UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INVESTIGACIÓN

EN TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

Diseñador de espacios 3D con realidad aumentada, realidad virtual, y edición online del entorno

Autor:

D. Álvaro Cid Rodríguez

Tutor:

Dr. D. Mario Martínez Zarzuela

Valladolid, Septiembre de 2012

TÍTULO:	**Diseñador de espacios 3D con realidad aumentada, realidad virtual, y edición online del entorno**
AUTOR:	**D. Álvaro Cid Rodríguez**
TUTOR:	**Dr. D. Mario Martínez Zarzuela**
DEPARTAMENTO:	TEORÍA DE LA SEÑAL Y COMUNICACIONES E INGENIERÍA TELEMÁTICA

TRIBUNAL

PRESIDENTE:	**Dr. D. Francisco Javier Díaz Pernas**
VOCAL:	**Dr. D. Carlos Gómez Peña**
SECRETARIO	**Dr. D. Miriam Antón Rodríguez**
FECHA:	**4 de Septiembre de 2012**
CALIFICACIÓN:	

Resumen de TFM

El objetivo de este Trabajo Fin de Máster es desarrollar e implementar un sistema capaz de diseñar espacios tridimensionales de forma intuitiva y funcional, añadiendo para ello el uso de tecnologías como la Realidad Aumentada (RA) y la Realidad Virtual (RV). Para ello, se han evaluado distintas tecnologías para seleccionar las idóneas y realizar la integración de ellas de la forma más eficaz posible, consiguiendo así un sistema más robusto, funcional y fácil de usar. Además, se ha incorporado un filtrado personalizado en la representación de objetos virtuales.

El lenguaje base de programación empleado para su desarrollo ha sido C++. La parte de captura de imágenes en movimiento se llevará a cabo con OpenCV, mientras que el reconocimiento de marcadores correrá por parte de la librería Alvar. Para la parte de representación de gráficos por ordenador se utilizará OpenSceneGraph. Por último, a la hora de acercar todas éstas tecnologías al usuario, se empleará Qt para crear una interfaz gráfica de usuario sencilla y fácil de usar. Todas estas soluciones permiten su utilización de forma gratuita y se adaptan correctamente a las necesidades del sistema.

Palabras clave

Realidad Aumentada, Realidad Virtual, espacios 3D, marcadores.

Abstract

The aim of this Master Thesis is to develop and implement a system able to design 3D spaces intuitively and functionally, adding technologies like augmented reality and virtual reality. Some technologies have been evaluated to select best ones and make technology integration as effective as possible. With this, we reach a more robust, functional and easy to use system. In addition, a custom filter has been built to smooth model vibrations towards the representation of the scene.

C++ is the single-based programming language. Data capture in carried out through OpenCV library. Markers detection is done by Alvar library, and all graphic content is performed by OpenSceneGraph. Moreover, Qt is used to create all the simple controls inside a functional graphic user interface. With this, all complex operation will be done de forma transparente para el user.

Keywords

Augmented Reality, Virtual Reality, 3D spaces, markers.

Agradecimientos

A mis padres, mi abuela y mi novia, por aguantarme durante la realización de éste TFM y ayudarme a la hora de tomar algunas decisiones importantes,

a mis compañeros del Grupo de Telemática e Imagen de la Uva, Rober, Antonio, Freddy… y otros tantos miembros del grupo, por los buenos momentos que hemos pasado a lo largo de estos meses,

a mi tutor Mario, por el enorme apoyo que me ha dado durante este tiempo y el buen trato que he recibido en todo momento,

y a todos los compañeros del MUI-TIC de la Uva, con los que he compartido momentos de mucho estudio, investigación y dedicación.

Índice de contenidos

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Muchos aspectos de nuestra vida han cambiado hacia una era digital, siendo el ejemplo más claro el de la televisión [46]. Así mismo, tecnologías como la Realidad Aumentada (RA) y la Realidad Virtual (RV) hacen mella en campos como la reconstrucción virtual de espacios ambientales [3] o el teatro [2]. La decoración no iba a ser menos. La capacidad de poder ver cómo va a quedar un espacio diseñado antes de su implantación física –y, por tanto, real- adquiere cada vez más importancia. Actualmente nos encontramos a medio camino del paso que dejaría atrás una mera explicación verbal o un simple dibujo, y que nos brindaría la capacidad de ver in-situ una superposición de objetos digitales a la realidad que estamos observando. Esto es lo que se denomina Realidad Aumentada (RA).

Además, a pesar de que el modelado de toda clase de objetos cuenta con nombres muy reconocidos –como 3DS Max [50], Maya [51], Blender [52], LightWave [53]- que, incluso, permiten la creación de escenarios completos y

composiciones de lo más complejas, la cantidad de controles y opciones de las que disponen muchas veces nos hacen reticentes a su uso.

En los últimos años, algunas industrias de los videojuegos han creado aproximaciones con el uso de la llamada realidad aumentada. Un ejemplo lo podemos ver en la Figura 1. Es el caso de Sony en su consola PSP en el año 2009, con el videojuego Invizimals [54], en el que se tienen que capturar una serie de monstruos representados sobre unos marcadores. Si llevamos esa idea al área que nos ocupa, podemos obtener representaciones de espacios tridimensionales dotadas de cierto realismo, las cuales podemos superponer en el propio espacio real.

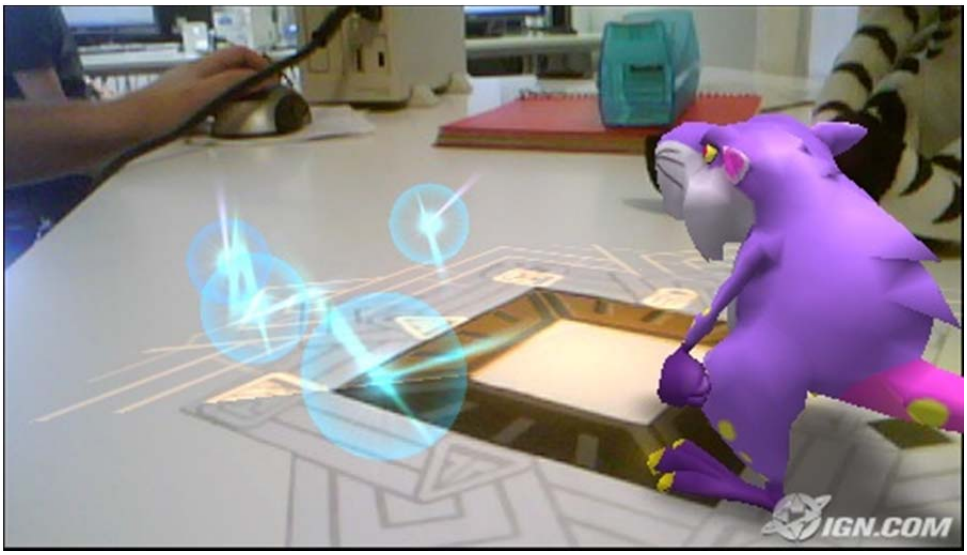

Figura 1.- Videojuego Invizimals para Sony PSP [41]

Por tanto, si reunimos las tecnologías que tenemos a nuestro alrededor, y aprovechamos tecnologías de código abierto como OpenCV [55], Alvar [56], OSG [57] y Qt [58], podemos alcanzar nuestro objetivo y desallorar un sistema innovador y fácil de usar.

1.- Motivación y objetivos

La capacidad actual para producir modelos virtuales muy similares a los reales es altísima. Así mismo, la cercanía de la tecnología a la mayor parte de la población y la disposición de manuales, hace que la competencia en cuanto a producción de modelos y software de diseño sea altísima. Los sofwares más representativos en el campo del modelado los podemos ver en la Tabla 1, donde aparecen indicadas las primeras y últimas versiones de cada uno.

Nombre	Primera versión	Última versión estable
Autodesk 3ds Max [50]	3D Studio (1990)	Autodesk 3ds Max 2013 (2012)
Autodesk Maya [51]	Maya 1998 (1998)	Autodesk Maya 2013 (2012)
Blender [52]	1.0 (1995)	2.63a (2012)
LightWave 3D [53]	LightWave 3D 1.0 (1990)	LightWave 11 (2012)

Tabla 1.- Principales softwares de modelado 3D

Por otra parte, el presupuesto necesario para llevar a cabo un sistema tan complejo y fiel a la realidad ha de ser importante. Por tanto, lo que puede hacer destacar a uno u otro es el ahorro de costes, la facilidad de uso, la capacidad de personalización, la interactuación con el diseño generado, y la incorporación de nuevas tecnologías.

Nuestro objetivo principal es el **desarrollo de un diseñador de espacios 3D con realidad aumentada, realidad virtual, y edición online del entorno**. Dado que la parte más innovadora con respecto a lo existente en el ámbito del diseño de espacios en tres dimensiones es la Realidad Aumentada (RA), nos centraremos en explotar esta característica. Según Azuma [30], la Realidad Aumentada (RA) mejora la percepción del usuario y la interacción con el mundo

real. Además, la realidad aumentada es un ejemplo específico de lo que Fred Brooks llama "Amplificación de la Inteligencia" [47], es decir, el uso del ordenador para ayudar al ser humano a realizar una tarea de una forma más fácil.

Con toda esta información, y atendiendo a la importancia que adquiere la realidad aumentada en el presente TFM, incorporamos los siguientes objetivos parciales o sub-objetivos:

- La investigación de nuevas técnicas de detección y reconocimiento de marcadores, conociendo el tipo de marcadores necesarios y el hardware compatible.
- Especificación de características básicas a cumplir por la interfaz del programa, atendiendo a las máximas de facilidad de uso, simplicidad e inmersión del usuario en el diseño generado.
- Recogida de datos provenientes de la cámara integrada en las gafas Vuzix Wrap 920AR maxReality mediante las librerías Alvar y OpenCV (Open Source Computer Vision).
- Implementación de un sistema de almacenamiento temporal que permita la edición online del sistema diseñado, que afecte a todos los ámbitos de la aplicación –vistas de diseño, vista de realidad virtual y vista de realidad aumentada-.
- Realización de la interfaz gráfica de usuario en lo alto de las tecnologías subyacentes, cerrando así la aplicación. Se han de integrar todas las soluciones tecnológicas y permitir al usuario el diseño de espacios en tres dimensiones con facilidad.

2.- Fases y métodos

El TFM consta de distintas fases:

- En una fase inicial –puesto que éste TFM nace como una nueva aplicación de la realidad aumentada– se realiza un estudio de las aplicaciones de dicha realidad aumentada en los distintos campos,

su evolución, y las posibilidades que tiene el sistema desarrollado a lo largo de TFM en cada campo.

- Posteriormente, se realiza una valoración general de las tecnologías –tanto software como hardware– disponibles en cuanto a captura de imágenes, representación de gráficos, y desarrollo de interfaces gráficas.
- Después, se realiza una lista de requisitos que deberá cumplir la aplicación, teniendo en cuenta todos los objetivos y sub-objetivos, así como la revisión de tecnologías previamente hecha.
- A continuación se desarrolla la parte de procesamiento de datos, es decir, captura de imágenes, reconocimiento de marcadores, carga de modelos para los diseños, texturas, etc.
- Después se desarrolla e implementa la interfaz gráfica de usuario, finalizando así por completo la aplicación.
- Por último, se establecen unas posibles líneas futuras de investigación y desarrollo.

3.- Medios disponibles en el TFM

El TFM se desarrolla en el Grupo de Telemática Industrial de la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universidad de Valladolid.

La cámara que utilizaremos para la captura de imágenes en movimiento -y poder detectar así los marcadores- una de las dos cámaras idénticas incorporadas en las gafas Vuzix WRAP 920AR + maxReality Bundle:

- Resolución de 640 x 480 pixeles (VGA).
- Captura de imágenes a 30 frames por segundo a una resolución de 640 x 480 pixeles.

El desarrollo del código lo llevaremos a cabo bajo el entorno de desarrollo Microsoft Visual Studio 2008 con un ordenador con las siguientes características:

- Procesador: AMD Phenom II X6 1055T 2.80 GHz.

- Memoria RAM: 3,25 GB.
- Tarjeta gráfica: nVidia GeForce GT 430.

También contamos con el SDK de VUZIX para el correcto funcionamiento de las gafas WRAP 920AR bajo Windows 7 32 bits, así como las librerías siguientes:

- OpenCV versión 2.4.0.
- Alvar versión 2.0.
- Qt versión 4.8.1.
- OpenSceneGraph versión 3.0.1.

Por último, y a nivel de documentación teórica, se dispone de las entradas correspondientes al apartado bibliogafía de éste documento. Muchos de los artículos consultados para la realización de éste TFM han sido accesibles gracias a la Universidad de Valladolid, la cual proporciona acceso a varias bases de datos como IEEE Xplore, ScienceDirect o SpringerLink.

4.- Organización del documento

El presente TFM está estructurado en tres grandes bloques muy diferenciados unos de otros para hacer más comprensible lo aquí expuesto. Veamos cuales van a ser estos bloques y una breve descripción de los mismos para dar cuenta sobre lo que tratan:

- **Capítulo 2: Alicaciones de la realidad amentada en los distintos campos:** En éste capítulo veremos el estado del arte de la realidad aumentada. Aplicaciones actuales, últimas innovaciones, etc.
- **Capítulo 3: Revisión de Tecnologías:** Aquí veremos las distintas opciones disponibles para la realización de éste TFM en lo que a software y hardware se refiere.

- **Capítulo 4: Sistema propuesto:** Este apartado contará con una explicación de los módulos en los que se divide el sistema propuesto, así como una primera aproximación o prototipo de integración de tecnologías. Por último, veremos el desarrollo completo del sistema.
- **Capítulo 5: Conslusiones y líneas futuras:** se abordarán las conclusinoes finales obtenidas en el TFM y se expondrán las sucesivas líneas futuras que se podrían abordar a raíz de este TFM.

Capítulo 2

Aplicaciones de la Realidad Aumentada

Aunque la Realidad Aumentada es una tecnología en auge, utilizada en los últimos años en infinidad de aplicaciones móviles, lo cierto es que desde hace algunos años ya se habla de ella. En éste capítulo veremos cómo nació el concepto de realidad aumentada, cómo evolucionó hasta nuestros días, y cómo afecta a las distintas áreas de la ciencia, la tecnología y la sociedad.

1.- Introducción

Podemos ver la realidad aumentada como una tecnología que nace de una necesidad de evolución de la realidad virtual. Mientras que la realidad virtual lo que pretende es sustituir el mundo real por uno virtual, la realidad aumentada no aleja al usuario del mundo real, sino que superpone información virtual sobre la realidad existente, permitiendo al usuario interactuar de una u otra manera con los objetos virtuales sobreimpresos en el mundo real.

El término "Realidad Aumentada" fue definido en 1997 por Ronald T. Azuma [30], donde habla de la realidad aumentada como una variación de la realidad virtual, en la que se permite al usuario ver el mundo real en lugar de reemplazarlo, tal y como la realidad virtual hace. Un ejemplo se puede ver en la Figura 2, donde una lámpara y dos sillas virtuales se superponen en un escenario real con una mesa y un teléfono reales.

Figura 2.- Mesa real con una lampara y dos sillas virtuales [30]

Además, para evitar limitar la realidad aumentada al uso de los llamados HMD (Head-Mounted Display) –de los cuales hablaremos en el siguiente apartado, y de los que podemos ver una imagen en la Figura 3– Azuma [30] define Realidad Aumentada (RA) como cualquier sistema con las siguientes tres características:

- Combina real y virtual.
- Interactúa en tiempo real.
- Los objetos han de estar en 3D.

Así, con estas tres características, se incluye el uso de monitores para la visualización, sistemas monoculares, HMDs (Head-Mounted Displays), etc.

Figura 3.- Foto de un HMD (Head-Mounted Display) real [35]

2.- Evolución desde la realidad virtual

Desde la aparición de los primeros gráficos generados por un ordenador, la tecnología ha sufrido muchos cambios.

En agosto de 1957, Morton L. Heilig, motivado por la creciente demanda de la época de nuevos caminos o formas de enseñar y entrenar individualmente a individuos sin necesidad de exponerle a una situación real determinada, creó un simulador denominado Sensorama [10], y lo patentó en 1962 –pat. nº 3.050.870–. Así pues, partiendo del ejemplo de la armada estadounidense, en la que los hombres han de ser entrenados idealmente para manejar equipamento muy complicado y/o peligroso sin correr ningún tipo de riesgo y, a ser posible, con el menor coste posible, el simulador Sensorama permitía el uso de entre uno y cuatro usuarios. Cuenta con imágenes en color, sonido stereo, vibración del asiento, generador de viento, y olfato, con el fin de conseguir una experiencia lo

más realista posible. Podemos ver una imagen del invento en la Figura 4, proveniente de la promoción del invento en la época.

Figura 4.- Promoción de Sensorama [36]

En 1966, y tras su artículo acerca de un "The Ultima Display" [48], Iván E. Sutherland creó el primer HMD (Head-Mounted Display), visible en la Figura 5. Según Iván, su invento permite a los usuarios estar envueltos en un mundo físico en el que se pueden predecir sus propiedades. Pone como ejemplo la predición de dónde caen los objetos, las sombras que tienen desde distintos ángulos, cuánta fuerza es necesaria para empujar los objetos y superar la fricción.

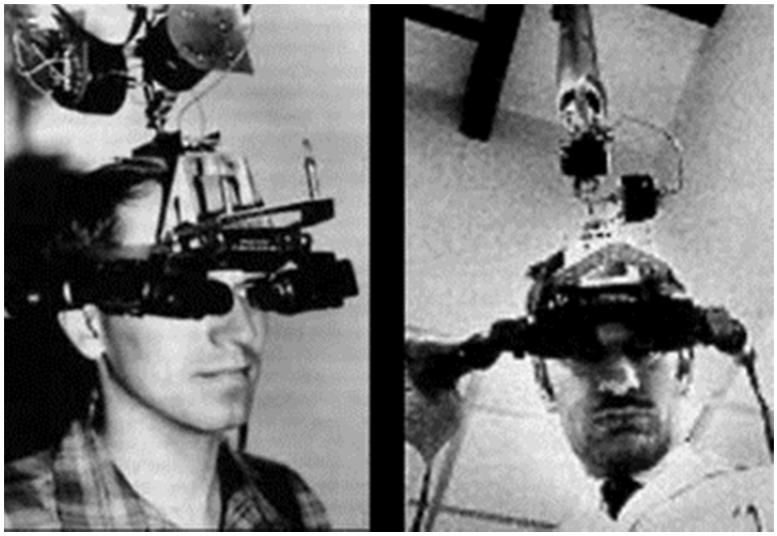

Figura 5.- HMD de Sutherland [37]

En definitiva, según palabras del mismo Sutherland, “es como una ventana en un mundo matemático”.

3.- Primera interactuación con objetos virtuales y realidad aumentada

Es Myron W. Krueger quien en 1975 crea Videoplace, el primer interfaz hombre-máquina que permite a los usuarios interactuar con objetos virtuales. La idea es creación de una realidad artificial que rodee al usuario y responda ante movimientos y acciones sin el uso de guantes o cualquier otro dispositivo intrusivo. La silueta del usuario se proyecta y responde ante los objetos artificiales proyectados, todo en tiempo real.

Podemos ver una serie de experimentos con buenos resultados en la Figura 6 y la Figura 7. Estos experimentos fueron realizados por W. Krueger et al. [12] utilizando VideoPlace para ellos.

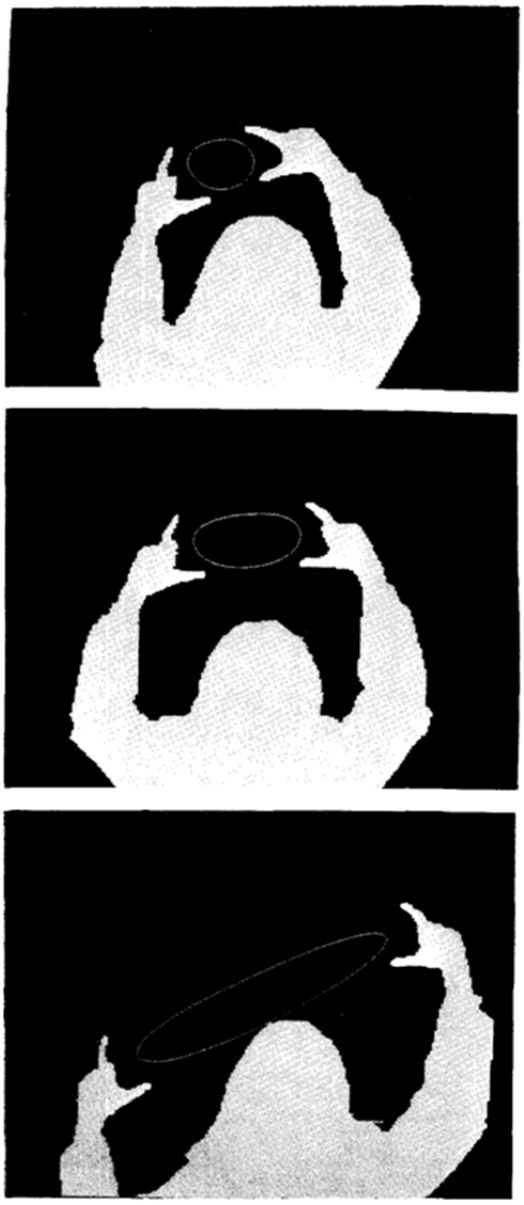

Figura 6.- Test 1 VideoPlace [12]

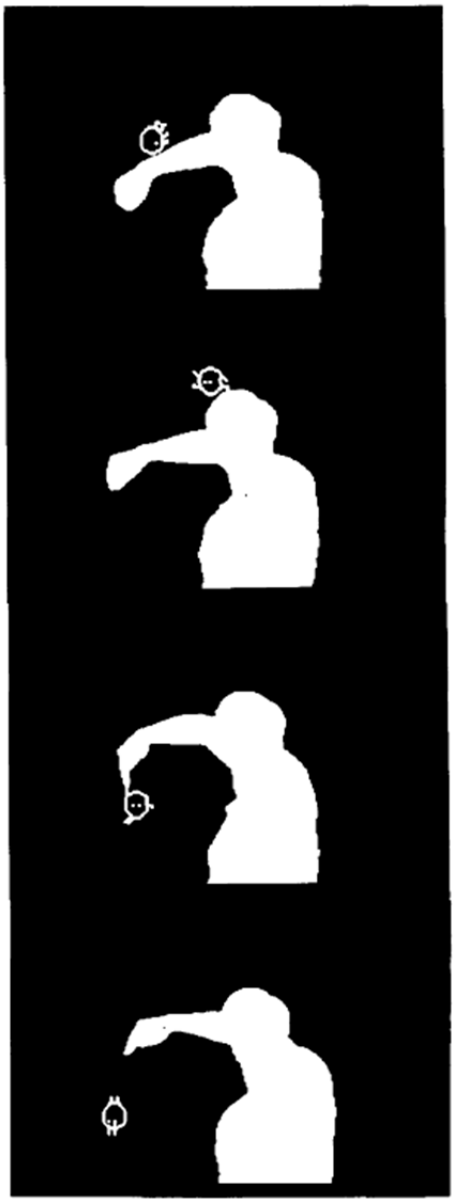

Figura 7.- Test 2 VideoPlace [12]

Siguiendo con la realidad virtual, Jaaron Lanier fue el primero en crear una aplicación comercial, aprovechamos dos de sus sistemas, como son Z-Glove y DateGlove. A lo largo de [14], [15] y [13] podemos ver desde su primera apararición en 1987, hasta su patente en 1991.

En el sistema propuesto en 1987 [14], se obtenía información acerca del gesto, la posición y la orientación de la mano, en la que se insertaba un guante con sensores analógicos para detectar la flexión de los dedos. La posición y orientación de la mano se detectan a través de dos sistemas. Por un lado está la

opción de los ultrasonidos, proveyendo cinco grados de libertad. Por otro, están los sensores magnéticos de flexión, los que proveen seis grados de libertad. Además, estos guantes tienen componentes vibratorios para dar realimentación al usuario acerca de los movimientos efectuados. Una imagen del guante sin el exterior y el resto del hardware con el software correspondiente funcionando se pueden apreciar en la Figura 8.

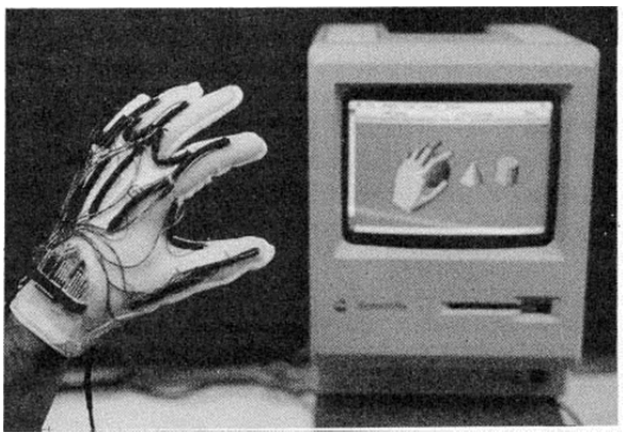

Figura 8.- Sensores visibles del guante DataGlove sin el recubrimiento exterior [14]

En una revisión del sistema propuesto, en [15], se realiza una adaptación del sistema inicial [14], para poder ser empleado por dos personas a la vez. En éste caso, existe una plataforma para diseñar e implementar realidades virtuales en tiempo real. Por ejemplo, si se aplica gravedad a un objeto de la escena, éste caerá inmediatamente en el escenario corriendo en ese momento.

Por último, en 1991 se llevó a cabo su patente [13], en la que describe el aparato como un sistema creado para manipular objetos virtuales en un sistema informático en función de los gestos y las posiciones de la mano del operador. En la Figura 9 apreciamos el diagrama de flujo principal del sistema.

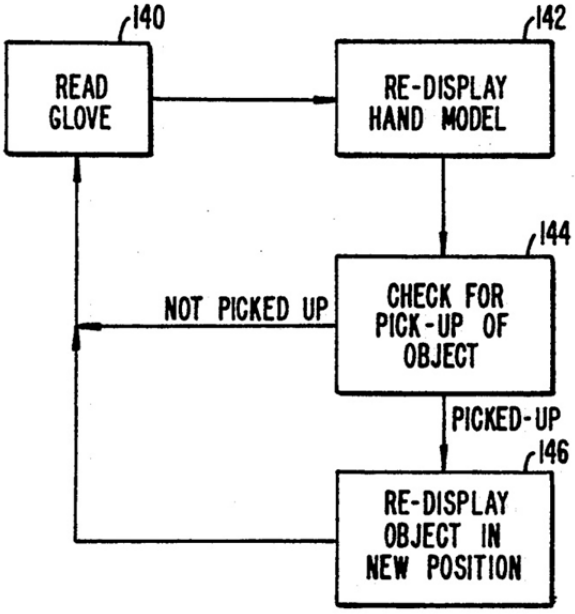

Figura 9.- Diagrama de flujo principal [13]

En la Figura 10 podemos observar la disposición del guante receptor con respecto al sistema informático que representa la parte gráfica y el procesamiento de datos. Observamos a través de los dispositivos situados en las esquinas del sistema informático cómo se realiza una especie de proyección del guante sobre la escena representada, para averiguar así la posición central de la mano tras hacer una traducción de coordenadas.

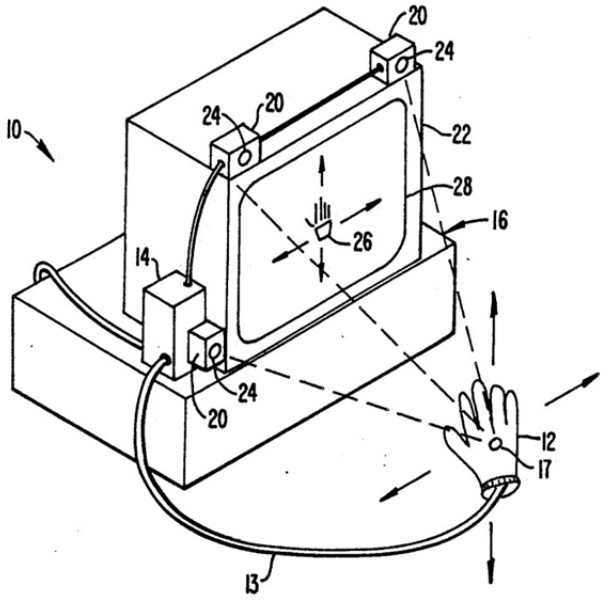

Figura 10.- Esquema del sistema patentado [13]

Si tomamos las posiciones de los receptores como A, B y C, y la posición del guante como P, las distancias desde el guante hasta A, B y C serán PA, PB y PC, respectivamente. Si las distancias AB, AC y BC son las distancias entre los distintos receptores, entonces,

$$x = \frac{PA2-PB2+AB2}{2AB} = K_1\left(PA - PB + \frac{1}{2}AB\right) \quad (2.0)$$

$$y = \frac{PC2-PA2+AC2}{2AC} = K_2\left(PC - PA + \frac{1}{2}AC\right) \quad (2.1)$$

$$z = \sqrt{PA^2 - X^2 - Y^2} = K_3(PA + PB + PC) \quad (2.2)$$

donde x, y y z son la distancia desde el origen de un sistema de coordenadas rectangulares. K_1, K_2 y K_3 son constantes de escalado. Puesto que las ecuaciones de posición sólo requieren cuadrados y raíces cuadradas, para aproximar la posición de la mano sólo son necesarias esas constantes [13].

Una de las primeras apariciones del término realidad aumentada se produce en [16], en la que se describe el diseño y prototipado para la implementación de un sistema de visión con un HMD (head-mounted display) que permite, a través de un sistema de detección de la posición de la cabeza en

un sistema de coordenadas en el mundo real, superponer imágenes sobre un objeto real. El objetivo de [16] es la tecnología denominada *HUDset* que permitirá –según los autores– la reducción de costes y mejoras de eficiencia en muchas de las operaciones involucradas en la fabricación de aeronaves, mediante la eliminación de plantillas, diagramas, etc.

Caudell y Mizell hablan de aumentar el campo de visión con información necesaria para completar una tarea determinada. Por ejemplo, en la Figura 11 se observa un operario al que se le indica mediante una flecha y un cartel generados de forma virtual, superpuestos a la realidad, toda la información necesaria para completar la tarea actual. Además, gracias a los seis grados de libertad con los que cuenta el dispositivo, se puede obtener la posición del operario y, si éste cambia su perspectiva con respecto al punto en el que desarrollar la tarea, los gráficos representados cambian para compensar ese cambio de posición y orientación visual.

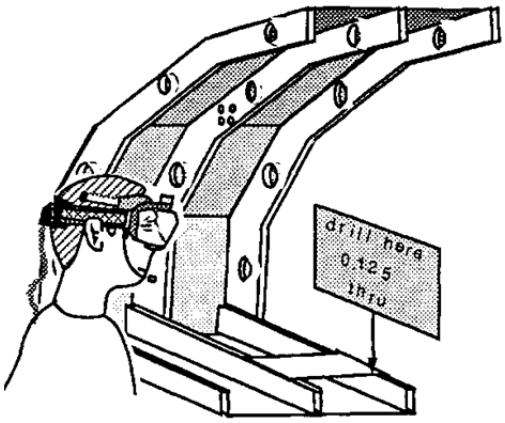

Figura 11.- Indicación de información a través de realidad aumentada para completar una tarea [16]

En ese año, el dispositivo para la representación simultánea de la realidad y de una generación virtual contaba con una fuente de imagen, un sistema de lentes de retransmisión, y un divisor de rayos. Un esquema puede verse en la Figura 12.

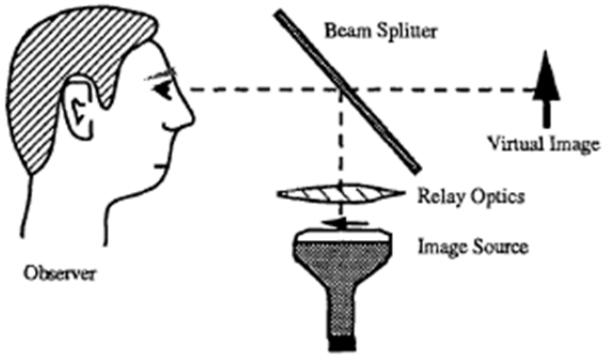

Figura 12.- Elementos básicos de un HUDset [16]

Al esquema básico se le incorpora el dispositivo de detección de posición y un micrófono para una posterior inclusión de comandos de voz. Podemos ver un esquema del montaje final en la Figura 13.

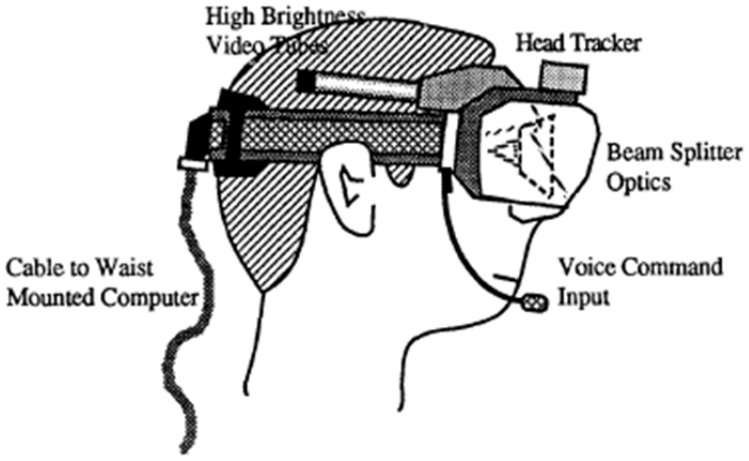

Figura 13.- Componentes de un HUDset [16]

Una vez conseguida la interación con objetos virtuales, la utilización de un dispositivo HMD en un sistema de Realidad Aumentada no tardó en sucederse. Con un sistema de teleconferencia moderadamente complejo [4], capaz de conseguir hasta 10 frames por segundo en una videollamada con visión directa entre participante y hasta 15 frames por segundo empleando una utilidad de pizarra compartida, en la Figura 14, Figura 15 y Figura 16 podemos observar un sistema de teleconferencia empleando la Realidad Aumentada.

Figura 14.- Utilizando el interfaz de Realidad Aumentada [4]

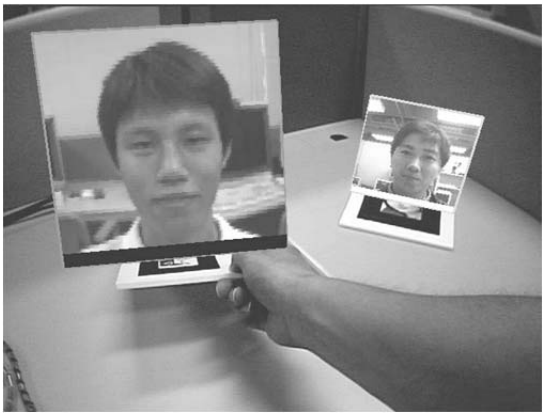

Figura 15.- Presentación remota en Realidad Aumentada [4]

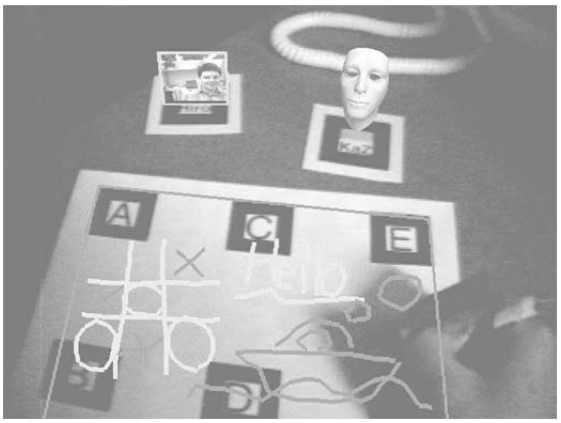

Figura 16.- Pizarra virtual compartida tilizando Realidad Aumentada [4]

Sin embargo no todo iba a ser tan sencillo, pues la posición de la cámara con respecto a los marcadores empleados y la posible variación de la misma, hace compleja la traducción de coordenadas virtuales a coordenadas del mundo real [4][17]. Podemos observar éste error en la traducción de coordenadas dentro de [4], en la Figura 16, donde la esquina inferior izquierda de la pizarra compartida representada no concide con la esquina delimitada por el marcador. Por tanto, para una correcta representación del mundo virtual generado, en la que los objetos se mimeticen con el mundo real, se hace necesaria la búsqueda de las matrices de transformación de perspectiva, rotación y translación.

Así, tras un proceso denominado calibración, en la que la cámara identifica las deformidades que producen sus lentes sobre la realidad, se realizaron pruebas que determinaron la efectividad de este método. Los dos test

consisten en sostener un marcador, situarlo en una mesa a 40 cm del ojo y situarlo en una mesa a 80 cm del ojo. Además, para cada test se comparan los parámetros estándar con los parámetros calibrados. Los resultados pueden verse en la siguiente tabla:

Usuario	Tiempo (min)	Calibrado	Test 1 (mm)	Test 2 (mm)
A	3	No	20	20
		Sí	0	5
B	2	No	30	35
		Sí	5	5
C	2	No	2	2
		Sí	2	2
D	2	No	5	10
		Sí	0	0
E	2	No	10	10
		Sí	0	0

Tabla 2.- Resultados de la calibración del HMD (Head-Mounted Display) [4]

La Tabla 2, con los resultados obtenidos tras la calibración del dispositivo HMD, muestra el error de cada test antes y después de la calibración, observando una importante mejora tras calibrar la cámara.

Para apreciar de una forma más detallada la variación del error, podemos representar el mismo para cada Test con una gráfica.

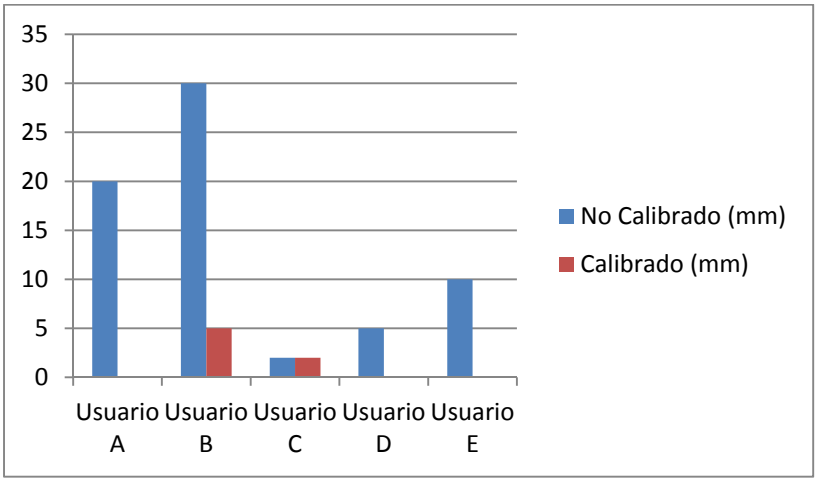

Figura 17.- Resultado del test1 para comprobar la mejora producida por la calibración de un dispositivo HMD

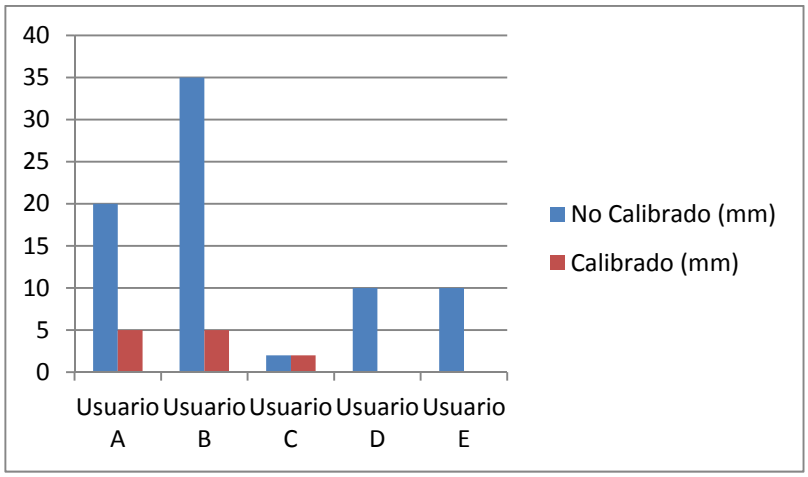

Figura 18.- Resultado del test2 para comprobar la mejora producida por la calibración de un dispositivo HMD

4.- Representación de la realidad aumentada: HMD (Head-Mounted Display)

Una decisión básica a la hora de trabajar con realidad aumentada es como mostrar la combinación de lo que es real y lo virtual. Dos elecciones básicas son la que funciona a través de una óptica y la que funciona a través de un video. A pesar de que Rolland, J. et al. realizaron una comparativa completa de estos dos sistemas [44], podemos encontrar unas pinceladas acerca de cada sistema en Kato, H. y Billinghurst, M. [4] y en Azuma, R.T. [30].

Un HMD puede ser, en general, de dos tipos. Los HMD de visión cerrada no permiten una visión directa del mundo real, mientras que los HMD de visión abierta permiten al usuario ver el mundo real, con objetos sobreimpresionados a través de tecnologías ópticas o de video.

Los HMDs de visión abierta basados en óptica colocan una combinación de ópticas en frente de los ojos del usuario. Estas lentes son por una parte transparentes, permitiendo ver la realidad a través de ellas, pero por otra tiene una capacidad de reflexión [30], lo que hace que se muestren los objetos virtuales. Un diagrama conceptual de un HMD de este tipo puede verse en la Figura 19.

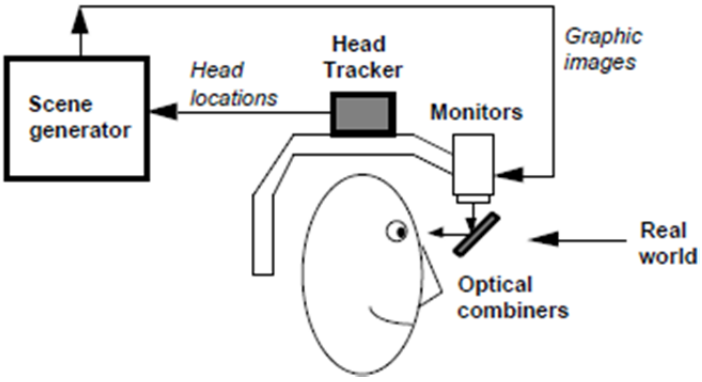

Figura 19.- Diagrama de un HMD (Head-Mounted Display) de visión abierta basado en ópticas

Por otra parte, los HMDs de visión abierta basados en video funcionan cobinando un HMD de visión cerrada con una o dos cámaras incorporadas. El video capturado por estas cámaras se combina con las imágenes gráficas generadas, obteniendo así la combinación del mundo real y el virtual. Esta combinación se muestra en dos monitores situados justo delante de los ojos del usuario [4]. La Figura 20 muestra un diagrama de éste tipo de HMD.

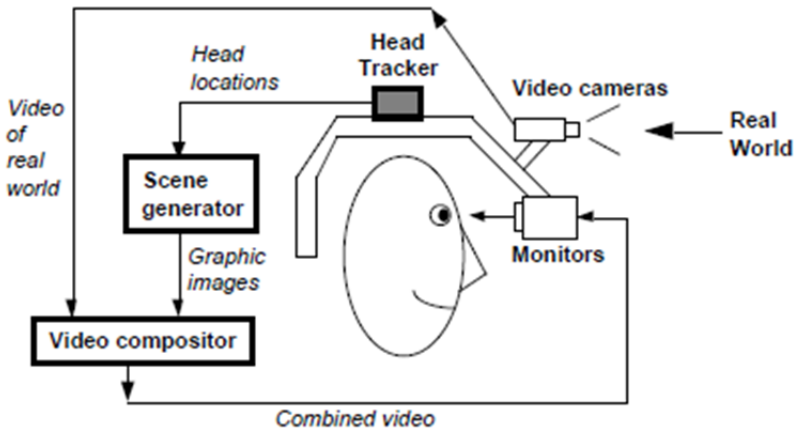

Figura 20.- Diagrama de un HMD (Head-Mounted Display) de visión abierta basado en video

Cada sistema tiene sus ventajas y sus desventajas, las cuales podemos ver detalladas a continuación, y a modo de resumen en la Tabla 3.

El HMD óptico sólo ha de trabajar con un solo canal de video, el de los gráficos generados de forma virtual. El mundo real se ve de forma directa a través de las ópticas del HMD, siendo el retraso entre lo real y lo virtual de tan solo unos pocos nanosegundos. Por otra parte, en los HMD basados en video se trabaja con dos canales de vídeo separados, el del mundo real capturado por las cámaras y el del mundo virtual. Además, la extracción de imágenes del video conlleva un retraso, por lo que se debe cuidar la sincronización de todos los canales para conseguir un efecto lo más realista posible.

Los HMDs basados en video limitan la visión del usuario a la resolución de las pantallas incorporadas en el dispositivo. Por su parte, los HMDs ópticos únicamente tienen la limitación en la resolución de los objetos virtuales, puesto que el límite de la visión del mundo real es el ojo humano.

Los HMDs basados en video están sometidos a la necesidad de una alimentación eléctrica continua. Si la alimentación desaparece, las pantallas se volverán negras y el usuario se quedará literalmente sin visión alguna. Por su parte, si la alimentación se desvanece en los HMDs ópticos, la única visión que desaparece es la correspondiente al mundo virtual, pues el usuario seguirá viendo el mundo real a través de las lentes.

Los HMDs basados en video muestran el mundo real a través de las imágenes tomadas por las cámaras exteriores. En esencia, los ojos del usuario se transladan donde las cámaras están situadas. En muchos casos, las cámaras no se sitúan exactamente donde los ojos, creando un offset o desplazamiento entre las cámaras y los ojos reales. También existe la posibilidad de que la distancia entre las cámaras sea diferente a la distancia interpupilar, provocando desplazamientos entre lo que el usuario ve y lo que realmente debería ver. Para corregir este problema, se pueden colocar una serie de espejos que se adapten a la posición de los ojos del usuario, aunque con un coste adicional. En el caso de los HMDs ópticos, si se emplea el centro de rotación del ojo como punto de vista del modelo gráfico generado por ordenador, se elimina la necesidad de seguimiento del ojo para corregir el offset en éste tipo de HMDs [33].

Un problema básico de los HMDs basados en ópticas es la combinación de varias fuentes de luces: la del mundo real y las generadas en el virtual. Por tanto, la combinación de ambas siempre se va a producir y la incorporación de algún filtro para poder ocultar la iluminación del mundo real puede provocar efectos muy poco fotorealistas. En el caso de los HMDs basados en video, al ser ambas fuentes de información digitales, resulta mucho más fácil la combinación de fuentes de iluminación, o la selección de una u otra fuente de iluminación.

Como hemos comentado antes, una imagen digital siempre va a producir un retardo, aunque sea muy pequeño [34]. En un sistema HMD basado en video, es más fácil corregir retrasos entre el mundo real y el virtual. Sin embargo, en los HMDs basados en ópticas siempre se va a ver la realidad al instante junto a la representación del mundo virtual creado –y su retraso–. Por tanto, en éste último caso, es más complejo realizar una sincronización entre lo real y lo virtual.

En los HMDs basados en ópticas, la única información que obtenemos de la cabeza del usuario proviene del "rastreador" situado sobre la misma. Sin embargo, en los HMDs basados en video, al tomar imágenes del mundo real, éstas se pueden procesar para obtener información adicional del entorno que rodea al usuario.

El enfoque puede ser un problema para ambos sistemas HMD. Idealmente, la imagen virtual debe coincidir con la real. En un sistema basado en vídeo, el combinado de imagen virtual y real se proyecta a la misma distancia de la óptica del HMD. Sin embargo, dependiendo de la cámara de vídeo, de la profundidad de campo y de los ajustes de enfoque, partes del mundo real puede estar desenfocados. En un software de gráficos típico todos los objetos aparecen enfocados, sea cual sea la distancia a la que se encuentren. Para superar esto, los gráficos podrían ser renderizados limitando la profundidad de campo, y dotando a la cámara de video de una lente de enfoque automático. En el caso de un HMD óptico, la imagen virtual se proyecta a cierta distancia del usuario. Esta distancia puede ser ajustable, aunque normalmente se deja fija. Por lo tanto, mientras los objetos reales están a diferentes distancias del usuario, los objetos virtuales son todos proyectados a la misma distancia. Si las distancias reales y virtuales de los objetos que el usuario está mirando, puede que no sea posible ver claramente ambos de forma simultánea.

Característica	HDM óptico	HDM video
Simplicidad	Un solo canal de video	Dos canales de video
Resolución	Virtual limitada	Virtual y Real limitada
Seguridad	Visibilidad asegurada	Visibilidad limitada a la alimentación eléctrica
Offset del ojo	Se corrige muy fácilmente y es mínimo	Muy acusado
Flexibilidad en la composición	Menos flexible	Más flexible
Desincronización entre lo real y lo virtual	Sólo podemos actuar sobre lo virtual	Capacidad para jugar con la fuente real y virtual
Estrategias de registro adicionales	Un solo disponsitivo localizador para la cabeza	Localizador + imagen de la realidad para su procesado
Ajuste de brillo	Más dificil	Más fácil

Tabla 3.- Resumen de la comparativa HMD basado en ópticas vs. HMD basado en video

Una vez seleccionado el dispositivo para la representación de imágenes virtuales, podemos dar paso a las diversas aplicaciones prácticas de la Realidad Aumentadad en las distintas áreas como la medicina, el montaje y la reparación de equipos –como ya hemos visto en [16]–, la muestra del recorrido de un robot, el entretenimiento o la aviación militar [30].

Uno de los campos con mayor repercusión social es la medicina. El personal sanitario puede usar la Realidad Aumentada para la recolección de datos en tres dimensiones provenientes de sistemas poco intrusivos como Resonancias Magnéticas, procesarlos en tiempo real, renderizarlos y combinarlos con una vista real del paciente, como podemos apreciar en la Figura 21, donde un feto se representa sobre el vientre de su madre a partir de la información recogida por una ecografía.

Figura 21.- Feto virtual representado con Realidad Aumentada sobre un paciente real [30]

Además de esta aplicación, la realidad aumentada puede ser muy útil para el entrenamiento del personal médico [49], o como asistencia a la hora de practicar la cirujía, minimizando el impacto sobre el paciente al localizar sobre el mismo órganos que no van a ser intervenidos [42]. En la Figura 22 podemos ver una guía para indicar el punto exacto donde se debe realizar la incisión para una biopsia.

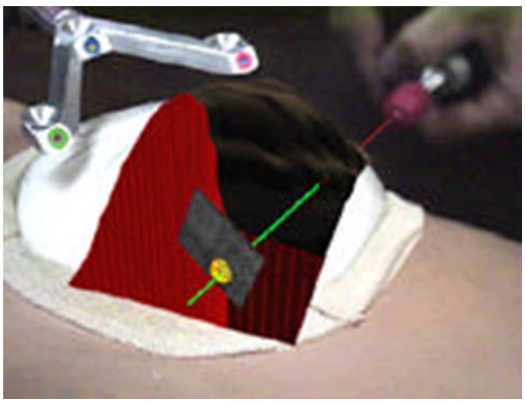

Figura 22.- Gráficos 3D para guiar la inserción necesaria en una biopsia [30]

Aparece también en [30] la posibilidad no sólo de añadir objetos al mundo real, sino de eliminarlos. Por ejemplo, para eliminar una mesa real, se puede dibujar una representación de las paredes y el suelo por detrás de la mesa real superpuestos sobre la mesa real.

Además se pueden incluir otro tipo de dispositivos no visuales. La posibilidad de incluir unos cascos y unos micrófonos exteriores permite detectar determinados sonidos reales y generar a través de los cascos una señal que cancele el sonido real [42]. Otros dispositivos interesantes pueden ser unos

guantes para proporcionar más realismo al vibrar cuando, por ejemplo, se toca un objeto virtual situado sobre una mesa real [43].

Hasta ahora, como hemos visto en las Figuras 14, 15 y 16, para localizar sobre qué parte de la realidad hay que mostrar la parte virtual se han empleado distintos marcadores especiales. Pero no necesariamente la Realidad Aumentada necesita de marcadores. Simon G. et al. emplean superficies planas como referencia para superponer elementos virtuales a la escena [3]. La pega de este sistema es –a parte de la existencia de al menos una superficie plana– que la calibración y el seguimiento comienzan con una selección manual con el ratón del ordenador. Su funcionamiento se basa en seguir un plano establecido y recuperar la posición de la cámara a través del análisis de la imágen. Además, el plano seleccionado es el encargado de proporcionar un sistema de coordenadas naturales para proceder a "aumentar" la realidad. En la Figura 23 se aprecian las 3 etapas del algoritmo que emplean: la selección del plano a seguir con el ratón del ordenador, la detección de características peculiares para el seguimiento del plano y el establecimiento de un sistema de coordenadas al indicar 4 puntos en un rectángulo.

Figura 23.- Fases del algoritmo que utiliza superficies planas como marcadores [3]

Aunque es una técnica con buenos resultados, lo cierto es que las nuevas tecnologías, capaces de crear dispositivos móviles equipados con sistemas de localización, hacen que los marcadores pasen a un segundo plano, conocido como identificación "Markerless". Cuando los sistemas de localización se emplean para sustituir a los marcadores, aparecen aplicaciones relacionadas con la recreación de yacimientos arqueológicos [7][9], la educación [22], o sistemas de información general [5], como podemos ver en la Figura 24.

Figura 24.- Aplicación para móviles WikiTude, de información general [5]

Lo cierto es que, aunque las técnicas de detección "Markerless" son bastante llamativas y permiten el ahorro de la colocación de marcadores en la escena, muchas aplicaciones novedosas siguen empleando marcadores en campos como la recuperación de edificaciones antiguas [8] o el guiado en el montaje y mantenimiento de aparatos [18].

Para finalizar, queremos dar a conocer la Realidad Aumentada como parte de la vida cotidiana, puesto que en muchos eventos deportivos televisados, se superponen elementos virtuales a la imagen de la realidad para añadir información –Figura 25–, o bien para destacar alguna parte de la imagen [31].

Figura 25.- Uso de la Realidad Aumentada en un evento deportivo televisado

5.- Realidad aumentada e investigación

Si juntamoslos artículos presentados para la conferencia ISMAR desde el año 2002 hasta el año 2007, y los artículos de eventos como IWAR ’98 y ’99, ISMR ’99 y ’01, y ISAR ’00 e ISAR ’01, tenemos un total de 276 artículos acerca de Realidad Aumentada y Realidad Mixta a lo largo de 10 años [45].

Figura 26.- Realidad Mixta [45]

Durante esos 10 años se emplean tres preguntas para analizar los artículos presentados:

- ¿Qué o cuáles áreas de la realidad aumentada han sido explorados?
- ¿Cuáles son los desarrollos más populares y los problemas detectados en esas áreas?
- ¿Qué temas son importantes para investigaciones futuras sobre realidad aumentada?

Para determinar la importancia de un documento, se recurre a un cociente formado por el número de citas en Google Scholar dividido entre el número de años desde que se publicó el artículo. Aunque Google Scholar tiene algunos fallos en las citas [45], es una importante fuente de información con respecto a la importancia o popularidad de un artículo.

Basándonos en los trabajos previos de Azuma R. et al [30] y [31], así como en [45], para tener cierto éxito y una experiencia efectiva en realidad aumentada, hay un número de factores que han de ser tenidos en cuenta,

- Renderizado de gráficos tanto por software como por hardware, capaz de crear contenido virtual para superponerlo al mundo real.
- Tecnologías de seguimiento para que los cambios en la posición del usuario se vean correctamente reflejados en los gráficos renderizados.

- Calibración del "tracker" -o herramienta de seguimiento- y herramientas de registro para alinear precisamente las vistas real y virtual cuando se forma la vista.
- Pantallas hardware para poder combinar imágenes virtuales con vistas del mundo real.
- Hardware de procesamiento para poder reproducir aplicaciones de Realidad Aumentada con soporte para dispositivos de entrada y/o salida.
- Técnicas de interactuación especificando detalladamente cómo puede el usuario manipular el contenido virtual.

Así, el número de artículos publicados a lo largo de esos 2010 años en cada categoría aparecen en la siguiente tabla.

Año	***98***	***99***	***00***	***01***	***02***	***03***	***04***	***05***	***06***	***07***		
Categoría											**Total**	**%**
Seguimiento	6	6	2	7	7	5	9	5	8	9	63	20.1
Interación	2	9	2	6	3	1	3	8	9	7	46	14.7
Calibración	5	6	4	5	6	3	2	1	3	6	44	14.1
RA App	6	7	2	9	5	8	2	2	1	4	45	14.4
Pantalla	0	4	5	7	2	3	3	4	1	8	37	11.8
Evaluaciones	0	4	1	3	2	2	0	3	5	4	18	5.8
RA Móvil	1	0	1	1	0	1	1	1	3	4	19	6.1
Autoría	0	0	0	1	2	3	3	2	0	1	12	3.8
Visualización	0	0	0	2	1	3	0	2	3	5	15	4.8
RA Multimodal	0	2	0	0	0	0	1	0	3	2	8	2.6
Renderizado	0	2	1	2	0	1	0	0	0	0	6	1.9
Total	20	40	18	43	28	30	24	28	35	47	313	100

Tabla 4.- Artículos publicados [45]

Para poder observar los cambios temporales con respecto a los temas tratados, podemos observar las figuras 27, 28 y 29.

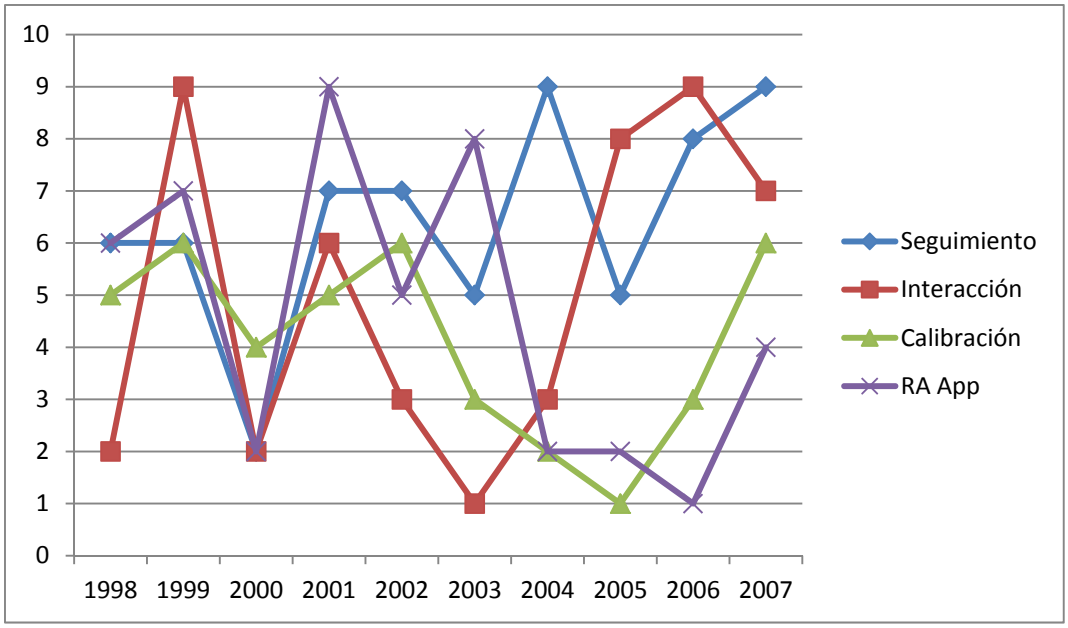

Figura 27.- Artículos publicados 1

Podemos observar como la porción de artículos de calibración y aplicaciones de Realidad Aumentada van poco a poco decreciendo, mientras que los de interación van creciendo, dándonos cuenta de los temas que van adquiriendo cada vez más relevancia.

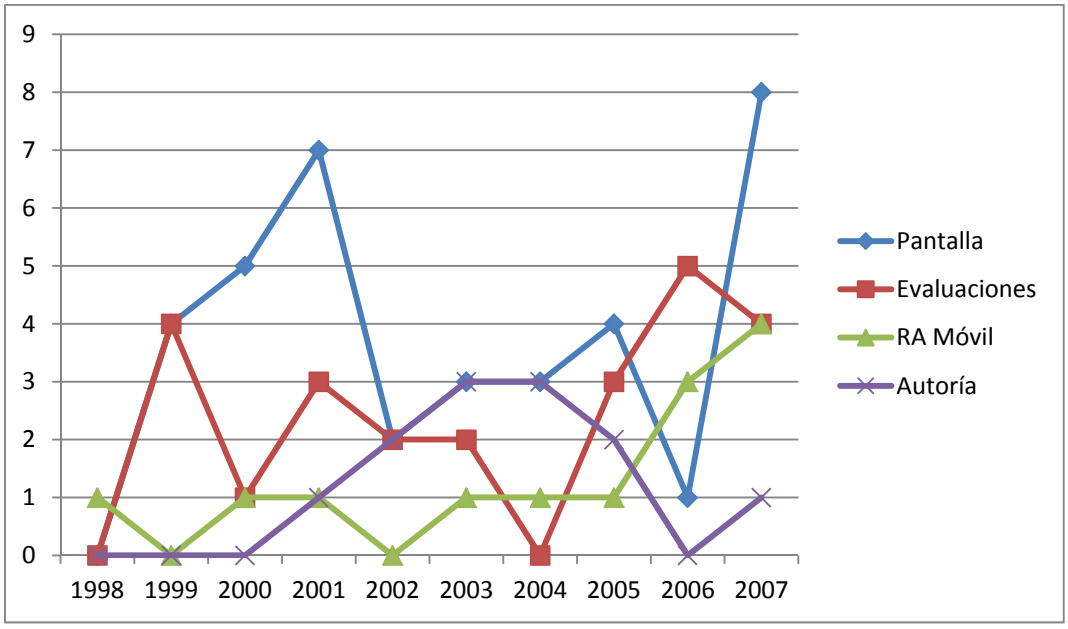

Figura 28.- Articulos publicados 2

Tal y como nos indica esta gráfica –y como vimos anteriormente– la Realidad Aumentada en dispositivos móviles va continuamente aumentando, a partir de 2002, de forma exponencial.

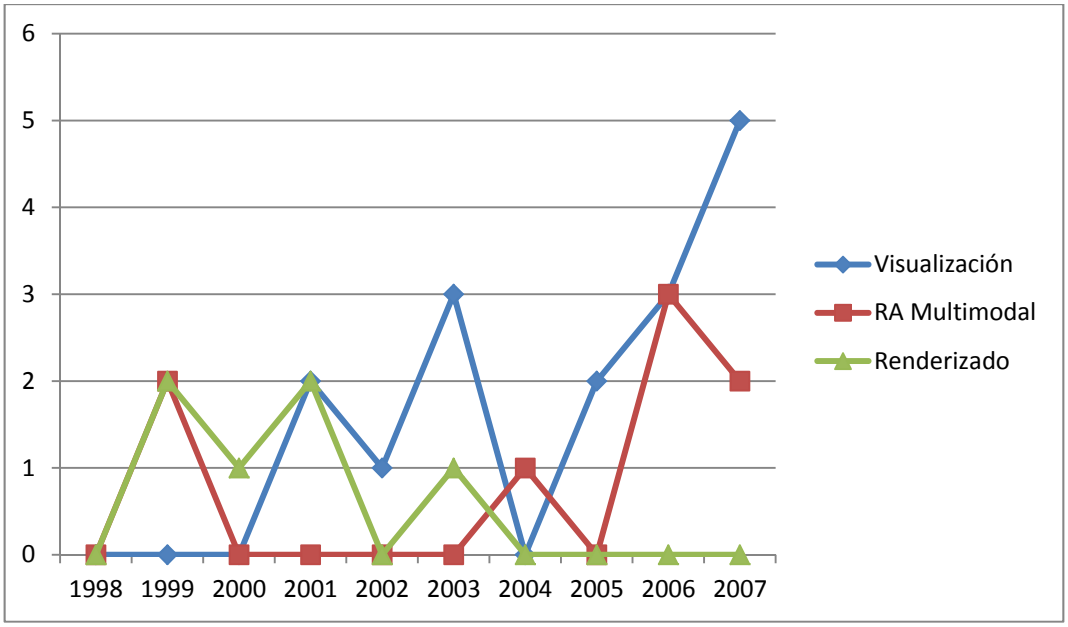

Figura 29.- Artículos publicados 3

Si observamos estas dos últimas gráficas, podemos ver cómo los temas de Visualización y Pantalla aumentan más o menos en la misma proporción, lo cual es bastante lógico dado que van en cierto modo ligado. Cuanto mayor sea la calidad de visualización, mejor se podrán aprovechar los avances y las investigaciones en pantallas.

Además, dado que una parte importante del análisis corresponde a la relevancia de cada artículo en cuando al número de citas que tiene, podemos ver los temas más citados en la siguiente tabla.

Tema	% artículos	% citas
Seguimiento	20.1	32.1
Interación	14.7	12.5
Calibración	14.1	12.5
Aplicación de RA	14.4	12.5
Pantalla	11.8	5.4
Evaluación	5.8	1.8
AR móvil	6.1	7.0
Autoría	3.8	8.9
Visualización	4.8	5.4
RA Multimodal	2.6	0.0
Renderizado	1.9	1.8

Tabla 5.- Proporcion de artículos citados

Con todos estos datos, podemos observar que los temas con más peso en temas de investigación incluyen técnicas de seguimiento, interacción del usuario con el mundo virtual y calibración, con un emergente boom en aplicaciones para dispositivos móviles, como así demuestran las más de 1000 aplicaciones registradas en Google Play [39] –el mercado de aplicaciones y juegos para Android– con Realidad Aumentada

Capítulo 3

Revisión de Tecnologías

En este capítulo se va a mostrar las características de cada tecnología –tanto hardware como software– disponible para la realización de este Trabajo Fin de Máster, así como la selección de las tecnologías empleadas en el desarrollo del sistema propuesto.

1.- Introducción

Dado que éste Trabajo Fin de Máster se centra en el desarrollo de un sistema que incorpore ciertas funcionalidades básicas similares a las de los softwares de modelado en tres dimensiones, pero aplicado al diseño de espacios en lugar de al modelado, necesitaremos una tecnología para elaborar interfaces gráficas de usuario.

Así mismo, para la representación de gráficos y, por tanto, de la escena que se diseñe en el editor, se necesita un motor de renderizado de gráficos. Éste será empleado tanto en el editor de la escena como en la navegación por Realidad Virtual y en la Realidad Aumentada.

Para la Realidad Aumentada, se necesitará un sistema de detección y seguimiento de marcadores, así como un elemento hardware para la captura de imágenes en movimiento en la que mostrar dichos marcadores.

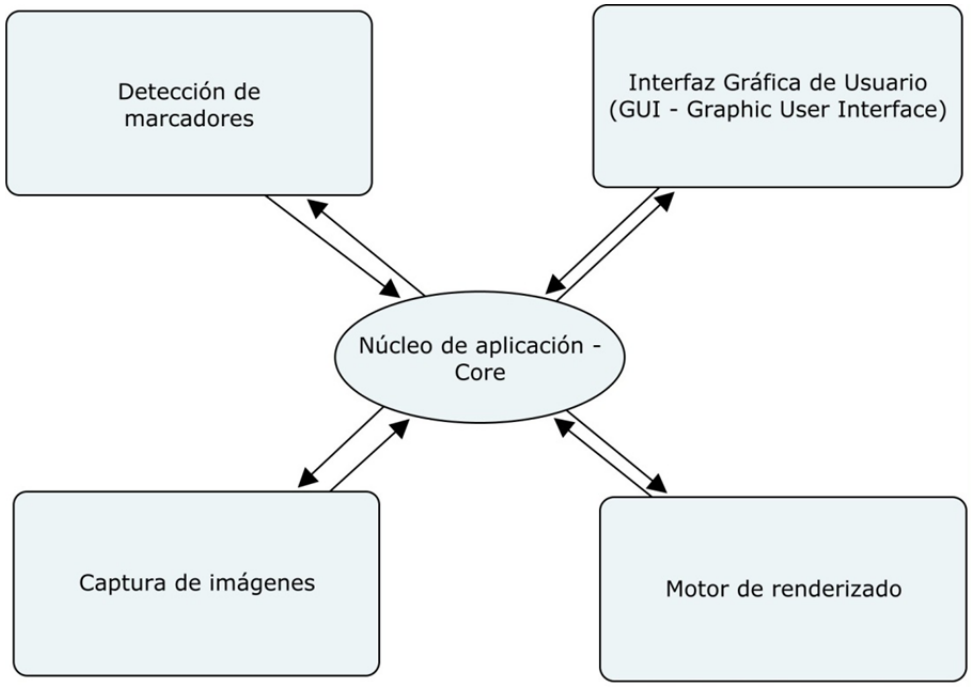

Figura 30.- Módulos de la aplicación

Atendiendo a los distintos módulos de los que se compone nuestra aplicación, interfaz gráfica de usuario, captura de imágenes, reconocimiento de marcadores y representación de modelos gráficos –ver Figura 30–, veremos las diferentes tecnologías empleadas en el modelo propuesto. En cuanto a la parte software veremos una visión general de las distintas librerías empleadas y de las características más importates de cara al desarrollo del sistema propuesto en este TFM.

Con esto, se procederá a lo largo del apartado a presentar la base teórica en lo que a tecnología empleada se refiere.

2.- Head-Mounted Display Vuzix WRAP920AR

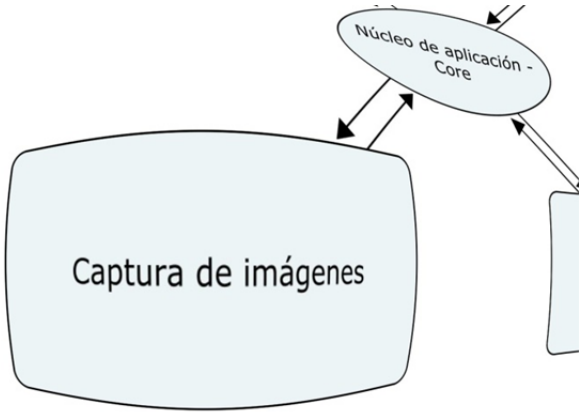

Figura 31.- Representación del módulo de captura de imágenes

Dado que contamos con éste HMD, podemos usarlo para lograr una mayor inmersión del usuario en la escena. La inmersión del usuario mediante el HMD sólo se llevará a cabo en la visualización de la escena en Realidad Aumentada y Realidad Virtual, puesto que a la hora de diseñar el espacio en tres dimensiones es mucho más cómodo trabajar sobre un monitor de un ordenador.

Éste modelo en concreto, visible en la Figura 32, cuenta con dos cámaras exteriores que se conectan por usb al ordenador. Cada cámara es capaz de capturar video con una resolución de 640 x 480 pixeles (VGA) y una tasa de 30 fotogramas por segundo. Además, no necesitan de drivers, con conectarlas por USB a un ordenador funcionan correctamente.

Además, ambas cámaras tienen un rango que va desde +2 hasta -5 dioptrías para ajustar el enfoque manualmente.

Figura 32.- HMD Vuzix Wrap 920AR [38]

Con respecto a los dispositivos para mostrar imágenes, estas “gafas” cuentan con dos pantallas de cristal líquido (LCD) –visibles en la Figura 33– con una resolución de 640 x 480 pixeles.

Estas pantallas se colocan de forma que coincidan con la disposición de los ojos, a una distancia tal, que la sensación que producen es equivalente al de una pantalla de 67 pulgadas vista desde una distancia aproximada de tres metros.

Figura 33.- Muestra de las pantallas LCD del HMD Vuzix Wrap 920AR [21]

El campo de visión que proporcionan, medido en diagonal, tiene una apertura de 31 grados, con una profundidad de color de 24 bits, es decir, 16 millones de colores.

Ambas pantallas son ajustables en brillo, contraste, tono y saturación, y tienen una tasa de refresco de 60 Hz.

Para conectar las pantallas a un ordenador, tan sólo es necesario tener una tarjeta gráfica compatible con más de un monitor, así como un puerto de salida VGA o DVI, mostrados en la Figura 34.

Figura 34.- Conectores VGA y DVI [20]

Por último, este HMD también incluye un dispositivo que dota al HMD con un sistema de seguimiento con seis grados de libertad. Este “tracker”, de tan sólo 30x20x15 mm de tamaño no supone ninguna molestia a la hora de utilizar el HMD. Podemos observarlo en la siguiente Figura.

Figura 35.- Tracker incorporado en el HMD Vuzix Wrap 920AR [38]

De este dispositivo emplearemos no sólo las cámaras y las pantallas LCD, sino además haremos uso de la información que nos proporcione acerca de los giros que se efectúen con la cabeza. Los movimientos que detectaremos son los que se incluyen en la Figura 36.

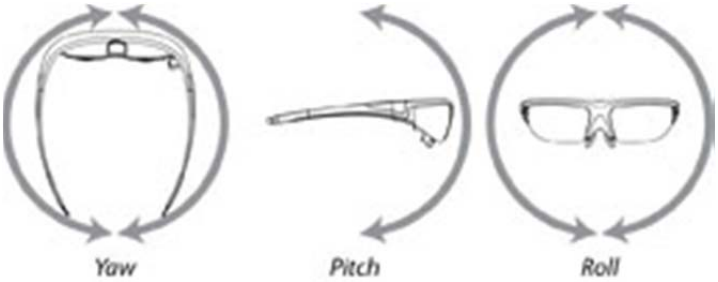

Figura 36.- Movimientos detectados por el HMD Vuzix empleados en la aplicación [38]

Naturalmente, el dispositivo que nos proporciona la información sobre los giros de la cabeza, puede ser calibrado con la aplicación incluída por Vuzix, permitiéndonos así obtener información de forma más precisa.

La aplicación de calibración se puede ver en la Figura 37.

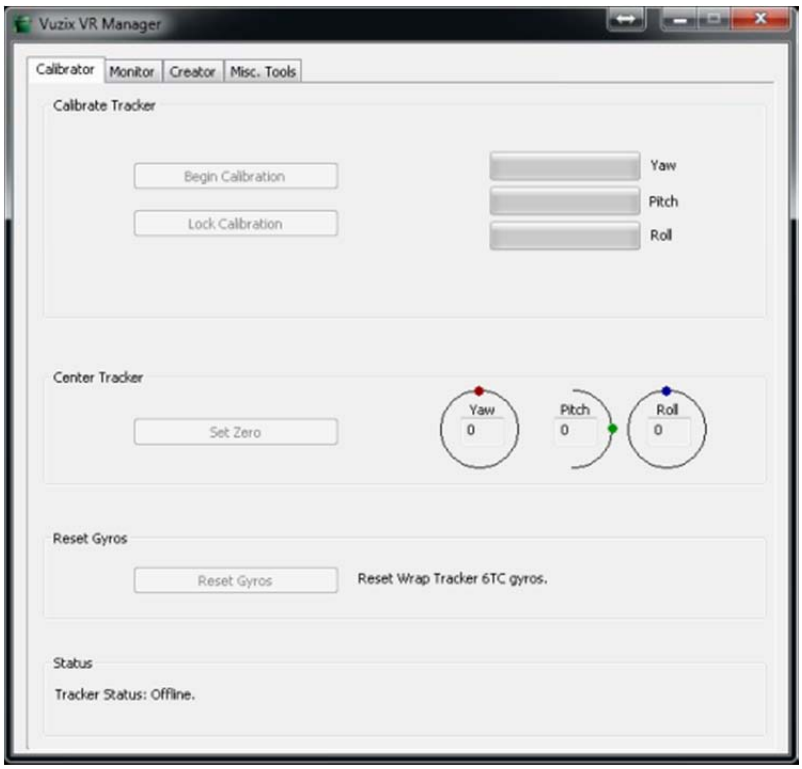

Figura 37.- Aplicación de calibración Vuzix [38]

Algunos de los métodos incluídos en el SDK (Software Development Kit) [40] de Vuzix que se emplearán en el sistema desarrollado se pueden ver descritos en la Tabla 6.

Función	Descripción
IWROpenTracker (void)	Inicia una sesión en el dispositivo de seguimiento (tracker).
IWRBeginCalibrate (void)	Empieza el proceso de calibración del tracker para adaptar los sensores a la localización del usuario.
IWRSetFilterState (BOOL on)	Activa o desactiva el filtro paso-bajo digital.

IWRSetMagAutoCorrect (BOOL on)	Activa o desactiva la corrección del valor del movimiento yaw a través del valor devuelto por los sensores magnéticos.
IWREndCalibrate (BOOL save)	Termina el proceso de calibración del tracker y guarda o descarta la misma.
IWRCloseTracker (void)	Termina una sesión en el dispositivo de seguimiento (tracker)
IWRZeroSet (void)	Utiliza la posición actual de la cabeza como referencia para futuras lecturas de posición.
IWRGet6DTracking (LONG *yaw, LONG *pitch, LONG *roll, LONG *xtrn, LONG *ytrn, LONG *ztrn)	Recibe los giros de la cabeza (yaw hacia izquierda y derecha, pitch hacia arriba y abajo y roll la inclinación hacia los hombros) y el movimiento en los distintos ejes (x-xtrn, y-ytrn y z-trn).

Tabla 6.- Métodos básicos incluídos en el SDK de Vuzix

3.- Librería Alvar

Figura 38.- Representación del módulo de detección de marcadores

Alvar es una conocida librería destinada a la creación de todo tipo de aplicaciones de Realidad Aumentada (RA) y Realidad Virtual (RV). Desarrollada por el Centro de Investigación Técnica VTT de Finlandia, está publicada bajo una licencia del tipo LGPL.

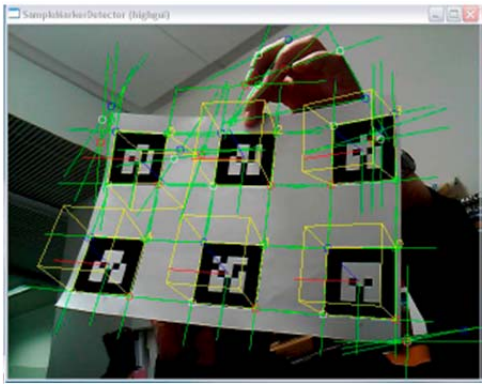

Figura 39.- Ejemplo de seguimiento de marcadores simples [11]

Su diseño, según los desarrolladores, pretende hacer de Alvar una librería lo más flexible posible, con herramientas de alto nivel que facilitan la tarea a la hora de crear aplicaciones de Realidad Aumentada. Funciona sobre Windows y Linux y la única dependencia de terceros que tiene es OpenCV.

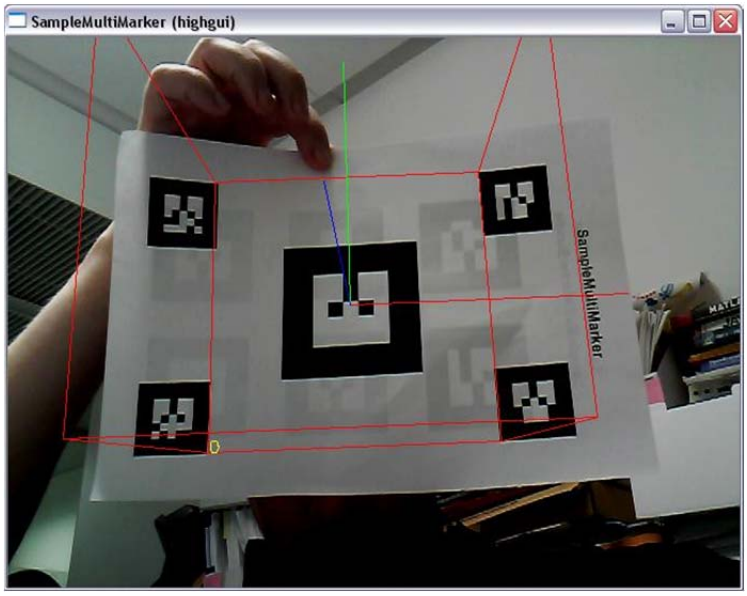

Figura 40.- Ejemplo de seguimiento basado en multimarcadores [11]

Alvar incorpora el seguimiento por marcadores simples –Figura 39– y multimarcadores –Figura 40–, pero también tiene incorporada la tecnología markerless, donde se emplea una clase propia de Alvar para entrenar un clasificador de Fern para realizar un seguimiento basado en imágenes en lugar de marcadores –Figura 41–.

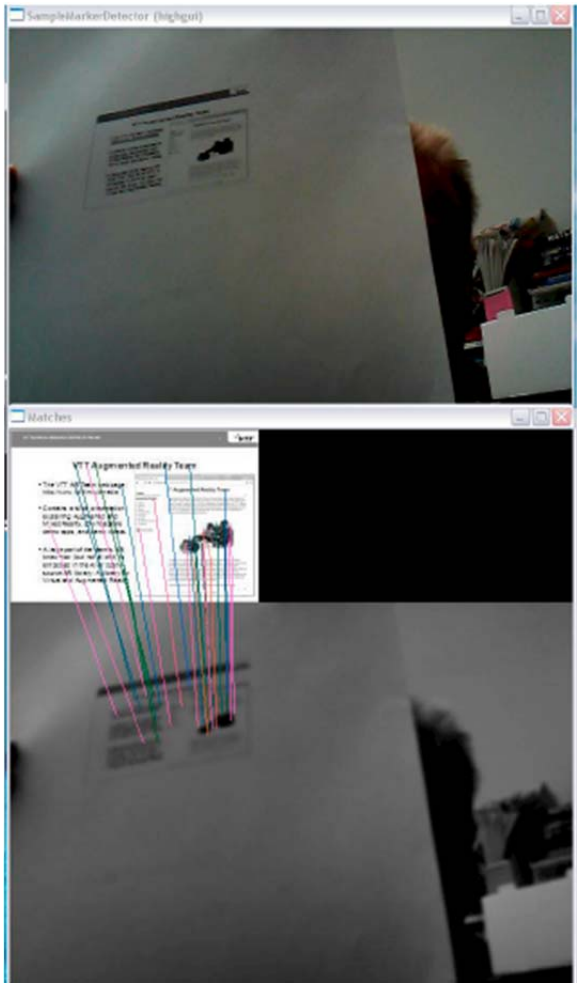

Figura 41.- Ejemplo de seguimiento de imágen como marcador (markerless) [11]

Otra característica importante de Alvar, es la posibilidad de ocultar los marcadores, y hacer una estimación del contenido situado justo detrás del marcador en base a lo que rodea al mismo. Podemos ver un par de ejemplos de su funcionamiento en la Figura 42.

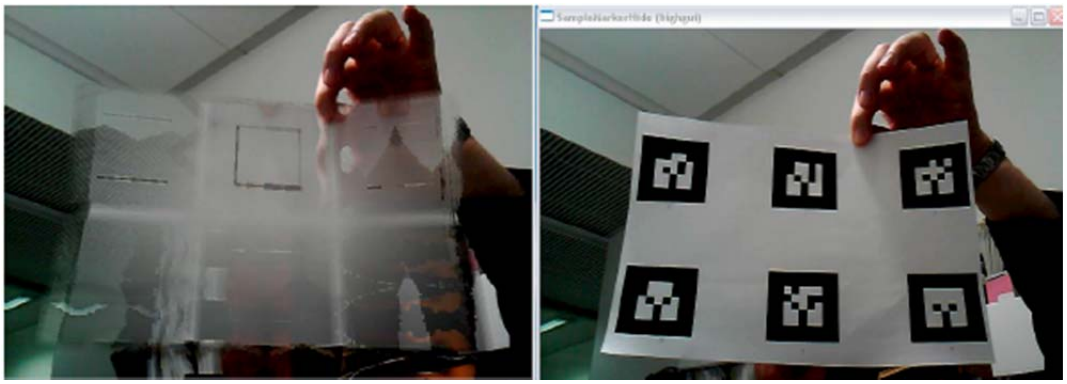

Figura 42.- Ejemplo de ocultación de marcadores [11]

Como podemos observar, la herramienta funciona relativamente bien, en función de las características del entorno. En el caso anterior, ocultar el marcador no nos serviría de mucha utilidad, pero, si observamos el ejemplo llevado a cabo en la Figura 43, los resultados son sorprendentes.

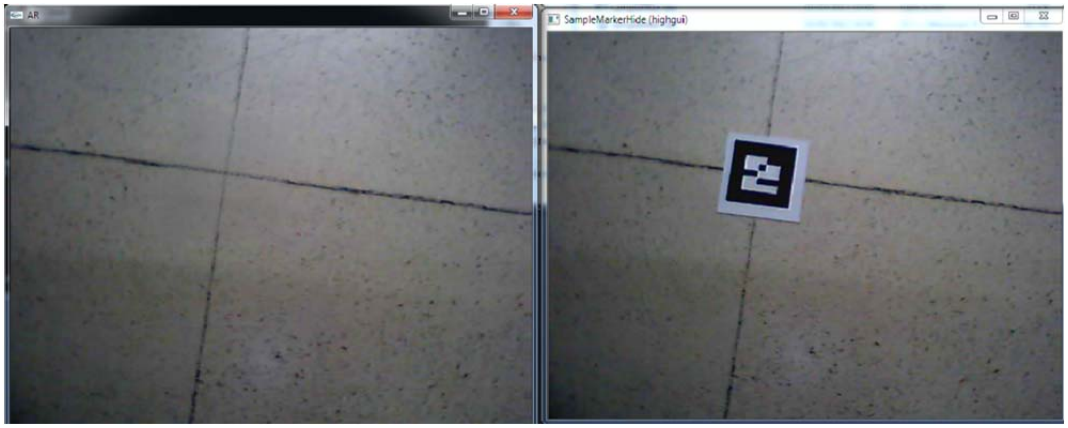

Figura 43.- Ejemplo de ocultación de marcadores

4.- Librería OpenCV

OpenCV (*Open Source Computer Vision Library*) es una librería de tratamiento de imágenes, para aplicaciones de visión artificial en tiempo real. Originalmente desarrollada por Intel en el año 1999, actualmente es Willow Garage [55] quien le da soporte. Se podrán cargar imágenes codificadas en casi todos los formatos, guardarlas a disco duro, modificarlas,…Con esta librería se puede trabajar tanto con imágenes estáticas como con flujos de imágenes

capturadas de una cámara. En la Figura 40 hemos visto cómo OpenCV se integra con Alvar para dibujar líneas, o detectar la cámara y permitir así la captura.

OpenCV tiene versiones para Windows, Linux y MacOSX. Cada una de las versiones cuenta con un módulo básico y varios módulos auxiliares. Una aproximación a su estructura básica se puede ver en la Figura 44, obtenida a partir de [32].

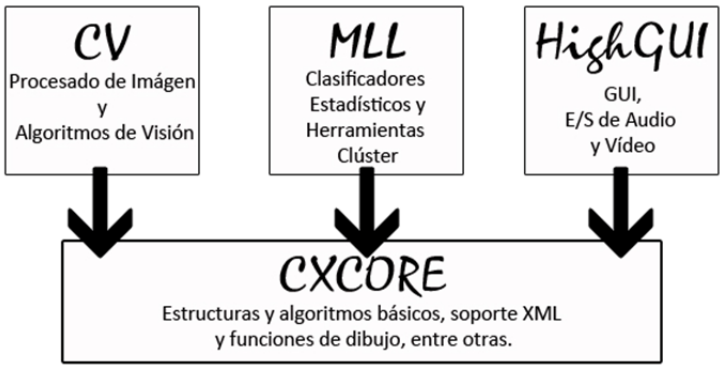

Figura 44.- Estructura básica de OpenCV

Aquí se darán unas nociones muy básicas de OpenCV, con el fin de que el lector tenga una visión general de la librería.

Se dividirá este apartado en otros sub apartados más pequeños para diferenciar las partes más importantes, como son los tipos de datos y las funciones más importantes de la librería.

4.1.- Tipos de datos en OpenCV

Aunque hay unos pocos tipos de datos en OpenCV, aquí solo se verán los tipos “CvMat” y “IplImage”, ambos datos estructura.

CvMat es un tipo de estructura con cabecera y datos. En el fondo se trata de una matriz de 1 a N dimensiones.

Por su parte, IplImage es una estructura derivaba de CvMat con algunos elementos más que hacen a una matriz ser capaz de representarse como una

imagen. Con éste tipo de dato se representan las imágenes sean del tipo que sean: ya sean en el modelo RGB, como en el BGR (es como el anterior lo único que se guarda en diferente orden), como en luminancia, como en los diferentes formatos de compresión de imágenes existentes.

En la Tabla 7 y la Tabla 8 pueden verse de forma resumida los componentes principales de las estructuras CvMat y IplImage, respectivamente.

Campo	Descripción
int type	Tipo de datos que contiene. En éste proyecto usaremos números en coma flotante de un solo canal, en OpenCV representados por CV_32FC1.
data	Contiene punteros a enteros, a coma flotante, etc., que son los datos que contiene la estructura.
int rows	Número de filas de los datos que contiene la estructura.
int cols	Número de columnas de los datos que contiene la estructura.

Tabla 7.- Campos de la cabecera de la estructura CvMat

Campo	Descripción
int nChannels	Proporciona el número de canales de la imagen.
int depth	Información de la profundidad, valor de los píxeles.
char colorModel[4]	Modelo utilizado en la imagen cargada.
char channelSeq[4]	Proporciona información sobre cómo están guardados los diferentes canales en el array, entrelazados o no.
int widthStep	Contiene el número de bytes que hay a mayores después de acabar la fila. Esta información de alineamiento se utiliza en algunos casos para obtener mayores velocidades de procesamiento.
int width	Anchura de la imagen y por tanto de los canales.
int height	Altura de la imagen y por tanto de los canales de la imagen.
char *imageData	Puntero a la imagen propiamente dicha.

Tabla 8.- Campos de la cabecera de la estructura IplImage

4.2.- Funciones de OpenCV

En la Tabla 9 se pueden ver algunas de las funciones principales de la librería OpenCV empleadas en alguna de las fases de desarrollo del sistema propuesto en este Trabajo Fin de Master.

Función	Descripción
cvNamedWindow(char name, int type)	Crea una ventana para el visualizado de las imágenes. Type a 1 para auto escala.
cvShowImage(char name, CvArr* img)	Representa la imagen indicada en la ventana correspondiente.
cvDestroyAllWindows()	Elimina todas las ventanas de visualizado que se hayan creado.
cvCreateImage(CvSize size, int depth, int channels)	Devuelve un puntero a una estructura IplImage, de tamaño size, profundidad depth y número de canales channels.
cvReleaseImage(& CvArr* img)	Libera espacio en memoria de una imagen no utilizada.
cvLoadImage(fileName,flag)	Carga la imagen para la posterior utilización de esta. Flag < 0 para cargar la imagen tal cual, con sus canales.
cvSaveImage(outFileName,img)	Salva imágenes a disco duro con el nombre de outFilename.
cvCreateMat(int rows, int cols, int type)	Devuelve un puntero a una estructura CvMat, inicializada con rows filas, cols columnas, y con datos del tipo type.
cvSetReal2D(CvArr* arr, int idx0, int idx1, double value)	Establece el valor value en la fila idx0 y columna idx1 de la matriz bidimensional arr.
cvGetReal2D(CvArr* arr, int idx0, int idx1)	Devuelve el valor de la fila idx0 y columna idx1 de la matriz bidimensional arr.

Tabla 9.- Algunos métodos de OpenCV

5.- Nokia Qt

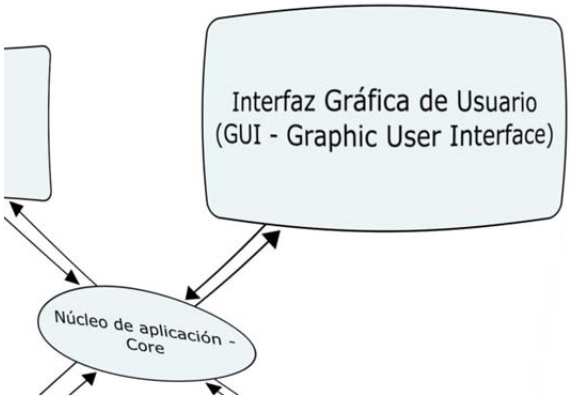

Figura 45.- Representación del módulo de Interfaz Gráfica de Usuario

Qt es un framework para la creación de interfaces gráficas de usuario y aplicaciones multiplataforma. Incluye una biblioteca de clases multiplataforma, herramientas de desarrollo integradas y un IDE multiplataforma –ver Figura 46–. Numerosas aplicaciones de escritorio como Skype, Google Earth o VLC utilizan Qt para la parte de interfaz gráfica de usuario.

Fue desarrollado inicialmente por Trolltech en 1992 siguiendo un desarrollo basado en el código abierto, pero no completamente libre. Usado activamente en el desarrollo del escritorio de GNU/Linux KDE (entre 1996 y 1998), en el año 2000, Trolltech comenzó a ofrecer la biblioteca Qt en su versión 2.1 bajo la licencia GPL en su versión para Linux, en 2003 para Mac OS X y en 2005 para Windows.

En 2008 Nokia adquirió Trolltech y, por tanto, Qt, hasta que, éste año 2012 Diga ha anunciado un acuerdo con Nokia para la adquisición de Qt.

Qt cuenta actualmente con una triple licencia [58]: GPL v2/v3 para el desarrollo de software de código abierto y libre, licencia de pago QPL para el desarrollo de aplicaciones comerciales, y una licencia gratuita para aplicaciones comerciales, LGPL.

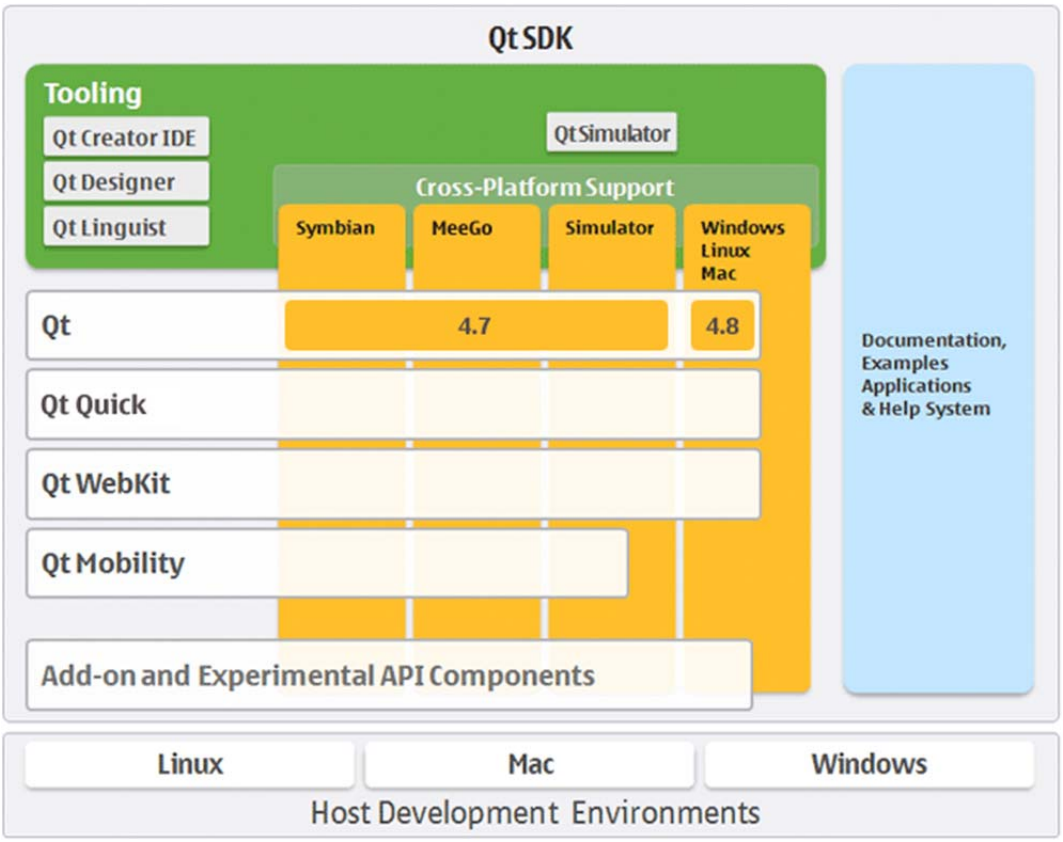

Figura 46.- Organización del SDK de Qt

El lenguaje nativo de Qt es C++, aunque los archivos de formulario –con extensión .ui– están escritos internamente en XML. Qt tiene varias opciones de descarga en función de la aplicación que se quiera dar a Qt. Para la realización de éste TFM tan sólo utilizatemos las clases precompiladas para VisualStudio 2008, un conector que integra perfectamente las librerías de Qt con VisualStudio, y el diseñador Qt Designer –Figura 47–, una ayuda imprescindible de cara a crear los interfaces gráficos de usuario. La última versión estable es la 4.8.2, pero en este desarrollo se ha empleado la versión 4.8.1, última versión estable disponible en el comienzo del desarrollo.

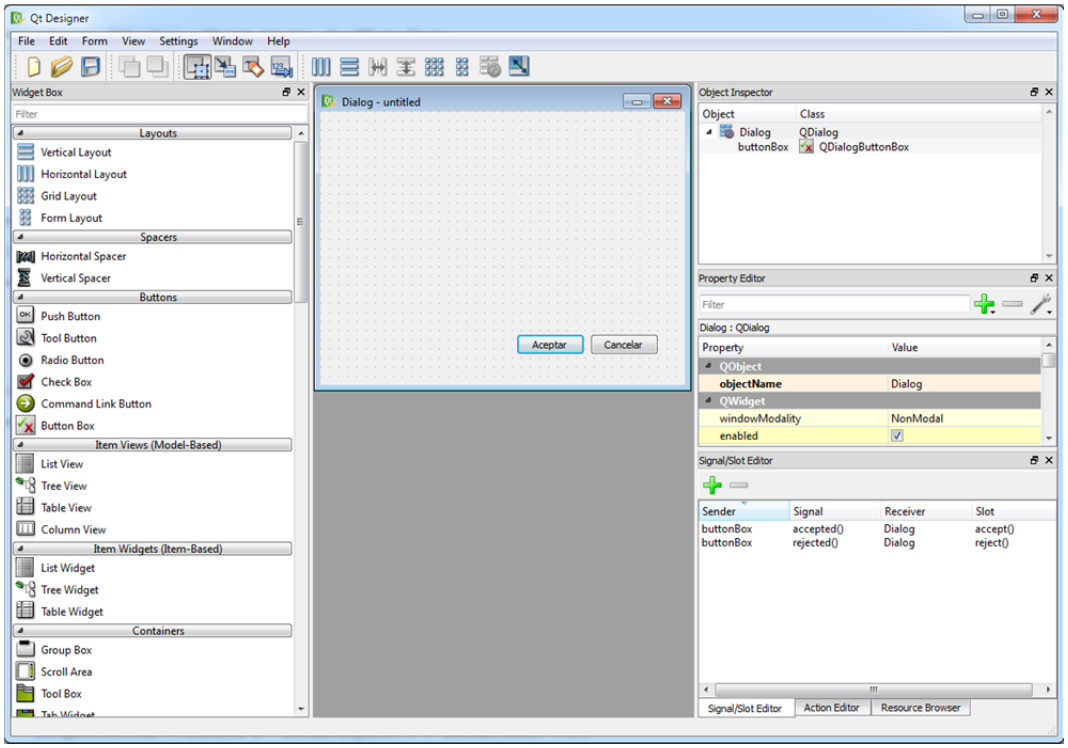

Figura 47.- Captura de pantalla de Qt Designer

6.- OpenSceneGraph

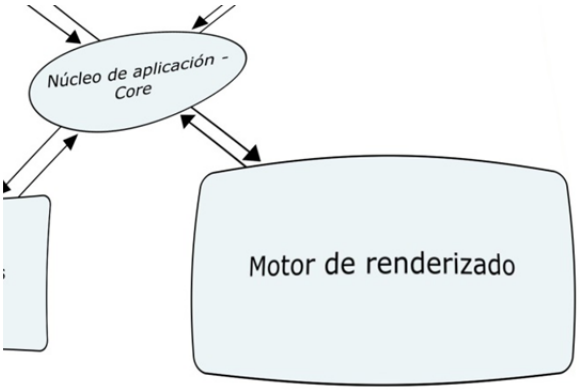

Figura 48.- Representación del módulo Motor de renderizado

OpenSceneGraph, o más conocido como OSG [57], es un conjunto de herramientas de alto rendimiento para gráficos en tres dimensiones, y de código abierto [57]. Su uso es muy extendido, desde simulación hasta modelado, pasando por juegos y un extenso número de campos más.

Figura 49.- Logotipo OpenSceneGraph [57]

Este motor gráfico, escrito completamente en C++ y OpenGL, funciona prácticamente en todos los sistemas operativos actuales. Está basado en el concepto de grafo de escena o SceneGraph [24] y se sitúa en lo alto de OpenGL. El grafo de escena principal encapsula la mayor parte de la funcionalidad de OpenGL, sus extensiones y un amplio abanico de optimizaciones. De éste modo, proporciona al desarrollador la libertad de trabajar tanto optmizando gráficos de bajo nivel, como a alto nivel para desarrollar aplicaciones rápidamente.

Una de sus máximas es el rendimiento, incorporando multitud de tipos de culling. El culling es una forma de extraer cierta parte de información de la escena. Si tenemos un escenario lo suficientemente grande y complejo, si nuestra aplicación intenta renderizar toda la escena durante todo el tiempo, el rendimiento se verá notablemente afectado. Sin embargo, si únicamente renderizamos la parte del escenario que se encuentra visible de cara al usuario, el rendimiento de nuestra aplicación aumentará notablemente. Otra característica relacionada con el rendimiento es la capacidad de establecer distintos niveles de detalle para cada nodo –Figura 50–, o para un mismo nodo pero variando en función de la distancia a la que se encuentre, la iluminación que tenga, si tiene sombras proyectadas sobre él, etc.

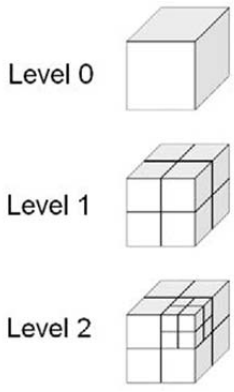

Figura 50.- Concepto de nivel de detalle o Level Of Detail [23]

Además, cuenta con compatibilidad con numerosos formatos de imágenes y modelos en tres dimensiones. Su funcionamiento se basa en plugins gestionados por la librería osgDB, la cual cuenta con más de 50 formatos soportados. Algunos de los formatos de modelos 3D soportados son LightWave (.lwo), Alias Wavefront (.obj), OpenFlight (.flt), TerraPage (.txp), Carbon Graphics GEO (.geo), 3D Studio MAX (.3ds), Peformer (.pfb), AutoCad (.dxf), Quake Character Models (.md2), Direct X (.x), Inventor Ascii 2.0 (.iv)/ VRML 1.0 (.wrl), Designer Workshop (.dw), AC3D (.ac) y su formato nativo, .osg ASCII. Entre los formatos de imágenes soportados destacan .rgb, .gif, .jpg, .png, .tiff, .pic, .bmp, .dds, .tga y quicktime (éste sólo bajo OSX). Por supuesto, también permite, a través de otro par de plugins, cargar fuentes personalizadas.

En la carrera por perseguir una amplia compatibilidad, y siguiendo una estructura modular, el núcleo de osg consta de cuatro librerías, descritas en la Tabla 10.

Función	Descripción
osg	Elementos básicos para construir una escena, como nodos, geometrías, estados de renderizado, texturas…Así como herramientas y métodos para trabajar con ellos. Además, cuenta con métodos matemáticos para trabajar con vectores y matrices.
OpenThreads	Proporciona una interfaz de hilos completa basada en orientación a objetos para programadores en C++.
osgDB	Destinada a las relaciones de entrada y salida, y a la carga de archivos 2D y 3D.
osgUtil	Es la base del renderizado de OSG. Convierte el grafo formado por OSG en unas series de llamadas a OpenGL. También proporciona algoritmos para la modificación de intersecciones y polígonos.

Tabla 10.- Núcleo de OSG

Además, cuenta con una serie de clases, módulos o utilidades adicionales, que se pueden cargar al compilar la aplicación o bien en tiepo de ejecución. Estos nodos aparecen presentados en la Tabla 11.

Función	Descripción
osgViewer	Define un conjunto de clases relacionadas con la visión. Contiene soporte para algunas clases externas de contrucción de GUIs
osgText	Para fuentes con anti-aliasing
osgFX	Framework para efectos especiales
osgGA	Ayuda en la captura de eventos desde dispositivos periféricos como teclado y ratón
osgShadow	Framework de sombras
osgParticle	Hace posible el renderizado de explosiones, fuego, humo y otros efectos basados en partículas
osgManipulator	Controles interactivos 3D
osgSim	Efectos visuales de simulación
osgTerrain	Renderizado de terrenos
osgAnimation	Animación de cuerpos
osgWidget	Extiende la funcionalidad del núcleo de OSG con un conjunto de widgets en 2D
osgQt	Conexión con Qt. Permite incrustar widgets de Qt en la escena
osgVolume	Renderizado de volúmenes

Tabla 11.- NodeKits de OpenSceneGraph [57]

El sistema de coordenadas empleado por el motor de gráficos OpenSceneGraph aparece representado en la Figura 51 junto al sistema de coordenadas empleado por el famoso software de modelado 3D Autodesk 3DS [50]. Si observamos la figura, podemos ver que la disposición de los ejes no coincide con la de los sistemas de coordenadas empleados por algunas de las soluciones software de modelado 3D más conocidas. Por tanto, hay que tener este dato en cuenta a la hora de diseñar aplicaciones con él, ya que un modelo, por ejemplo, de una botella llena, exportado desde un software de modelado con otro sistema de coordenadas en posición vertical, va a ser cargado en OSG en una posición completamente girada.

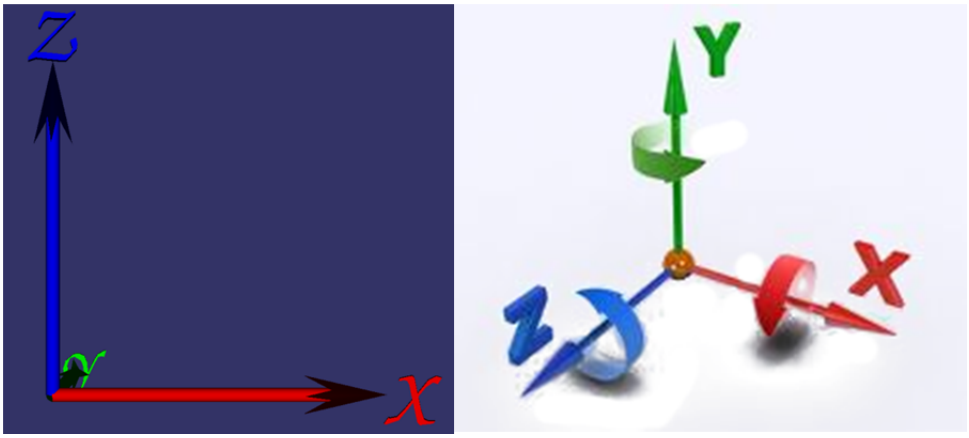

Figura 51.- Sistema de coordenadas de OSG (izquierda) frente al empleado por 3DS max (derecha)[19]

7.- Integración de tecnologías

Tras un studio detallado de las librerías y el hardware necesario, podemos seleccionar éstas candidatas como las tecnologías a emplear por varias razones:

- En primer lugar todas permiten su uso sin ningún coste alguno, por lo que concuerda perfectamente con nuestro presupuesto nulo.
- Alvar, la librería que se encarga de la detección de los marcadores, viene perfectamente integrada con OpenCV, es más, la necesita para poder funcionar.
- Por su parte OpenCV tiene un módulo –highgui– que cuenta con un plugin para detectar cámaras e iniciar la captura de imágenes. Tras probar las cámaras del HMD de Vuzix, se comprobó que eran reconocidas sin problemas por éste módulo de OpenCV.
- Siguiendo con Alvar, en su kit de desarrollo encontramos ejemplos que funcionan con OpenSceneGraph (OSG), por lo que la parte de integración de la Realidad Aumentada con el motor gráfico está solventada.
- Por su parte, OSG cuenta entre sus NodeKits –Tabla 11– con la librería osgQt, la cual permite conectar OSG con Qt, incrustando widgets de Qt en renderizados de OSG.

Por tanto, no es de extrañar que de una forma relativamente fácil, todas estas tecnologías se puedan integrar en un primer prototipo, únicamente destinado a dicha integración de tecnologías.

En la Figura 52 podemos ver la integración completada. La parte de interfaz gráfica con Qt contiene los controles para cargar y modificar la rotación, el tamaño y el desplazamiento del modelo, así como para lanzar la vista en Realidad Aumentada. Además, también integra cuatro vistas en las que está trabajando OSG, y sobre las que actúa.

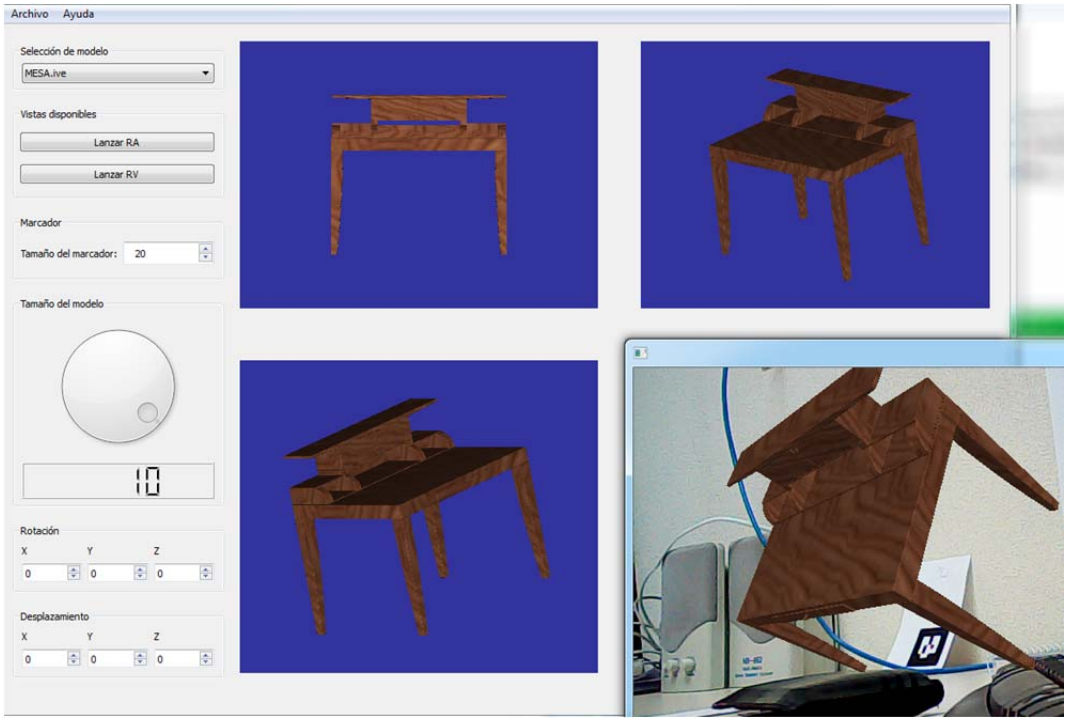

Figura 52.- Ejemplo de integración de todas las tecnologías propuestas

Por otra parte, observamos una ventana independiente que nos muestra la parte de Realidad Aumentada. En dicha ventana aparece el video que está siendo capturado por la cámara –gracias a OpenCV–. Además, podemos apreciar al fondo la existencia de un marcador, con la forma de la Figura 53. Este marcador es detectado por Alvar, el cual indica a OSG que debe renderizar, en base a ese marcador, el modelo de la mesa.

Figura 53.- Ejemplo de marcador utilizado por Alvar

Capítulo 4

Sistema Propuesto

Éste apartado nos mostrará el sistema planteado. Veremos el funcionamiento interno del núcleo del sistema, así como las distintas partes del interfaz gráfico. Además, veremos a fondo el filtrado que se ha creado específicamente para éste desarrollo.

1.- Introducción

Una vez hemos visto de forma general de qué consta el sistema a desarrollar, el estado del arte en lo que a Realidad Aumentada se refiere y la tecnología empleada, y hemos comprobado que es viable la realización del mismo, en este capítulo veremos el desarrollo del propio sistema, con una explicación más a fondo de las funcionalidades del programa.

El programa desarrollado para este Trabajo Fin de Master se puede dividir en cuatro bloques perfectamente diferenciados, más un quinto bloque que será el núcleo del programa.

Hay un primer bloque encargado de la detección de las cámaras disponibles en el sistema y la captura de imágenes en movimiento de una de ellas. Existe un segundo bloque, muy relacionado con este primero, encargado de detectar marcadores y, en base a ellos, mandar información a un tercer módulo de renderizado, encargado de representar la escena virtal sobre la vista de la cámara. El cuarto bloque sería la parte de interación con el usuario, donde se encuentra la interfaz gráfica, que permite hacer modificaciones sobre la escena virtual, así como lanzar y ocultar determinadas vistas.

El último bloque es el principal. El núcleo contiene toda la información de la escena que se está diseñando, sobre él actúa la interfaz de usuario para modificar dicha información, y a él le consultan los demás módulos cómo está la escena en ese momento.

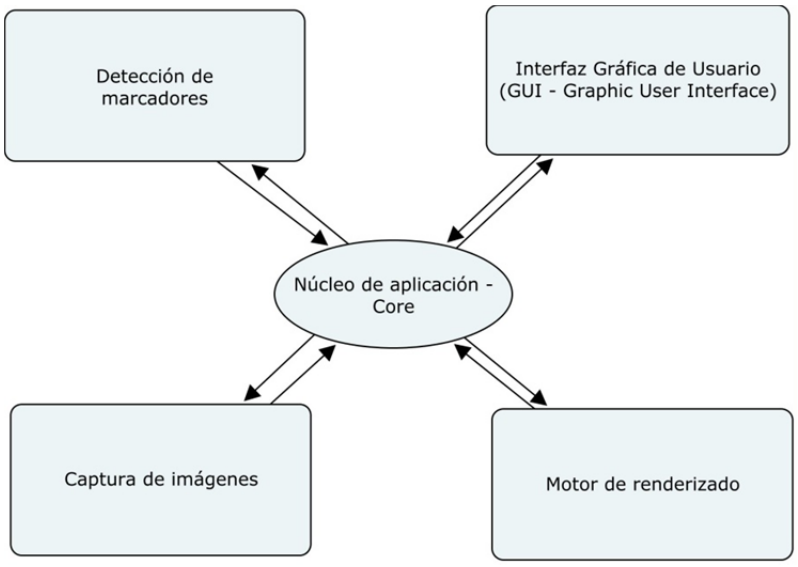

Figura 54.- Distribución de los módulos de la aplicación

2.- Funcionamiento básico del programa

Con el fin de realizar un programa de diseño de espacios en tres dimensiones, cuatro elementos básicos para el diseño son unas vistas de alzado, planta, perfil y perspectiva. Así, las tres primeras serán en dos dimensiones, mientras que la última tendrá vista en tres dimensiones. La disposición de éstas vistas será la habitual en los programas de diseño, dibujo, etc., que corresponde con el de la Figura 55. Dos vistas adicionales están disponibles a petición del usuario, como son la vista en Realidad Aumentada y en Realidad Virtual.

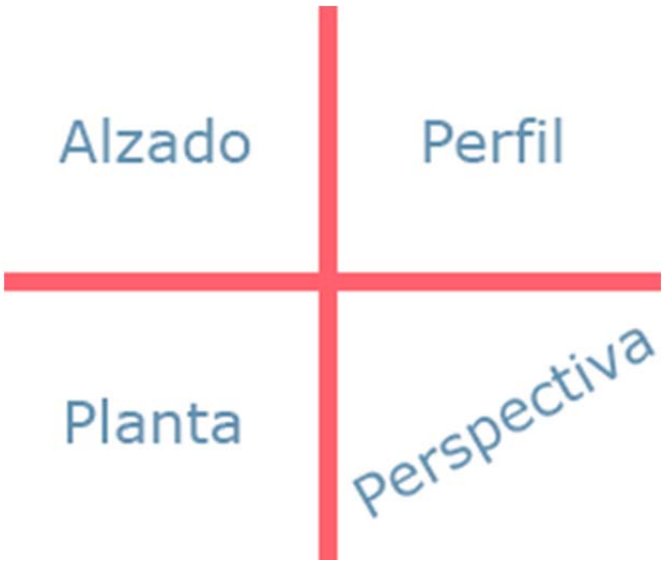

Figura 55.- Disposición de vistas en el entorno de diseño

Además debemos de poder añadir a la escena distintos elementos como modelos, marcadores, planos, luces o cámras, cada una con sus propiedades particulares. Todos los elementos, a excepción de las cámaras, son ocultables en la escena, pero conservables en el editor.

2.1.- Modelos

Los modelos representan las geometrías que introduciremos en la escena, tales como muebles, farolas, botellas, carteles, coches, etc. Las propiedades de un modelo vienen determinadas en la Tabla 12.

Propiedad	Descripción
Visible	Posibilidad de ocultar modelo de la escena en RA
Posición	Posición que ocupa en cada eje de coordenadas
Rotación	Rotación que tiene en cada eje de coordenadas
Escala	Tamaño del modelo
Marcas	Marcadores que tiene asociados el modelo
Imágenes	Imágenes asociadas al modelo
Texto	Texto asociado al modelo
Figura asociada	Geometría asignada al modelo

Tabla 12.- Propiedades de un modelo

Las imágenes y el texto serán opcionales ya que son a título descriptivo. Los marcadores serán independientes a los que veremos en el siguiente apartado 2.2, pues estos se podrán mover a lo largo de la escena, y situarlos exactamente donde queremos que aparezca el modelo. Además, la figura asociada se puede intercambiar por otra, conservando así el resto de propiedades que éste ya tuviera.

2.2.- Marcadores

Estos marcadores, a los que podemos llamar de referencia, representan la escena al completo, es decir, todo lo contenido en la escena que hayamos diseñado.

En este punto puede darse el problema de cómo actuar si observamos varios marcadores de referencia, o qué hacer si perdemos el marcador de referencia. Para solventar estos dos problemas, una vez se detecte el marcador y la escena se renderiza, el programa bloquea el escenario en ese marcador hasta que éste deja de ser visible. Si cuando esto ocurra hay otro marcador de referencia visible, la escena se renderiza en base a este marcador y el proceso se

repite. Si no hay ningún otro marcador de referencia visible, la posición de la escena se estima y se va variando en función de los movimientos efectuados con la cámara.

Tanto el paso de la escena de un marcador de referencia a otro como la reasignación de la escena a un marcador -tras desaparecer de la vista de la cámara todos los marcadores de referencia- se realizan de forma suave gracias al filtro que se ha implementado para tal fin.

Las propiedades de los marcadores de referencia las podemos ver en la siguiente tabla.

Propiedad	Descripción
Visible	Posibilidad de ocultar el marcador de la escena en RA
Posición	Posición del marcador en cada eje de coordenadas
Rotación	Rotación del marcador en cada eje de coordenadas
Identificador	Número que permite saber de qué marcador se trata

Tabla 13.- Propiedades de los marcadores de referencia

2.3.- Cámaras

No se trata de un elemento que aparezca de forma fija en la escena como los modelos y los marcadores, sino como capturas de la vista en perspectva. Una vez guardamos una cámara, ésta se puede recuperar en cualquier momento y ser visualizada en la vista en perspectiva. Las propiedades que tiene una cámara las podemos observar en la Tabla 14.

Propiedad	Descripción
Posición	Posición de la cámara en cada eje de coordenadas
Dirección	Dirección hacia dónde está orientada la cámara
Orientación	Indica hacia dónde está la vertical de la cámara

Tabla 14.- Propiedades de una cámara

2.4.- Luces

Las luces pueden ser de dos tipos, ambiental y direccional, y tener varias temperaturas de color. Pueden coexistir hasta ocho fuentes de luz de forma simultánea debido a una limitación de OpenSceneGraph. Sus propiedades pueden verse en la siguiente tabla.

Propiedad	Descripción
Visible	Posibilidad de ocultar la luz de la escena en RA
Tipo	Indican si la fuente de luz es ambiental o direccional
Intensidad	Entero entre 0 y 100 que especifica la fuerza de la luz
Temperatura	Indica el color de la luz
Posición	Posición de la luz en cada eje de coordenadas
Dirección (sólo en luces direccionales)	Indica la dirección hacia donde está dirigida la luz
Apertura (solo en luces direccionales)	Nos dice lo abierto que está el foco

Tabla 15.- Propiedades de las luces

2.5.- Planos

La utilidad de los planos viene a la hora de crear modelos como carreteras, paredes, techos, espejos, etc. En definitiva, superficies que sólo necesiten un espacio delimitado por tres o cuatro puntos en dos dimensiones y una textura, ahorrando así el coste de desarrollar un modelo en tres dimensiones con forma de carretera, pared, techo, etc.

Las propiedades de los planos aparecen en la siguiente tabla.

Propiedad	Descripción
Visible	Posibilidad de ocultar el plano en la escena en RA
Posición	Posición del plano en cada eje de coordenadas
Rotación	Rotación del plano en cada eje de coordenadas
Dimensiones	Anchura y altura del plano
Textura	Textura asignada al plano

Tabla 16.- Propiedades de un plano

3.- Lógica del núcleo

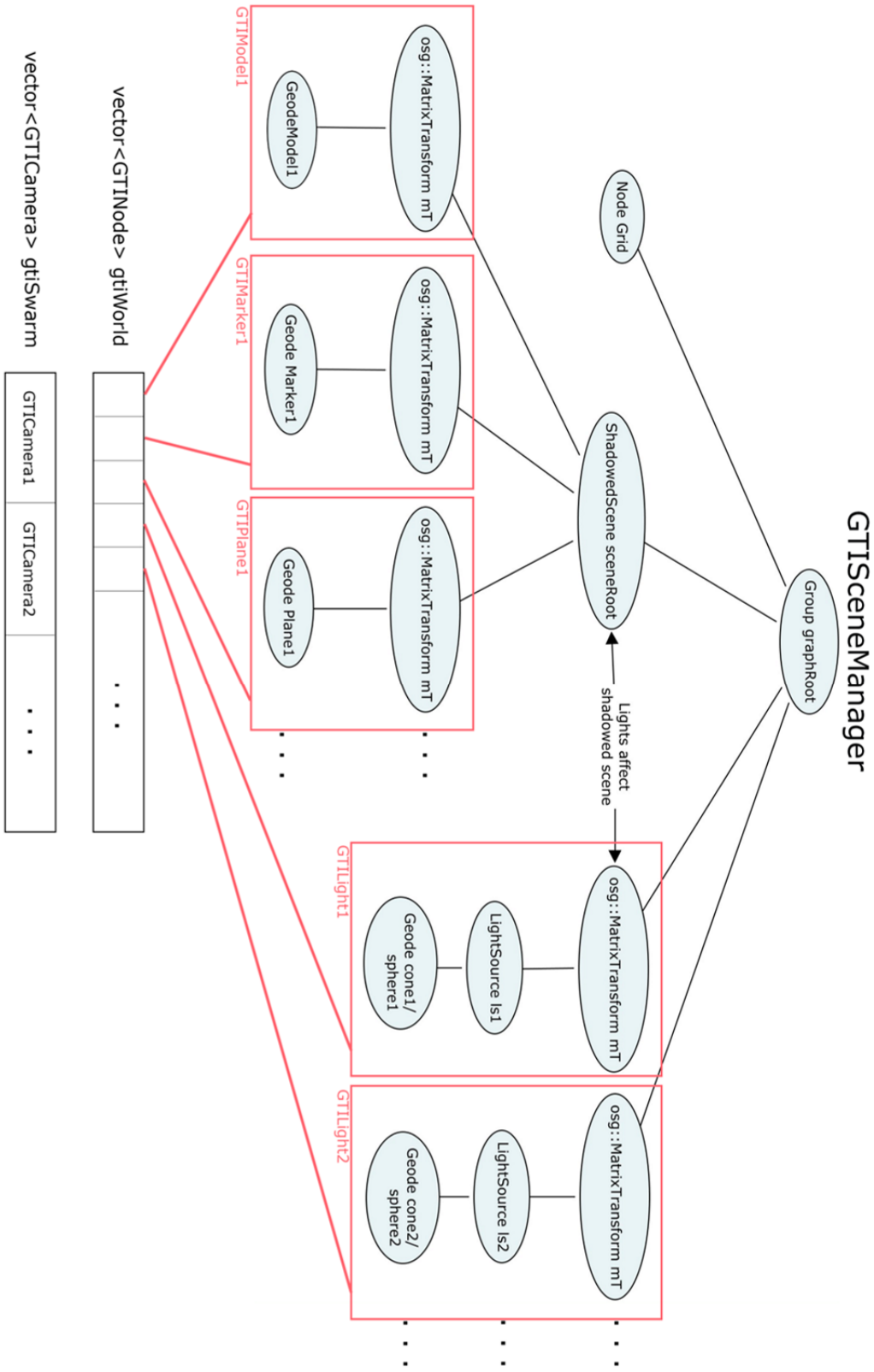

Figura 56.- Organización del núcleo del sistema propuesto

A la vista de la Figura 56, podemos describir la organización del núcleo con respecto a los elementos gráficos de la escena a diseñar, llamado GTISceneManager.

Todo el grafo de escena cuelga de un nodo raíz -graphRoot-, del que a su vez cuelgan una cantidad indefinida de nodos. Hay un nodo especial llamado Grid, encargado de mostrar una malla predeterminada a modo de referencia –como en la Figura 57– para empezar a construir el diseño particular. Este nodo no es editable con el fin de conservar su estado como referencia.

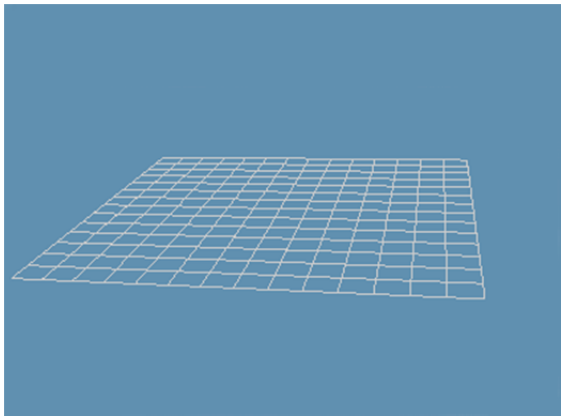

Figura 57.- Malla de referencia para el diseño

Después existen al mismo nivel que la malla de referencia, un nodo padre denominado sceneRoot. De él colgarán todos los modelos, los marcadores y los planos, todos con la misma estructura, primero la matriz de transformación mT y después la geometría en sí. La matriz mT es la encargada de cambiar la posición, la rotación y el escalado de la geometría.

Al mismo nivel que el nodo llamado sceneRoot se sitúan las distintas luces que se vayan creando, primero su matriz de transformación, después la fuente de luz, y por último una geometría –para identificar la luz en la escena– que varía en función del tipo de luz –una esfera para las ambientales y un cono para las direccionales–. Las luces se disponen así en el grafo para que afecten al nodo padre del resto de nodos, es decir, el nodo sceneRoot.

Por último, todas las partes del grafo de la Figura 56 que están enmarcado en rojo se almacenan, con sus propiedades, en el vector de GTINodes gtiWorld. Un GTINode es la clase de la que heredan los modelos, los planos, los marcadores de referencia y las luces, por tener propiedades similares. Por su parte, las cámaras se almacenan en el vector de GTICameras gtiSwarm.

La interfaz se encarga de manejar estos dos vectores, y los cambios se ven reflejados en tiempo real tanto en el editor, como en las vistas de Realidad Aumentada como Realidad Virtual.

4.- Filtrado de modelos

Puesto que las primeras pruebas realizadas con Realidad Aumentada mostraban cierta imperfección en la actualización de la posición de los modelos con respecto a los marcadores, se pensó en instalar un filtro que nos proporcionara estabilidad en visiones estáticas, así como ante los cambios de posición.

El filtro de Kalman [29] estima un sistema lineal cuyo modelo se quiere conocer y que además tiene ruido. Con este filtro se puede llegar a hacer una aproximación del estado del sistema a partir de una cierta cantidad de información del modelo. Éste filtro ha aparecido en multitud de investigaciones en áreas como la navegación marítima, la energía nuclear o la visión artificial [28]. La variante del filtro de Kalman que linealiza en torno a la media y a la covarianza actual se conoce como Filtro de Kalman Extendido.

El filtro de Kalman extendido desacoplado [26][27] se basa en el filtro de Kalman extendido para considerar el entrenamiento como un problema de filtrado óptimo en el que se encuentra recursivamente una solución al problema de los mínimos cuadrados. Es decir, se busca la curva que mejor aproxima un conjunto de datos de manera que se minimice la distancia media entre los datos y la curva. En el proceso original de mínimos cuadrados, es necesario utilizar toda la información suministrada hasta el instante actual. Sin embargo, con esta variante del filtro de kalman solo es necesario almacenar los resultados de la última iteración, de modo que la carga computacional en cada iteración es la misma.

2.1.- El filtro de Kalman

El filtro de Kalman [29] intenta estimar el estado w$[t] \in R^n$de un sistema dinámico lineal de tiempo discreto que está gobernado por la siguiente ecuación:

$$w[t\ +\ \ 1] = Aw[t] + Bu[t] + \omega[t] \tag{4.1}$$

Siendo $u\ [t]$ la entrada del sistema, con una medida d$[t] \in R^m$ de la forma:

$$d\ [t\ +\ \ 1] = Hw[t] + v[t] \tag{4.2}$$

Donde *A*, *B* y *H* son conocidas. Las variables aleatorias $\omega[t]$ y $v[t]$ representan el ruido del proceso y de la medición, respectivamente. Se asume que se trata de ruido blanco de media cero y con matrices de covarianza diagonales $Q[t]$ y $R[t]$:

$$\langle \omega[t]\omega^T[t] \rangle = Q[t] \tag{4.3}$$

$$\langle v[t]v^T[t] \rangle = R[t] \tag{4.4}$$

En cada paso, el filtro proyecta la estimación del estado actual y de la covarianza actual hacia adelante en el tiempo. De esta forma se obtiene una estimación a priori para el siguiente paso. Después, utiliza los resultados de la medición real para mejorar esta estimación y obtener una estimación a posteriori. A este proceso se le puede ver como un ciclo de predicción y corrección.

Sea $\hat{\mathrm{w}}^-[t]$ la estimación a priori del estado en el instante *t*, a partir del conocimiento anterior al paso *t*:

$$\hat{w}^-[t] = A\hat{w}\ [t-1\ ] + Bu\ [t-1] \tag{4.5}$$

La estimación a posteriori del estado, $\hat{\mathrm{w}}[t]$, se obtiene como una combinación lineal de la estimación a priori $\hat{\mathrm{w}}^-[t]$, y la diferencia ponderada entre la medición real $d[t]$ y una predicción de la medidad $H\hat{\mathrm{w}}^-[t]$.

$$\hat{w}[t] =\ \hat{w}^-[t] + K[t](d[t] - H\hat{w}^-[t]) \tag{4.6}$$

La expresión $(d[t]H\widehat{w}^{-}[t])$ se denomina residuo o innovación de la medida y refleja la discrepancia entre la medición predicha y la real.

Consideremos ahora los errores de la estimación a priori y de la estimación a posteriori:

$$e^{-}[t] = w[t] - \widehat{w}^{-}[t] \quad (4.7)$$
$$e[t] = w[t] - \widehat{w}[t] \quad (4.8)$$

La covarianza a priori del error de estimación es:

$$\langle e^{-}[t]\,(e^{-}[t]\,)^{T}\rangle = P^{-}[t] \quad (4.9)$$

Y a posteriori:

$$\langle e[t](e[t])^{T}\,\rangle = P\,[t] \quad (4.10)$$

La matriz de ganancia K se elige de manera que se minimice la covarianza del error a posteriori (4.10). Una posibilidad es:

$$K[t] = P^{-}[t]H^{T}(HP^{-}[t]H^{T} + R[t])^{-1} \quad (4.11)$$

Debido al ruido, la covarianza del error a priori y a posteriori de la ecuación 4.5 se calcula, respectivamente, de la siguiente forma:

$$P^{-}[t] = AP[t-1]A^{T} + Q[t] \quad (4.12)$$
$$P[t] = (I - K\,[t]H)\,P^{-}\,[t] \quad (4.13)$$

Un ciclo del algoritmo consiste en evaluar las ecuaciones (4.5), (4.12), (4.11), (4.6) y (4.13). La naturaleza recursiva del filtro hace que la estimación del estado del sistema esté en función de todas las mediciones del pasado pero sin tenerlas que considerar explícitamente. Podemos resumir los pasos para realizar un filtrado de kalman de la siguiente manera:

- Cálculo de la estimación a priori (4.5).
- Cálculo de la covarianza del error a priori (4.12).
- Cálculo de la matriz de ganancia (4.11).
- Cálculo de la estimación a posteriori (4.6).
- Cálculo de la covarianza del error a posteriori (4.13).

El rendimiento del filtro puede mejorarse mediante el control de las matrices $Q\ [t]$ y $R[t]$. Estas matrices pueden fijarse antes del funcionamiento del filtro o pueden ir cambiándose dinámicamente. Así, $R[t]$ tendrá que ser ajustada en función de nuestra confianza en el mecanismo responsable de la medicion. Por otra parte, con $Q\ [t]$ se puede modelizar nuestra incertidumbre en el modelo (4.1). A veces, un modelo aproximado o incluso alejado del real puede ser util si se introduce suficiente ruido en la matriz $Q[t]$.

En la versión de Alvar [56] que hemos empleado existe un filtro de Kalman intuitivo a lo largo de una serie de funciones, permitiendo en pocos pasos la modificación de las matrices $Q[t]$ y $R[t]$. La función que configura los parámetros del filtro se muestra a continuación. Además, podemos ver su funcionamiento en la Figura 58.

```
void filter_kalman(double x, double y, double *fx, double
*fy)
{
	static bool init=true;
	static KalmanSensor sensor(4,2);
	static Kalman kalman(4); // x, y, dx, dy
	if (init)
	{
		init = false;
		cvZero(sensor.H); // H
		cvmSet(sensor.H, 0, 0, 1);
		cvmSet(sensor.H, 1, 1, 1);
		cvSetIdentity(sensor.R, cvScalar(10)); // R
		cvSetIdentity(kalman.F); // F
		cvmSet(kalman.F, 0, 2, 1);
		cvmSet(kalman.F, 1, 3, 1);
		cvmSet(kalman.Q, 0, 0, 0.0001); // Q
		cvmSet(kalman.Q, 1, 1, 0.0001);
		cvmSet(kalman.Q, 2, 2, 0.000001);
		cvmSet(kalman.Q, 3, 3, 0.000001);
		cvSetIdentity(kalman.P, cvScalar(100)); // P
	}
	cvmSet(sensor.z, 0, 0, x);
	cvmSet(sensor.z, 1, 0, y);
	kalman.predict_update(&sensor, GetTickCount());
	*fx = cvmGet(kalman.x, 0, 0);
	*fy = cvmGet(kalman.x, 1, 0);
}
```

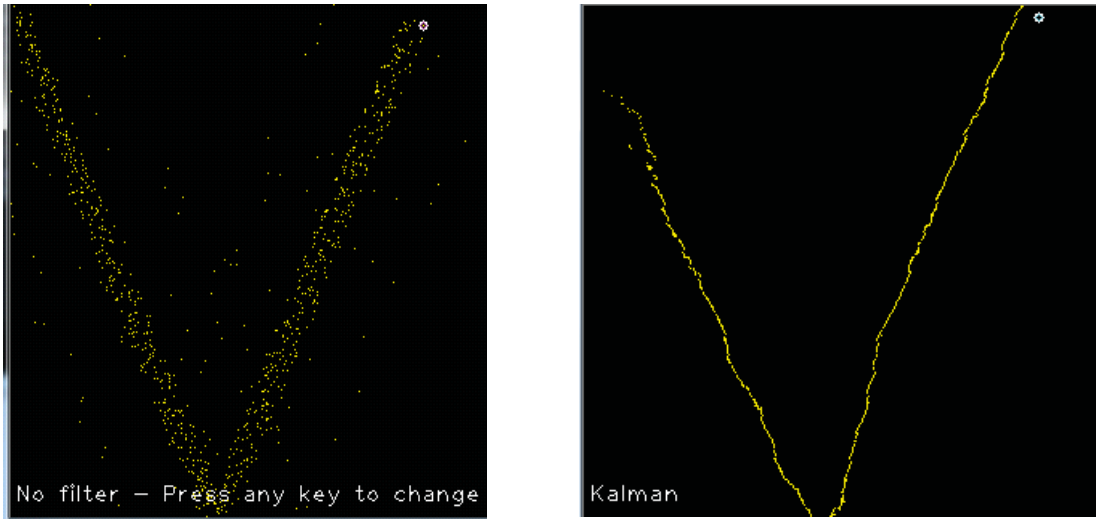

Figura 58.- Señal sin filtrar (izquierda) frente a señal filtrada (derecha)

2.2.- Filtro de Kalman Extendido

El filtro de Kalman extendido (Extended Kalman Filter, EKF) es una adaptación del filtro de Kalman que permite trabajar sobre sistemas no lineales. Esta variante linealiza en torno a la media y a la covarianza actual.

En este caso, en vez de encontrarnos las matrices *A*, *B* y *H*, tenemos las funciones $f(\mathrm{w}[\mathrm{t}], \mathrm{u}\,[\mathrm{t}])$ y $\mathrm{h}(\mathrm{w}[\mathrm{t}])$ que entregan la transición de estado y la observación, respectivamente. A partir de estas funciones, el vector de estado $\mathrm{w}[t] \in R^n$ que se pretende estimar tiene el siguiente aspecto:

$$w[t+1] = f\,(w[t], u\,[t]) + \omega[t] \qquad (4.14)$$

Con una medida $\mathrm{d}[t] \in R^m$ de la forma:

$$d[t] = h(\omega[t]) + u[t] \qquad (4.15)$$

donde las variables aleatorias $\omega[t]$ y $u[t]$ representan el ruido de media cero del proceso y de la medida, respectivamente (con matrices de covarianza $Q[t]$ y $R[t]$). Las funciones f y h son funciones no lineales que relacionan el estado en el instante t con el estado en el instante $t+1$ y con la medición $d[t]$, respectivamente.

Mediante la linealización de las ecuaciones de estado y de medición se llega a unas ecuaciones equivalentes a las del caso lineal [25]. Así, la estimación a priori del estado (4.5) se aproxima de la forma:

$$\widehat{w}^{-}[t] = f\ (\widehat{w}\ [t-1],\ u[t-1]) \tag{4.16}$$

y la covarianza del error a priori (4.12) se calcula con

$$P^{-}[t] = A[t-1]\ P\ [t-1](A[t\ -1]\ )^{T}\ + W[t-1]Q[t-1]\ (W[t-1])^{T} \tag{4.17}$$

Donde $A\ y\ W$ son matrices nuevas. La matriz A se define como la matriz jacobiana de f respecto al estado:

$$A[t] = \begin{bmatrix} \frac{\partial f_1}{\partial w_1}[t] & \frac{\partial f_1}{\partial w_2}[t] & \cdots & \frac{\partial f_1}{\partial w_n}[t] \\ \frac{\partial f_2}{\partial w_1}[t] & \frac{\partial f_2}{\partial w_2}[t] & \cdots & \frac{\partial f_2}{\partial w_n}[t] \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial w_1}[t] & \frac{\partial f_n}{\partial w_2}[t] & \dots & \frac{\partial f_n}{\partial w_n}[t] \end{bmatrix} \tag{4.18}$$

Por otra parte, W es la matriz de derivadas parciales de f respecto al ruido ω:

$$W[t] = \begin{bmatrix} \frac{\partial f_1}{\partial \omega_1}[t] & \frac{\partial f_1}{\partial \omega_2}[t] & \cdots & \frac{\partial f_1}{\partial \omega_n}[t] \\ \frac{\partial f_2}{\partial \omega_1}[t] & \frac{\partial f_2}{\partial \omega_2}[t] & \cdots & \frac{\partial f_2}{\partial \omega_n}[t] \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial \omega_1}[t] & \frac{\partial f_n}{\partial \omega_2}[t] & \dots & \frac{\partial f_n}{\partial \omega_n}[t] \end{bmatrix} \tag{4.19}$$

En esta ocasió n, la matriz de ganancia (4.11) se obtiene para el Filtro de Kalman Extendido de la siguiente forma:

$$K[t] = P^{-}[t]\,(H[t])^{T}\,(H\,[t]\,P^{-}[t](H[t])^{T} + \;V[t]\,R\,[t]\,(V[t])^{T})^{-1} \qquad (4.20)$$

Siendo H el jacobiano de las derivadas parciales de h respecto al estado:

$$H[t] = \begin{bmatrix} \frac{\partial h_1}{\partial w_1}[t] & \frac{\partial h_1}{\partial w_2}[t] & \cdots & \frac{\partial h_1}{\partial w_n}[t] \\ \frac{\partial h_2}{\partial w_1}[t] & \frac{\partial h_2}{\partial w_2}[t] & \cdots & \frac{\partial h_2}{\partial w_n}[t] \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial w_1}[t] & \frac{\partial h_m}{\partial w_2}[t] & \dots & \frac{\partial h_m}{\partial w_n}[t] \end{bmatrix} \qquad (4.21)$$

Y V la matriz de derivadas parciales de f respecto a v:

$$V[t] = \begin{bmatrix} \frac{\partial h_1}{\partial v_1}[t] & \frac{\partial h_1}{\partial v_2}[t] & \cdots & \frac{\partial h_1}{\partial v_m}[t] \\ \frac{\partial h_2}{\partial v_1}[t] & \frac{\partial h_2}{\partial v_2}[t] & \cdots & \frac{\partial h_2}{\partial v_m}[t] \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_m}{\partial v_1}[t] & \frac{\partial h_m}{\partial v_2}[t] & \dots & \frac{\partial h_m}{\partial v_m}[t] \end{bmatrix} \qquad (4.22)$$

La predicción a posteriori, al igual que en 4.6, emplea K con el fin de ponderar la diferencia entre la medición real y una predicción de la medida:

$$\widehat{w}[t] = \widehat{w}^{-}[t] + K[t]\left(d[t] - h(\widehat{w}^{-}[t])\right) \qquad (4.23)$$

Para terminar, la covarianza del error tiene una forma similar a la del caso lineal visto en (4.13), aunque debe tenerse en cuenta que ahora $H[t]$ se calcula de manera diferente al caso lineal:

$$P[t] = (I - K[t]H[t])P^{-}[t] \qquad (4.24)$$

Los pasos básicos para realizar el filtrado con el EKF se puede resumir de la siguiente forma: En primer lugar, se proyectan las estimaciones del estado y de la covarianza del error del instante t al $t+1$ haciendo uso de las ecuaciones (4.16) y (4.17).

Después, se utilizan estas estimaciones a priori para obtener otras corregidas debo a la consideración de $d[t]$. Las ecuaciones (4.20), (4.23) y (4.24) nos dan las estimaciones a posteriori del estado y de la covarianza del error. Al igual que ocurria en el filtro de Kalman, $R[t]$ y $Q[t]$ son parámetros ajustables del algoritmo.

De la misma forma que ocurria en el caso anterior, Alvar [56] proporciona funciones que permiten realizar este filtrado sobre señales. Sin embargo, el filtro incorporado sólo funciona con dos señales de forma simultánea. Como en éste TFM se trabaja con objetos en tres dimensiones, tenemos en cada instante tres datos y, por tanto, necesitamos que el filtro procese una señal más. Se modificó así el filtro proporcionado, quedando el código que a continuación se muestra.

```
void filter_ekf(double x, double y,double z, double *fx,
double *fy, double *fz)
{
	static bool init=true;
	static KalmanSensorOwn sensor(6,3);
	static KalmanOwn kalman(6); // x, y, z, dx, dy, dz
	if (init)
	{
		init = false;
		cvSetIdentity(sensor.R, cvScalar(100)); // R
		cvmSet(kalman.Q, 0, 0, 0.001); // Q
		cvmSet(kalman.Q, 1, 1, 0.001);
		cvmSet(kalman.Q, 2, 2, 0.001);
		cvmSet(kalman.Q, 3, 3, 0.01);
		cvmSet(kalman.Q, 4, 4, 0.01);
		cvmSet(kalman.Q, 5, 5, 0.01);
		cvSetIdentity(kalman.P, cvScalar(100)); // P
	}
	cvmSet(sensor.z, 0, 0, x);
	cvmSet(sensor.z, 1, 0, y);
	cvmSet(sensor.z, 2, 0, z);
	kalman.predict_update(&sensor, GetTickCount());
	*fx = cvmGet(kalman.x, 0, 0);
	*fy = cvmGet(kalman.x, 1, 0);
	*fz = cvmGet(kalman.x, 2, 0);
}
```

De esta forma, los movimientos de los elementos virtuales ante cambios de perspectiva se realizan de manera suave sin que el usuario se percate de transiciones bruscas entre diferentes renderizados de la escena. El resultado de

aplicar el filtro sobre dos señales puede verse en la Figura 59, donde se aprecia la diferencia de no usar ningún filtro a usarlo.

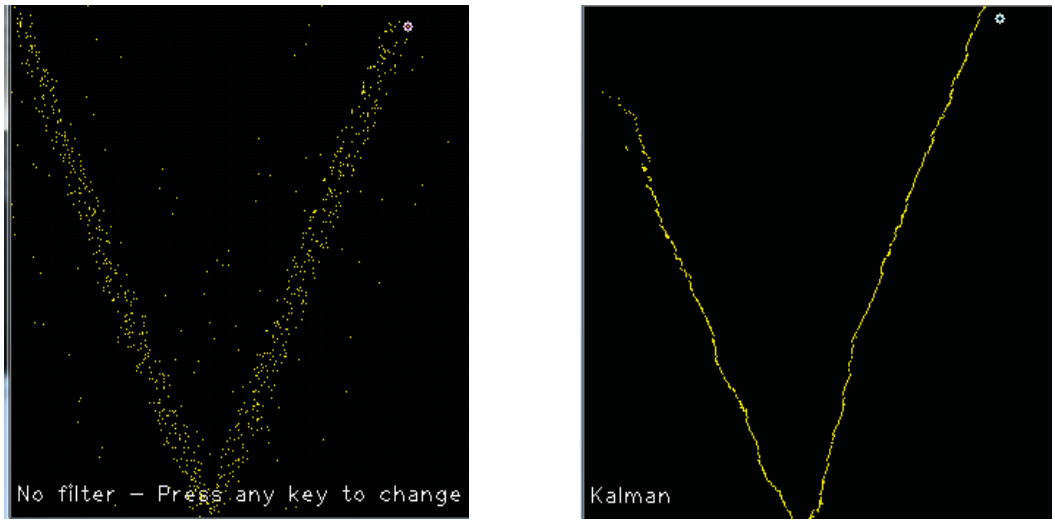

Figura 59.- Señal sin filtrar (izquierda) frente a señal filtrada con Filtro de Kalman Extendido (derecha)

Además, si comparamos el funcionamiento de los dos filtros, observamos como el Filtro de Kalman Extendido –Figura 60 derecha– funciona mejor que el Filtro de Kalman –Figura 60 izquierda–.

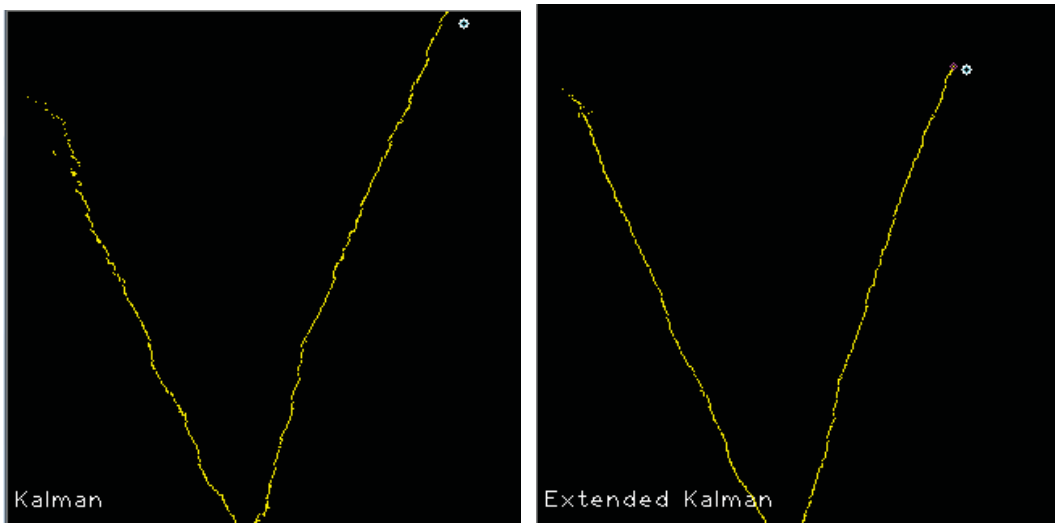

Figura 60.- Señal filtrada con el Filtro de Kalman (izquierda) y el Filtro de Kalman Extendido (derecha)

Capítulo 5

Conclusiones y Líneas Futuras

En este capítulo final se abordarán las principales conclusiones extraídas del presente Trabajo Fin de Master, las posibles mejoras sobre el mismo, en cuanto a optimización, ampliación y actualización, y algunas de las potenciales líneas futuras de investigación a seguir a raíz de este trabajo.

1.- Conclusiones

Las tecnologías evolucionan muy rápidamente y junto a ellas, las soluciones de código abierto o de uso gratuito. Así, aprovechándonos de ellas, podemos combinarlas para crear sistemas que ayuden en el día a día de las personas, ya sea ahorrando gastos, facilitando determinadas tareas o bien sirvan de entretenimiento.

Haciendo uso de éstas tecnologías, hemos logrado cumplir todos los objetivos de éste Trabajo Fin de Master, cuyo objetivo principal era el desarrollo de un diseñador de espacios 3D con realidad aumentada, realidad virtual, y edición online del entorno. Para llegar a cumplir dicho objetivo, se creyó conveniente añadir al objetivo global una serie de objetivos parciales:

La investigación de nuevas técnicas de detección y reconocimiento de marcadores, conociendo el tipo de marcadores necesarios y el hardware compatible.

Para responder a este objetivo parcial se han revisado las distintas técnicas de detección y reconocimiento de marcadores, dando un repaso a sistemas tan novedosos como la ocultación de marcadores o el sistema markerless, capaz de, a través de un algoritmo, extraer puntos característicos de imágenes y utilizarlos en lugar de los marcadores “convencionales”.

Especificación de características básicas a cumplir por la interfaz del programa, atendiendo a las máximas de facilidad de uso, simplicidad e inmersión del usuario en el diseño generado.

Para dar solución a este objetivo parcial nos hemos guiado por las máximas especificadas en el mismo, y hemos desarrollado a lo largo del Capítulo 4 las propiedades básicas disponibles para cada elemento de la escena, así como la disposición y funcionamiento de las vistas de las que se compone la interfaz gráfica de usuario.

Recogida de datos provenientes de la cámara integrada en las gafas Vuzix mediante las librerías Alvar y OpenCV (Open Source Computer Vision).

Una vez comprobado que la librería OpenCV, y, más concretamente, su módulo highgui, reconocía sin problemas las cámaras integradas en el HMD Vuzix, se comprobó como Alvar era capaz de, a través de ese módulo, abrir una ventana emergente con video procedente de una de las cámaras.

Implementación de un sistema de almacenamiento temporal que permita la edición online del sistema diseñado, que afecte a todos los ámbitos de la aplicación –vistas de diseño, vista de realidad virtual y vista de realidad aumentada-.

Ante este objetivo surgió la necesidad de crear un núcleo para la aplicación capaz de gestionar la integración de los distintos módulos de los que se compone la aplicación. A lo largo del apartado 3 del Capítulo 4 podemos observar cómo se organiza, las partes de las que se compone, y cómo interactúan unas con otras-

Realización de la interfaz gráfica de usuario en lo alto de las tecnologías subyacentes, cerrando así la aplicación. Se han de integrar todas las soluciones tecnológicas y permitir al usuario el diseño de espacios en tres dimensiones con facilidad.

A lo largo de este TFM se lleva a cabo una integración de tecnologías de diversa índole, de forma que el producto final resulta ser una aplicación sencilla, fácil de utilizar y muy útil, que abstrae al usuario de toda la tecnología subyacente, y con una interfaz gráfica de usuario básica pero que cumple a la perfección con su cometido.

La cantidad de aplicaciones posibles para las que el sistema desarrollado puede suponer una mejoría es muy amplia, pues el diseño de espacios en tres dimensiones abarca prácticamente todas las áreas que podamos imaginarnos. Podemos emplearlo para realizar visitas a museos, diseñar un parque, realizar maquetas arquitectónicas, entornos de trabajo, escenarios de entrenamiento, o diseñar interiores, todo ello con la ventaja de poder ver cómo quedará todo in situ con un coste prácticamente nulo.

2.- Líneas futuras

Dadas las posibilidades que ofrecen dos tecnologías como son la Realidad Aumentada y el renderizado de gráficos, las mejoras que se pueden insertar en el sistema desarrollado son prácticamente infinitas.

Con respecto al rendimiento, resulta muy interesante la paralelización de procesado. Haciendo uso de la librería OpenThreads incluída en OSG, podemos

aprovechar el potencial de algunos equipos para mejorar notablemente el rendimiento y la fluidez de la aplicación. Además, tenemos al alcance otra técnica como es el renderizado selectivo, es decir, renderizar únicamente lo que aparece en la escena. En la aplicación actual, la escena entera está continuamente siendo renderizada, por lo que tenemos una pérdida de rendimiento que aumenta en función de la complejidad del escenario que estemos diseñando.

Pensando en la compatibilidad y la escalabilidad, sería interesante la elaboración de un plugin –aprovechando la apertura de código de OSG–, el cual permitiera mediante alguna acción en la interfaz, conectar con un software de modelado 3D instalado en el ordenador, y editar un modelo en cuestión.

Además, aprovechando el tirón de la tecnología móvil, el desarrollo de una aplicación móvil capaz de cargar un escenario diseñado desde la aplicación desarrollada en éste TFM y mostrarlo de igual forma que lo hace el HMD actualmente.

Centrándonos en la Realidad Aumentada, dos líneas de investigación resultan muy interesantes. Por un lado tenemos la oclusión selectiva. Cuando se renderiza un escenario en tres dimensiones, éste se presenta sobre la imagen de la realidad, es decir, oculta todo lo que hubiera en esa misma posición. Por tanto, la oclusión tan sólo de ciertas partes de la escena permitiría colocar una farola dentro de unas verjas que en la realidad existen, o un cuadro dentro de una vitrina ya existente en la realidad. Por otro lado, se podrían investigar más a fondo los marcadores, y el modo en que éstos se reconocen, pudiendo mejorar la tecnología markerless.

Para terminar, y siguiendo con la idea de que las posibles mejoras son casi infinitas, las mejoras en cuando a características visuales y de usabilidad son de lo más variado: acciones para deshacer y rehacer movimientos, cambiar los nombres a los distintos tipos de datos que forman la escena, agrupar disintos modelos para que una acción se vea siempre reflejada sobre la totalidad del grupo, etc.

Bibliografía

[1] Bell, J. (2002). "Cómo hacer tu primer trabajo de investigación: guía para investigadores en educación y ciencias sociales", ISBN: 8474329310, Barcelona: Gedisa.

[2] Boj, C. y Díaz D. (2007). "La hibridación a escena: Realidad aumentada y Teatro", Revista Digital Universitaria, vol. 8, nº 6, ISSN: 1067-6079.

[3] Simon, G. et al. (2000). "La hibridación a escena: Realidad aumentada y Teatro", Revista Digital Universitaria, vol. 8, nº 6, ISSN: 1067-6079.

[4] Kato, H. y Billinghurst, M. (1999). "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System", en Proceedings. 2nd IEEE and ACM International Workshop on Augmented Reality, 1999.

[5] WikiTude. Página Web Oficial: http://www.wikitude.com/. Última visita: Agosto 2012.

[6] Ruiz, A. et al. (2004). "Ideación Arquitectónica Asistida mediante Realidad Aumentada", Innovación en Telecomunicaciones, V-1-V-8, 2004.

[7] Alonso, N. et al. (2001). "Análisis de escenarios de futuro en realidad aumentada. Aplicación al yacimiento arqueológico de Els Vilars", en En: Interacción'2001, 2º Congreso Internacional de Interacción Persona-Ordenador, Salamanca, Mayo 2001.

[8] Redondo, E. (2010). "Un caso de estudio de investigación aplicada. La recuperación de la trama viaria del barrio Judío de Girona mediante Realidad Aumentada", en Revista EGA: Revista de Expresión Gráfica Arquitectónica, 16:70-81.

[9] Granollers, T. et al. (2002). "Vilars. Un nuevo modelo de diálogo aplicando Realidad Aumentada, en Congreso Internacional de Diseño, Especificación y Verificación de Sistemas Interactivos, Rostock (Alemania).

[10] Heilig, Morton L. (1962). "Sensorama Simulator", United States Patent Office. Pat. no. 3050870. 28 de Agosto, 1962.

[11] VTT Finland. (2012). "ALVAR – A Library for Virtual and Augmented Reality User's Manual (v.2.0)".

[12] W. Krueger et al. (1985). "Videoplace – An Artificial Reality", Proceedings of ACM SIGCHI Bulletin '85.

[13] Zimmerman, T. G. y Lanier, J. Z. (1991). "Computer data entry and manipulation apparatus and method", United States Patent. Pat. No 4988981, 29 de Enero, 1991.

[14] G. Zimmerman, T. et al. (1987). "A Hand Gesture Interface Device" en Proceeding CHI '87: Proceedings of the SIGCHI/GI conference on Human factors in computing systems and graphics interface 1987. Pages 189 - 192. ISBN:0-89791-213-6.

[15] Lanier, J. et al. (1990). "Reality Built For Two: A Virtual Reality Tool", en Proceedings of the 1990 symposium on Interactive 3D graphics, pages 35 – 36. ISBN:0-89791-351-5.

[16] Caudell, T.P. y Mizell, D.W. (1992). "Augmented Reality: An Application of Heads-Up Display Technology to Manual Manufacturing Processes", en Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences, 1992. Pages 659 - 669 vol.2.

[17] Janin, A. L. et al. (1993). "Calibration of Head-Mounted Displays for Augmented Reality Applications", en IEEE Virtual Reality Annual International Symposium, 1993. Pages 246 – 255.

[18] Hincapié, M. et al. (2011). "An Introduction to Augmented Reality with Applications in Aeronautical Maintenance", en IEEE 13th International Conference on Transparent Optical Networks (ICTON). Pages 1 - 4.

[19] TeclasRápidas, Página Web Oficial: http://eclipseplugin.blogspot.com.es/. Última Visita: Agosto 2012.

[20] MuchoModding. Página Web Oficial: http://muchomodding.wordpress.com/. Última Visita: Agosto 2012.

[21] Pocket-Lint. Página Web Oficial: http://www.pocket-lint.com. Última Visita: Agosto 2012.

[22] Chang, W. y Tan, Q. (2010). "Augmented Reality System Design and Scenario Study for Location-based Adaptive Mobile Learning", en IEEE International Conference on Computational Science and Engineering (CSE). Pages 20 – 27.

[23] Wang, R. y Qian, X.. (2010), "OpenSceneGraph 3.0 Beginner's Guide", Packt Publishing, Birmingham-Mumbai, ISBN 978-1-849512-82-4.

[24] Woolford, D. (2003), "Understanding and using scene graphs", COMP4201 Lectures, 2003.

[25] Welch, G. y Bishop, G. (2002), "An Introduction to the Kalman Filter". Department of Computer Science, University of North Carolina at Chapel Hill, 2002.

[26] Haykin, S. (1999), "Neural networks: a comprehensive foundation", New Jersey: Prentice-Hall, 1999.

[27] Puskorius, G.V. y Feldkamp, L.A. (1991), "Decoupled extended kalman filter training of feedforward layered networks", In Neural Networks, 1991, IJCNN-91-Seattle International Joint Conference on, volume i, pages 771 –777, vol.1, jul 1991.

[28] Loebis, D. et al. (2004), "Adaptive tuning of a kalman filter via fuzzy logic for an intelligent auv navigation system", Control Engineering Practice, 12(12):1531 – 1539, 2004.

[29] Kalman, R. E. (1960), "A new approach to linear filtering and prediction problems", Transaction of the ASME - Journal of Basic Engineering,, pages 35 – 45.

[30] Azuma, R.T. (1997). "A Survey of Augmented Reality", In Presence: Teleoperators and Virtual Environments 6, 4 (August 1997), 355-385.

[31] Azuma, R.T. et al. (2001). "Recent Advances in Augmented Reality", en IEEE Computer Graphics and Applications.

[32] Bradski, Gary & Kaebler, Adrian. (2008), "Learning OpenCV", Computer Vision with the OpenCV Library. O'Reilly Media Inc.

[33] Holloway, R. (1995) "Registration Errors in Augmented Reality". Ph.D. dissertation. UNC Chapel Hill Department of Computer Science technical report TR95-016.

[34] Allison, R.S. (2001) "Tolerance of temporal delay in virtual environments". Proceedings of the IEEE Virtual Reality 2001.

[35] GadgetGrid Página Web Oficial: http://www.gadgetgrid.com/. Última Visita: Agosto 2012.

[36] Telepresence Options, Página Web Oficial: http://www.telepresenceoptions.com. Última Visita: Agosto 2012.

[37] Omnispace. Página Web Oficial: http://www.omnispace.org/. Última Visita: Agosto 2012.

[38] Vuzix Wrap 920AR. Página Web Oficial: http://www.vuzix.com/augmented-reality/products_wrap920ar.html. Última Visita: Agosto 2012.

[39] Búsqueda "Realidad Aumentada" en Google Play. Página Web Oficial: https://play.google.com/store/search?q=augmented+reality&c=apps. Última Visita: Agosto 2012.

[40] Vuzix Eyewear. "Vuzix Eyewear Software Development Kit Version 3.1." Documentación official del software de desarrollo para sistemas operativos Windows.

[41] ING Entertainment. Página Web Oficial: http://uk.ign.com/. Última visita: Agosto 2012.

[42] Durlach, Nathaniel I. and Anne S. Mavor. "Virtual Reality: Scientific and Technological Challenges". (Report of the Committee on Virtual Reality Research and Development to the National Research Council) National Academy Press (1995). ISBN 0-309-05135-5.

[43] Wellner, P. (1993) "Interacting with Paper on the DigitalDesk". CACM 36, 7, 86-96.

[44] Rolland, J. et al. (1993) "A Comparison of Optical and Video See-Through Head-Mounted Displays". SPIE Proceedings volume 2351: Telemanipulator and Telepresence Technologies (Boston, MA, 31 October - 4 November 1994), pp. 293-307.

[45] Zhou, F. et al. (2008). "Trends in Augmented Reality Tracking, Interaction and Display: A Review of Ten Years of ISMAR", en IEEE International Symposium on Mixed and Augmented Reality 2008.

[46] Suárez Candel, R. (2007). "The Migration towards Digital Terrestrial Television (DTT): Challenges for Public Policy and Public Broadcasters". Observatorio (OBS*) Journal, 1 (pp 185-203). ISSN: 1646-5954.

[47] Brooks, Frederick P. Jr. (1996). "The Computer Scientist as Toolsmith II", Communications of the ACM, 1996.

[48] E. Sutherland, Ivan. (1965). "The Ultimate Display", en Proceedings of IFIP Congress, pp. 506-508, 1965.

[49] Kancherla, A. R. et al. "Dynamic 3D Anatomy. Proceedings of Computer Vision, Virtual Reality, and Robotics in Medicine '95 (CVRMed '95) (Nice, France, 3-6 April 1995), 163-169.

[50] Autodesk 3ds Max. Página Web Oficial: http://www.autodesk.es/adsk/servlet/pc/index?siteID=455755&id=14626995. Última visita: Agosto 2012.

[51] Autodesk Maya. Página Web Oficial: http://usa.autodesk.com/maya/. Última visita: Agosto 2012.

[52] Blender. Página Web Oficial: http://www.blender.org/. Última visita: Agosto 2012.

[53] LightWave. Página Web Oficial: https://www.lightwave3d.com/. Última visita Agosto 2012.

[54] Invizimals. Página Web Oficial: http://www.invizimals.com/home.php. Última visita Agosto 2012.

[55] OpenCV: Open Source Computer Vision. Página Web Oficial: http://opencv.willowgarage.com/wiki/. Última visita Agosto 2012.

[56] Alvar: Augmented Reality Library from VTT Technical Research Center of Finland. Página Web Oficial: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar.html. Última visita Agosto 2012.

[57] OSG: Open Scene Graph. Página Web Oficial: http://www.openscenegraph.org/projects/osg. Última visita Agosto 2012.

[58] Nokia Qt. Página Web Oficial: http://qt.nokia.com/products/. Última visita Agosto 2012.