



TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DE COMPUTADORES



Seguimiento y alineación 3D para la visualización de modelos anatómicos mediante realidad aumentada e impresión 3D

Estudiante:	Brais Barboza Ordóñez
Dirección:	José Antonio Iglesias Gutián
	Francisco Javier Taibo Pena
	Emilio José Padrón González

A Coruña, agosto de 2025.

A mis padres y mi novia

Agradecimientos

Dedico este logro académico a aquellos que me han brindado su apoyo incondicional en mi camino. A mis tutores Emilio, Jose y Javi, quienes personificaron la guía necesaria para completar este proyecto con éxito. Gracias por vuestras enseñanzas valiosas y por creer en mí.

A mi familia, especialmente a mis padres, por su amor incondicional y por ser mi constante motivación. Gracias por estar a mi lado en cada decisión y por ser mi fuente de inspiración.

A mis amigos, por su amistad y apoyo en momentos difíciles. Gracias por hacer la vida más alegre e incitarme a alcanzar nuevos horizontes.

Gracias a todos por ayudarme a alcanzar este objetivo. Este trabajo es el resultado de vuestro apoyo y esfuerzo constante, y está dedicado con todo mi agradecimiento y cariño.

Resumen

La imagen médica se ha beneficiado a lo largo del tiempo de los avances en las técnicas de visualización de contenido con el fin de poder brindar una mejor comprensión de los datos mostrados que se traduzca en una mejor atención médica. Es por ello que en este trabajo se analizan métodos para combinar la imagen médica con la realidad aumentada, otro campo cuyo auge en los últimos años para aplicaciones tanto industriales como de ocio no han pasado desapercibidas. Se lleva a cabo un análisis completo para proporcionar una solución funcional, desde la segmentación de los modelos desde una Tomografía Computerizada (TC), el desarrollo de un marcador para el seguimiento y la alineación en 3D hasta la implementación de una solución para llevarlo a cabo. Con el objetivo de servir como discusión de futuros desarrollos, se concibe teniendo como piedra angular su libre disposición y el software libre como alternativa a métodos existentes tras licencias restrictivas.

Abstract

Medical imaging has benefited over time from advances in content visualization techniques in order to provide a better understanding of the displayed data, leading to improved healthcare. Therefore, this project examines methods for combining medical imaging with augmented reality, another field whose recent rise for both industrial and recreational applications has not gone unnoticed. A comprehensive analysis is carried out to provide a functional solution, including model segmentation from a computerized tomography (CT) scan, marker development for tracking and 3D alignment, and the implementation of a solution to execute the process. In order to serve as a discussion of future developments, this work is conceived with the cornerstone of open availability and open-source software as an alternative to existing methods with restrictive licenses.

Palabras clave:

- Realidad aumentada
- Impresion 3D
- Imagen médica

Keywords:

- Augmented reality
- 3D Printing
- Medical imaging

Índice general

1	Introducción	1
1.1	Contexto y motivación	1
1.2	Objetivos	2
1.3	Estructura de la memoria	3
2	Fundamentos Teóricos y Técnicos	4
2.1	Tomografía Computerizada	4
2.2	Impresión 3D	6
2.3	Realidad Extendida	7
2.4	Visión Artificial	8
3	Estado del Arte	9
3.1	Visualización	9
3.2	Realidad Extendida	9
3.3	Impresión 3D	11
4	Herramientas y Software	12
4.1	Lenguajes de programación	12
4.1.1	C++	12
4.2	Sistema Operativo	12
4.2.1	Windows 11	12
4.3	Control de versiones	12
4.3.1	GitLab	12
4.4	Entorno de desarrollo	13
4.4.1	Visual Studio Community	13
4.5	Herramientas	13
4.5.1	DICOM	13
4.5.2	3D Slicer	13

4.5.3	Meshmixer	13
4.5.4	Blender	13
4.5.5	UltiMaker Cura	13
4.6	Frameworks	14
4.6.1	OpenXR	14
4.6.2	OpenCV	15
4.6.3	ARuco	16
4.6.4	Funcionamiento de Aruco	17
4.7	Hardware	20
4.7.1	Impresión 3D	20
4.7.2	Head Mounted Display (HMD)	20
5	Metodología y Gestión del proyecto	21
5.1	Metodología	21
5.2	Sprints	23
5.2.1	Extracción de un modelo a partir de un TC	23
5.2.2	Diseño del marcador fiduciario y software de tracking	23
5.2.3	Integrar solución de tracking en Exposure Render	23
5.2.4	Integrar passthrough en Exposure Render	23
6	Ejecución del proyecto	25
6.1	Análisis	25
6.2	Generación de volúmenes a partir de TC	26
6.3	Impresión 3D del volumen	27
6.4	Desarrollo del marcador fiduciario	28
6.5	Implementación del Passthrough en Exposure Render	36
6.6	Integración del software de tracking en Exposure Render	36
7	Conclusiones y trabajo futuro	37
7.0.1	Enriquecimiento Formativo	37
7.0.2	Trabajo futuro	38
	Lista de acrónimos	40
	Bibliografía	41

Índice de figuras

2.1	Representación del funcionamiento de un TC	5
2.2	Representacion simplificada del concepto de <i>virtuality continuum</i>	7
3.1	Aproximaciones a la realidad extendida para aplicaciones biomédicas.(fuente: [1])	10
4.1	Ciclo de vida de OpenXR. (fuente: https://www.khronos.org/)	15
4.2	Explicación de la API por capas. (fuente: https://registry.khronos.org)	16
4.3	Marcador Generado con ArUco	16
4.4	Ejemplos de marcadores fiduciarios previos (fuente: [2])	17
4.5	Proceso de extracción de Bits	19
5.1	Diagrama de la planificación del proyecto por sprints	24
6.1	Flujo de integración del seguimiento en Exposure Render.	26
6.2	TC de partida y obtención del modelo 3D final con 3DSlicer.	27
6.3	Modelo de cráneo 3D con geometrías erróneas.	28
6.4	Proceso de recuperación de material.	29
6.5	Esquema de un marcador.	31
6.6	Esquema de una tabla de marcadores.	31
6.7	Layout de las caras del marcador fiduciario.	32
6.8	Diseño final del marcador fiduciario.	34
6.9	Imágenes sobre las que se estima la pose del cubo.	34
6.10	Diagrama de flujo de la solución de tracking.	35

Índice de tablas

5.1 Tareas llevadas a cabo	22
--------------------------------------	----

Capítulo 1

Introducción

El constante progreso en medicina, y en particular en imagen médica, hace necesario contar con sistemas cada vez más avanzados para la representación y visualización de los datos obtenidos. El progreso en esta área ha sido patente en los últimos 30 años [3]. Más recientemente, en esta búsqueda de métodos que sean capaces de explotar las mejoras tecnológicas, aparece la aplicación de la realidad aumentada (AR, por sus siglas en inglés) y de otras técnicas englobadas dentro de lo que se conoce como realidad extendida (XR, compendio de realidad virtual, aumentada y mixta) en la medicina [4]. Un uso efectivo de este tipo de tecnologías en el campo de la imagen médica supone un complemento de gran utilidad para muchas de las tareas del personal médico: diagnóstico, planificación preoperatoria, explicación a pacientes, cirugía guiada por imagen, formación médica, etc.

Este proyecto plantea el uso de piezas creadas con una impresora 3D a partir de imagen médica en un entorno de realidad aumentada, abordando la problemática de su detección y seguimiento para un correcto alineamiento con un modelo 3D virtual, con el objetivo de integrar una imagen sintética sobreimpresa en la imagen real capturada por la cámara.

1.1 Contexto y motivación

Varias son las razones que explican el actual auge en la aplicación de técnicas de XR en medicina. Por un lado, la superposición en tiempo real de información digital sobre imagen real facilita la visualización de datos médicos y su interpretación. Esta información extra ayuda al personal sanitario a tener una comprensión más clara y detallada de la anatomía de un paciente, lo que resulta especialmente útil tanto en las fases de diagnóstico como durante procedimientos quirúrgicos complejos, proporcionando información relevante en el campo de visión que sirve como guía en tiempo real durante una intervención. Además, la posibilidad de superponer modelos tridimensionales de estructuras anatómicas en el paciente facilita la planificación precisa de una cirugía, así como la comunicación con el paciente sobre el procedimiento.

miento. Por otro lado, la realidad extendida ofrece una herramienta efectiva para la educación y formación médica. Los estudiantes y profesionales sanitarios pueden utilizarla para practicar y simular procedimientos médicos en un entorno virtual realista antes de realizarlos en pacientes reales. Esto brinda la oportunidad de adquirir experiencia y habilidades sin riesgo para los pacientes.

En este proyecto proponemos el uso de piezas creadas con una impresora 3D a partir de imágenes médica en un entorno de realidad aumentada que permita superponer una imagen sintética (un *render* 3D) sobre la visualización de la pieza en la imagen real capturada por la cámara. La idea fundamental es facilitar la manipulación de lo que podría ser una prótesis médica en un entorno de realidad extendida completo. Dentro de ese entorno, la pieza de AR desarrollado en este proyecto se encargaría de la correcta detección y seguimiento de la prótesis, junto a la integración de una imagen virtual superpuesta sobre la misma.

Para el correcto seguimiento y alineación de una pieza física con su imagen virtual en un flujo de vídeo es preciso conocer los parámetros de la cámara y la posición de la misma respecto al objeto. Para la impresión puede extraerse un modelo 3D a partir de una [Tomografía Computerizada \(TC\)](#). Con el fin de facilitar el seguimiento es posible que sea preciso añadir marcadores de referencia sobre la pieza, para utilizarlas como guía. En el proyecto se cubrirá el flujo de trabajo completo, desde el análisis y la extracción de un modelo a partir de la [TC](#), pasando por el diseño de un marcador fiduciario que se usará para hacer el seguimiento del objeto, hasta la manipulación física de la pieza en un entorno de realidad aumentada en la que se proyectará un modelo virtual sobre la pieza impresa vista en imágenes reales.

1.2 Objetivos

Los objetivos principales de este proyecto son:

- Extraer un modelo 3D a partir de imágenes capturadas mediante [Tomografía Computerizada \(TC\)](#) para su impresión.
- Diseñar un marcador fiduciario que sirva como guía para facilitar el seguimiento de la pieza.
- Hacer detección, seguimiento y alineamiento 3D de la pieza impresa o guía en el flujo de vídeo capturado por un sistema de realidad aumentada, como puede ser un [HMD](#) que incorpore cámaras de vídeo.
- Integrar elementos sintéticos en la imagen real de la visualización 3D.
- El objetivo final es disponer de un software capaz de resolver el problema del seguimiento y la estimación de pose en 3D y que pueda ser fácilmente integrable en un sistema

de realidad extendida completo.

1.3 Estructura de la memoria

TODO

Capítulo 2

Fundamentos Teóricos y Técnicos

EN este capítulo se repasan los principios básicos sobre los que se establece este trabajo.

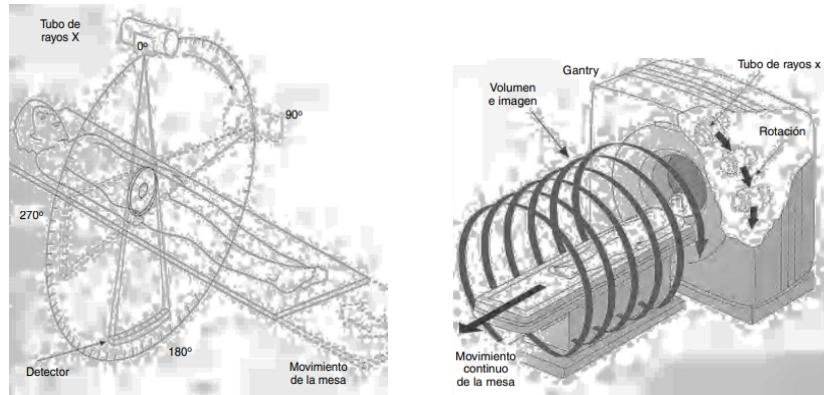
2.1 Tomografía Computerizada

Una **Tomografía Computerizada (TC)**, también conocida como escáner, es una técnica de diagnóstico médico que utiliza rayos X, detectores y un ordenador para obtener imágenes detalladas de estructuras internas del cuerpo. La **TC** combina una serie de imágenes radiográficas en secciones transversales para crear imágenes en 2D y 3D del área estudiada.

Durante una **TC**, el paciente se coloca en una mesa que se desliza dentro de un dispositivo circular llamado tomógrafo. Un tomógrafo es, en esencia, una máquina de rayos X en la cual se ha sustituido la placa por una serie de detectores [5]. La fuente de rayos X y los detectores efectúan un movimiento circular y avanzan lentamente hasta cubrir el área deseada, como se puede ver en Figura 2.1. Los detectores capturan la radiación después de que ha atravesado el cuerpo. La información recopilada se envía a un ordenador que procesa los datos y los convierte en imágenes transversales o en secciones longitudinales del área de interés.

La tomografía computarizada proporciona imágenes detalladas de tejidos blandos, huesos y órganos internos, lo que permite a los médicos diagnosticar y evaluar una amplia variedad de condiciones y enfermedades. Se utiliza comúnmente en el diagnóstico y seguimiento de enfermedades como cáncer, lesiones traumáticas, enfermedades cardiovasculares, trastornos pulmonares, afecciones cerebrales y abdominales, entre otros.

Teorema de Radon. Propuesto por el matemático austriaco Johann Radon en 1917, el Teorema de Radon es un resultado fundamental en la teoría de la tomografía computerizada. Este teorema establece que es posible reconstruir una función bidimensional a partir de sus pro-



(a) Orientación del tubo de rayos X respecto al eje corporal.

(b) Tomografía axial computerizada convencional.

Figura 2.1: Representación del funcionamiento de un TC

yecciones a lo largo de diferentes ángulos. En el contexto de la tomografía computerizada, las proyecciones se obtienen mediante la medición de la atenuación de la radiación a medida que atraviesa el objeto en estudio. Una vez que se han recopilado todas las proyecciones, se utiliza un algoritmo de reconstrucción para combinarlas y generar una imagen transversal detallada del área estudiada. La base matemática de ese proceso de reconstrucción de imágenes en la tomografía computerizada la proporciona el teorema de Radon y ha llevado al desarrollo de diversos algoritmos de reconstrucción.

Algoritmos de reconstrucción de imágenes. Los algoritmos de reconstrucción de imágenes son métodos computacionales utilizados para reconstruir imágenes bidimensionales o tridimensionales a partir de proyecciones adquiridas en la tomografía computerizada. Estos algoritmos se basan en el Teorema de Radon y pueden clasificarse en dos categorías principales: métodos analíticos [6] y métodos iterativos [7].

1. Métodos analíticos: Estos algoritmos, como la retroproyección filtrada (FBP, por sus siglas en inglés), procesan las proyecciones de manera directa para obtener la imagen reconstruida. La FBP es el algoritmo más utilizado en la práctica clínica debido a su rapidez y eficiencia.

2. Métodos iterativos: Estos algoritmos, como el de máxima verosimilitud de la expectativa-maximización (MLEM) y el de mínimos cuadrados conjugados (CGLS), utilizan un enfoque iterativo para mejorar la calidad de la imagen reconstruida. Aunque estos métodos suelen ser más lentos que los analíticos, pueden proporcionar imágenes de mayor calidad y son especialmente útiles en aplicaciones donde la cantidad de datos de proyección es limitada o ruidosa.

2.2 Impresión 3D

La impresión 3D es una tecnología de fabricación aditiva que permite crear objetos tridimensionales a partir de un modelo digital. Esta tecnología ha revolucionado la forma en que se fabrican piezas y productos, ya que permite la creación de objetos complejos con geometrías que serían difíciles o imposibles de lograr con métodos de fabricación tradicionales. La impresión 3D se utiliza en una amplia variedad de aplicaciones, desde la fabricación de piezas de repuesto hasta la creación de prótesis médicas personalizadas.

Fabricación aditiva. La impresión 3D es un ejemplo de fabricación aditiva, que se refiere a manipulación y depósito de un material a escala micrométrica de forma muy precisa para construir un sólido. La fabricación aditiva es una alternativa a los métodos de fabricación tradicionales, como el fresado y el torneado, que implican la eliminación de material de una pieza bruta [8].

Esta técnica de fabricación presenta una serie de ventajas. La complejidad de la geometría de la figura no encarece la fabricación de la misma (a expensas de la necesidad de material como soporte de la geometría principal) sino que permite generar piezas con geometrías previamente inviables, o con un alto coste. Otra de las ventajas es la posibilidad de generar prototipos de piezas cuyas versiones finales presentan un alto coste, por un precio muy reducido, y una alta fidelidad acelerando así el proceso iterativo del diseño.

Modelado. El proceso de impresión 3D comienza con el modelado de la pieza o producto que se desea imprimir. Este modelo puede crearse utilizando software [Computer-Aided Design \(CAD\)](#) o mediante la digitalización de un objeto existente utilizando un escáner 3D. En nuestro caso, el modelo tridimensional se obtiene de la reconstrucción 3D a partir de una [TC](#).

Slicing (rebanado) En impresión 3D, el proceso de slicing (o rebanado, en español) se refiere a la preparación del modelo tridimensional, típicamente en formato STL, para su impresión en capas sucesivas. Es un paso crucial que convierte el modelo en una serie de capas planas y delgadas que la impresora 3D puede imprimir una por una. El *slicer* permite además realizar ajustes en el modelo 3D, con el fin de obtener el resultado deseado, generando como resultado una serie de instrucciones precisas que la impresora entiende para elaborar capa a capa el volumen.

GCODE. Las instrucciones que se generan a partir del proceso de *slicing* son provistas en forma de GCODE. El GCODE es el lenguaje utilizado para describir paso a paso que movimientos y acciones debe tomar la impresora en cada momento. Este lenguaje tiene distintas

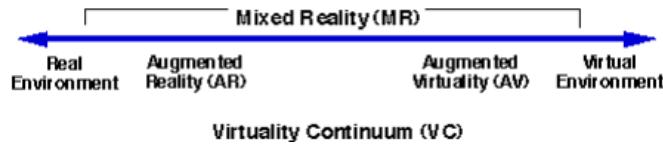


Figura 2.2: Representación simplificada del concepto de *virtuality continuum*.

implementaciones dependiendo del fabricante del equipamiento, ya que se trata de un lenguaje utilizado en múltiples aplicaciones de control numérico.

2.3 Realidad Extendida

Realidad Extendida (XR) es un término general que engloba todo el espectro de tecnologías inmersivas, incluyendo Realidad Virtual (VR), Realidad Aumentada (AR) y Realidad Mixta (MR). XR se refiere a la fusión de los mundos físico y digital, creando un entorno inmersivo que puede incluir objetos virtuales, información digital y elementos del mundo real.

En 1994, Milgram and Kishino [9] acuñan el término de *virtuality continuum*, que representa una escala que comprende desde la realidad física pura hasta la realidad virtual total, abarcando diferentes niveles de inmersión e interacción. Este *continuum* describe así la gama completa de experiencias, desde la realidad física no modificada hasta entornos completamente virtuales, como se muestra en la Figura 2.2:

- Entorno Real: consiste únicamente en objetos reales (extremo izquierdo de la figura).
- Realidad Aumentada: mundo real aumentado o compuesto por elementos digitales.
- Virtualidad Aumentada: mundo digital aumentado por objetos reales o físicos.
- Entorno Virtual: entorno puramente virtual (extremo derecho de la figura).

Realidad aumentada. Posteriormente Azuma [10] definía la Realidad Aumentada como una variación de la Realidad Virtual, en la cual el usuario es capaz de ver el mundo real, con objetos virtuales superpuestos, o compuestos por el mundo real. La posibilidad de crear objetos y prototipos de forma rápida, sobre los cuales poder iterar y componer imágenes virtuales, demuestra el potencial de estas tecnologías inmersivas trabajando a la par. En la actualidad sus principales aplicaciones se encuentran, además de en juegos y entretenimiento, en educación, medicina, arquitectura, ingeniería e interpretación del patrimonio.

2.4 Visión Artificial

Se denomina visión artificial al campo que incluye los métodos necesarios para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información procesable por un ordenador. Esta fuertemente vinculado con la realidad aumentada, ya que es imprescindible para poder transmitir la información del medio al ordenador encargado de generar imágenes correspondientes. Entre los objetivos de la visión artificial se encuentra la capacidad de reconocer patrones dentro de una imagen o vídeo con el fin de poder extraer las características de los objetos dentro de dicho medio y procesarlas. Otro objetivo es la reconstrucción 3D a partir de imágenes, que pretende generar volúmenes 3D desde las imágenes obtenidas, esto es especialmente importante en la realidad aumentada por que permite una mayor percepción de la profundidad sobre el medio generado por ordenador.

Capítulo 3

Estado del Arte

Este capítulo se centra en detallar el estado actual de las tecnologías utilizadas en el proyecto así como en analizar soluciones similares al proyecto.

3.1 Visualización

Los componentes artificiales que deseamos incluir en las imágenes del mundo real proceden del framework desarrollado por Kroes et al. que aplica técnicas de Monte Carlo Ray Tracing (MCRT) sobre Direct Volume Rendering (DVR).

El término DVR se utiliza para referirse a las técnicas que producen una imagen directamente a partir de datos de un volumen, sin realizar pasos intermedios. Para que esto sea posible es necesario implementar modelos físicos que indiquen cómo se genera, refleja, dispersa o oculta la luz [12]. Estos modelos con el paso del tiempo han evolucionado en modelos más y más complejos que han probado ser beneficiosos para la visualización científica de modelos 3D [13], [14], [15] .

La implementación de estos modelos conlleva altos tiempos de renderizado, o en su defecto, un equipo increíblemente costoso para poder obtener una experiencia interactiva [16]. Para enfrentar esta casuística, Iglesias-Guitian et al. implementan un algoritmo de reducción de ruido basado en Recursive Least Squares (RLS) que permite una experiencia interactiva en tiempo real, sobre GPUs comerciales. Este proyecto se fundamenta en [16] para el renderizado de las imágenes que posteriormente se apliquen en realidad aumentada sobre las imágenes reales.

3.2 Realidad Extendida

Para llevar a cabo el proyecto existen varias aproximaciones en el estado del arte [1]. Se seleccionaron aquellas que más se ajustaban al proyecto. Uno de los objetivos principales es

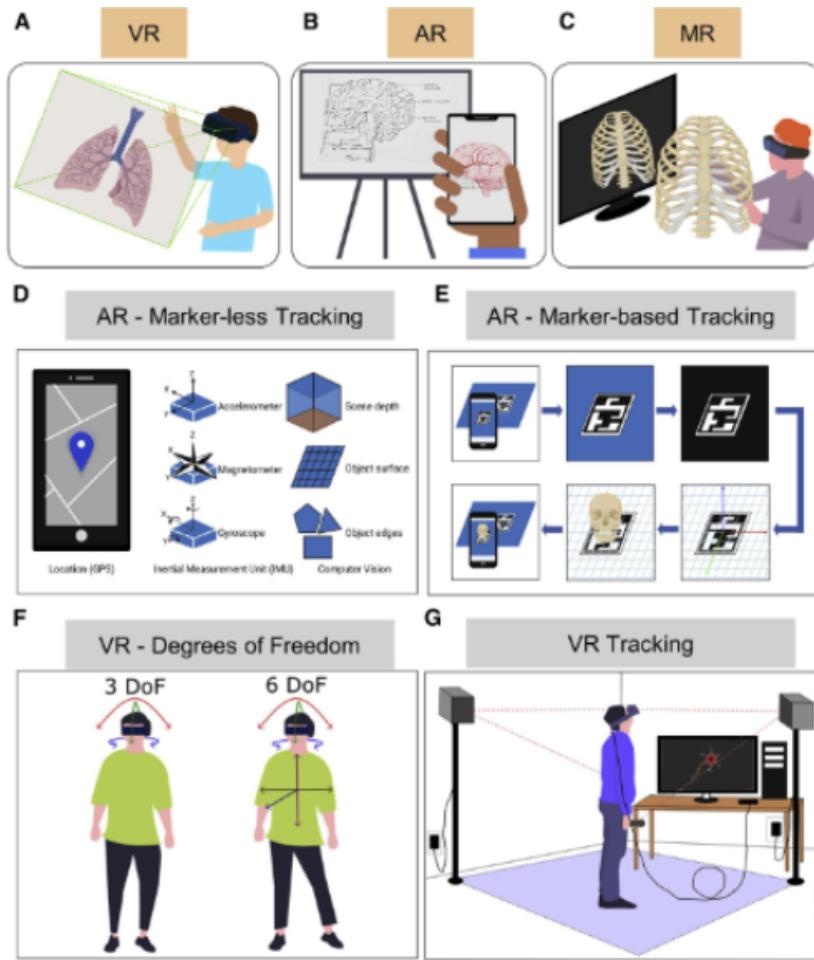


Figura 3.1: Aproximaciones a la realidad extendida para aplicaciones biomédicas.(fuente: [1])

el seguimiento de piezas extraídas de un TC y diseñar marcadores que facilitasen el registro de las mismas, por lo que se optó por un seguimiento basado en marcadores. No obstante, en lo que a la visualización se refiere, es necesario un equipo con gran potencia computacional, o en su defecto, un visor que permita la reproducción de vídeo renderizado por un tercer equipo. Dadas estas restricciones se decidió por utilizar un Head Mounted Display (HMD) HTC VIVE PRO. Utilizando la Figura 3.1 [1] como referencia , el sistema implementado se compondría de un seguimiento basado en marcadores (E) visualizando estos marcadores en realidad aumentada (B). Para este proyecto se escogieron las tecnologías E y G que se observan en la Figura 3.1

Destacar también la solución presentada por [17]. Este proyecto presenta un método para diseñar aplicaciones de realidad aumentada para la visualización de modelos anatómicos en 3 dimensiones mediante el uso de un marcador fiduciario. En el se explica un método para extraer una figura a partir de una TC. Posteriormente, provee instrucciones detalladas para

la impresión 3D del marcador fiduciario. Finalmente utilizando la extensión de Vuforia para Unity crea una aplicación móvil que permite la visualización en realidad aumentada de la pieza.

3.3 Impresión 3D

La impresión 3D o fabricación aditiva es un proceso para manufacturar objetos de cualquier forma o tamaño a partir de un modelo 3D o otro tipo de fuente electrónica a partir de procesos aditivos en los que sucesivas capas de material se depositan bajo control de un ordenador [18]. Desde 1984 [19] hasta la actualidad, la impresión 3D ha afrontado una revolución que se fundamenta en pilares sólidos como el abaratamiento de los costes de producción de impresoras 3D, la mejora de su precisión y velocidad, el software libre, la comercialización de los productos a usuarios finales, documentación, formación extensa corroborada y creada por terceros que facilita y realimenta el desarrollo de nuevos proyectos. Producto de esto, ha sido la implementación de esta tecnología en el ámbito sanitario, tanto para la producción de herramientas especializadas [19] como para el diseño y la implantación de prótesis personalizadas para cada paciente [20].

Capítulo 4

Herramientas y Software

En este capítulo se explican las herramientas y librerías utilizadas para llevar a cabo el proyecto.

4.1 Lenguajes de programación

4.1.1 C++

El proyecto se desarrolló en su totalidad en C++. Esto se debe a que como se menciona previamente, este trabajo forma parte del esfuerzo académico de Iglesias-Guitian et al. y por coherencia se decidió seguir la línea de trabajo. C++ es un lenguaje de programación que se beneficia de programación orientada a objetos sobre la sintaxis de C y ha sido utilizado para implementar librerías gráficas intrínsecas en el proyecto.

4.2 Sistema Operativo

4.2.1 Windows 11

Windows 11 es un sistema operativo desarrollado por Microsoft. Se utilizó debido a la familiaridad del proyecto con el mismo.

4.3 Control de versiones

4.3.1 GitLab

Para llevar a cabo el control de versiones se utilizó GitLab ya que el código implementado formaba parte de le proyecto previamente mencionado, y este se almacena en GitLab.

4.4 Entorno de desarrollo

4.4.1 Visual Studio Community

Es el Entorno de desarrollo de C++ por excelencia en Windows.

4.5 Herramientas

4.5.1 DICOM

DICOM es la denominación de un estándar utilizado principalmente para la visualización, impresión, almacenamiento y transmisión de imágenes y datos de propósito médico. Los ficheros DICOM consisten en una cabecera con campos estandarizados y de forma libre, y un cuerpo con datos de imagen. Un objeto DICOM simple puede contener solamente una imagen, pero esta imagen puede tener múltiples fotogramas, permitiendo el almacenamiento de bloques de datos con varios fotogramas.

4.5.2 3D Slicer

3D Slicer es un programa de software libre diseñado para solventar los problemas más avanzados de la computación de imagen relacionados con las aplicaciones clínicas y biométricas. Las capacidades del mismo se encuentran la posibilidad de implementar scripts de python, segmentación de imágenes y volúmenes, la posibilidad de añadir extensiones para aumentar su funcionalidad y la interoperabilidad del estándar DICOM entre otras.

4.5.3 Meshmixer

Al procurar generar un modelo a partir de una nube de puntos de un TAC, es común encontrarse con que los modelos exportados en STL contienen errores y no pueden ser impresos directamente. Se probaron varias herramientas para solventar estos errores en la estructura de los modelos 3D, pero finalmente Meshmixer (<https://www.meshmixer.com>) resultó dar los mejores resultados a la hora de arreglar las geometrías con esta casuística tan específica.

4.5.4 Blender

Blender es la herramienta de software libre para la creación 3D por excelencia.

4.5.5 UltiMaker Cura

UltiMaker Cura es el programa desarrollado por UltiMaker para generar el GCODE necesario para imprimir modelos en una impresora de dicha marca. Se utilizó ya que permite

importar ajustes específicos de la impresora sobre la que se trabaja de forma sencilla.

4.6 Frameworks

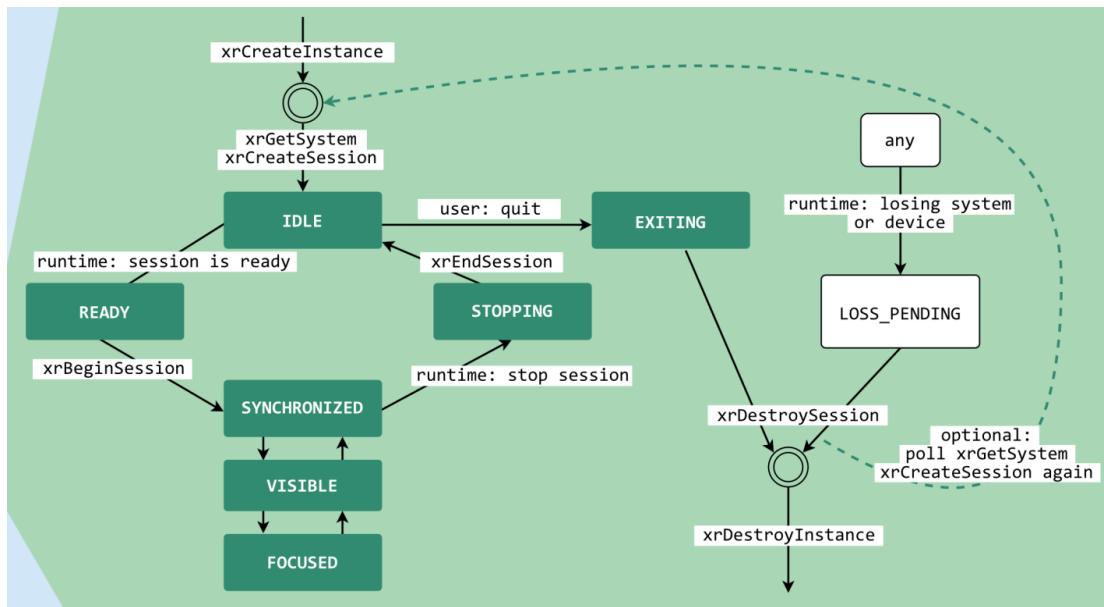
4.6.1 OpenXR

OpenXR es una [Interfaz de programación de aplicaciones \(API\)](#) multi plataforma, que permite el desarrollo de medios en el *virtual continuum* mediante ordenadores a través de interacción humano-máquina. Esta [API](#) es la interfaz con un runtime para llevar a cabo operaciones comunes como puede ser acceder al estado de un mando o periférico, obtener o predecir la posición del sistema o enviar frames para ser renderizados.

Ciclo de Vida

En la Figura 4.1 se muestra la máquina de estados de la sesión:

1. La aplicación crea una sesión escogiendo un sistema y una API gráfica. En un primer momento esta se encuentra en estado IDLE.
2. Se monitorea la sesión en busca de cambios de estados mediante eventos.
3. Cuando el runtime determina que el sistema está listo para empezar con el contenido [XR](#) de la sesión, se recibe un cambio de estado a READY.
4. Mientras que la sesión está corriendo, se espera que la aplicación ejecute continuamente el frame loop, estableciendo así sincronización con el runtime, lo que provoca un cambio de estado a SYNCHRONIZED.
5. Una vez que el runtime está listo para mostrar frames de la aplicación, se notifica con el estado VISIBLE.
6. Si el runtime detecta que es posible recibir entradas desde un mando, reconocimiento facial o demás, notifica con un estado FOCUSED.
7. Estos estados, como se ve en la Figura 4.1, también tienen carácter retroactivo, de forma que cuando las características dejan de estar disponibles se va cambiando de estado, hasta que se desee parar o cerrar la aplicación.

Figura 4.1: Ciclo de vida de OpenXR. (fuente: <https://www.khronos.org/>)

API Layers

OpenXR está diseñado como una API por capas, lo que quiere decir que una aplicación puede insertar más o menos capas entre la aplicación y la implementación del runtime seleccionada. Estas capas proveen de funcionalidades adicionales interceptando las funciones de OpenXR de la capa superior, y posteriormente llevando a cabo operaciones distintas a las que se llevarían a cabo en caso de que no estuviese presente la capa. En el más sencillo de los casos una capa simplemente llama a la inferior con los mismos argumentos, pero en casos más elaborados se pueden implementar funcionalidades no disponibles en las capas o incluso runtime inferiores (Figura 4.2).

Esta arquitectura permite el desarrollo multiplataforma con mayor simplicidad, pero es dependiente de que los vendedores implementen sus propias capas API de OpenXR, lo que limita en cierta medida las

4.6.2 OpenCV

Open Source Computer Vision Library (OpenCV) es una biblioteca de código abierto que implementa principalmente funciones de visión artificial en tiempo real. Se utilizó para la generación y el seguimiento de los marcadores ARuco que forma parte de los paquetes adicionales de la biblioteca.

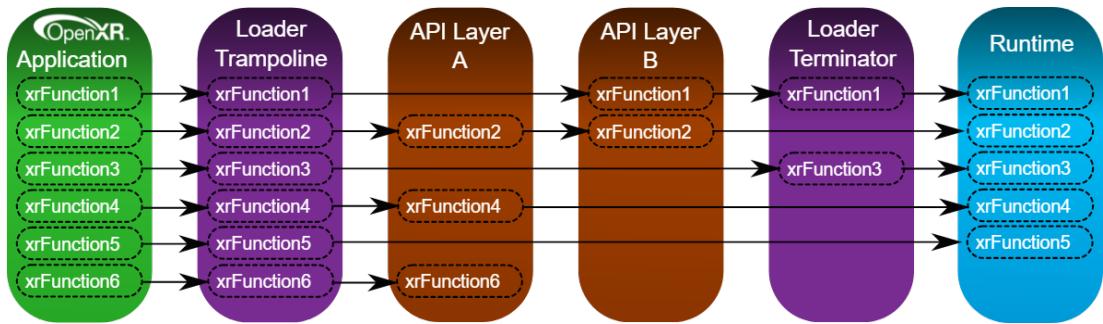
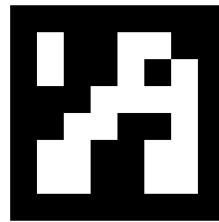
Figura 4.2: Explicación de la API por capas. (fuente: <https://registry.khronos.org>)

Figura 4.3: Marcador Generado con ArUco

4.6.3 ARuco

Los últimos años, los desarrollos de nuevos marcadores han tendido a un cuadrado negro con distintos patrones interiores, como en el ejemplo de la Figura 4.4, ya que permiten extraer la pose de la cámara a partir de sus 4 esquinas, asumiendo que esta esté adecuadamente calibrada [2]. Esencialmente estos marcadores comparten ciertas características comunes en cuanto a su funcionamiento, entre todas las opciones disponibles se escogió ArUco como solución a nuestro proyecto por varios motivos:

- Diccionarios generados dinámicamente.
- Posibilidad de crear tablas de marcadores lo que incrementa la resistencia a las occlusiones.
- Software para la calibración de cámara: De forma sencilla se puede calibrar cualquier cámara.
- La librería ha soportado el paso de los años sin problema, existiendo ejemplos y documentación extensa sobre el funcionamiento de la misma, facilitando así el desarrollo.

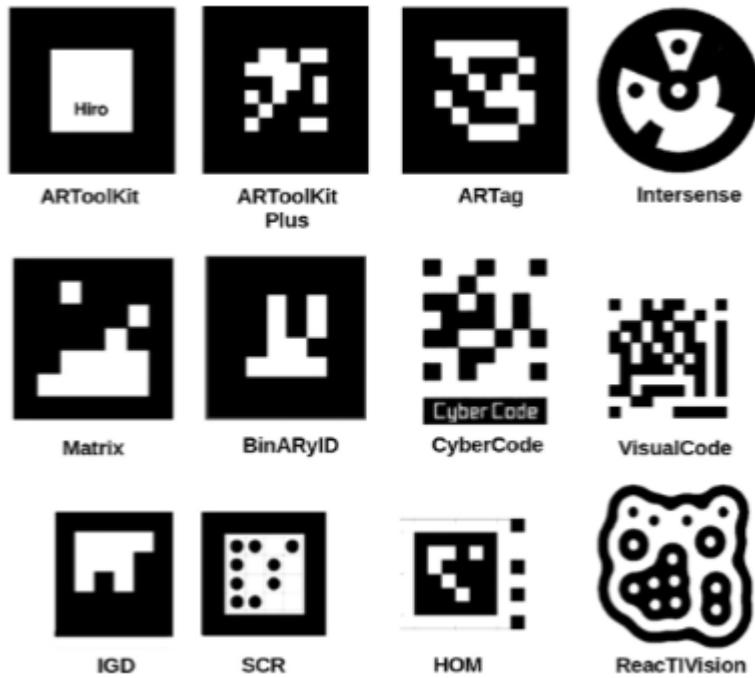


Figura 4.4: Ejemplos de marcadores fiduciarios previos (fuente: [2])

4.6.4 Funcionamiento de Aruco

Captura de imágenes o videos

La captura de imágenes o vídeos se realiza mediante un dispositivo específico, como una cámara digital o un dispositivo móvil con cámara incorporada. En este paso se espera obtener una serie de imágenes o un vídeo sobre el que se espera encontrar uno o varios marcadores ARuco.

Conversión a escala de grises

La conversión a escala de grises se realiza mediante el algoritmo de promedio ponderado de los canales RGB. Esta conversión reduce la cantidad de información a procesar y mejora la velocidad de procesamiento al trabajar con un único canal de información.

Aplicación de un filtro de bordes

Para resaltar los bordes de los marcadores en la imagen se aplica un filtro de bordes. Un ejemplo común es el algoritmo de Canny, que se basa en la detección de gradientes y utiliza un umbral para determinar que bordes son relevantes y cuales no.

Detección de contornos

Se utiliza un algoritmo de detección de contornos adaptativo para encontrar los contornos de los marcadores en la imagen. Un algoritmo común es el Transformada de Hough que permite detectar contornos circulares y lineales en la imagen. Este algoritmo busca patrones en la imagen que se correspondan con los contornos de los marcadores ARuco. En un sistema de thresholding tradicional, se elige un umbral global para toda la imagen. Cualquier píxel con un valor de brillo superior al umbral se considera activo (p.ej. negro) y cualquier píxel con un valor de brillo inferior al umbral se considera inactivo (p.ej. blanco). Sin embargo, en muchas imágenes, el nivel óptimo de umbral puede variar entre diferentes partes de la imagen. El thresholding adaptativo se utiliza para solucionar este problema.

El thresholding adaptativo se divide en dos pasos:

- Selección de una región de interés (ROI) en la imagen. Esta región puede ser de cualquier tamaño y forma.
- Selección del umbral para cada pixel dentro de la ROI. El umbral se calcula a partir de la distribución de los niveles de gris dentro de la ROI.

Existen varios métodos para calcular el umbral adaptativo, algunos de los mas conocidos son:

- Método de media global
- Método de la desviación estándar
- Método de Otsu

Cada uno de estos métodos tiene sus propios pros y contras y en función de la aplicación y el tipo de imagen, se puede elegir uno u otro. Además es posible modificar los siguientes parámetros para adecuar la librería a nuestro caso de uso, los más importantes son:

- **markerBorderBits:** El número de bits que se utilizan para representar el borde de un marcador. El borde de un marcador es el área blanca que rodea el patrón de código de barras en un marcador ARuco. El valor predeterminado es 4.
- **adaptiveThreshWinSizeMin and adaptiveThreshWinSizeMax:** El tamaño mínimo y máximo de la ventana utilizada para la umbralización adaptativa. La umbralización adaptativa es un método para determinar automáticamente el valor de umbral óptimo para una imagen. Estos parámetros se utilizan para especificar el tamaño de la ventana en píxeles que se utilizará para la umbralización adaptativa.
- **adaptiveThreshWinSizeMax:** Especifica el paso o incremento con el cual se variará el tamaño de la ventana utilizada en la umbralización adaptativa.

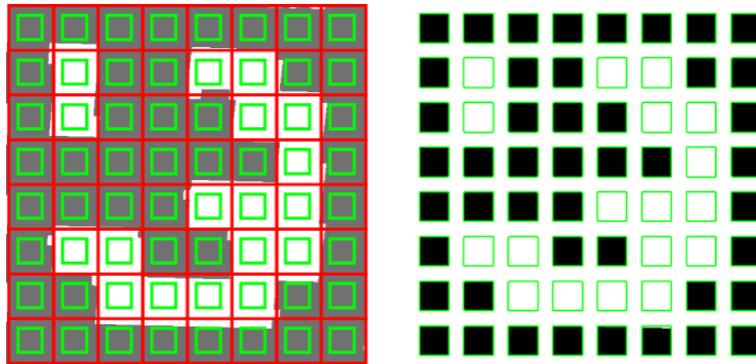


Figura 4.5: Proceso de extracción de Bits

- **adaptiveThreshConstant:** especificar una constante que se utilizará en el cálculo del valor de umbral para cada subregión de la imagen.
- **minMarkerPerimeterRate and maxMarkerPerimeterRate:** El porcentaje mínimo y máximo del perímetro de un marcador en relación con su área. Estos parámetros se utilizan para especificar el tamaño mínimo y máximo de los marcadores que se detectarán en la imagen.
- **minCornerDistanceRate:** La relación entre la distancia entre las esquinas de un marcador y su longitud de lado. Esto es utilizado para ignorar marcadores que tengan esquinas muy cercanas entre sí.
- **minDistanceToBorder:** La distancia mínima desde el borde de la imagen hasta el borde de un marcador. Esto se utiliza para ignorar marcadores que estén demasiado cerca del borde de la imagen.
- **minMarkerDistanceRate:** La relación entre la distancia entre los marcadores y su longitud de lado. Esto se utiliza para ignorar marcadores que estén demasiado cerca entre sí.

Extracción de Bits

Una vez que se han detectado los contornos de los posibles candidatos en una imagen, se analizan los bits extraídos de cada uno de estos para determinar si en efecto son marcadores o no. Para ello se somete cada sección de la imagen a una corrección de perspectiva. A continuación se subdivide el candidato en la cantidad previamente establecida en el diccionario de bits que componen cada marcador. Dado que en un bit pueden encontrarse píxeles de los bits contiguos o errores en la imagen capturada por la cámara, se establece el valor del bit en función de la desviación típica de los píxeles de dicho bit, como en el ejemplo de la Figura 4.5.

Identificación de Marcadores

Una vez determinados los bits de los que esta compuesto cada candidato a marcador es necesario determinar si el código extraído pertenece al diccionario y en caso de ser necesario aplicar algoritmos de corrección de errores. La primera operación consiste en determinar la cantidad de bits erróneos permitidos en el borde de un marcador, ya que todos los marcadores ARuco cuentan al menos con un bit de borde. En lo que a corrección de errores se refiere, cada diccionario cuenta con un límite teórico de bits que pueden ser corregidos.

Refinado de Esquinas

Una vez se han detectado e identificado todos los marcadores es posible realizar un refinado a nivel de subpíxel de las esquinas para favorecer a la precisión del sistema. Este ultimo paso es altamente costoso a nivel computacional, pero se recomienda en aplicaciones en las que prima la precisión como es el caso.

Esta librería trabaja con un Pinhole Camera Model lo que quiere decir que se considera como el origen de coordenadas el punto en el que todos los rayos de lux convergerían en una supuesta cámara estenopeica ideal.

Las coordenadas se definen de la siguiente forma: Z crece frente a la camara mientras que X e Y se encuentran en el plano ortogonal de Z. X aumenta de derecha a izquierda e Y de abajo a arriba.

4.7 Hardware

4.7.1 Impresión 3D

Se utilizaron dos impresoras 3D a lo largo del proyecto puesto que eran necesarios distintos requisitos para cada pieza. Los modelos anatómicos debido a su complejidad se imprimieron en una impresora Fuse 1+ 30W que utiliza polvo de nylon para llevar a cabo las piezas. Para los marcadores fiduciarios, se utilizo la Ultimaker 3, ya que se trata de figuras mas simples en las que la posibilidad de imprimir en distintos colores era especialmente importante.

4.7.2 Head Mounted Display (HMD)

Para la implementación del sistema de captación de imagen y tracking se utilizó un casco HTC Vive Pro 2.

Capítulo 5

Metodología y Gestión del proyecto

Este capítulo se centra en la organización del proyecto, la metodología usada y todo lo que a gestión de proyecto se refiere.

5.1 Metodología

Para el desarrollo del proyecto se ha decidido por implementar una metodología *Ágil*. Este término se refiere a una metodología regida por una serie de principios que permiten ajustar la forma de trabajo a las condiciones del proyecto. Se debe priorizar a los individuos e interacciones sobre procesos y herramientas, asegurando no perder el valor humano de la comunicación no verbal en el proceso. También se debe atender antes a soluciones funcionales sobre la documentación exhaustiva. Lejos de procurar dejar de lado la documentación, es necesario tener en cuenta que el valor del desarrollo está en la funcionalidad del mismo, por lo que sacrificar tiempo de trabajo por documentar en exceso un proyecto puede acarrear resultados no deseados. Otro pilar en el que se basa esta metodología es la respuesta al cambio sobre los planes preestablecidos, lo que asegura la versatilidad del proyecto aumentando las posibilidades de éxito. Dada la naturaleza incremental e iterativa de la metodología *Ágil* es posible dividir el ciclo de vida del proyecto en tareas para facilitar el desarrollo. Esta aproximación permite aplicar el principio de “Divide y Vencerás” fragmentando la estimación de las tareas y su desarrollo con el fin de reducir la dificultad en el desarrollo. En la Tabla 5.1 se pueden ver todas las tareas que se llevaron a cabo junto con la estimación en horas de la misma. Destacar que dada la naturaleza del trabajo de fin de grado, se asume un único recurso humano, y por lo tanto no nos referiremos a las horas como horas por hombre o h x h.

Para llevar a cabo las tareas se optó por un desarrollo en *Sprints*. Los sprints son los períodos de tiempo en los que se llevan a cabo las tareas. Estos se acotaron en el tiempo, dándoles siempre una longitud determinada, y a su vez se acotaron en funcionalidad procurando siempre que finalicen alcanzando algún hito del proyecto. Estos bloques o *Sprints* se fueron

Nº	Funcionalidad	Estimación (H)
1	Importar datos en formato DICOM en 3D Slicer.	2
2	Generar secciones para poder exportar de 3DSlicer.	40
3	Arreglar el modelo para que se viables para impresión.	12
4	Imprimir modelo.	15
5	Instalar librerías necesarias para el desarrollo.	4
6	Familiarizarse con la librería.	12
7	Prototipar pruebas para tests.	40
8	Implementar herramienta para tracking del marcador.	40
9	Testear prototipos.	25
10	Generar modelo a partir el prototipo final.	33
11	Imprimir prototipo final.	16
12	Instalar y compilar Exposure Render.	80
13	Familiarizarse con el software y sistema de trabajo.	19
14	Integrar software de tracking en Exposure Render	24
15	Instalar software necesario para el HMD	2
16	Familiarizarse con el uso del HMD	24
17	Ejecutar pruebas de passthrough	2
18	Integrar passthrough en Exposure Render	32

Tabla 5.1: Tareas llevadas a cabo

definiendo a lo largo del proyecto, adaptándose a las necesidades y riesgos del mismo.

5.2 Sprints

Tras introducir la metodología usada, vamos a detallar los Sprints que tuvieron lugar en el proyecto, las tareas, los riesgos y recursos disponibles para cada uno.

5.2.1 Extracción de un modelo a partir de un TC

En este sprint, se tuvo como objetivo el desarrollo de las tareas 1, 2, 3 y 4. Dado que en el estudio previo se encontraron varios caminos disponibles para alcanzar las tareas objetivo, el sprint se consideró viable en todo momento. Este sprint se llevaron a cabo dentro del tiempo estimado de 2 semanas. El principal riesgo fue la complejidad de la pieza, que en una impresora 3D de filamento común podría haber alargado la impresión, o incluso fallar en el proceso. Esto se minimizó utilizando la impresora Fuse 1+ 30W capaz de imprimir modelos más complejos a pesar de tener una superficie de impresión más reducida.

5.2.2 Diseño del marcador fiduciario y software de tracking

Durante este Sprint, se llevaron a cabo las tareas de la 5 a la 10. En este caso el mayor riesgo fue el tiempo de prototipado, ya que para una única pieza a imprimir, el tiempo de impresión oscila las 15 horas. Para minimizar este riesgo se optó por realizar prototipos sobre papel como se comenta en la ejecución del proyecto. El tiempo estimado para este sprint fue de 3 semanas y media, pero debido a la dificultad de las tareas 7 y 8 se alargó a 5 semanas.

5.2.3 Integrar solución de tracking en Exposure Render

Las tareas de la 11 a la 14 se llevaron a cabo en este Sprint. La compilación e instalación de todo el software necesario para ejecutar Exposure Render fue el riesgo por excelencia de este sprint. A pesar de esto se realizó dentro del tiempo estimado para ello ya que se conocía en el momento de la planificación.

5.2.4 Integrar passthrough en Exposure Render

Se trata del último sprint del proyecto, se llevaron a cabo las tareas de la 14 a la 18. En este sprint, nos encontramos con el mayor imprevisto del proyecto. El **SDK** del **HMD** no funcionaba correctamente, lo que imposibilitaba la reconstrucción de imágenes y causó un gran retraso en el proyecto respecto a la planificación inicial.

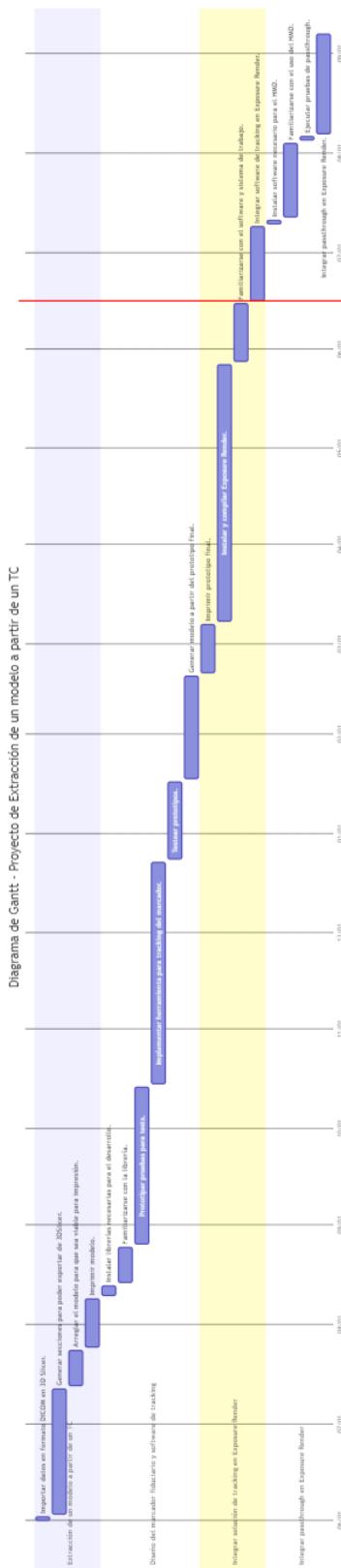


Figura 5.1: Diagrama de la planificación del proyecto por sprints

Capítulo 6

Ejecución del proyecto

Este capítulo tiene como objetivo tratar el desarrollo del proyecto en sí mismo, así como discutir las opciones disponibles durante el progreso y las decisiones tomadas para llevarlo a cabo.

6.1 Análisis

Se llevó a cabo un estudio para definir la hoja de ruta del proyecto. Dada la problemática a solventar, este trabajo alcanza a tocar áreas bien diferenciadas entre si que se pueden destacar como los pasos a seguir del mismo:

- Extracción de volúmenes 3D a partir de un [TC](#) válidos para su impresión.
- Diseñar un marcador fiduciario que permita el seguimiento de una pieza en 3 dimensiones y un método de acople al volumen previamente impreso.
- Implementar una solución que permita el seguimiento de dicho marcador.
- Integrar la solución sobre un [HMD](#).

Fruto de la investigación surge el artículo de [Moreta-Martinez et al.](#); que expone una solución existente a los objetivos de este trabajo mediante el uso de software bajo licencia o de pago. Es por ello que se toma una aproximación similar al problema, sobre todo en las fases iniciales, para la generación de los volúmenes a pesar de implementar una solución propia para lo que a seguimiento se refiere. Este estudio también permitió especificar los requisitos necesarios para el software de tracking.

Uno de los principales requisitos debe ser la robustez del sistema frente a las occlusiones del marcador. Es necesario que el seguimiento sea posible a pesar de la occlusión parcial del marcador. Además, es preciso que a partir de una fuente de vídeo se puedan extraer las coordenadas del marcador, así como su rotación en el espacio. Destacar también que el seguimiento

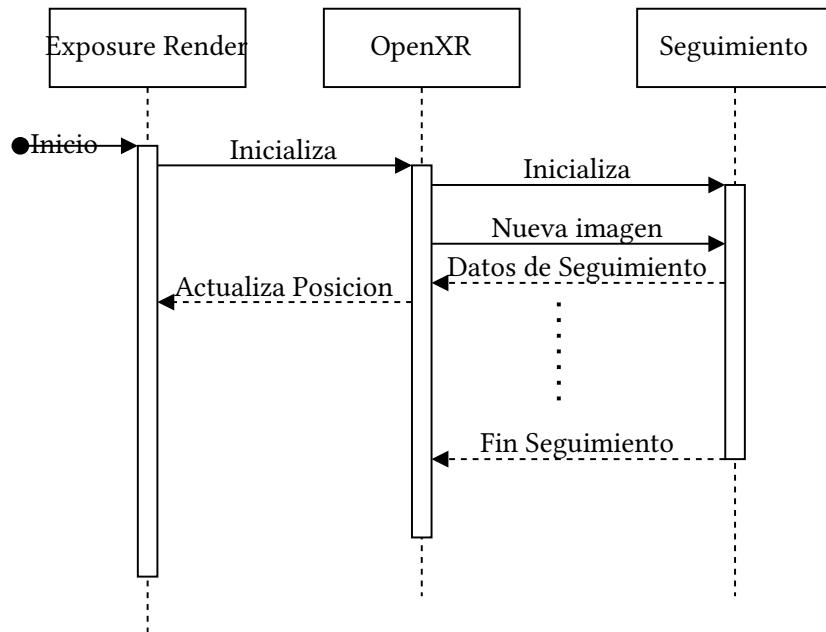


Figura 6.1: Flujo de integración del seguimiento en Exposure Render.

debe ocurrir en un segundo plano, entorpeciendo lo menos posible las operaciones del hilo principal de ejecución, ya que parte de estas operaciones tiene una latencia crítica.

Este trabajo nace como un desarrollo del proyecto troncal de Iglesias-Guitian et al., y como tal, debe ceñirse a ciertas pautas del mismo. Exposure Render cuenta con un módulo de realidad virtual, en el que se integrará la solución de tracking para lograr un control mas natural sobre el modelo a tratar, moviéndose en las imágenes renderizadas a la par que se mueve en la realidad, como se muestra en el diagrama de secuencia de la Figura 6.1.

6.2 Generación de volúmenes a partir de TC

Con el fin de facilitar la validación del progreso del proyecto, se utilizó una TC de pruebas. Dichos datos contienen la sección superior (hombros y cabeza) de un sujeto, mostrado en la Figura 6.2a. Durante el desarrollo se sugirió como posible caso práctico seleccionar el cráneo del sujeto en los datos de prueba y trabajar en la alineación 3D sobre el mismo.

Con el fin de seleccionar una sección concreta para exportar, se utilizaron las herramientas para segmentar volúmenes de Slicer3D. Al abrir el programa se pueden ver las vistas, en las que se representará el TC una vez cargado, como se muestra en la captura de pantalla de la Figura 6.4. Se utilizó principalmente la herramienta de “Thresholding” que permite seleccionar partes del modelo cuyas intensidades se comprenden en un intervalo o “threshold” (ver Figura 6.2c). Posteriormente, para la eliminación de las partes del modelo no deseadas, se



Figura 6.2: TC de partida y obtención del modelo 3D final con 3DSlicer.

utilizó la herramienta de borrado hasta alcanzar el volumen deseado.

6.3 Impresión 3D del volumen

Dado que el modelo no se genera a partir de figuras geométricas previas, es posible que presente geometrías rotas. Las cuales a la hora del Slicing provocarían errores y no permitirían que se genere el GCODE correctamente. Por ello es necesario importar el modelo en un programa que nos facilite arreglar estas geometrías como es Meshmixer. En la Figura 6.3 se aprecia el modelo exportado, y cada marcador corresponde a errores producto de la exportación. Una vez reparados, se procede a imprimir la pieza.

Como se comenta en el Capítulo 4, para una pieza con una geometría tan compleja, se requeriría una gran cantidad de soportes. Debido a esto, se optó por la impresora Fuse 1 para la impresión de este modelo. A diferencia de una impresora 3D al uso, esta impresora utiliza un láser para fijar capa a capa el polvo de nylon, lo que garantiza una gran resolución en la pieza final y una gran durabilidad de la misma.

Posterior al trabajo de impresión es necesario retirar el material sobrante en la cámara de

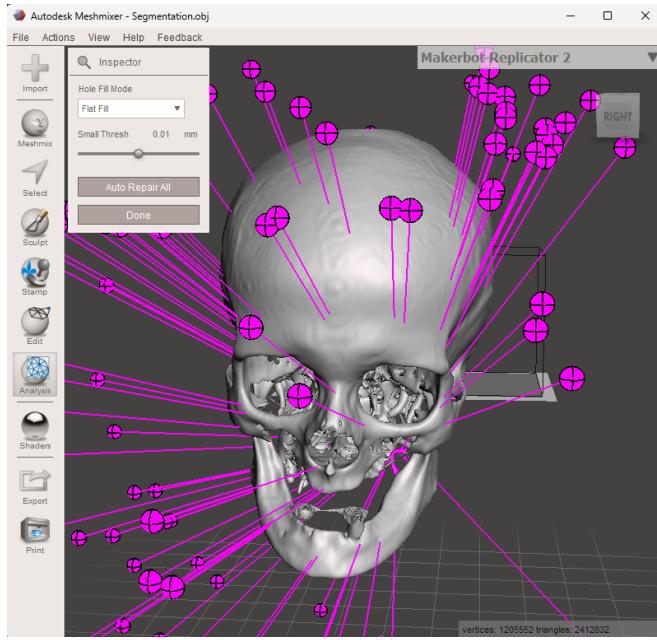


Figura 6.3: Modelo de cráneo 3D con geometrías erróneas.

recuperación que cuenta con distintos utensilios para evitar malgastar el material sobrante ya que puede ser reutilizado.

6.4 Desarrollo del marcador fiduciario

Obtener la posición y rotación de una figura desconocida en el espacio es uno de los problemas principales a la hora de implementar soluciones de realidad virtual o aumentada, ya que requiere encontrar correspondencias entre objetos conocidos en el espacio y sus proyecciones en el vídeo.

Si bien existen aproximaciones que buscan puntos claves de las figuras o reconocen sus geometrías mediante técnicas de visión artificial e inteligencia artificial, se optó por el uso de marcadores fiduciarios por varios motivos.

Primeramente, permite replicar el seguimiento del objeto independientemente del hardware utilizado, ya que una vez calibrada la cámara no se requiere ningún otro tipo de ajuste en el sistema. Otra ventaja es la robustez del sistema, ya que permite mantener el seguimiento a pesar de que parte del marcador se encuentre oculto o no esté en el campo de visión de la cámara. Dados los recursos disponibles, se optó por utilizar la librería ARuco para generar y seguir el marcador.

ARuco a la hora de detectar la posición de un marcador, trabaja con un modelo de coordenadas pin-hole, donde las coordenadas y rotación de los objetos detectados se expresan en



(a) Pieza en el proceso de recuperación
del material.

(b) Pieza final.

Figura 6.4: Proceso de recuperación de material.

función de la posición de la cámara. El calibrado de la cámara permite determinar la proyección de cualquier punto en las 3 dimensiones del espacio en el sensor de la cámara. En una cámara ideal, un punto 3D (X, Y, Z) en el espacio se proyectaría en el píxel:

$$x = \frac{X \cdot fx}{Z} + cx \quad y = \frac{Y \cdot fy}{Z} + cy$$

Donde:

- fx, fy : Es la longitud focal de la lente de la cámara en ambos ejes.
- cx, cy : Es el centro óptico del sensor (expresado en píxeles).
- $k1, k2, p1, p2, k3$: Son los coeficientes de distorsión.

Asumiendo que la ubicación tridimensional del punto con respecto al sistema de referencia de la cámara es conocida. Si se desea conocer la proyección de un punto referido a un sistema de referencia arbitrario, entonces deben mencionarse parámetros extrínsecos. Los parámetros extrínsecos consisten básicamente en las rotaciones tridimensionales ($Rvec = Rx, Ry, Rz$) y las traslaciones tridimensionales ($Tvec = Tx, Ty, Tz$) requeridas para trasladar el sistema de referencia de la cámara al sistema arbitrario. Los elementos de rotación se expresan mediante la fórmula de Rodrigues [21], por lo que es posible obtener la matriz de rotación equivalente de 3x3 utilizando la función cv::Rodrigues() de OpenCV.

Cada marcador detectado devuelve como coordenadas la esquina superior izquierda del mismo, o lo que se etiqueta en el ejemplo de la Figura 6.5 como *corner 0*, en forma de ($Rvec = Rx, Ry, Rz$) como vector de rotación y ($Tvec = Tx, Ty, Tz$) como vector de translación.

Detectar un solo marcador puede fallar por diferentes razones, como malas condiciones de iluminación, movimiento rápido de la cámara, obstrucciones, etc. Para superar ese problema, ArUco permite el uso de tablas de marcadores como la mostrada en la Figura 6.6. Cada tabla de marcadores está compuesta por varios marcadores en ubicaciones conocidas. Presentan dos ventajas principales. Primero, dado que hay más de un marcador, es menos probable perderlos todos al mismo tiempo. Segundo, cuanto más marcadores se detecten, más puntos están disponibles para calcular los parámetros extrínsecos de la cámara. Como consecuencia, se obtiene una mayor precisión.

Debido a la versatilidad de las piezas con las que se pretende usar el marcador, se implementó priorizando eliminar las occlusiones del marcador por la pieza, por este motivo se diseñó como un cubo, de forma que al menos una cara sería visible en todo momento.

Se llevaron a cabo pruebas con distintos diccionarios de marcadores, modificando las tolerancias para los márgenes entre marcadores y los bordes del cubo con el fin de poder mantener unas dimensiones manejables sin comprometer el tamaño de cada marcador individual. Final-

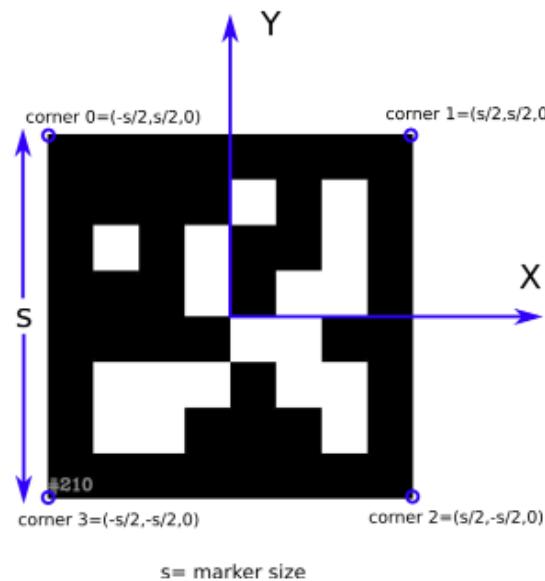


Figura 6.5: Esquema de un marcador.

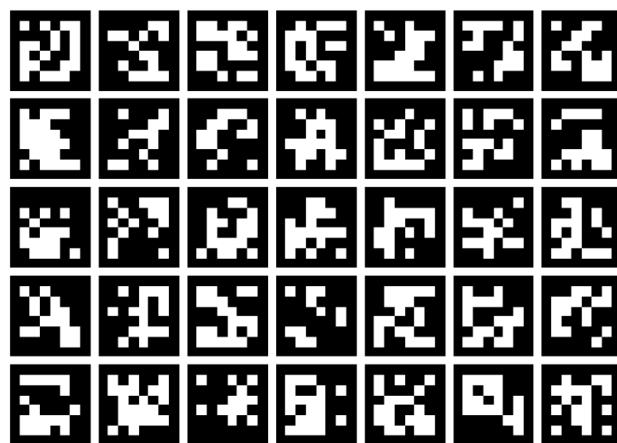


Figura 6.6: Esquema de una tabla de marcadores.

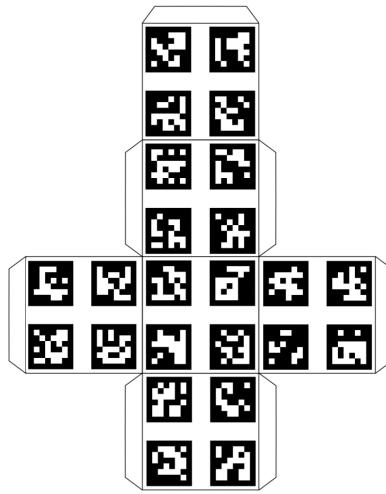


Figura 6.7: Layout de las caras del marcador fiduciario.

mente, fruto de los tests, se llegó al cubo de la Figura 6.8, que cuenta con una matriz 2×2 de marcadores del mismo diccionario en cada cara.

El cubo se descompone como se puede ver en la Figura 6.7. En un primer momento la detección devuelve la posición y rotación de cada cara de forma individual. Dado que se busca la posición del centro del cubo, sobre cada valor obtenido, se aplica una transformada para obtener el vector de translación del centro del cubo. Se calcula como la posición original menos la distancia del lado de media tabla en cada eje, como se ve a continuación :

```

1 cv::Vec3d moveAxis(cv::Vec3d& tvec, cv::Vec3d rvec, double
2   distance, int axis)
3 {
4   cv::Mat rotationMatrix, rotationMatrixTransposed;
5   Rodrigues(rvec, rotationMatrix);
6   rotationMatrixTransposed = rotationMatrix.t();
7   double* rz = rotationMatrixTransposed.ptr<double>(axis); // 
8   x=0, y=1, z=2
9   tvec[0] -= rz[0] * distance;
10  tvec[1] -= rz[1] * distance;
11  tvec[2] -= rz[2] * distance;
12  return tvec;
13 }
14 tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 0);
15 tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 1);
16 tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 2);

```

No obstante, cada cara precisa de una rotación específica para mantener la rotación del

cubo congruente con el resto de caras, excepto una que se tome como referencia:

```

1
2
3 void cubeCoordinates(int id, cv::Vec3d& rvecs, cv::Vec3d& tvecs,
4   float sideLength)
5 {
6   tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 0);
7   tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 1);
8   tvecs = moveAxis(tvecs, rvecs, -SIDELENGTH, 2);
9   switch (id)
10  {
11    case 1://cara 1
12      rvecs = rotateXAxis(rvecs, -M_PI / 2);
13      break;
14    case 2://cara 2
15      rvecs = rotateXAxis(rvecs, M_PI);
16      break;
17    case 3://cara 3
18      rvecs = rotateYAxis(rvecs, M_PI / 2);
19      rvecs = rotateZAxis(rvecs, M_PI);
20      break;
21    case 4://cara 4
22      rvecs = rotateXAxis(rvecs, M_PI);
23      rvecs = rotateYAxis(rvecs, -M_PI / 2);
24      break;
25    case 5://cara 5
26      rvecs = rotateXAxis(rvecs, M_PI / 2);
27      break;
28    default://La cara 0 no precisa rotar
29      break;
30  }

```

En caso de que únicamente se detectase una cara del cubo, los vectores de rotación y translación finales serían los obtenidos llegados a este punto, pero es posible que se detecten hasta 3 caras en una única imagen. En estos casos se selecciona como resultado la media de los valores obtenidos en cada cara como se ve en la figura Figura 6.9.

Hasta ahora, se ha descrito el comportamiento de la aplicación para una única imagen. Sin embargo, aplicar este comportamiento a una fuente de vídeo apenas modifica el comportamiento. Una vez obtenidos los valores, simplemente hay que esperar a la siguiente imagen y volver a ejecutar el bucle de detección sobre la siguiente hasta que se desee terminar el programa. En la figura Figura 6.10 se puede apreciar el flujo aquí descrito.

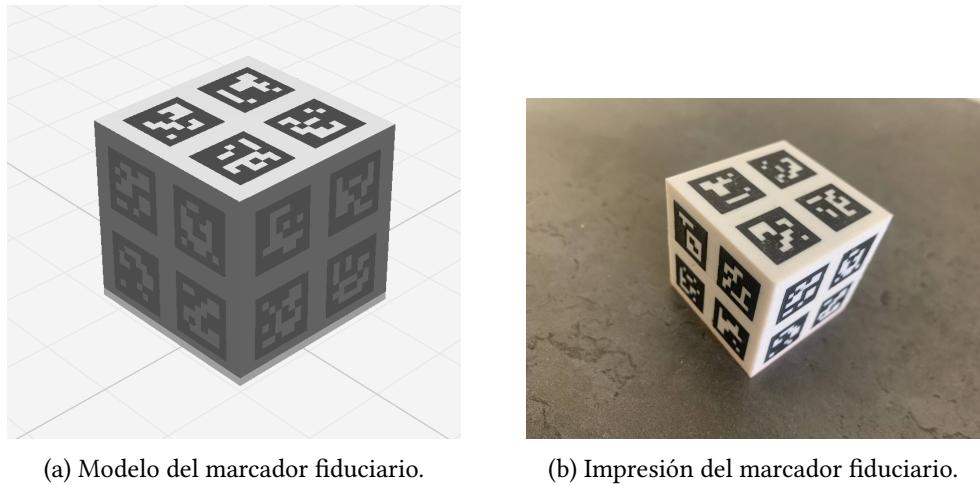


Figura 6.8: Diseño final del marcador fiduciario.

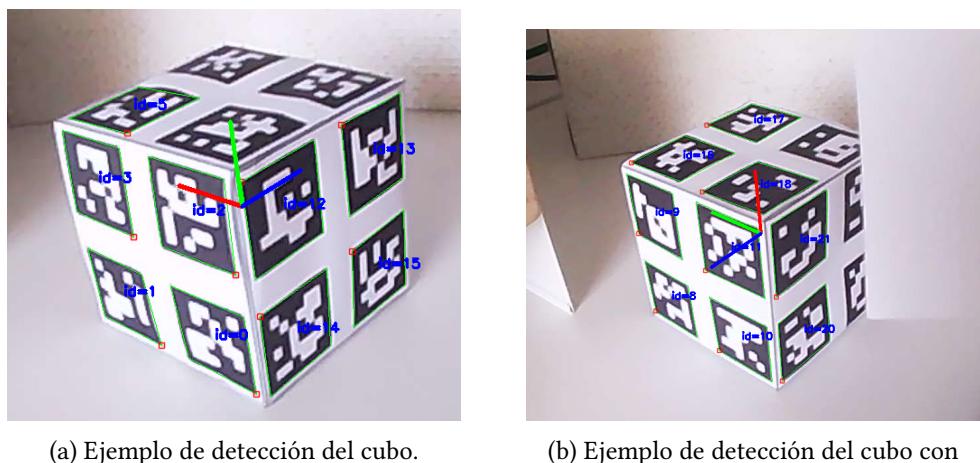


Figura 6.9: Imágenes sobre las que se estima la pose del cubo.

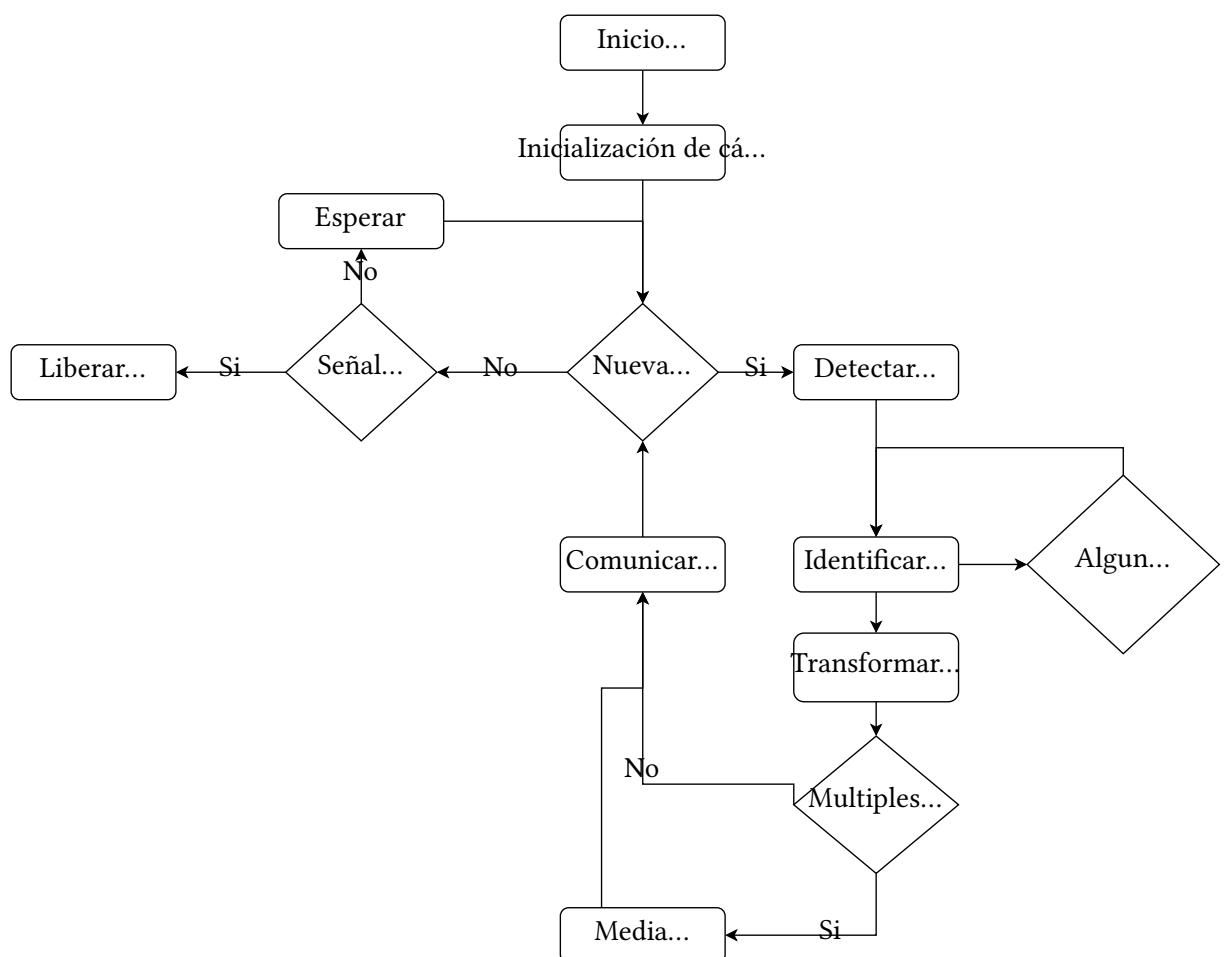


Figura 6.10: Diagrama de flujo de la solución de tracking.

6.5 Implementación del Passthrough en Exposure Render

Para la obtención de imágenes sobre las que poder trabajar se utilizaron las propias cámaras frontales del HTC Vive Pro 2. Se trata de un par de cámaras colocadas longitudinalmente a lo largo del frontal del casco, que permiten su uso en aplicaciones de realidad aumentada y realidad mixta. Para acceder a estas cámaras se debe hacer uso de SRworks C++ SDK.

Este software en el momento del desarrollo del trabajo, en su versión nativa, presentaba errores que imposibilitaron la reconstrucción de la imagen para ser visualizada en el casco.

6.6 Integración del software de tracking en Exposure Render

Durante la integración, se procuró mantener al mínimo la latencia introducida al analizar cada imagen. Por ello se implementó la solución como un thread. Este thread está sincronizado con la tasa de refresco de las cámaras; de forma que en cuanto una nueva imagen es recibida, se procesa, se devuelven los datos de seguimiento pertinentes, y esperan a la siguiente imagen.

Capítulo 7

Conclusiones y trabajo futuro

Como cierre del trabajo, cabe reflexionar sobre el desarrollo del mismo así como sobre su estado actual, y su futuro. Dado por finalizado el trabajo, se han alcanzado buena parte de los objetivos fijados en su concepción:

- Se identificaron métodos para la extracción de secciones de [TC](#) así como para el refinado de las mismas. También se analizaron las posibilidades de impresión optimizando el uso de los materiales y de las capacidades de las impresoras. Esto ha permitido trabajar sobre piezas en un nivel más visual e interactivo.
- Se ha llevado a cabo un estudio de las soluciones existentes para el seguimiento 3D realizando pruebas de las mismas sobre los datos propios para extraer las ventajas y desventajas de cada solución en un nivel práctico.
- Se han diseñado e impreso una gran cantidad de marcadores fiduciarios sobre los que se han realizado pruebas de forma iterativa con el fin de refinar y optimizar el diseño hasta obtener un resultado que satisface las características del proyecto. Todo esto mediante el uso de software libre creando así un proceso adaptable a otros casos de uso.
- Se ha integrado la solución en el proyecto de [Iglesias-Guitian et al.](#) desvirtualizando un posible caso de uso.

7.0.1 Enriquecimiento Formativo

Es imprescindible destacar la vertiente formativa que presenta este trabajo dada su naturaleza como trabajo de fin de grado. El autor ha tenido la posibilidad de trabajar sobre una serie de campos de los más variados que comprenden la imagen médica, la impresión 3D, la visión artificial, la realidad virtual, el renderizado en un motor de trazado de rayos. Por otra parte, mencionar la exposición a un ambiente investigador en el [CITIC](#) del que poder empaparse de

la forma de trabajar y la cooperación entre iguales. También se trató de una primera puesta en práctica de los conceptos aprendidos sobre la gestión de proyectos que resultó enriquecedora.

7.0.2 Trabajo futuro

En la actualidad el proyecto tiene distintas vertientes que pueden ser desarrolladas en el futuro. Por una parte esta la mejora del sistema actual de seguimiento, si bien este en la actualidad es completamente funcional, en sistemas en los que la tasa de refresco de la cámara no es lo suficientemente alto el seguimiento puede dar una sensación de escalonado. Una posible solución sería aplicar algún tipo de filtrado de señales como puede ser un filtro de Kalman, que bien implementado, ayudaría a estimar pasos intermedios entre imágenes para proporcionar una experiencia mas confortable [22].

Una segunda vertiente de mejora, consistiría en reconstruir sobre las imágenes extraídas del casco, un ambiente de realidad aumentada, en el que se pudieran ver los modelos extraídos de las TC sobre las mismas, donde aprovechándose de los desarrollos de la interfaz de realidad virtual del proyecto de Iglesias-Guitian et al. se pudiera trabajar con la pieza virtual modificando sus parámetros visuales.

Para terminar, otro posible desarrollo, consistiría en tratar de aumentar la resolución de las imágenes de las cámaras del casco, utilizando las imágenes de cada cámara para crear una imagen compuesta por ambas que permita una mejor detección del marcador fiduciario.

Apéndices

Lista de acrónimos

API Interfaz de programación de aplicaciones. 14, 15

AR Realidad Aumentada. 7

CAD Computer-Aided Design. 6

CITIC Centro de Investigación en Tecnologías de la información y Comunicaciones. 37

DICOM Digital Imaging and Communication In Medicine. 13, 22

DVR Direct Volume Rendering. 9

GPU Graphical Processing Unit. 9

HMD Head Mounted Display. ii, 2, 10, 20, 22, 23, 25

MCRT Monte Carlo Ray Tracing. 9

MR Realidad Mixta. 7

OpenCV Open Source Computer Vision Library. 15

RLS Recursive Least Squares. 9

SDK Software Development Kit. 23

TC Tomografía Computerizada. ii, iii, 2, 4–6, 10, 23, 25–27, 37, 38

VR Realidad Virtual. 7

XR Realidad Extendida. 7, 14

Bibliografía

- [1] M. Venkatesan, H. Mohan, J. R. Ryan, C. M. Schürch, G. P. Nolan, D. H. Frakes, and A. F. Coskun, “Virtual and augmented reality for biomedical applications,” *Cell Reports Medicine*, vol. 2, no. 7, p. 100348, Jul. 2021. [Online]. Available: <https://doi.org/10.1016/j.xcrm.2021.100348>
- [2] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, Jun. 2014. [Online]. Available: <https://doi.org/10.1016/j.patcog.2014.01.005>
- [3] C. P. Botha, B. Preim, A. E. Kaufman, S. Takahashi, and A. Ynnerman, “From individual to population: Challenges in medical visualization,” in *Mathematics and Visualization*. Springer London, 2014, pp. 265–282. [Online]. Available: https://doi.org/10.1007/978-1-4471-6497-5_23
- [4] T. Sielhorst, M. Feuerstein, and N. Navab, “Advanced medical displays: A literature review of augmented reality,” *Journal of Display Technology*, vol. 4, no. 4, pp. 451–467, Dec. 2008. [Online]. Available: <https://doi.org/10.1109/jdt.2008.2001575>
- [5] S. H. Muñiz and M. M. Casanovas, “Introducción a la tomografía computarizada,” *Revista Española de Medicina Nuclear*, vol. 25, no. 3, pp. 206–214, 2006.
- [6] G. Kontaxakis, J. J. Vaquero López, and A. Santos, “Reconstrucción de imagen en tomografía por emisión de positrones,” 2002.
- [7] M. J. Willemink, P. A. de Jong, T. Leiner, L. M. de Heer, R. A. J. Nievelstein, R. P. J. Budde, and A. M. R. Schilham, “Iterative reconstruction techniques for computed tomography part 1: Technical principles,” *European Radiology*, vol. 23, no. 6, pp. 1623–1631, Jan. 2013. [Online]. Available: <https://doi.org/10.1007/s00330-012-2765-y>
- [8] M. Zahera, “La fabricación aditiva, tecnología avanzada para el diseño y el desarrollo de productos,” 2012.

- [9] P. Milgram and F. Kishino, “A taxonomy of mixed reality visual displays,” *IEICE Transactions on Information and Systems*, vol. 77, pp. 1321–1329, 1994.
- [10] R. T. Azuma, “A survey of augmented reality,” *Presence: Teleoperators and Virtual Environments*, vol. 6, no. 4, pp. 355–385, Aug. 1997. [Online]. Available: <https://doi.org/10.1162/pres.1997.6.4.355>
- [11] T. Kroes, F. H. Post, and C. P. Botha, “Exposure render: An interactive photo-realistic volume rendering framework,” *PLoS ONE*, vol. 7, no. 7, p. e38586, Jul. 2012. [Online]. Available: <https://doi.org/10.1371/journal.pone.0038586>
- [12] N. Max, “Optical models for direct volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, Jun. 1995. [Online]. Available: <https://doi.org/10.1109/2945.468400>
- [13] J. Díaz, T. Ropinski, I. Navazo, E. Gobbetti, and P.-P. Vázquez, “An experimental study on the effects of shading in 3d perception of volumetric models,” *The Visual Computer*, vol. 33, no. 1, pp. 47–61, Sep. 2015. [Online]. Available: <https://doi.org/10.1007/s00371-015-1151-6>
- [14] R. Englund and T. Ropinski, “Evaluating the perception of semi-transparent structures in direct volume rendering techniques,” in *SIGGRAPH ASIA 2016 Symposium on Visualization*. ACM, Nov. 2016. [Online]. Available: <https://doi.org/10.1145/3002151.3002164>
- [15] F. Lindemann and T. Ropinski, “About the influence of illumination models on image comprehension in direct volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1922–1931, Dec. 2011. [Online]. Available: <https://doi.org/10.1109/tvcg.2011.161>
- [16] J. A. Iglesias-Guitian, P. Mane, and B. Moon, “Real-time denoising of volumetric path tracing for direct volume rendering,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 28, no. 7, pp. 2734–2747, Jul. 2022. [Online]. Available: <https://doi.org/10.1109/tvcg.2020.3037680>
- [17] R. Moreta-Martinez, D. García-Mato, M. García-Sevilla, R. Pérez-Mañanes, J. A. Calvo-Haro, and J. Pascau, “Combining augmented reality and 3d printing to display patient models on a smartphone,” *Journal of Visualized Experiments*, no. 155, Jan. 2020. [Online]. Available: <https://doi.org/10.3791/60618>

- [18] Q. Yan, H. Dong, J. Su, J. Han, B. Song, Q. Wei, and Y. Shi, “A review of 3d printing technology for medical applications,” Aug 2018. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2095809917306756>
- [19] F. Chana-Rodríguez, R. P. Mañanes, J. Rojo-Manaute, P. Gil, J. M. Martínez-Gómiz, and J. Vaquero-Martín, “3d surgical printing and pre contoured plates for acetabular fractures,” *Injury*, vol. 47, no. 11, pp. 2507–2511, Nov. 2016. [Online]. Available: <https://doi.org/10.1016/j.injury.2016.08.027>
- [20] A. G. Alvarez, P. L. Evans, L. Dovgalski, and I. Goldsmith, “Design, additive manufacture and clinical application of a patient-specific titanium implant to anatomically reconstruct a large chest wall defect,” *Rapid Prototyping Journal*, vol. 27, no. 2, pp. 304–310, jan 2021. [Online]. Available: <https://doi.org/10.1108%2Frpj-08-2019-0208>
- [21] J. E. Mebius, “Derivation of the euler-rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations,” 2007.
- [22] G. F. Welch, “Kalman filter,” *Computer Vision: A Reference Guide*, pp. 1–3, 2020.