiento. El ejemplo más claro es el de la utilización de una cámara en conjunto con un marcador visual, en el momento que el marcador es tapado, aunque sea ligeramente o solo una parte, el sistema será incapaz de reconocer el objeto. Interferencia con el censado puede significar problemas para calibrar la posición una vez que la información se tiene nuevamente

disponible. En teoría se espera que la aplicación pueda seguirte de manera correcta en 360 grados, pero la realidad es que la señal se perderá en el momento que te des la espalda.

El otro problema, más específico de los sensores visuales como son las cámaras, es que el censado depende de las limitaciones de las cámaras. Por ejemplo, si el dispositivo tiene un campo limitado de visión, el rango de movimiento será más limitado o si por otra parte tiene dificultades para ver en la obscuridad, el sistema requerirá una buena iluminación sobre el marcador todo el tiempo.

5.2.6 Inside-out tracking

El método de Inside-out tracking incluye los sensores dentro de los objetos que deben ser seguidos. Estos sensores se encargan de determinar que tanto ha cambiado la posición en relación con el ambiente en el que se desenvuelve. Esto permite que los movimientos de dichos objetos sean fácilmente trasladados al mundo virtual. Por lo general comienzan en una posición inicial fija que se denomina el origen, a partir de ahí una vez que la simulación comienza simplemente estos dispositivos calculan con sus sensores integrados los cambios de traslación y rotación.

El ejemplo más clásico de esta configuración es como sigue: se coloca una cámara dentro del dispositivo u objeto que deseamos seguir en el espacio, esta cámara mira lo que hay fuera, calcula y almacena de alguna forma el ambiente que logra captar y determina su propia posición según los cambios que detecta en relación con el exterior. Una vez calculados estos cambios de posición los refleja directamente en el ambiente virtual, inmediatamente después se reposiciona en cero por decirlo de alguna manera, comenzando el ciclo de nuevo.

5.2.7 Ventajas y desventajas del inside-out tracking

Por una parte, importante, este tipo de seguimiento puede hacerse con o sin marcadores. A diferencia del outside-in tracking que aun que puede lograrse sin marcadores esto por lo general entorpece el proceso. En el caso del inside-out tracking el uso de marcadores no impacta de forma significativa el rendimiento del sistema.

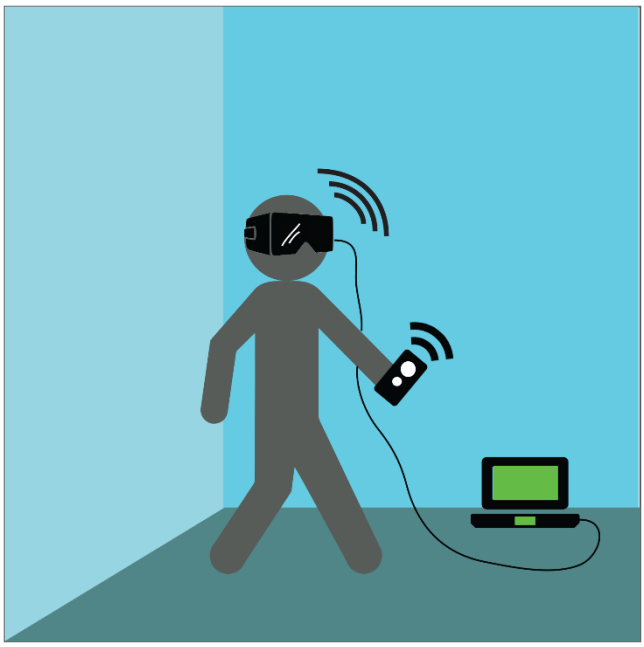

FIGURA 35 Modelo inside-out tracking.

Una de las principales ventajas que ofrece al eliminar los marcadores, es que el espacio de movimiento deja de estar restringido, así como el tipo de movimientos, pues se elimina al cien por ciento el problema de la oclusión. Para la realidad aumentada y mixta este tipo de tecnología es el más utilizado, por ser comúnmente más barato, portable y de bajo consumo energético.

Sin embargo, la exactitud disminuye mientras la latencia aumenta, pues requiere mayor poder computacional para lograrse correctamente y comúnmente todos los cálculos deben hacerse dentro del casco mismo como es el caso del casco de realidad mixta utilizado en este trabajo. El casco recibe y procesa la información el mismo, y la manda posteriormente al ordenador para hacer los cambios.

5.2.8 Software

El ordenador, recibe como entrada toda la información anteriormente descrita. Esta información es procesada dentro del motor de juegos Unity 3D, mezclando la información visual de las cámaras con la información virtual y produciendo como resultado una única

escena la cual por medio de una cámara virtual que funge como una cámara estéreo calibrada con el casco de realidad mixta, da como resultado una imagen estereoscópica de la escena final integrada.

En esta parte es donde se pueden integrar otras herramientas destinadas a los otros sentidos a parte de la vista, como son el sonido estéreo, estímulos hápticos, etc. Los cuáles serán enviados ya sea a través del casco u otros medios al mismo tiempo que los estímulos visuales, a través de sus dispositivos de salida específicos como pueden ser audífonos en el caso de audio o vibraciones en los controles en el caso de estímulos hápticos.

6 IMPLEMENTACIÓN DEL SISTEMA DE REALIDAD MIXTA

El desarrollo de este proyecto consta de tres etapas principales: la conexión adecuada de los componentes físicos al ordenador y la configuración para su funcionamiento. En segundo término, la creación del código de conexión entre los dispositivos y otras herramientas para su utilización dentro del motor de juegos Unity 3D que en conjunto forman el entorno de trabajo para realidad mixta. Por último, el proceso de compactar lo desarrollado en las etapas anteriores a modo de plugin para Unity, para que pueda ser migrado y utilizado en otros proyectos.

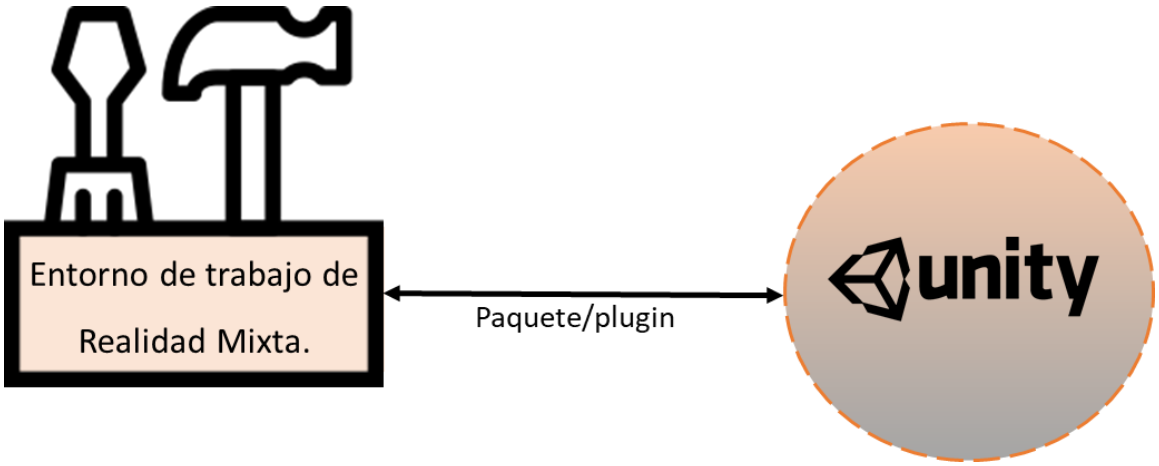

FIGURA 36 Resultado final esperado del proyecto.

En el capítulo anterior se describió el equipo necesario junto con sus características principales individuales de cada uno de sus elementos para la construcción del proyecto. Ahora se presenta las modificaciones e interacción de estos para crear el entorno de trabajo de realidad mixta y el papel que cumple cada uno de ellos en el funcionamiento general del sistema.

6.1 CONEXIÓN DE DISPOSITIVOS

Hoy en día se pueden encontrar distintos dispositivos de realidad aumentada y realidad virtual en el mercado, desde aquellos enfocados al entretenimiento hasta algunos desarrollados con fines específicos como puede ser médicos o educativos. Sin embargo, los dispositivos para realidad mixta apenas están surgiendo como alternativas a las tecnologías clásicas, existiendo al alcance del público general a partir del 2016 aproximadamente.

Con el fin de cumplir el objetivo de crear un entorno de realidad mixta, se propone utilizar varios dispositivos en conjunto. En particular la integración de un casco de realidad mixta que cuenta con dos cámaras estéreo frontales y sensores de movimiento, así como con

pantallas LCD para hacer el despliegue de imágenes estéreo. En conjunto se añaden cámaras externas y sus respectivos controles como información adicional a la captada por las cámaras.

Los dispositivos (casco y cámaras) se eligieron por su accesibilidad. Especialmente las cámaras que fueron utilizadas únicamente para hacer pruebas, pues el sistema permite la conexión de cualquier cámara con el ambiente virtual.

Todos estos elementos son autónomos, sin embargo, es posible coordinar sus funciones a través del motor de juegos Unity 3D, que es al final de cuentas el objetivo del trabajo presente.

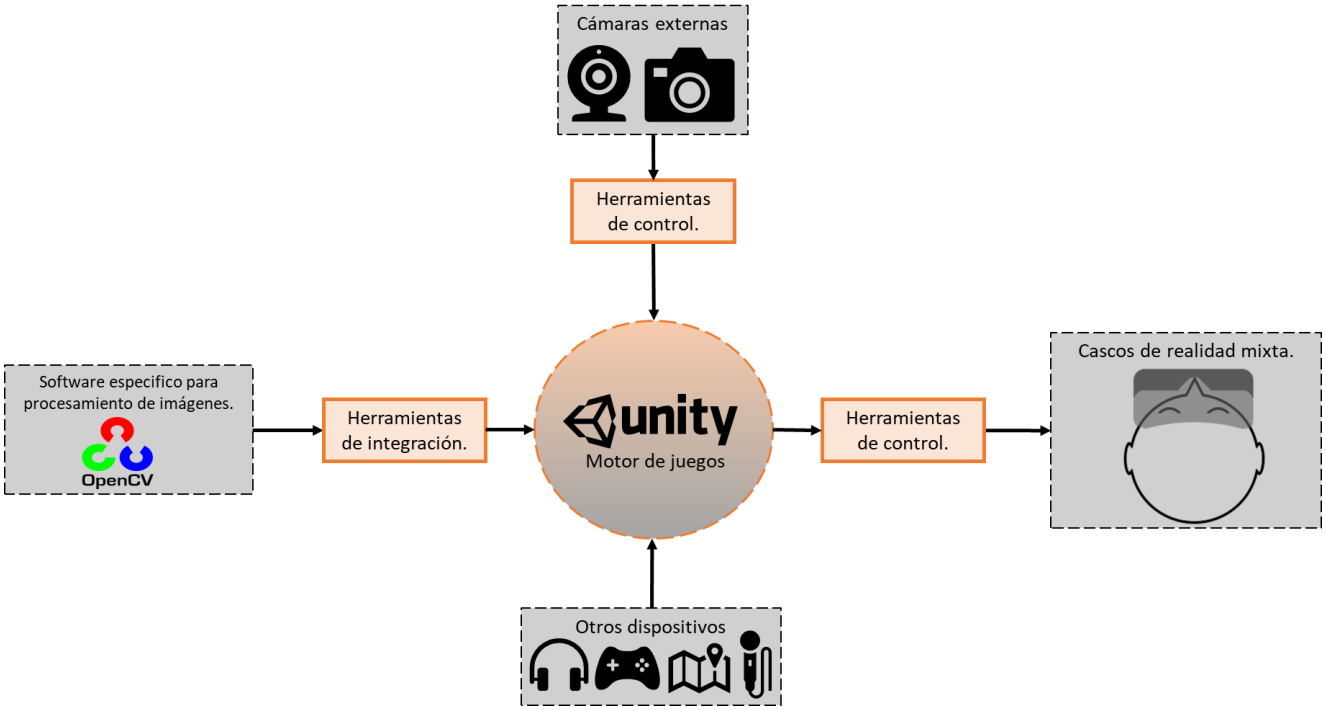

FIGURA 37 Diagrama general de la integración del sistema.

6.1.1 Conexión de las cámaras externas.

En primer lugar, las cámaras externas sean del tipo que sean deben ser reconocidas por el ordenador como dispositivos que transmiten imágenes en tiempo real. Cualquier dispositivo para adquisición que no cumpla con esta característica no podrá ser utilizado posteriormente en conjunto con las herramientas desarrolladas.

Por lo general cualquier cámara web es reconocida de manera automática. Existen sin embargo algunas otras como es el caso de la cámara Intel RealSense descrita en el capítulo anterior que necesita de un software proporcionado por el mismo fabricante para ser reconocida adecuadamente y por lo tanto para ser utilizada.

Este tipo de situaciones es importante tenerla en cuenta si se quiere añadir un nuevo dispositivo de video.

Otro ejemplo claro es la cámara del teléfono. Esta puede ser utilizada como cámara externa, sin embargo, para esto es necesario por lo general instalar una aplicación ajena tanto en el dispositivo móvil como en la computadora que por medio de un pin y la conexión de internet añadirá una instancia de dispositivo de adquisición de video en el ordenador, simulando una conexión física. La única desventaja de este tipo de configuración es que la calidad del video adquirido dependerá de la velocidad de conexión con la que se cuente y la capacidad de transferencia de la aplicación utilizada.

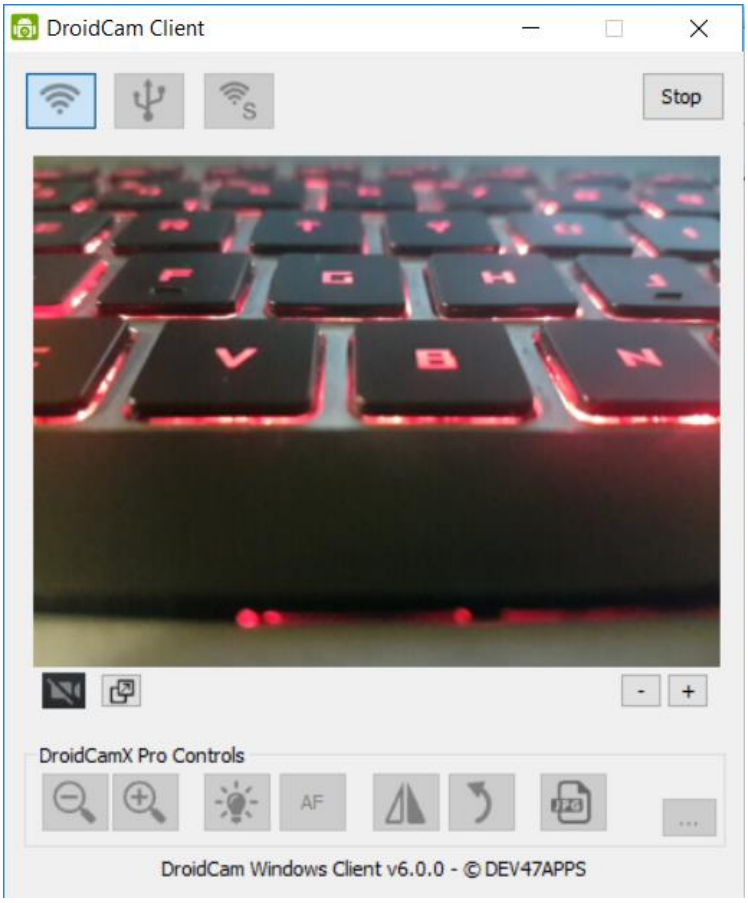

FIGURA 38 Aplicación DroidCam que permite conectar la cámara del dispositivo móvil al ordenador.

Una de las aplicaciones que demostró mejor rendimiento para utilizar la cámara del celular fue DroidCam creada por Dev47Apps, requiere llevar a cabo la instalación de un programa en el ordenador junto con la instalación de una aplicación en el teléfono. Tiene una interfaz fácil de usar y lo único que se necesita es escribir en ambas aplicaciones la IP y puerto del router que se desea utilizar, así como que ambos dispositivos estén conectados al mismo. Esta aplicación ofrece varias opciones como conexión a través de Wi-Fi o puerto USB. El resultado final es el mostrado en la Figura 37. Además de transferir datos de imagen,

también transfiere audio si así se requiere. La única desventaja además de las ya mencionadas anteriormente para este tipo de métodos es que utiliza una configuración de cámara fija. Es decir, utiliza la cámara en una resolución estándar con una frecuencia de actualización específica, sin permitir al usuario acceder a las imágenes de la cámara con su total capacidad de resolución y tamaño.

Existen otros métodos para conectar las cámaras del celular en la computadora. El que demuestra tener mejor funcionalidad es aquel que ocupa el cable para transferir los datos. Sin embargo, de igual manera se necesita una aplicación externa que permita este tipo de configuraciones.

Una última posible conexión es a través de un servidor. La cámara se conecta con cualquiera de los métodos anteriores mencionados a un servidor. La computadora entonces pide acceso al servidor para descargar las imágenes en él. Es posible que con este tipo de configuración exista algún tipo de latencia, pero es una excelente herramienta para acceder a un dispositivo a distancia. Incrementando incluso las posibilidades de las aplicaciones del software propuesto.

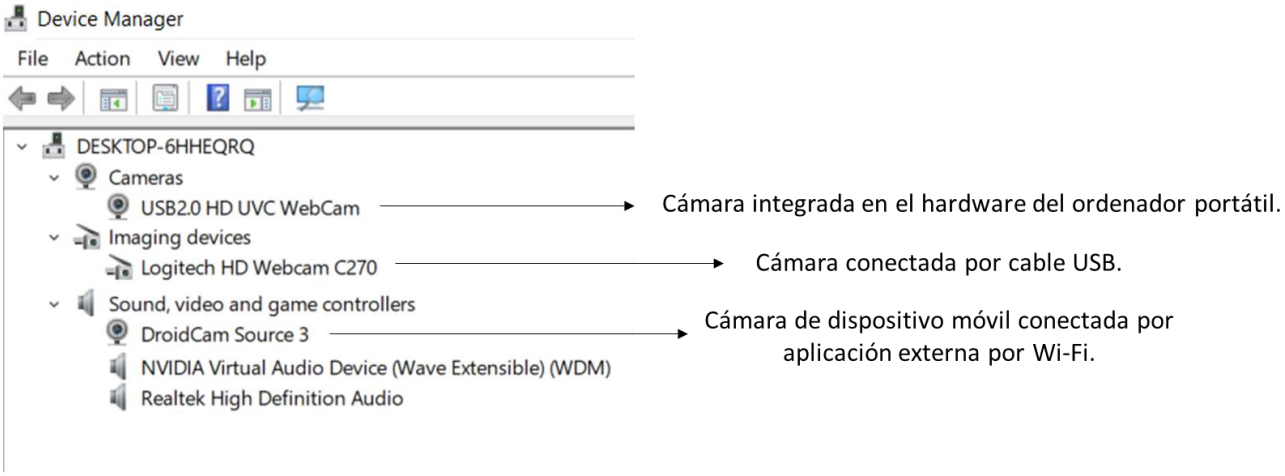

FIGURA 39 Cámaras conectadas a la computadora utilizando varios métodos.

En la figura anterior se muestran tres ejemplos de cámaras conectadas de distintas formas. La primera es la cámara integrada en el hardware del ordenador portátil, la mayoría de esta índole de ordenadores cuenta con una cámara de este tipo. En segundo lugar, se puede ver la cámara web de Logitech conectada por cable USB y por último la cámara del dispositivo móvil utilizando la aplicación descrita con anterioridad.

Todas y cada una son detectadas por el ordenador como dispositivos de entrada de imagen, en una categoría u otra. Cuestión que como se mencionaba anteriormente es de vital importancia para el correcto funcionamiento de las cámaras externas en el proyecto.

Una vez que los dispositivos deseados se encuentran conectados y funcionando, es posible comenzar con su utilización desde Unity.

Unity funciona a través de scripts independientes. Es decir, cada pedazo de código se considera una clase independiente. Dichos scripts son asignados a un objeto en particular que a su vez forma parte de su propia clase. Cuando un pedazo de código es asignado a un objeto, el objeto "absorbe" la nueva clase como propia y esta hace efecto únicamente sobre el objeto asignado. De esta forma un mismo script puede ser reutilizado para una infinidad de objetos diferentes sin modificar las propiedades de otros.

Los scripts pueden tener variables públicas que se consideran como parámetros modificables por el usuario. De esta manera cuando se declara una variable de esta forma, en el panel de atributos del objeto al que se le asigne dicho script aparecerá la opción referente para modificar el parámetro o parámetros correspondientes a las variables públicas.

A continuación, se muestra uno de los scripts (funciones) que forman parte del entorno de trabajo. Quizá el más sencillo de todos, se encarga de listar los dispositivos de video conectados y compatibles.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class webcam : MonoBehaviour {
6
7      public WebCamDevice[] GetDevices ()
8      {
9          return WebCamTexture.devices;
10     }
11
12     // Use this for initialization
13     void Start () {
14
15         //Print available webcam devices on console
16         WebCamDevice[] devices = WebCamTexture.devices;
17         for (int i = 0; i < devices.Length; i++)
18         {
19             Debug.Log(devices[i].name);
20         }
21     }
22 }
```

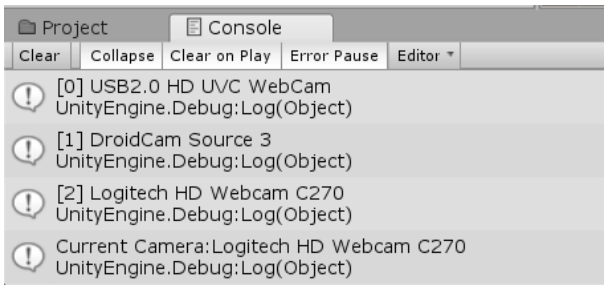

FIGURA 40 Código y ejemplo/resultado del Script PrintDevices.cs

El código se encarga de encontrar e imprimir en la consola de Unity los dispositivos encontrados que cumplen las características necesarias para ser utilizados como cámaras de video. En este caso el script simplemente fue importado en un proyecto nuevo, y arrastrado a un objeto en la escena (en este caso a la cámara principal). De forma automática cumplió su función sin mayor percance.

6.1.2 Cámaras como textura.

La parte más importante de este trabajo es la combinación de las cámaras dentro de Unity, una vez que se ha logrado hacer la conexión de estas, se necesitan una serie de herramientas para controlarlas y desplegar su contenido de alguna forma. En general existen dos formas de mostrar las imágenes dentro del mundo virtual.

- Como capa superior al mundo virtual.
- En combinación con el mundo virtual.

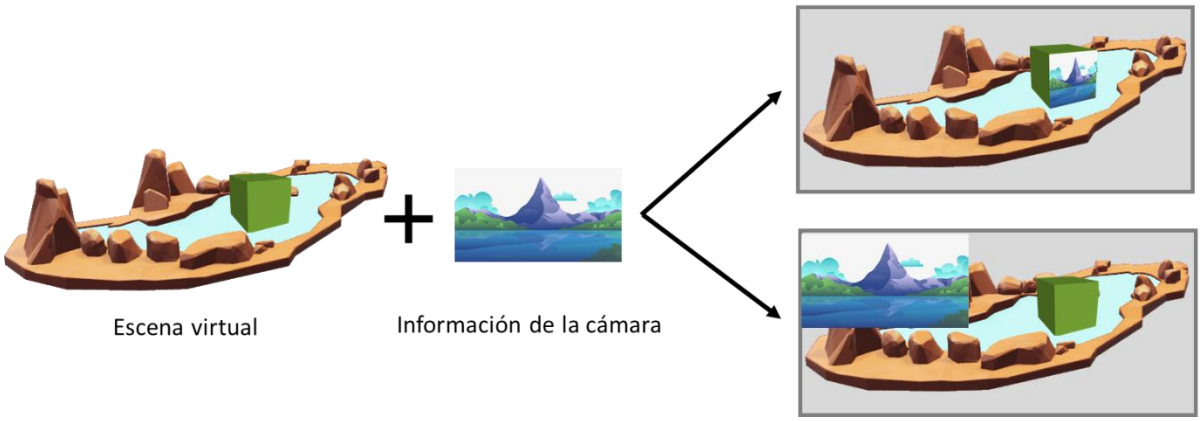

FIGURA 41 Formas de desplegar el contenido de las cámaras en el mundo virtual.

Para utilizar la información de las cámaras como capa superior es necesario añadir el contenido a la parte de UI (Interfaz de Usuario por sus siglas en ingles) de Unity. Esta capa es independiente del mundo virtual y tiene como objetivo mostrar información al usuario superponiéndola en la escena final. Ningún objeto virtual puede interactuar u modificar dicha capa. Para este propósito se crearon dos scripts encargados de tomar la información

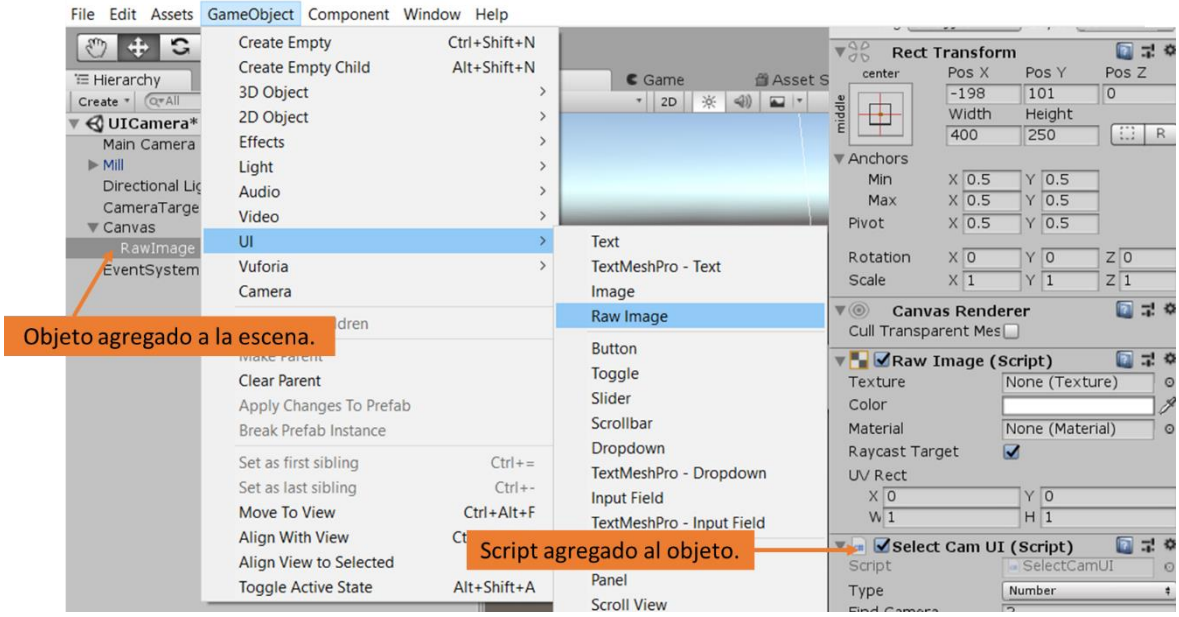

FIGURA 42 Script SelectCamUI.cs agregado a un objeto tipo RawImage.

de video proveniente de la cámara y mostrarla en esta capa. Los scripts es necesario colocarlos sobre un objeto de Unity llamado RawImage como se ilustra en la figura anterior. A continuacion se describen ambos scripts y sus funciones, asi como un breve ejemplo de su funcionamiento.

SelectCamUI.cs

Permite al usuario seleccionar una de los dispositivos de video disponibles, ya sea por el nombre con que el ordenador identifica el dispositivo o por el numero del lugar en que fue reconocido y despliega el contenido en el objeto especificado.

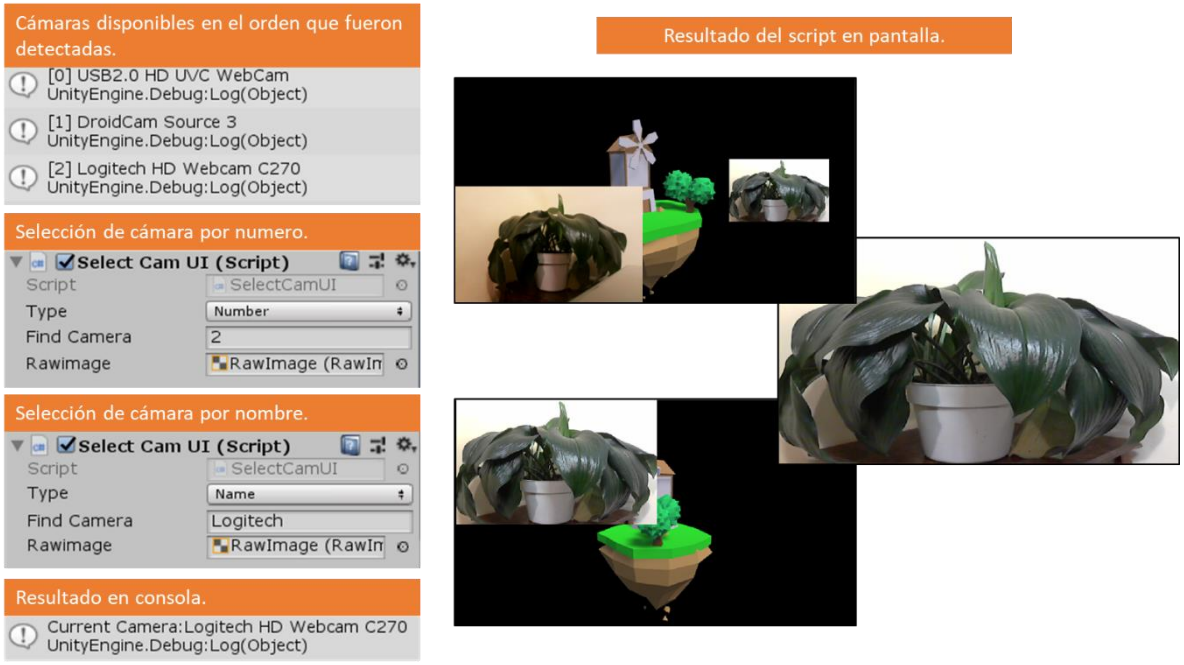

FIGURA 43 Script SelectCamUI.cs en funcionamiento.

En la figura anterior se puede apreciar un ejemplo de la funcion descrita anteriormente. El resultado (informacion de la camara) puede ser desplegado en pantalla como mejor convenga, siendo posicionado, escalado o rotado. Es posible mostrar el contenido de una o mas camaras al mismo tiempo, e incluso se puede desplegar el contenido en pantalla completa.

CamUI.cs

La funcion es similar al script anterior, la unica diferencia es que despliega la informacion de la primer camara que encuentra disponible.

El codigo de todos los scripts mencionados, asi como otros incluidos en el entorno de trabajo, se pueden encontrar en el anexo al final de este documento.

La informacion de las camaras externas tambien puede ser requerido como parte del mundo virtual. Para esto se desarrollaron tres scripts diferentes, que en combinacion permiten utilizar la informacion de video como textura sobre cualquier objeto virtual.

La peculiaridad de estos scripts es que permiten la interaccion del mundo virtual con la informacion obtenida de las camaras, de tal forma que una accion en la escena virtual (por ejemplo tomar un objeto de un color específico) puede afectar, si asi se desea, la visualizacion del mundo real.

SelectCamTexture.cs

Permite al usuario seleccionar una de los dispositivos de video disponibles, ya sea por el nombre con que la computadora identifica el dispositivo o por el numero del lugar en que fue reconocido y despliega el contenido como textura en el objeto virtual especificado. Requiere de un material sobre el cual desplegar la textura, el cual a su vez debe ser incluido en las propiedades de un objeto en la escena.

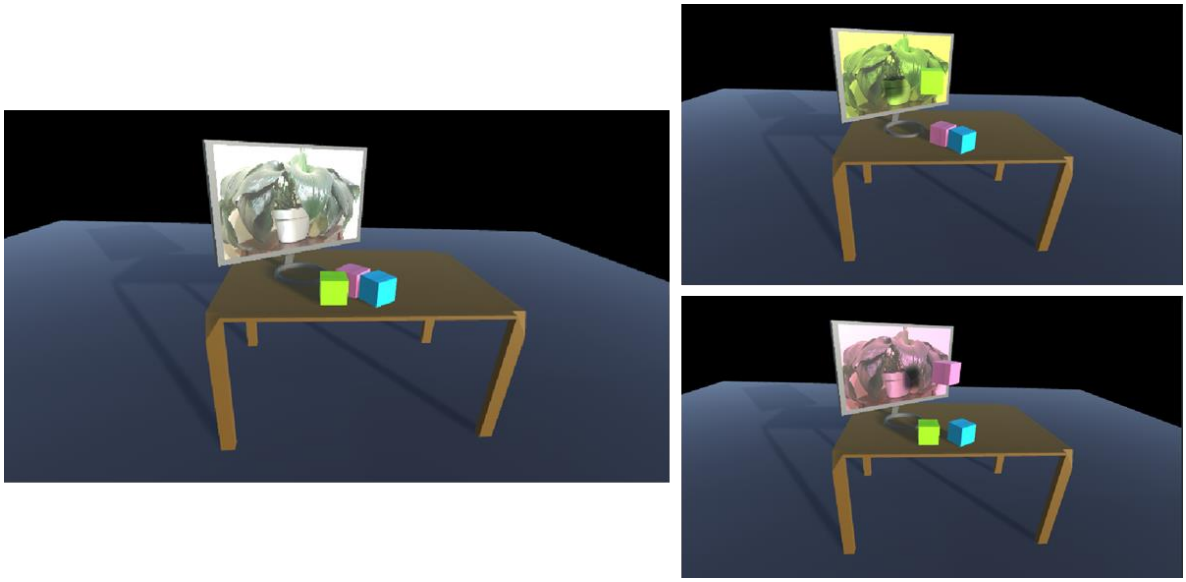

FIGURA 44 Información de cámara en combinación con el mundo virtual y ejemplo de interacción, script SelectCamTexture.cs

FirstCamTexture.cs

La funcion es similar al script anterior, la unica diferencia es que despliega la informacion de la primer camara que encuentra disponible.

CopyCam.cs

La informacion de la camara siempre se mostrara unicamente en el ultimo objeto a la que fue asignada dicha textura con alguno de los scripts anteriores. Sin embargo puede haber

casos en que se desea colocar la informacion en dos o mas objetos, el ojetivo de este pedazo de codigo es precisamente ese. Solo se requiere introducir en la interfaz del script el material al que fue asignada la camara que se desea copiar.

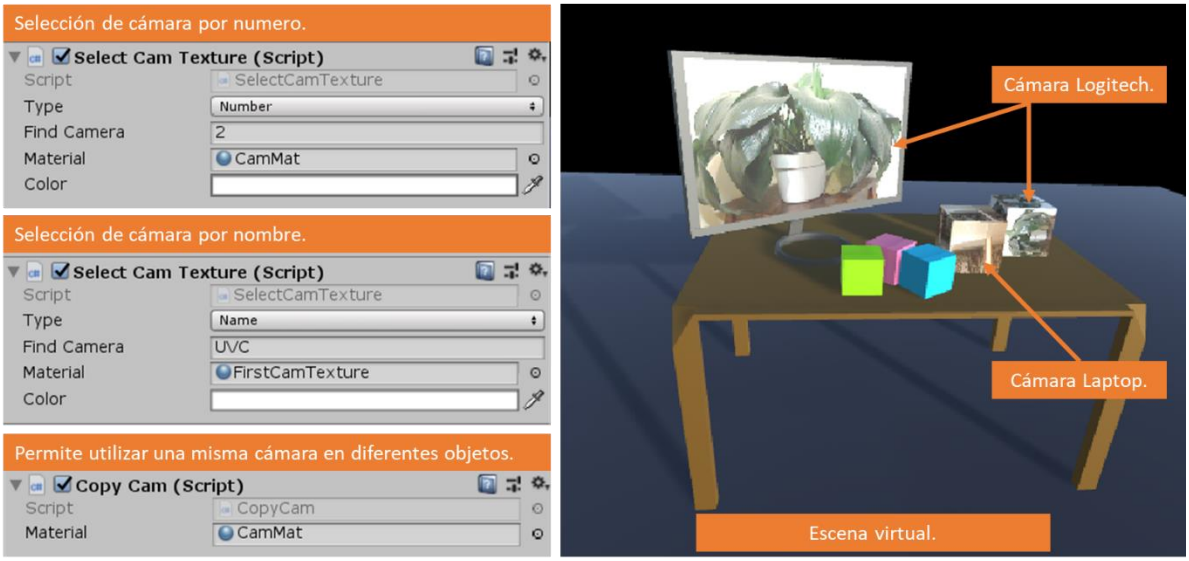

FIGURA 45 Información de una cámara en dos objetos distintos y dos cámaras en funcionamiento en una misma escena.

Tambien se desarrollaron otros scripts para este trabajo. Cada uno tiene una funcion demostrativa en mayor o menor medida. Aun que forman parte del entorno de trabajo, no se consideran escenciales para este. Algunas de las funciones mencionadas incluyen el poder apagar y prender una camara con una tecla que el desarrollador puede seleccionar de una lista desplegable y funciones de navegacion relacionadas con la camara virtual como navegar en el mundo virtual utilizanod el raton o algun otro tipo de control o asignar comportamientos a objetos, por ejemplo, que un objeto se encuentre centrado frente a la camara constantemente.

En el paquete final se incluyen dos escenas, junto con sus elementos como son modelos 3D, scripts asignados a objetos, materiales y texturas que demuestran el funcionamiento del entorno de trabajo. Son escenas pequeñasa forma de ejemplo que contienen distintos elementos necesarios para crear una aplicación de realidad mixta. Ambas escenas son mostradas en las figuras 43 y 44 del trabajo presente.

6.1.3 Conexión del casco de realidad mixta.

Para poder trabajar con los lentes de realidad mixta es necesario hacer algunas modificaciones en Unity en cuanto a su configuración. En las opciones para compilar el proyecto (File>Build Settings) debe seleccionarse la opción de “Universal Windows Platform”, dispositivos objetivo: cualquiera, tipo de construcción: D3D haciendo referencia a DirectX 3D. También es necesario configurar dentro de File>Build Settings>Player Settings

Scripting Backend a .NET para asegurar que la aplicación funcione en los lentes. Por último, es necesario asegurarse de seleccionar la opción “soportar realidad virtual”.

Existen dos formas de generar y probar la aplicación, una de ellas requiere compilar el proyecto para abrirlo posteriormente en Visual Studio e iniciar la visualización desde ahí (lo único que hay que hacer para esto es seleccionar la opción “Unity C# projects” en las opciones de Build, la ventaja de usar este método es que modificar los scripts o cualquier otro código es más fácil y la compilación del proyecto es más rápida, la desventaja es que desde Visual Studio no se pueden modificar otros aspectos gráficos como son las posiciones de los objetos tridimensionales.

La segunda forma consiste en utilizar la opción de compilar y ejecutar desde Unity, igualmente se generará un proyecto de Visual Studio la primera vez que se seleccione, pero se permanecerá dentro del editor de Unity y la aplicación se iniciara al hacer clic en el botón de play incluido en el motor de juegos.

Ambas opciones iniciaran la plataforma de realidad mixta de Microsoft y comenzaran con la ejecución de la aplicación.

Al crear un nuevo proyecto en Unity este se inicializa con una cámara virtual y una luz direccional en la escena. La cámara principal de Unity soporta el seguimiento de la cabeza y visualización estereoscópica, sin embargo, su configuración inicial está pensada para realidad virtual y requiere de algunas modificaciones para trabajar con realidad mixta. Dependiendo del dispositivo con el que se trabaja, quizá sea necesario modificar las opciones “Clear Flags” que determina lo que se mostrara en las áreas vacías del mundo virtual, las configuraciones más comunes son un color solido comúnmente situado en negro o la opción por default de Unity que es un SkyBox. En segundo lugar, están los “Clipping planes” que son las distancias a las que la cámara dejara de renderizar (es decir de generar una imagen en dos dimensiones a partir de los objetos virtuales).

6.1.4 Elementos de procesamiento de imágenes.

Como parte del trabajo presente, se realizó la conexión entre OpenCV y Unity3D. Dicha integración se realiza a través de un plugin. En computación un complemento o mejor conocido como plugin es, como su nombre lo indica, un programa independiente que se integra a otro para aumentar las capacidades de este ultimo.

Unity3D permite la integración de plugins desde C++ y C#. Sin embargo, a pesar de que OpenCV ya es un complemento en sí mismo, para poder utilizarlo dentro de Unity es necesario crear la conexión que comunique ambos.

Para la integración, se creó un .dll propio e independiente, cuya función es proveer un ambiente donde se puedan utilizar las diferentes funciones contenidas dentro de OpenCV directamente en C++, y servir como traductor entre este y el ambiente de Unity3D.

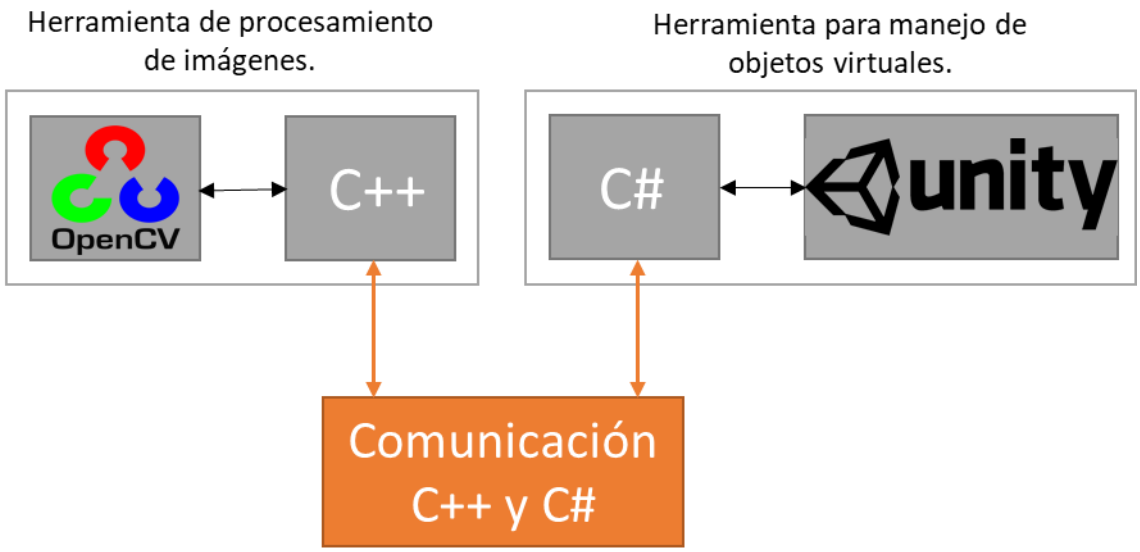

FIGURA 46 Flujo de información entre OpenCV y Unity.

El plugin creado cuenta de distintas funciones que invocan OpenCV desde C++ y un script interpretador de C# para Unity, las que se describirán a detalles más adelante. En general este plugin junto con sus funciones puede ser incluido dentro de Unity en la carpeta de Plugins, en la misma carpeta deberán ser incluidos las herramientas de OpenCV que se desee utilizar. El script de Unity se encarga de llamar las funciones del plugin, que a su vez se conecta con OpenCV e interpreta los datos enviados por estas para ser utilizados dentro del ambiente de Unity normalmente.

Las herramientas proporcionadas se enfocan en la adquisición de video, proveyendo una segunda opción para la conexión con las cámaras externas, además de permitir utilizar las herramientas de procesamiento de imágenes propias de OpenCV sobre las imágenes obtenidas. El complemento junto con el script interpretador está pensado y diseñado para funcionar en conjunto con las funciones de Unity.

C++

A continuación, se desglosan las funciones principales (el código de estas puede ser encontrado en la última sección del trabajo presente dentro de los Anexos).

- Init. Encargada de inicializar la cámara, contiene tres parámetros, dos por referencia para enviar el ancho y alto de la cámara (necesario para poder interpretar posteriormente la imagen en Unity) y uno numérico de entrada para indicar la cámara a la que se desea hacer el acceso (similar a las funciones descritas con anterioridad). Esta función devuelve un numero comunicar a Unity el estado de la cámara: 0 para indicar que todo salió correctamente, -1 para indicar una falla de conexión con OpenCV y -2 para indicar un problema de acceso a la cámara.
- GetVideoData. Encargada de tomar un cuadro o imagen desde la cámara y enviar la información de imagen en crudo. Contiene un parámetro de referencia indicando el tamaño total de pixeles contenidos en la imagen, nuevamente para poder ser interpretado desde Unity. En esta función es donde se pueden realizar todas las operaciones de procesamiento de imagen que se desee sobre la información obtenida desde las cámaras. OpenCV utiliza un formato para almacenar y modificar imágenes llamado Mat, el mat resultante (ya sea directo de la cámara o después de ser modificado) se transforma a RGB (espacio de color reconocido por Unity, y se convierte posteriormente en un arreglo de una dimensión del tipo unsigned char para poder así guardar números de 8 bits (rango de 0 a 255) correspondientes con el espacio de color RGB. Dicho arreglo es devuelto por la función.
- GetByteArray. Muy similar al anterior, con la única diferencia de que sirve únicamente como un interpretador entre los espacios de color y el tipo de datos de OpenCV y Unity. De esta manera si se tiene cualquier otra imagen independiente de las cámaras, puede ser interpretada para ser visualizada en Unity.
- printInfo. Que sirve como método de depuración. Los complementos tipo dll son códigos ya compilados por lo que depurarlos para encontrar errores es imposible. Este método permite imprimir en archivos de texto a modo de log, la distinta información que pueda necesitar el usuario con este fin.
- ReleaseMemory. A diferencia de C# donde se exime al desarrollador de la complejidad del manejo de memoria, en C++ es necesario gestionarla manualmente. Por lo que es necesario proveer una herramienta para ello. Esta función simplemente recibe un puntero y libera la memoria ocupado por este.

C#

Se presenta ahora el interpretador para poder utilizar la información proporcionada por OpenCV dentro de Unity. Dicho interpretador cuenta con las mismas funciones del bloque anterior, donde son invocadas las funciones de C++ e interpretadas a objetos y clases que Unity reconozca. De esta forma Init devuelve un vector que contiene el ancho y alto de los cuadros obtenidos desde la cámara, así como permitir indicar la cámara a la que se desea tener acceso desde el inspector grafico de Unity. GetVideoData regresa una textura2D (tipo exclusivo de Unity) que puede ser utilizada sobre cualquier objeto de la misma forma que

con las funciones del bloque anterior, tanto como elemento de interfaz o sobre un objeto tridimensional, esta misma función se encarga de manejar la memoria.

En resumen, se pueden llamar todas las funciones del complemento de OpenCV ya interpretadas para funcionar directamente con Unity, desde cualquier script de la aplicación en que se necesiten. Este archivo se puede incluir en cualquier objeto de la escena principal, sin embargo, se recomienda colocarlo sobre un objeto vacío para optimizar el rendimiento.

```
//Importar las funciones del DLL
internal static class OpenCV
{
    [DllImport("UnityOpenCV", CallingConvention = CallingConvention.Cdecl)]
    public static extern IntPtr getVideoData(ref int datasize);

    [DllImport("UnityOpenCV", CallingConvention = CallingConvention.Cdecl)]
    public static extern int ReleaseMemory(IntPtr ptr);
}

Texture2D GetVideoData(int camWidth, int camHeight)
    {
        int datasize = 0;
        IntPtr returnedPtr = OpenCV.getByteArray(ref datasize);
        byte[] returnedResult = new byte[datasize];
        Texture2D texture = new Texture2D(camWidth, camHeight,
TextureFormat.RGB24, false);
        if (datasize > 0)
        {
            Marshal.Copy(returnedPtr, returnedResult, 0, datasize);
            int succes = OpenCV.ReleaseMemory(returnedPtr);
            if (succes != 0)
                return texture = Resources.Load<Texture2D>("NoCam");
            texture.LoadRawTextureData(returnedResult);
        }
        else
        {
            texture = Resources.Load<Texture2D>("NoCam");
        }
        texture.Apply();

        return texture;
    }
..
//Desde un script asignado a algun objeto 3D
GetComponent<Renderer>().material.mainTexture = GetVideoData(680, 420);
```

FIGURA 47 Ejemplo de código del interpretador entre OpenCV y Unity

6.2 INTEGRACIÓN DE ELEMENTOS (SOFTWARE)

Se parte del supuesto de que se tiene un arreglo de cámaras estéreo para acceder al mundo real, los lentes de Realidad Mixta de Lenovo, a pesar de contar con un par de cámaras VGA

utilizadas para lograr el inside-out tracking, no permiten el acceso a dichas cámaras, que por su parte son de muy baja calidad.

Sin embargo, con la conexión de cámaras externas que permite el sistema, es posible crear el mismo efecto con una sola cámara que accediendo a un arreglo de cámaras especializadas en visión estereoscópica.

En el paquete de Unity realizado para este trabajo, se incluye un prefab llamado CameraUI para lograr dicho efecto. Es importante recordar que cualquier elemento de la interfaz de usuario está hecho para ser renderizado sobre la pantalla, por encima de los demás elementos que puedan existir, sin embargo, en cuanto a los lentes de realidad virtual concierne, esta pantalla no existe como tal, pues la imagen final que el usuario percibe es la combinación de distintos elementos como son lentes y pantallas para lograr el efecto 3D. Por esta razón es necesario incluir el prefab anteriormente mencionado en la escena si se quiere lograr el resultado de una cámara estereoscópica (ya sea que se tenga el arreglo de cámaras o una sola cámara). Este prefab junto con el script CamUI.cs sobre una imagen en la escena de Unity crea una proyección que es “vista” por la cámara principal como el fondo de la escena.

La desventaja de utilizar estos elementos en conjunto particularmente con una sola cámara es que se pierde información y el campo de visión se ve reducido considerablemente. Al tener una única imagen es necesario partir está en dos partes para lograr el efecto estéreo, en lugar de realmente adquirir dos puntos de vista distintos de la escena.

Esta información puede o no ser necesaria, dependiendo del tipo de aplicación que se desee realizar, sin embargo, se vuelve de peculiar importancia cuando de un sistema de realidad mixta se trata y por ese motivo se incluyeron las herramientas ya mencionadas.

Recapitulando se tiene la información del mundo real de ser necesario, la cual será proyectada como el fondo del mundo, logrando el efecto de mostrar a la persona sus alrededores.

Por otra parte, se puede incluir información adicional proveniente de otras fuentes, dicha información se puede combinar en la escena con las herramientas del sistema. La imagen proveniente de la cámara se puede intervenir utilizando un software de procesamiento de imágenes, y ser enviada una vez procesada posteriormente a Unity o utilizarse sin modificaciones directamente.

Estas imágenes enviadas por la cámara se actualizan constantemente en la escena tridimensional, dando la sensación de ser recibidas en tiempo real, como la información del mundo real es percibida por nuestros ojos. La velocidad a que se actualicen (medida en cuadros por segundo) es independiente de la velocidad a la que la escena final es

actualizada. Ambas velocidades son independientes una de otra, lo que quiere decir que, aunque la cámara envié imágenes a una baja velocidad, la escena final compuesta por todos los elementos puede ser renderizada y procesada por Unity a otra velocidad completamente diferente.

La velocidad con que se actualiza la información recibida por una cámara dependerá de varios factores como son las capacidades físicas de la cámara, el tipo de conexión que se esté utilizando y procesamientos extras que se realicen sobre la imagen antes de ser recibida por Unity.

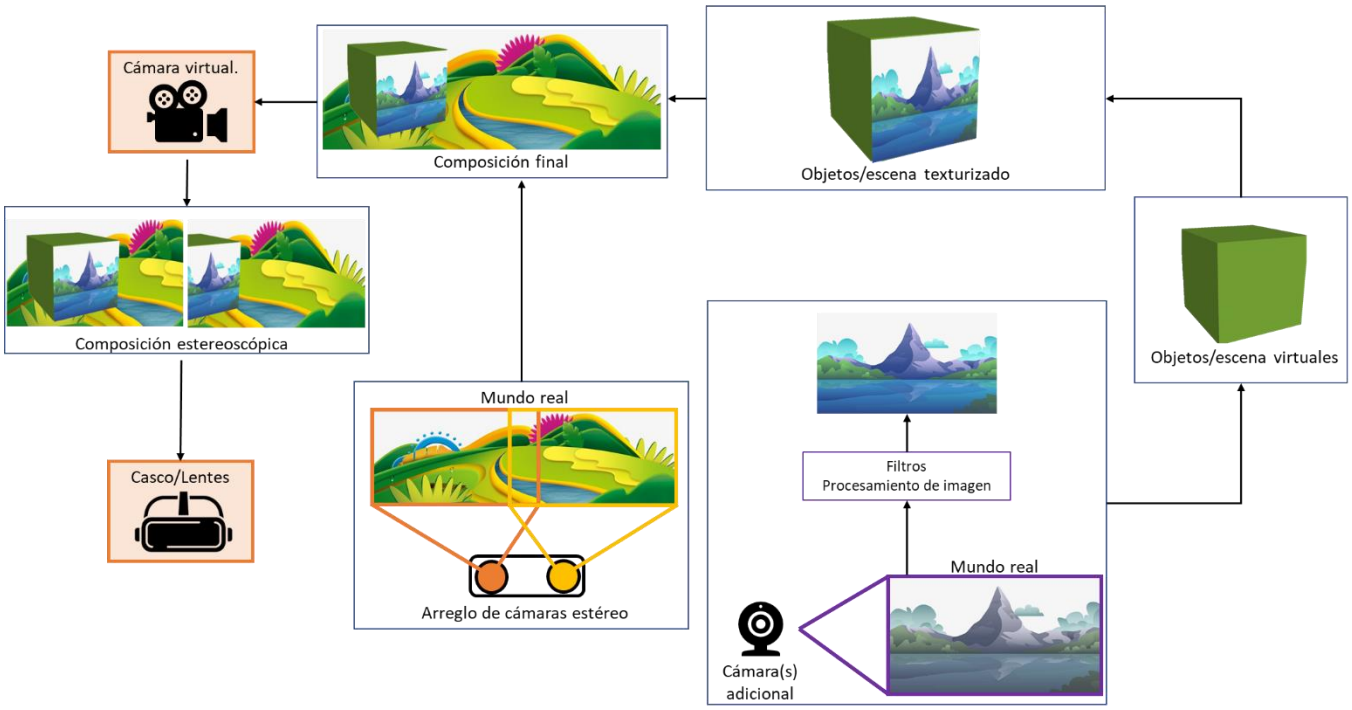

FIGURA 48 Diagrama general del funcionamiento del sistema.

Posteriormente como se ilustra en la figura 48, tanto la información del mundo real (estéreo) como la información de otras cámaras y cualquier objeto virtual perteneciente a la escena, se mezclan finalmente a través de una cámara virtual que es la encargada de generar la composición final. Esta cámara virtual es compatible con el despliegue estéreo, que siguiendo los pasos descritos en la sección “Conexión del casco de realidad mixta” del trabajo presente, crea y envía a los lentes de realidad virtual las imágenes correspondientes para cada ojo. Por último, dicha información es desplegada sobre los lentes.

6.3 PAQUETE DE UNITY 3D PARA REALIDAD MIXTA

El paquete final generado, puede ser añadido a cualquier proyecto de Unity, por medio de la opción Assets->Import package->Custom package que abrirá una ventana de dialogo permitiendo elegir el archivo descrito. Una vez el paquete es agregado, se creará en el entorno de trabajo de Unity una carpeta nueva con el nombre del paquete donde se podrán encontrar las distintas herramientas contenidas para proceder con su inmediata utilización.

El paquete final propuesto en el presente trabajo contiene las siguientes subcarpetas donde se podrán encontrar distintos elementos que se explican a continuación.

Carpeta scripts.

Donde se podrán encontrar todos los scripts mencionados con anterioridad para el control de las cámaras externas. Además, incluye algunas funciones básicas de control destinados a los objetos virtuales utilizados en los ejemplos incluidos en el mismo proyecto. Estas últimas funciones cubren comportamientos básicos como rotar el objeto sobre el eje Y o Z, hacer que un objeto permanezca fijo frente a la cámara en todo momento, permitir la exploración de la escena 3D utilizando el mouse del ordenador, cambiar el material de un objeto con una tecla y prender y apagar la cámara de un objeto.

Además se incluye en esta parte el script interpretador, que como se comenta anteriormente solo es necesario arrastrarlo sobre cualquier objeto de la escena para poder llamar las funciones desde cualquier otro script.

Carpeta scenes.

Esta carpeta contiene dos escenas de Unity a modo de ejemplo, donde se utilizan los elementos mencionados anteriormente. Una está enfocada específicamente en los elementos de la interfaz de usuario, mientras la otra se centra en los elementos de la escena virtual, ejemplos de ambas se pueden ver en las figuras 43 y 44 respectivamente. Este tipo de elementos es común encontrarlos en los paquetes de Unity pues por lo general facilitan al usuario explorar y utilizar las herramientas incluidas, así como darse una idea inicial de su alcance.

Carpetas Materials, Models, Resources.

En estas carpetas se incluyen algunos materiales, modelos e imágenes ya sea utilizadas dentro de los ejemplos o necesarias para el correcto funcionamiento del sistema, como es el caso de la textura que se muestra cuando se detecta algún problema con la cámara.

No se puede garantizar la conexión con las cámaras en todo momento debido a factores externos, como son que sean desconectadas, que se queden sin batería o que exista alguna falla de cualquier otro tipo, por lo tanto, en estos casos, el juego o proyecto continuara funcionando normalmente, sin embargo, en el lugar donde anteriormente se proveía la información de la cámara se mostrara una textura que avise al usuario que dicha cámara no está disponible.

FIGURA 49 Textura de cámara no disponible.

Carpeta Test.

Dentro de la carpeta test se incluyen tres escenas y tres scripts, cada uno de ellos con la finalidad de proporcionar una herramienta para medir la velocidad de actualización (en cuadros por segundo) de varios elementos:

- Velocidad a que la escena final está siendo actualizada.
- Velocidad con que la escena es mostrada en los lentes de realidad virtual.
- Velocidad a la que Unity recibe la información proveniente de alguna cámara (tanto en el caso de ser mostrada en la interfaz de usuario o sobre un objeto tridimensional).

Carpeta Plugins.

Donde se incluyen todos los complementos de OpenCV para su versión 3.4.3 ("OpenCV library", 2018) ya compilados tanto para Debug y Release a 64 bits. Junto con los complementos de C++ que forman parte del interpretador, presentados en este trabajo.

7 RESULTADOS Y DISCUSIÓN

En primer lugar, se cuenta con el paquete anteriormente descrito, listo para ser importado directamente en cualquier proyecto de Unity.

En segundo lugar, se realizaron algunas pruebas sobre la velocidad real de actualización de las imágenes ya sea de las cámaras o de la escena final. Para dichas pruebas, se utilizaron los elementos mínimos necesarios (que serán explicados en cada caso más adelante) para ponerlas en funcionamiento, con el fin de evitar que cualquier otro proceso pudiera entorpecer o influir en los resultados.

La configuración básica de cualquier proyecto de Unity 3D incluye la cámara principal encargada de captar y traducir la escena virtual para el usuario. Esta cámara, como se mencionaba anteriormente soporta elementos de realidad virtual como son los cascos de realidad mixta, produciendo en consecuencia el par de imágenes estereoscópicas para cada ojo y en algunos casos siguiendo el movimiento de la cabeza. También se puede configurar para mostrar el resultado final en cualquier pantalla. Este elemento es imprescindible pues es el traductor entre la escena virtual y el mundo real, de tal forma que todos los proyectos y medidas presentadas a continuación fueron realizadas con este asset incluido y funcionando.

Es importante tener presente que estos datos de velocidad son independientes de la velocidad a la que funcione la pantalla en que se visualice el resultado final. Por ejemplo, Unity puede producir 60 cuadros por segundo, sin embargo, si la pantalla que se utiliza únicamente funciona a 30 cuadros, el resultado final estará en 30 cuadros. De igual forma en todos los casos es deseable lograr una velocidad de actualización por parte de Unity siempre mayor a la pantalla de destino, garantizándonos que las imágenes serán mostradas a la máxima velocidad permitida por la pantalla.

Las velocidades mostradas a continuación son las finales producidas sin tomar en cuenta la tasa de refresco de la pantalla en que se muestre el proyecto. Todos los resultados están en cuadros por segundo (abreviado de ahora en adelante como fps, frames per second por sus siglas en ingles).

En total para cada prueba se tomaron un total de 1200 muestras por cada caso presentado más adelante, para poder obtener una visión más general de las capacidades reales del proyecto.

Por otra parte, es importante mencionar que cada una de las muestras es una medida real de los fps que fueron generados y no un aproximado. Es decir, para tomar dichas medidas se dejó la escena funcionando por un total de 20 minutos, en bloques de dos minutos (120

segundos) cada sesión, por un total de 10 sesiones. La forma de medir fue utilizando un temporizador que cada segundo transcurrido contaba los cuadros generados durante ese periodo de tiempo.

7.1 FRAME RATE IDEAL.

Los videos y en general cualquier representación visual digital que pretenda dar la ilusión de movimiento, no es más que una serie de imágenes presentadas secuencialmente en cierto periodo de tiempo. El frame rate o tasa de refresco hace referencia a los cuadros (imágenes) que pueden ser mostradas en un tiempo determinado (comúnmente se miden por segundo).

Los experimentos propuestos, sugieren la pregunta ¿Cuál es el mejor frame rate? Para poder compararlos con algún punto de referencia. Sin embargo, esta es una pregunta aparentemente sin respuesta puesto que hasta ahora no se ha encontrado un óptimo.

El ojo humano no es una cámara, de tal forma que perciba una serie de imágenes, sino que parece percibir un continuo de información en lugar de cuadros fijos independientes. Este efecto se cree se debe a dos fenómenos distintos, por una parte, a la persistencia visual, que es producido dentro de la retina, donde una imagen permanece aproximadamente 40 milisegundos y se desvanece lentamente, dando lugar a la nueva información recibida del exterior. Por otra parte, está el fenómeno phi que es un tipo de ilusión óptica donde el cerebro completa la información entre dos imágenes estáticas diferentes dando la sensación de movimiento (Gibson, 1950).

El cerebro está capacitado para ayudarnos a percibir movimiento proveniente de una serie de imágenes estáticas, siempre y cuando el cambio entre una y otra sea lo suficientemente rápido. Pero ¿qué tan rápido debe ser este cambio para que el ojo humano pueda percibir movimiento o para acercarse lo mas posible a la percepción continua de la realidad?

Se han realizado varios estudios sobre este tema a lo largo de los años, aun que en si mismos no estudian el fenómeno aislado de la velocidad que debe llevar una sucesión de imágenes, se logro extraer de estos la siguiente tabla que es una aproximación de la relación entre cuadros por segundo y la percepción humana (Apteker, Fisher, Kisimov, & Neishlos, 1995; Emoto, Kusakabe, & Sugawara, 2014; Mccarthy, Sasse, & Miras, 2004).

FRAME RATE	PERCEPCIÓN HUMANA
ENTRE 6 Y 10 FPS	Se considera el mínimo para lograr la ilusión de movimiento, el frame rate mínimo real varia de persona a persona.
16 FPS	Es suficiente para percibir movimiento correctamente, pero puede producir incomodidad al usuario.

24 FPS	Es el estándar en la industria del cine, aun que los movimientos muy rápidos se suelen percibir borrosos, se considera suficiente para lograr la percepción correcta de movimiento y por lo general es cómodo para las personas.
30 FPS	A pesar de la poca diferencia con el anterior los objetos en movimiento se tienden a percibir más nítidos.
60 FPS	La mayoría de las personas es incapaz de percibir movimiento más suave con mayores frame rates a este.

FIGURA 50 Tabla comparativa de cuadros por segundo y la percepción humana de movimiento.

A pesar de que los 60 fps podrían ser considerados como el frame rate ideal, existen otros factores como el tamaño de la pantalla, brillo, contraste, la calidad de la imagen y el tipo de movimiento entre otros que pueden influir en el resultado final, como se expone en los trabajos citados anteriormente. Por ejemplo, en cuanto al cambio de brillo si se observa una pantalla negra, y esta cambia a blanco durante 0.017 milisegundos (60 fps), el ser humano será capaz de detectar el cambio, a diferencia de si el cambio de color fuera menos drástico.

Por otra parte, existen teorías que indican que el cerebro es capaz de procesar cierta cantidad de "momentos consientes" por segundo. Es decir, los ojos tienen la capacidad de adquirir información a mayor velocidad, pero el cerebro solo puede procesar cierta cantidad. Sin embargo, aun no existen pruebas contundentes para determinar la forma y velocidad con que el cerebro procesa dicha información recibida.

Otros fenómenos como el de Uncanny Valley en psicología, que indica que estadísticamente si se mide la reacción de una persona a algo que está viendo, su aceptación hacia este objeto será buena si parece completamente artificial e ira disminuyendo conforme su apariencia se vaya acercando a la realidad.

7.2 COMPARACIÓN DE VELOCIDAD DE ACTUALIZACIÓN EN UNITY EN PANTALLA Y EN EL CASCO DE REALIDAD VIRTUAL.

Por una parte, se midió la velocidad a la que Unity puede actualizar una escena con su elemento más simple que es la cámara principal sin realizar ningún proceso que requiera un despliegue estéreo. Esta velocidad puede variar dependiendo de los elementos que se incluyan en la escena, por ejemplo, si se tienen una gran cantidad de objetos 3D y luces, todos los procesos necesarios para generar el render final pueden alentar la velocidad resultante. Sin embargo, esta medida es utilizada únicamente como punto de referencia para otras posteriores. En este caso la velocidad a que se logra el mismo efecto, pero renderizando para los lentes de realidad mixta.

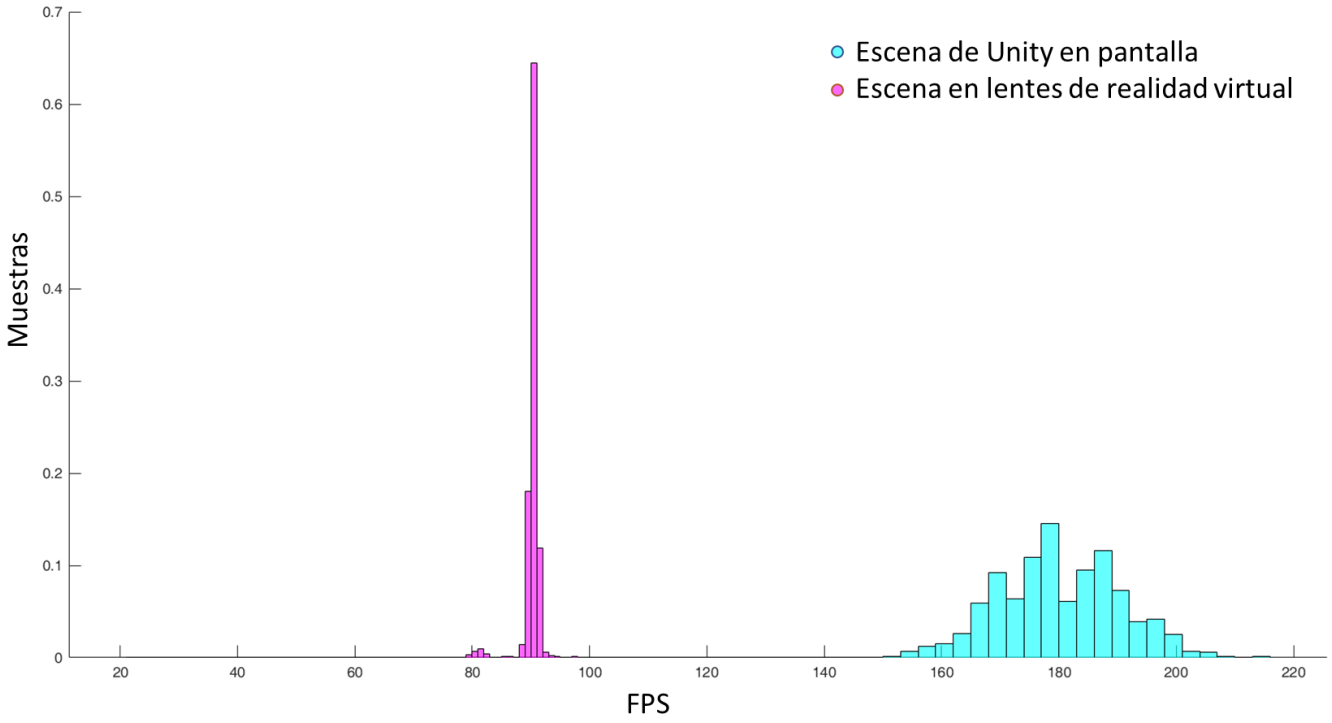

FIGURA 51 Comparación de cuadros por segundo entre una escena en pantalla y en lentes de realidad virtual.

A continuación, se presentan algunos datos extraídos de las muestras tomadas correspondientes al frame rate mínimo y máximo, promedio y desviación estándar normalizada. En los últimos tres apartados, el porcentaje de muestras correspondientes a frame rates menores a 10 fps, mayores a 24 y 60 fps respectivamente.

	Min (fps)	*Max (fps)*	*Promedio (fps)*	*σ normalizada*	*<10fps (%)*	*>24fps (%)*	*>60fps (%)*
Pantalla	21	215	179	0.058	0	99.92	99.92
Lentes	20	98	90	0.033	0	99.92	99.92

FIGURA 52 Tabla comparativa de fps entre una escena en pantalla y sobre lentes de realidad mixta.

Podemos observar que se pierde un número importante de fps al utilizar los lentes de realidad virtual, esto es debido a todos los cálculos necesarios para generar las imágenes estéreo que se enviaran a estos. Es importante mencionar, que el frame rate generado por Unity no es muy estable pues depende de muchos factores como se mencionó anteriormente, existen herramientas dentro del motor de juegos que permiten controlar esta tasa de refresco, en cuyo caso Unity se limitara a entregar tantos frames por segundo como sean solicitados siempre que sea posible.

7.3 VELOCIDAD DE ACTUALIZACIÓN DE CÁMARAS EXTERNAS DENTRO DE UNITY.

Como se comentaba anteriormente, la velocidad a la que Unity genera las imágenes finales (composición final de la escena virtual) es independiente a la velocidad a la que las cámaras son actualizadas. La velocidad de las cámaras depende de diversos factores como el tipo de conexión, la capacidad de la cámara y su resolución. Es difícil comparar los resultados de unas con otras, pues cada cámara es distinta y fue conectada por un medio distinto al ordenador. A continuación, se presentan las cámaras utilizadas para las pruebas y los datos de estas.

Cámara	Descripción	Conexión	Resolución	FPS
Laptop	Esta cámara web se encuentra embebida al ordenador montada sobre el monitor.	USB 2.0	680x420	30 fps
Logitech	Es una cámara web externa, que se conecta por cable USB.	USB 3.0	1280x720	30 fps
Trasera Samsung S7	Cámara de dispositivo móvil conectada a través de una conexión de internet, utilizando DroidCam (aplicación).	Router	1280x720	30 fps

FIGURA 53 Datos generales de las cámaras utilizadas en las pruebas.

Recordemos que la cámara del dispositivo móvil se utiliza a través de una aplicación que permite el acceso e intercambio de imágenes entre el dispositivo y el ordenador a través de una conexión Wi-Fi. El router utilizado para realizar las pruebas es uno comercial sencillo, del tipo que se utiliza en la mayoría de los hogares en la actualidad y es proporcionado por el proveedor de internet, se pueden ver sus especificaciones en la sección 4 del trabajo presente.

Las condiciones para las pruebas con las cámaras se realizaron con los elementos mínimos necesarios. En este caso para el entorno virtual se utilizo la cámara virtual (indispensable) y un único objeto 3D sobre el que se embebió el contenido proveniente de la cámara a modo de textura. Todas las pruebas de velocidad se hicieron mostrando la escena en los lentes de realidad virtual para tomar en consideración la configuración final de un proyecto de realidad mixta.

En el caso de la cámara del dispositivo móvil, el ordenador se conecto vía Ethernet al Router, mientras que el dispositivo se mantuvo a un lado de este, para evitar que en los datos se reflejara una disminución de la velocidad por elementos físicos que intervengan como pueden ser paredes o distancia al Router. Es importante comentar que los frames que se

midieron de esta última cámara son el resultado de los frames entregados por la aplicación DroidCam (utilizada para hacer la conexión), que pueden distar del frame rate real entregado por el dispositivo. La velocidad real de la cámara dependerá del método que se utilice para conectarla, lo cual esta fuera de los márgenes de este trabajo.

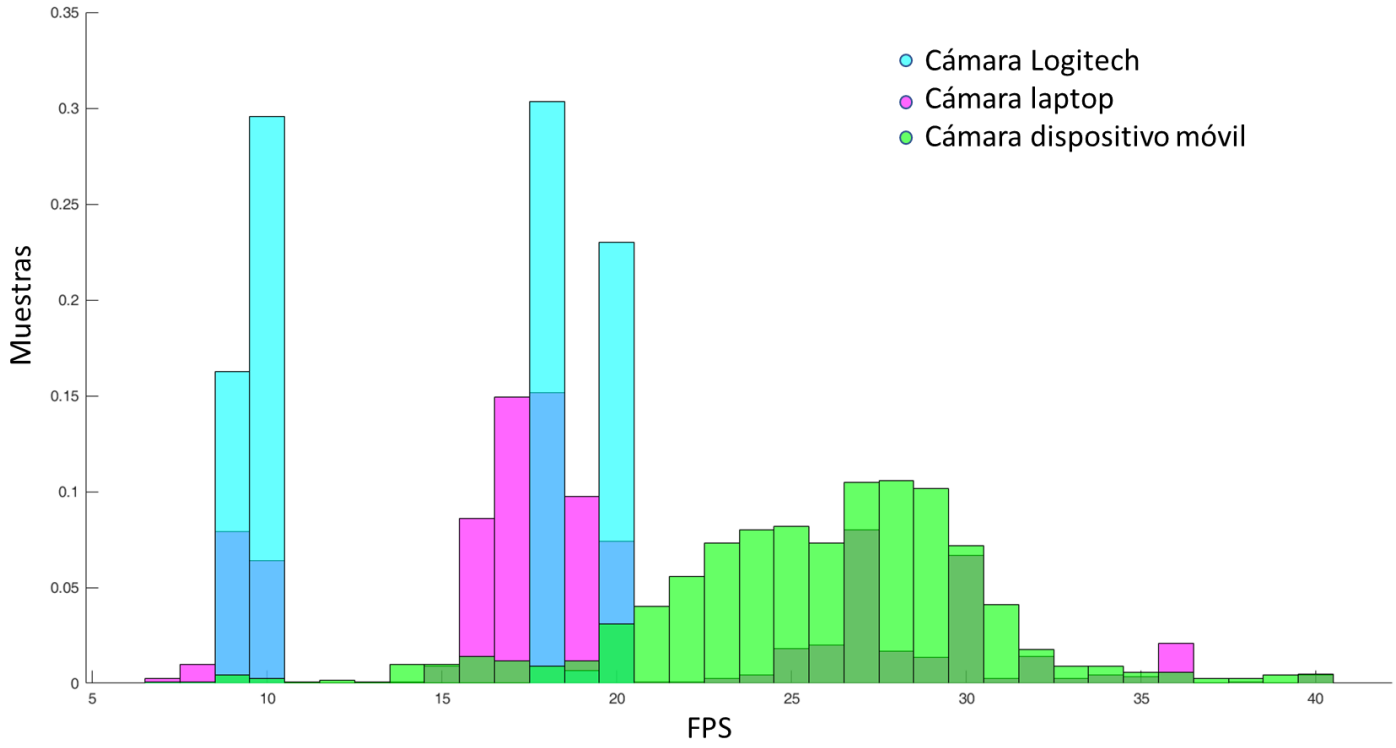

FIGURA 54 Comparación de cuadros por segundo entre tres cámaras distintas.

A continuación, se presentan algunos datos extraídos de las muestras tomadas para las cámaras descritas anteriormente, en comparación con el frame rate de la renderización de la escena final en los lentes de realidad virtual.

	Min (fps)	*Max (fps)*	*Promedio (fps)*	*σ normalizada*	*<10fps (%)*	*>24fps (%)*	*>60fps (%)*
Lentes	21	97	87	0.07	0	99.91	99.58
Laptop	7	30	19	0.21	9.17	27.17	0
Logitech	7	20	15	0.35	16.4	0	0
Android	7	40	26	0.14	0.58	72.16	0

Aunque los resultados de las velocidades de las cámaras son un poco bajos a lo esperado, tomando en cuenta que todas ellas tienen la capacidad de adquirir a 30 fps, se mantienen por encima de los 10fps que como se comentaba es lo mínimo necesario para lograr la ilusión de movimiento.

El sistema aun presenta algunos problemas de velocidad respecto a las cámaras, el tiempo que se toma en procesar los pixeles para convertirlos en una textura funcional dentro de Unity puede ser uno de los factores que afecten el rendimiento. El sistema podría mejorarse buscando hacer el acceso a la información de las cámaras más rápido y eficiente para poder aprovechar la velocidad que estas proporcionan al máximo.

Además de incluir un modulo que permita la conexión de cámaras por medio de un servidor o Wi-fi (Router) directamente desde Unity, sin la necesidad de utilizar aplicaciones externas que se encarguen de ello, con la finalidad de tener mas control sobre la conexión.

Las cámaras actuales en su mayoría cuentan con varios tipos de configuraciones para adquirir imágenes, esta peculiaridad es especialmente visible en las cámaras de los dispositivos móviles. Por ejemplo, en el caso de la cámara del celular Samsung S7, la adquisición de video se puede hacer en dos modos distintos, la configuración por defecto es a 30fps a una resolución de 1280x720, pero cuenta con un modo de 60fps a una resolución de 640x360. Un punto interesante por tratar seria la implementación de herramientas específicas para cada cámara en particular, que permitan gestionar los distintos modos de estas.

8 CONCLUSIONES

En este trabajo se desarrolló satisfactoriamente un entorno de trabajo para la creación de ambientes de Realidad Mixta, enfocado específicamente en la integración de imágenes tomadas en tiempo real provenientes de cámaras para su posterior combinación con elementos virtuales. Por supuesto queda mucho trabajo por hacer para la creación de una herramienta completa de realidad mixta, ya que esta tecnología exige la combinación no solo de elementos visuales como en los que se centra esta tesis, sino de todo tipo de hardware y herramientas para estimular los demás sentidos, como la integración de dispositivos hápticos u auditivos.

Lamentablemente la tecnología utilizada y desarrollada aun no permite el completo control sobre las cámaras en cuestión, así como la adquisición de video utilizando las capacidades máximas del hardware. Esto es en parte debido a los programas y herramientas externas utilizadas que pudieran entorpecer este proceso, a cambio de brindar al entorno de trabajo con la capacidad de ser multi plataforma.

Una solución inicial para este problema podría ser aumentar la capacidad computacional del sistema, sin embargo, este último punto se deja a consideración del lector, ya que este entorno de trabajo al ser multi plataforma, esta pensado para funcionar en dispositivos móviles además de escritorio, y aun que los teléfonos inteligentes cuentan hoy en día con grandes capacidades de procesamiento, aun no se comparan con las de los ordenadores.

Las cámaras utilizadas en este trabajo no cuentan con características distintivas, sin embargo, se pueden utilizar cualquier tipo de cámaras especializadas como infrarrojas o térmicas en conjunto con las herramientas desarrolladas para la creación de aplicaciones más específicas. Por supuesto este tipo de decisiones se dejan a discreción del usuario, entre muchas otras, como pueden ser la elección de los filtros adecuado o la calibración de las cámaras utilizadas.

Aun que no se encuentra de manera explícita, la conexión entre el entorno de trabajo y Unity es de dos vías. A primera vista parece que los datos tienen un flujo especifico que es Cámara – Plugin – Unity, sin embargo, el entorno de trabajo al servir como intermediario entre la información de las cámaras y Unity permite la interacción de los objetos virtuales con la información obtenida del mundo real, debido a que esta una vez traducida e incorporada al motor de juegos puede ser modificada e intervenida tanto como se desee.

El trabajo a futuro en esta área es basto, pues las herramientas que se pueden implementar para realidad mixta cubren una gran cantidad de áreas. Una de estas es el control de la simulación con gestos por poner un ejemplo. La inclusión de herramientas mas avanzadas de visión por computadora para mejorar la interacción y despliegue entre el mundo real y

el virtual es solamente otro ejemplo. Como estos casos existen muchos más, todos con la finalidad de enriquecer la experiencia final brindada al usuario, haciéndola lo más real, intuitiva e inmersiva posible.

9 REFERENCIAS

Apteker, R. T., Fisher, J. A., Kisimov, V. S., & Neishlos, H. (1995). Video Acceptability and Frame Rate. *IEEE MultiMedia*, *2*(3), 32–40. https://doi.org/10.1109/93.410510

Arisandi, R., Takami, Y., Otsuki, M., Kimura, A., Shibata, F., & Tamura, H. (2012). Enjoying virtual handcrafting with ToolDevice. *Adjunct Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology - UIST Adjunct Proceedings '12*, 17. https://doi.org/10.1145/2380296.2380306

Azuma, R. (1997). A survey of augmented reality presence. *Teleoperators and Virtual Environments*, *6*(4), 355–385. https://doi.org/10.1.1.30.4999

Bajura, M., Fuchs, H., & Ohbuchi, R. (1992). Merging Virtual Objects with the Real World : Seeing Ultrasound Imagery within the Patient. *Computer Graphics*, *26*(2), 203–210. https://doi.org/10.1145/142920.134061

Billinghurst, M., & Kato, H. (1999). Collaborative Mixed Reality. *Proceedings of the First International Symposium on Mixed Reality*, 261–284. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.682&rep=rep1&type=pdf

Bos, J. E., Bles, W., & Groen, E. L. (2008). A theory on visually induced motion sickness. *Displays*, *29*(2), 47–57. https://doi.org/10.1016/j.displa.2007.09.002

Bouchard, Stéphane; Wiederhold, B. K. (2014). *Advances in virtual reality and anxiety disorders*. *Advances in virtual reality and anxiety disorders. Series in anxiety and related disorders*. https://doi.org/10.1016/S1571-5078(08)00422-4

Brooks, F. P. (1999). What's real about virtual reality? *IEEE Computer Graphics and Applications*, *19*(6), 16–27. https://doi.org/10.1109/38.799723

Bur, J. W., McNeill, M. D. J., Charles, D. K., Morrow, P. J., Crosbie, J. H., & McDonough, S. M. (2010). Augmented Reality Games for Upper-Limb Stroke Rehabilitation. *2010 Second International Conference on Games and Virtual Worlds for Serious Applications*, 75–78. https://doi.org/10.1109/VS-GAMES.2010.21

Burdea, G. C., & Coiffet, P. (2003). Human Factors in VR. *Virtual Reality Technology*.

Carlin, A. S., Hoffman, H. G., & Weghorst, S. (1997). Virtual reality and tactile augmentation in the treatment of spider phobia: A case report. *Behaviour Research and Therapy*, *35*(2), 153–158. https://doi.org/10.1016/S0005-7967(96)00085-X

Carrer, M., & Gabriel, C. (2009). Mobile tagging and mixed realities. *SIGGRAPH '09: Posters on - SIGGRAPH '09*, 1–1. https://doi.org/10.1145/1599301.1599321

Chatzopoulos, Di., Bermejo, C., Huang, Z., & Hui, P. (2017). Mobile Augmented Reality Survey: From Where We Are to Where We Go. *IEEE Access*, *5*, 6917–6950. https://doi.org/10.1109/ACCESS.2017.2698164

Cruz-Neira, C., Sandin, D. J., & DeFanti, T. A. (1993). Surround-screen projection-based virtual reality. *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '93*, 135–142. https://doi.org/10.1145/166117.166134

Cullen, B., Collins, K., Hogue, A., & Kapralos, B. (2016). Sound and stereoscopic 3D: Examining the effects of sound on depth perception in stereoscopic 3D. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. https://doi.org/10.1109/IISA.2016.7785403

[IMAGEN] Dachis, A. (2018). How Skype Video Chatting Works on the HoloLens. [en linea] Microsoft Hololens. Disponible en: https://hololens.reality.news/news/skype-video-chatting-works-hololens-0171644/

Dipietro, L., Sabatini, A. M., & Dario, P. (2008). A survey of glove-based systems and their applications. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *38*(4), 461–482. https://doi.org/10.1109/TSMCC.2008.923862

Emoto, M., Kusakabe, Y., & Sugawara, M. (2014). High-Frame-Rate Motion Picture Quality and Its Independence of Viewing Distance. *Journal of Display Technology*, *10*(8), 635–641. https://doi.org/10.1109/JDT.2014.2312233

[IMAGEN] Fonrouge, S. (2018). Espectro electromagnético - Ondas electromagnéticas. [en linea] Disponible en: https://sites.google.com/site/ondaselecmag/espectro-electromagneticov

[IMAGEN] Google VR. (2018). Google Cardboard. [en linea] Disponible en: https://vr.google.com/cardboard/ [Ultimo acceso 25 Oct. 2018].

Gibson, B. J. J. (1950). *The perception of the visual world*. (L. Carmichael, Ed.) (1st ed.). Cambridge, Massachussetts: The riverside press.

Heikkila, J., Silven, O. (1997) A four-step camera calibration procedure with implicit image correction. IEEE International Conference on Computer Vision ans Pattern Recognition.

IJsselsteijn, W. a, de Kort, Y. a. W., Haans, A., Kort, Y. A. W. De, & Dickinson, E. (2006). Is This My Hand I See Before Me? The Rubber Hand Illusion in Reality, Virtual Reality, and Mixed Reality. *Presence: Teleoperators and Virtual Environments*, *15*(4), 455–464. https://doi.org/10.1162/pres.15.4.455

Intel® RealSense™. (2018). Depth Camera D435. [en linea] Disponible en: https://click.intel.com/intelr-realsensetm-depth-camera-d435.html

Jana, S., Molnar, D., Moshchuk, A., Dunn, A., Livshits, B., Wang, H. J., & Ofek, E. (2013). Enabling Fine-Grained Permissions for Augmented Reality Applications With Recognizers. In *USENIX security*.

Johnson, R. E., & Foote, B. (1988). Designing Reusable Classes. *Journal of Object-Oriented*

Programming, *1*(2), 22–35.

Jung, Y., Choi, E., Hong, H. (2014), Using Orientation Sensor of Smarthphoneto Reconstruct Environment Lights in Augmentes Reality. IEEE International Conference on Consumer Electronics (ICCE), 53-54.

Kawagoe, M., Otsuki, M., Shibata, F., & Kimura, A. (2016). Sharpen Your Carving Skills in Mixed Reality Space. *Proceedings of the 2016 Symposium on Spatial User Interaction - SUI '16*, *102*(2), 161–161. https://doi.org/10.1145/2983310.2989188

Lambooij, M., & İ, W. I. (2009). Visual Discomfort and Visual Fatigue of Stereoscopic Displays : A Review, *53*(3), 1–14. https://doi.org/10.2352/J.ImagingSci.Technol.2009.53.3.030201

Lebeck, K., Kohno, T., & Roesner, F. (2016). How to Safely Augment Reality : Challenges and Directions. *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, 45–50. https://doi.org/10.1145/2873587.2873595

Lenovo. (2018). Lenovo Explorer. [en linea] Disponible en: https://www.lenovo.com/mx/es/lenovo-explorer

Lenovo. (2017). Smartphone Phab 2 Pro. [en linea] Disponible en: https://www.lenovo.com/mx/es/serie-phab/Lenovo-Phab-2-Pro/p/WMD00000220

Livingston, M. A., & Ai, Z. (2008). The effect of registration error on tracking distant augmented objects. *Proceedings - 7th IEEE International Symposium on Mixed and Augmented Reality 2008, ISMAR 2008*, 77–86. https://doi.org/10.1109/ISMAR.2008.4637329

Livingston, M. A., Julier, S. J., Brown, D., Baillot, Y., Gabbard, J. L., & Hix, D. (2002). An Augmented Reality System for Military Operations, *89*, 1–8.

Logitech. (2018). HD Webcam C270. [en linea] Available at: https://www.logitech.com/es-es/product/hd-webcam-c270

[IMAGEN] Logitech H110. (2018). Disponible en: https://www.headsets.vn/san-pham/tai-nghe-logitech-h110/

Malek, S., Belhocine, M., Zenati-Henda, N., & Benbelakacem, S. (2011). Tracking chessboard corners using projective transformation for augmented reality. In *International Conference on Communication, Computing and Control Applications* (pp. 1–6).

María, J., & Haro, G. (2008). Diseño e implementación de un marco de trabajo (framework) de presentación para aplicaciones JEE Resumen, 1–173.

Mccarthy, J. D., Sasse, M. A., & Miras, D. (2004). Sharp or Smooth, Comparing the effects of quantization vs . frame rate for streamed video. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (Vol. 6, pp. 535–542). https://doi.org/10.1145/985692.985760

Microsoft HoloLens. (2018). Disponible en: https://www.microsoft.com/en-us/hololen

Milgram, P., & Colquhoun, H. (1999). A Taxonomy of Real and Virtual World Display Integration. *Mixed Reality-Merging Real and Virtual Worlds*, 5–30. https://doi.org/10.1.1.32.6230

Milgram, P., & Kishinott, F. (1994). Special Issue on Networked Reality. *IEICE Transactions on Information and Systems*, *E77–D*(12), 1321–1329.

Mine, M. R., Van Baar, J., Grundhofer, A., Rose, D., & Yang, B. (2012). Projection-based augmented reality in Disney theme parks. *Computer*, *45*(7), 32–40. https://doi.org/10.1109/MC.2012.154

Moser, K., Itoh, Y., Oshima, K., Swan, J. E., Klinker, G., & Sandor, C. (2015). Subjective Evaluation of a Semi-Automatic Optical See-Through Head-Mounted Display Calibration Technique. *IEEE Transactions on Visualization and Computer Graphics*, *21*(4), 491–500. https://doi.org/10.1109/TVCG.2015.2391856

Mulloni, A., Wagner, D., & Schmalstieg, D. (2008). Mobility and social interaction as core gameplay elements in multi-player augmented reality. *Proceedings of the 3rd International Conference on Digital Interactive Media in Entertainment and Arts - DIMEA '08*, 472. https://doi.org/10.1145/1413634.1413718

Munroe, C., Meng, Y., Yanco, H., & Begum, M. (2016). Augmented reality eyeglasses for promoting home-based rehabilitation for children with cerebral palsy. *ACM/IEEE International Conference on Human-Robot Interaction*, *2016–April*, 565. https://doi.org/10.1109/HRI.2016.7451858

Navab, N., Blum, T., Wang, L., Okur, A., & Wendler, T. (2012). First deployments of augmented reality in operating rooms. *Computer*, *45*(7), 48–55. https://doi.org/10.1109/MC.2012.75

[IMAGEN] New Jersey Institute of Technology. (2018). Benefits of Augmented Reality in Advertising. [en linea] Disponible en: https://graduatedegrees.online.njit.edu/resources/mba/mba-articles/3-benefits-of-augmented-reality-in-advertising/

OpenCV library. (2018). Disponible en: https://opencv.org/

Padilla, M., Frisoli, A., Pabon, S., & Bergamasco, M. (2014). The Modulation of Ownership and Agency in the Virtual Hand Illusion under Visuotactile and Visuomotor Sensory Feedback. *Presence*, *23*(2), 209–225. https://doi.org/10.1162/PRES_a_00181

Poussard, B., Loup, G., Christmann, O., Eynard, R., Pallot, M., Richir, S., … Loup-Escande, E. (2014). Investigating the main characteristics of 3D real time tele-immersive environments through the example of a computer augmented golf platform. *Proceedings of the 2014 Virtual Reality International Conference on - VRIC '14*, 1–8. https://doi.org/10.1145/2617841.2620720

Purdy, T. G., & Choi, Y. M. (2014). Enhancing augmented reality for use in product design. *Proceedings of the Extended Abstracts of the 32nd Annual ACM Conference on Human Factors in Computing Systems - CHI EA '14*, 1303–1308. https://doi.org/10.1145/2559206.2581251

Quintana, P., Bouchard, S., Serrano, B., & Cárdenas-López, G. (2014). Efectos secundarios negativos de la inmersión con realidad virtual en poblaciones clínicas que padecen ansiedad. *Revista de Psicopatologia Y Psicologia Clinica*, *19*(3), 197–207. https://doi.org/10.5944/rppc.vol.19.num.3.2014.13901

Qwake Tech, C-Thru. (2018). Disponible en: https://www.qwake.tech/

Rehder, J., Nikolic, J., Schneider, T., & Siegwart, R. (2017). A Direct Formulation for Camera Calibration. *Icra 2017*, 6479–6486.

Riehle, D. (2000). Framework design: a role modeling approach, (13509), 212. https://doi.org/10.1017/CBO9781107415324.004

Robertson, G., Card, S. K., Mackinlay, J. D., & Parc, X. (1993). Nonimmersive Virtual Reality. *Computer*, *26*(2), 81. https://doi.org/10.1109/2.192002

[IMAGEN] Rodríguez, C. (2018). Pantallas LCD. [[en linea] Solo electronicos. Disponible en: http://soloelectronicos.com/2016/11/15/tengo-un-arana-dentro-de-mi-tv/

[IMAGEN] Rodriguez, N. (2018). El ojo humano. [en linea] Laminas educativas. Disponible en: http://laminaseducativasde.blogspot.com/2018/04/el-ojo.html#.W70sdmNRdPY

Sabelman, E. E., & Lam, R. (2015). Real-LifeDANGERS OF AUGMENTED REALITY. *Spectrum, IEEE*, *52*, 48–53. https://doi.org/10.1109/MSPEC.2015.7131695

Samsung. (2018). Galaxy A9. [en linea] Disponible en: https://www.samsung.com/es/smartphones/galaxy-a9-a920/SM-A920FZBDPHE/

Samsung. (2018). Galaxy S9. [en linea] Disponible en: https://www.samsung.com/mx/smartphones/galaxy-s9/

Sharples, S., Cobb, S., Moody, A., & Wilson, J. R. (2008). Virtual reality induced symptoms and effects (VRISE): Comparison of head mounted display (HMD), desktop and projection display systems. *Displays*, *29*(2), 58–69. https://doi.org/10.1016/j.displa.2007.09.005

Sing, K. H., & Xie, W. (2016). Garden : A Mixed Reality Experience Combining Virtual Reality and 3D Reconstruction. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems - CHI EA '16* (pp. 180–183). https://doi.org/10.1145/2851581.2890370

Stanney, K. M., Mourant, R. R., & Kennedy, R. S. (1998). Human Factors Issues in Virtual Environments: A Review of the Literature. *Presence: Teleoperators and Virtual Environments*, *7*(4), 327–351. https://doi.org/10.1162/105474698565767

Steindecker, E., Stelzer, R., & Saske, B. (2014). *Requirements for virtualization of AR displays within VR environments*. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 8525 LNCS). https://doi.org/10.1007/978-3-319-07458-0_11

Sturman, D. J., & Zeltzer, D. (1994). A Survey of Glove-based Input. *IEEE Computer Graphics and Applications*, *14*(1), 30–39.

Tamura, H., Yamamoto, H., & Katayama, A. (2001). Mixed reality: Future dreams seen at the border between real and virtual worlds. *IEEE Computer Graphics and Applications*, *21*(6), 64–70. https://doi.org/10.1109/38.963462

Träskbäack, M., & Haller, M. (2004). Mixed reality training application for an oil refinery: user requirements. *Proceedings of the 2004 ACM SIGGRAPH International Conference on Virtual Reality Continuum and Its Applications in Industry*, *1*(212), 324–327. https://doi.org/10.1145/1044588.1044658

Tzovaras, D., Moustakas, K., Nikolakis, G., & Strintzis, M. G. (2009). Interactive mixed reality white cane simulation for the training of the blind and the visually impaired. *Personal and Ubiquitous Computing*, *13*(1), 51–58. https://doi.org/10.1007/s00779-007-0171-2

Vasileiou, P. G., & Psarakis, E. Z. (2015). A new depth camera calibration algorithm. *23rd International Conference on Robotics in Alpe-Adria-Danube Region, IEEE RAAD 2014 - Conference Proceedings*. https://doi.org/10.1109/RAAD.2014.7002236

Wang, X. (2011). *Mixed reality and human robot interaction*. (S. G. Tzafestas, Ed.) (Advisory B). Sidney, Australia: Springer.

Welch, R. B. (1972). The effect of experienced limb identity upon adaptation to simulated displacement of the visual field. *Perception & Psychophysics*, *12*(6), 453–456. https://doi.org/10.3758/BF03210933

[IMAGEN] Wikipedia. (2018). Stereoscope. [en linea] Disponible en: https://en.wikipedia.org/wiki/Stereoscope

[IMAGEN] Wikipedia. (2018). Virtual Boy. [en linea] Disponible en:https://es.wikipedia.org/wiki/Virtual_Boy

[IMAGEN] Wikipedia. (2018). Lucas the Spider. en linea] Disponible en: https://en.wikipedia.org/wiki/Lucas_the_Spider

Yamamoto, T., Kawagoe, M., Otsuki, M., Shibata, F., & Kimura, A. (2017). Enjoyable Carving with ChiselDevice in Mixed Reality Space. *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, *17*(1), 2–3. https://doi.org/10.1145/3139131.3143419

Zhang, S., Wang, C., & Chan, S. C. (2013). A New High Resolution Depth Map Estimation System Using Stereo Vision and Depth Sensing Device, 8–10.

Zhang, Z. (2000). A Flexible New Technique for Camera Calibration. IEEE TRansactions on Pattern Analysis and Machine Intelligence, 22(11), 1330-1334.

10 ANEXO

Código del entorno de trabajo.

C#	Unity	CamDevices.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CamDevices : MonoBehaviour {

    public WebCamDevice[] GetDevices() {
        return WebCamTexture.devices;
    }

    void Start () {
        //Print available webcam devices on console
        WebCamDevice[] devices = GetDevices();
        for (int i = 0; i < devices.Length; i++)
        {
            Debug.Log("["+i.ToString()+"] "+ devices[i].name);
        }
    }
}
```

C#	Unity	FirstCamTexture.cs

```
    private WebCamDevice[] deviceList;
    private string deviceName = "none";
    public Material material;
    public Color color = Color.white;

    void Start()
    {
        camDevices = gameObject.AddComponent(typeof(CamDevices)) as
CamDevices;
        devices = camDevices.GetDevices().Length;

        if (devices > 0)
        {
            deviceList = camDevices.GetDevices();
            deviceName = deviceList[0].name;

            try
            {
                webCamTexture = new WebCamTexture();
                material.mainTexture = webCamTexture;
                material.color = color;
                webCamTexture.Play();
            }
```

```
catch
            {
                Debug.Log("Unable to connect to " + deviceName + "
device");
                Texture2D texture =
Resources.Load<Texture2D>("NoCam");
                material.mainTexture = texture;
            }
            GetComponent<Renderer>().material = material;
        }
        else
        {
            Debug.Log("Camera devices not found.");
        }

    }
    private void OnDestroy()
    {
        webCamTexture.Stop();
    }
}
```

C#	Unity	SelectCamTeture.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public enum State
{
    number, name
}

public class SelectCamTexture : MonoBehaviour {

    public State type;
    public string findCamera = "0";
    public Material material;
    public Color color = Color.white;
    private WebCamTexture webcamtexture;

    // Use this for initialization
    void Start () {
        WebCamDevice[] devicelist = WebCamTexture.devices;
        string deviceName = "none";
        if (devicelist.Length > 0)
        {
            switch (type)
            {
                case State.name:
                    for (int i = 0; i < devicelist.Length; i++)
                    {
                        if (devicelist[i].name.Contains(findCamera))
                        {
```

```
deviceName = devicelist[i].name;
                            break;
                        }

                    }
                    break;
                case State.number:
                    int camindex = 0;
                    if (int.TryParse(findCamera, out camindex))
                    {
                        if (camindex > devicelist.Length - 1 ||
camindex < 0)
                        {
                            Debug.Log("Invalid index camera.");
                        }
                        else
                        {
                            deviceName = devicelist[camindex].name;
                        }
                    }
                    else
                    {
                        Debug.Log("Invalid camera number.");
                    }
                    break;
            }
            if (deviceName == "none")
            {
                Debug.Log("The selected camera was not found");
            }
            else
            {
                Debug.Log("Current Camera:" + deviceName);
                try
                {
                    webcamtexture = new WebCamTexture(deviceName);
                    material.mainTexture = webcamtexture;
                    material.color = color;
                    webcamtexture.Play();
                }
                catch
                {
                    Debug.Log("Unable to connect to " + deviceName +
" device");
                    Texture2D texture =
Resources.Load<Texture2D>("NoCam");
                    material.mainTexture = texture;
                }
                GetComponent<Renderer>().material = material;
            }
        }
        else
        {
            Debug.Log("Camera devices not found.");
        }
    }
```

```
private void OnDestroy()
    {
        webcamtexture.Stop();
    }
}
```

C#	Unity	CamUI.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CamUI : MonoBehaviour {

    // Use this for initialization
    public RawImage rawimage;
    private CamDevices camDevices;
    private WebCamDevice[] deviceList;
    private string deviceName = "none";
    private WebCamTexture webcamTexture;

    void Start () {
        camDevices = gameObject.AddComponent(typeof(CamDevices)) as
CamDevices;
        deviceList = camDevices.GetDevices();
        deviceName = deviceList[0].name;
        try
        {
            webcamTexture = new WebCamTexture();
            rawimage.texture = webcamTexture;
            rawimage.material.mainTexture = webcamTexture;
            webcamTexture.Play();
        }
        catch
        {
            Debug.Log("Unable to connect to " + deviceName + "
device");
            Texture2D texture = Resources.Load<Texture2D>("NoCam");
            rawimage.texture = texture;
            rawimage.material.mainTexture = texture;
        }

    }

    private void OnDestroy()
    {
        webcamTexture.Stop();
    }
}
```

C#	Unity	SelectCamUI.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public enum States
{
    number, name
}

public class SelectCamUI : MonoBehaviour
{
    public States type;
    public string findCamera = "0";
    public RawImage rawimage;
    private CamDevices camDevices;
    private WebCamDevice[] deviceList;
    private string deviceName = "none";
    private WebCamTexture webcamTexture;

    void Start()
    {
        camDevices = gameObject.AddComponent(typeof(CamDevices)) as
CamDevices;
        deviceList = camDevices.GetDevices();
        if (deviceList.Length > 0)
        {
            switch (type)
            {
                case States.name:
                    for (int i = 0; i < deviceList.Length; i++)
                    {
                        if (deviceList[i].name.Contains(findCamera))
                        {
                            deviceName = deviceList[i].name;
                            break;
                        }

                    }
                    break;
                case States.number:
                    int camindex = 0;
                    if (int.TryParse(findCamera, out camindex))
                    {
                        if (camindex > deviceList.Length - 1 ||
camindex < 0)
                        {
                            Debug.Log("Invalid index camera.");
                        }
                        else
                        {
                            deviceName = deviceList[camindex].name;
                        }
                    }
```

```
                    else
                        {
                            Debug.Log("Invalid camera number.");
                        }
                        break;
                }
                if (deviceName == "none")
                {
                    Debug.Log("The selected camera was not found");
                }
                else
                {
                    Debug.Log("Current Camera:" + deviceName);
                    try
                    {
                        webcamTexture = new WebCamTexture(deviceName);
                        rawimage.texture = webcamTexture;
                        rawimage.material.mainTexture = webcamTexture;
                        webcamTexture.Play();
                    }
                    catch
                    {
                        Debug.Log("Unable to connect to " + deviceName + "
device");
                        Texture2D texture =
Resources.Load<Texture2D>("NoCam");
                        rawimage.texture = texture;
                        rawimage.material.mainTexture = texture;
                    }
                }
            }
            else
            {
                Debug.Log("Camera devices not found.");
            }
        }

        private void OnDestroy()
        {
            webcamTexture.Stop();
        }
}
```

C#	Unity	CopyCam.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CopyCam : MonoBehaviour {

    public Material material;

    // Use this for initialization
    void Start () {
```

```
GetComponent<Renderer>().material = material;
    }

    // Update is called once per frame
    void Update () {

    }
}
```

C#	Unity	ObjFollowCam.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObjFollowCam : MonoBehaviour {

    public Transform cameraTransform;
    public float distanceFromCamera;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {
        Vector3 resultingPosition = cameraTransform.position +
cameraTransform.forward * distanceFromCamera;
        transform.position = resultingPosition;
    }
}
```

C#	Unity	ChangeMaterial.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ChangeMaterial : MonoBehaviour {

    public Material material;
    public KeyCode userKey = KeyCode.A;
    private Material orig_mat;
    // Use this for initialization
    void Start () {
        orig_mat = GetComponent<Renderer>().material;
        GetComponent<Renderer>().material = material;
        material = GetComponent<Renderer>().material;
    }

    // Update is called once per frame
```

```
void Update () {
        if (Input.GetKeyDown(userKey))
        {
            if(GetComponent<Renderer>().material == material)
            {
                GetComponent<Renderer>().material = orig_mat;
            }
            else if (GetComponent<Renderer>().material == orig_mat)
            {
                GetComponent<Renderer>().material = material;
            }
        }
    }
}
```

C#	Unity	Test_CamText.cs

```
private int cont = 0;

    private int U_frameCounter = 0;
    private int C_frameCounter = 0;
    private float U_timeCounter = 0.0f;
    private float U_lastFramerate = 0.0f;
    private float C_lastFramerate = 0.0f;
    public float refreshTime = 0.5f; //seconds (valor maximo 1)
    public float totalTime = 1.0f; //minutes

    private int instanceCount = 0;
    private int instance = 0;
    public string file_name = "01";
    private static string FILENAME1 = "";
    private static string FILENAME2 = "";

    // Use this for initialization
    void Start()
    {
        WebCamDevice[] devicelist = WebCamTexture.devices;
        string deviceName = devicelist[findCamera].name;
        webcamtexture = new WebCamTexture(deviceName);
        webcamtexture.requestedFPS = 35.0f;
        material.mainTexture = webcamtexture;
        material.color = Color.white;
        GetComponent<Renderer>().material = material;
        webcamtexture.Play();

        FILENAME1 = "D:/Users/Scarlett/Documents/Tesis_PCIC/Pruebas/"
+ file_name + "_U.txt";
        FILENAME2 = "D:/Users/Scarlett/Documents/Tesis_PCIC/Pruebas/"
+ file_name + "_C.txt";
        StreamWriter writer1 = File.CreateText(FILENAME1);
        writer1.Dispose();
        StreamWriter writer2 = File.CreateText(FILENAME2);
        writer2.Dispose();
        instance = (int)((totalTime * 60.0f) / (refreshTime)) + 1;
```

```
Debug.Log("Instancias: " + instance.ToString());
    }

    private void Update()
    {
        if (U_timeCounter < refreshTime)
        {
            U_timeCounter += Time.deltaTime;
            U_frameCounter++;
            if (webcamtexture.didUpdateThisFrame)
                C_frameCounter++;
        }
        else
        {
            U_lastFramerate = (float)U_frameCounter / U_timeCounter;
            C_lastFramerate = (float)C_frameCounter / U_timeCounter;
            if (instanceCount < instance)
            {
                cont++;
                Debug.Log(cont.ToString());
                StreamWriter writer1 = File.AppendText(FILENAME1);
                writer1.WriteLine(U_lastFramerate.ToString());
                writer1.Dispose();
                StreamWriter writer2 = File.AppendText(FILENAME2);
                writer2.WriteLine(C_lastFramerate.ToString());
                writer2.Dispose();
                instanceCount++;
            }
            else
            {
                Application.Quit();
            }
            U_frameCounter = 0;
            C_frameCounter = 0;
            U_timeCounter = 0.0f;
        }
    }

    private void OnDestroy()
    {
        webcamtexture.Stop();
    }
}
```

C#	Unity	Test_FPSScene.c

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class TestFPS_Scene : MonoBehaviour {

    private int cont = 0;
```

```
    private int U_frameCounter = 0;
    private float U_timeCounter = 0.0f;
    private float U_lastFramerate = 0.0f;
    public float refreshTime = 0.5f; //seconds (valor maximo 1)
    public float totalTime = 1.0f; //minutes

    private int instanceCount = 0;
    private int instance = 0;
    public string file_name = "01.txt";
    private static string FILENAME = "";


    // Use this for initialization
    void Start () {
        FILENAME = "D:/Users/Scarlett/Documents/Tesis_PCIC/Pruebas/" +
file_name;
        StreamWriter writer = File.CreateText(FILENAME);
        writer.Dispose();
        instance = (int)((totalTime * 60.0f)/(refreshTime))+1;
        Debug.Log("Instancias: " + instance.ToString());
    }

    // Update is called once per frame
    void Update () {
        if (U_timeCounter < refreshTime)
        {
            U_timeCounter += Time.deltaTime;
            U_frameCounter++;
        }
        else
        {
            U_lastFramerate = (float)U_frameCounter / U_timeCounter;
            if (instanceCount < instance)
            {
                cont++;
                Debug.Log(cont.ToString());
                StreamWriter writer = File.AppendText(FILENAME);
                writer.WriteLine(U_lastFramerate.ToString());
                writer.Dispose();
                instanceCount++;
            }
            else
            {
                Application.Quit();
            }
            U_frameCounter = 0;
            U_timeCounter = 0.0f;
        }
    }
}
```

C++ | OpenCV | CaptureVideo.h

```
#pragma once
#include <stdexcept>

namespace CaptureVideo {

    extern "C" {__declspec(dllexport) int Init(int& outCameraWidth,
int& outCameraHeight, int camera); }
    extern "C" {__declspec(dllexport) void Close(); }
    extern "C" {__declspec(dllexport) unsigned char* GetVideo(int&
arraysize); }

    extern "C" {__declspec(dllexport) unsigned char* getByteArray(int&
arraysize); }
    extern "C" {__declspec(dllexport) int ReleaseMemory(unsigned char*
pArray); }
    extern "C" {__declspec(dllexport) void printInfo(); }
}
```

C++ | OpenCV | CaptureVideo.cpp

```
#include "opencv2/objdetect.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"
#include "opencv2/videoio.hpp"
#include <iostream>
#include <fstream>
#include <stdio.h>
#include "CaptureVideo.h"

using namespace std;
using namespace cv;

namespace CaptureVideo {

    CascadeClassifier _faceCascade;
    String _windowName = "Unity OpenCV Interop Sample";
    VideoCapture _capture;

    int Init(int& outCameraWidth, int& outCameraHeight, int camera)
    {
        if (!_faceCascade.load("lbpcascade_frontalface.xml"))
            return -1;
        _capture.open(camera);
        if (!_capture.isOpened())
            return -2;

        outCameraWidth = _capture.get(CAP_PROP_FRAME_WIDTH);
        outCameraHeight = _capture.get(CAP_PROP_FRAME_HEIGHT);

        return 0;
```

```
}

    void Close()
    {
        _capture.release();
        destroyAllWindows();
    }

    unsigned char* GetVideo(int& arraysize)
    {

        v::Mat img;
        if (_capture.isOpened())
            _capture >> img;
        cv::cvtColor(img, img, CV_BGR2RGB, 0);
        cv::FileStorage file("mat.xml", cv::FileStorage::WRITE);
        file << "img" << img;

        arraysize = img.total()*img.channels();
        unsigned char* result = new unsigned char[arraysize];
        std::memcpy(result, img.data, arraysize);
        imshow(_windowName, img);
    }

    void printInfo()
    {
        cv::Mat img;
        if (_capture.isOpened())
            _capture >> img;
        cv::cvtColor(img, img, CV_BGR2RGB, 0);
        cv::FileStorage file("mat.xml", cv::FileStorage::WRITE);
        file << "img" << img;

        int datasize = img.total()*img.channels();
        unsigned char* result = new unsigned char[datasize];
        result = img.data;

    }

    int ReleaseMemory(unsigned char* pArray)
    {
        delete[] pArray;
        return 0;
    }
}
```

C#	Unity-OpenCV	OpenCV_Interpreter.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Runtime.InteropServices;
using UnityEngine;

// Define the functions which can be called from the .dll.
internal static class OpenCV
{
    [DllImport("UnityOpenCVSample", CallingConvention =
CallingConvention.Cdecl)]
    public static extern int Init(ref int outCameraWidth, ref int
outCameraHeight, int camera);

    [DllImport("UnityOpenCVSample", CallingConvention =
CallingConvention.Cdecl)]
    public static extern void Close();

    [DllImport("UnityOpenCVSample", CallingConvention =
CallingConvention.Cdecl)]
    public static extern IntPtr getByteArray(ref int datasize);

    [DllImport("UnityOpenCVSample", CallingConvention =
CallingConvention.Cdecl)]
    public static extern int ReleaseMemory(IntPtr ptr);

    [DllImport("UnityOpenCVSample", CallingConvention =
CallingConvention.Cdecl)]
    public static extern void printInfo();
}

public class OpenCV_interpreter : MonoBehaviour
{
    public static Vector2 CameraResolution;
    private int _findCamera = 0;
    private bool _ready;
    private int camWidth, camHeight, channels, datasize;

    void Start()
    {
        camWidth = 0;
        camHeight = 0;
        channels = 0;
        datasize = 0;
        int result = OpenCV.Init(ref camWidth, ref camHeight,
_findCamera);
        if (result < 0)
        {
            if (result == -1)
            {
                Debug.LogWarningFormat("[{0}] Failed to find cascades
definition.", GetType());
            }
            else if (result == -2)
```

```
{
                Debug.LogWarningFormat("[{0}] Failed to open camera
stream.", GetType());
            }

            return;
        }

        _ready = true;
        Debug.Log("Ready");
    }

    void OnApplicationQuit()
    {
        if (_ready)
        {
            OpenCVLIB.Close();
        }
        Debug.Log("Aplication Quit");
    }

    private void OnDestroy()
    {
        OpenCVLIB.Close();
        Debug.Log("Object destroided");
    }

    void Update()
    {
        if (!_ready)
            return;

        IntPtr returnedPtr = OpenCV.getByteArray(ref datasize);
        byte[] returnedResult = new byte[datasize];
        Marshal.Copy(returnedPtr, returnedResult, 0, datasize);
        OpenCV.ReleaseMemory(returnedPtr);

        Texture2D tex = new Texture2D(camWidth, camHeight,
TextureFormat.RGB24, false);
        tex.LoadRawTextureData(returnedResult);
        GetComponent<Renderer>().material.mainTexture = tex;
        tex.Apply();
    }
}
```

C#	Unity	Test_OpenCVCam.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.IO;

public class Test_OpenCVCam : MonoBehaviour
{

    //Test FPS
    private int cont = 0;

    private int frameCounter = 0;
    private float timeCounter = 0.0f;
    private float lastFramerate = 0.0f;
    public float refreshTime = 0.1f; //seconds (valor maximo 1)
    public float totalTime = 2.0f; //minutes

    private int instanceCount = 0;
    private int instance = 0;
    public string file_name = "opencvCam";
    private static string FILENAME = "";
    private Hash128 lasthash;

    // Use this for initialization
    void Start()
    {
        FILENAME = "D:/Users/Scarlett/Documents/Tesis_PCIC/Pruebas/" +
file_name + ".txt";
        StreamWriter writer = File.CreateText(FILENAME);
        writer.Dispose();
        instance = (int)((totalTime * 60.0f) / (refreshTime)) + 1;
        Debug.Log("Instancias: " + instance.ToString());
        lasthash =
GetComponent<Renderer>().material.mainTexture.imageContentsHash;
    }

    private void Update()
    {
        if (timeCounter < refreshTime)
        {
            timeCounter += Time.deltaTime;
            Hash128 actualhash =
GetComponent<Renderer>().material.mainTexture.imageContentsHash;
            if (actualhash != lasthash)
            {
                frameCounter++;
                lasthash = actualhash;
            }
        }
        else
        {
            lastFramerate = (float)frameCounter / timeCounter;
            if (instanceCount < instance)
            {
```

```
                cont++;
                Debug.Log(cont.ToString());
                StreamWriter writer = File.AppendText(FILENAME);
                writer.WriteLine(lastFramerate.ToString());
                writer.Dispose();
                instanceCount++;
            }
            else
            {
                Application.Quit();
            }
            frameCounter = 0;
            timeCounter = 0.0f;
        }
    }
}
```