



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Visualización inmersiva geoespacial mediante realidad
aumentada en visores tipo Cardboard*

Grado en Ingeniería Mecánica

ALUMNO: Pablo Rodríguez Torrente

DIRECTOR: Miguel Rodelgo Lacruz

CURSO ACADÉMICO: 2025-2026

Universida_{de}Vigo



Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

*Visualización inmersiva geoespacial mediante realidad
aumentada en visores tipo Cardboard*

Grado en Ingeniería Mecánica
Intensificación en Tecnología Naval
Cuerpo General

UniversidadeVigo

RESUMEN

Este Trabajo de Fin de Grado presenta el diseño y desarrollo, mediante el motor Unity, de un prototipo de bajo coste para la visualización inmersiva de información geoespacial marítima, formado por un visor de tipo Cardboard y un teléfono móvil Android. El objetivo principal del trabajo ha sido estudiar la viabilidad de representar datos AIS de embarcaciones dentro de un entorno binocular, analizando qué configuración de visualización resulta más adecuada para su uso en este tipo de dispositivos.

Para ello, se desarrollaron y compararon distintas alternativas de visualización estereoscópica, evaluando configuraciones basadas en cámaras virtuales, superficies de representación y una cámara estéreo externa. Paralelamente, se implementó el acceso a los sensores del dispositivo móvil, en particular GPS y brújula, con el fin de posicionar geoespacialmente los marcadores dentro de la escena sin recurrir a plataformas avanzadas de realidad aumentada. Asimismo, se integró una cadena de recepción de datos AIS reales mediante la aplicación AIS Catcher, un receptor RTL-SDR y una antena externa, permitiendo utilizar mensajes NMEA como entrada del sistema.

Los resultados obtenidos permiten valorar las posibilidades y limitaciones de este enfoque, especialmente en relación con la inmersión visual, la calibración, la latencia y la precisión en el posicionamiento de la información representada.

PALABRAS CLAVE

Realidad Aumentada, Unity, AIS, Visualización Geoespacial, Cardboard

AGRADECIMIENTOS

Primero de todo agradecer a mi tutor D. Miguel Rodelgo Lacruz, por su dedicación y paciencia durante todo el desarrollo del trabajo, aportando conocimientos y soluciones que han sido determinantes para el resultado del mismo.

A la promoción 426-156, lo mejor que me ha dado la Armada, han hecho que estos 5 años sean inolvidables.

Finalmente agradecer a mi familia y amigos, por ser el pilar fundamental en mi vida y acompañarme durante todo el camino.

CONTENIDO

Contenido	1
Índice de Figuras	3
Índice de Tablas.....	5
1 Introducción y objetivos	7
1.1 Contexto y motivación	7
1.2 Objetivos	8
1.3 Estructura de la memoria	8
2 Estado del arte	11
2.1 Realidad Aumentada (AR).....	11
2.1.1 Definición	11
2.1.2 AR Geoespacial	12
2.1.3 Hardware.....	13
2.1.4 Sensores	15
2.1.5 Visualización	16
2.1.6 Dispositivos comerciales	17
2.1.7 Teléfono móvil + Visor Tipo Cardboard	19
2.2 AIS	19
2.2.1 Contexto.....	19
2.2.2 Funcionamiento	20
3 Desarrollo del TFG.....	23
3.1 Fases de desarrollo	23
3.2 Entorno de desarrollo: Unity	23
3.3 Visualización inmersiva.....	26
3.3.1 Visualización inmersiva con visor tipo Cardboard.....	26
3.3.2 Visualización mediante proyección sobre pantalla virtual	27
3.3.3 Visualización mediante proyección monoscópica directa	30
3.3.4 Visualización mediante proyección estereoscópica.....	33
3.4 Creación de marcadores geoespaciales	35
3.4.1 Marcador 3D geoespacial	35
3.4.2 Visualización de marcadores	36
3.4.3 Anclaje geoespacial	37
3.5 Captación e integración de información AIS	39
3.5.1 Sintonizador RTL-SDR	40

3.5.2 AIS Catcher	40
3.5.3 Procesamiento de la información AIS en Unity	41
4 Resultados y validación.....	45
4.1 Resultados de visualización	45
4.1.1 Criterios de evaluación	45
4.1.2 Resultados visualización mediante proyección sobre pantalla virtual.....	46
4.1.3 Resultados visualización mediante proyección monoscópica directa	47
4.1.4 Resultados visualización mediante proyección estereoscópica	49
4.2 Resultados geoespaciales	50
5 Conclusiones y líneas futuras	53
5.1 Conclusiones y revisión de los objetivos	53
5.2 Líneas futuras	54
6 Bibliografía.....	57
Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales	61
Anexo II: Reflexiones Éticas y Sociales	63
Anexo III: Primera Versión Código GeoMarkerText	65
Anexo IV: Código GPS Compass Debug.....	69
Anexo V: Código AIS MarkerWorldAnchor	71
Anexo VI: Código AIS UDP Receiver.....	77
Anexo VII: Código Cámara Solución Monoscópica Directa	83
Anexo VIII: Uso Inteligencia Artificial	87

ÍNDICE DE FIGURAS

Figura 2-1 Visión HMD utilizada en aviación militar [9].....	13
Figura 2-2 Google Cardboard [10]	14
Figura 2-3 VR Box [11]	14
Figura 2-4 Holo Lens 2 [15].....	16
Figura 2-5 Video See-Through vs Optical See-Through [18].....	17
Figura 2-6 Representación AIS [25]	20
Figura 2-7 AIS signal [27].....	21
Figura 3-1 Vista general de la interfaz de Unity (Elaboración propia)	24
Figura 3-2 Versiones del editor Unity disponibles en Unity Hub (Elaboración propia).....	25
Figura 3-3 Unity installs (Elaboración propia).....	25
Figura 3-4 Vista estereoscópica mostrada en la pantalla del teléfono móvil mediante la aplicación Cardboard, ilustrando la disposición binocular empleada para su observación con visor (Elaboración propia).	27
Figura 3-5 Interfaz dos cámaras y un plano (Elaboración propia)	28
Figura 3-6 Esquema 1 (Elaboración propia)	28
Figura 3-7 Inspector Cámara izq (Elaboración propia).....	29
Figura 3-8 Salida visual de la configuración basada en dos cámaras virtuales y un único plano de representación. (Elaboración propia)	30
Figura 3-9 Interfaz dos cámaras y dos planos (Plano izq seleccionado).....	31
Figura 3-10 Esquema 2 (Elaboración propia)	31
Figura 3-11 Ajuste independiente de cada ojo. (Elaboración propia)	32
Figura 3-12 Salida visual de la configuración basada en dos cámaras virtuales y dos planos de representación. (Elaboración propia)	33
Figura 3-13 Cámara estéreo (Elaboración propia)	34
Figura 3-14 Esquema 3 (Elaboración propia)	34
Figura 3-15 Visualización Cámara estéreo (Elaboración propia)	35
Figura 3-16 Primer marcador geoespacial (Elaboración propia)	36
Figura 3-17 Variables configurables del marcador geoespacial en Unity para la asignación manual de datos de posición e identificación. (Elaboración propia)	36
Figura 3-18 Información visual mostrada solo parte izquierda (Elaboración propia)	37
Figura 3-19 Fragmento del script utilizado para activar la brújula e iniciar el servicio de localización del dispositivo. (Elaboración propia)	38
Figura 3-20 Lectura de latitud, longitud, rumbo magnético, rumbo verdadero y precisión de la brújula en el script. (Elaboración propia).....	38
Figura 3-21 Presentación calibración brújula (Elaboración propia).....	39

Figura 3-22 Sintonizador RTL-SDR con chips RTL2832U y R820T2 (Elaboración propia)	40
Figura 3-23 Interfaz principal de AIS Catcher mostrando la representación cartográfica de contactos AIS detectados (izquierda).....	41
Figura 3-24 Vista de mensajes NMEA en AIS Catcher, donde se muestran las tramas AIS recibidas en formato textual estructurado (derecha).	41
Figura 3-25 Recepción en Unity de sentencias AIS reenviadas por UDP y filtrado de mensajes NMEA. (Elaboración propia).....	42
Figura 3-26 Esquema del proceso de decodificación de sentencias AIS (Elaborado con IA)	42
Figura 4-1 Solución mediante proyección sobre pantalla virtual sin ajustar. (Elaboración propia)	46
Figura 4-2 Solución mediante proyección sobre pantalla virtual ajustada. (Elaboración propia)....	47
Figura 4-3 Solución mediante proyección monoscópica directa ajustada (Elaboración propia)	48
Figura 4-4 Lancha de Instrucción Chereguini entrando en puerto. (Elaboración propia).....	50
Figura 4-5 Lancha de Instrucción Godínez entrando en puerto. (Elaboración propia).....	51

ÍNDICE DE TABLAS

Tabla 4-1 Comparación resumida de las soluciones de visualización evaluadas. (Elaboración propia).	49
--	----

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto y motivación

La representación de información geoespacial en tiempo real constituye un aspecto de gran interés en numerosos ámbitos técnicos, especialmente en aquellos en los que el usuario necesita interpretar datos de posición, rumbo o movimiento dentro de un entorno físico real. En el ámbito marítimo, esta necesidad adquiere una relevancia particular, ya que la identificación y seguimiento de embarcaciones mediante sistemas como el AIS (Automatic Identification System) permite disponer de información útil para mejorar la conciencia situacional, apoyar la navegación y facilitar la supervisión del tráfico en zonas costeras o portuarias [1],[2]. Sin embargo, en la práctica, esta información suele presentarse mediante interfaces bidimensionales convencionales, como mapas o pantallas de monitorización, que obligan al usuario a interpretar de forma indirecta la relación entre los datos mostrados y el entorno observado.

A partir de esta situación, resulta de interés explorar nuevas formas de representación que permitan una percepción más intuitiva y espacialmente integrada de la información marítima. En este contexto, la realidad aumentada y la visualización inmersiva ofrecen una vía prometedora, ya que permiten superponer datos digitales sobre la escena real y presentarlos de una forma más cercana a la percepción natural del usuario [3]. No obstante, muchas de las soluciones comerciales actuales se basan en dispositivos específicos de alto coste o en arquitecturas complejas de sensorización y seguimiento, lo que limita su accesibilidad para aplicaciones de prototipado, investigación o validación experimental.

El presente Trabajo de Fin de Grado se plantea como una continuación y evolución técnica en cuanto a representación de información de anteriores trabajos como el realizado por Pablo Díaz García [4], precisamente en ese marco, con el objetivo de estudiar la viabilidad de una solución de bajo coste para la representación inmersiva de información geoespacial asociada a embarcaciones. Para ello, se propone el uso de un visor tipo Cardboard, entendido como un soporte con lentes en el que se inserta un teléfono móvil para generar una visualización estereoscópica de bajo coste, un teléfono móvil y el motor de desarrollo Unity como base de implementación [5],[6], aprovechando tanto la capacidad de visualización estereoscópica del sistema como los sensores integrados del dispositivo, especialmente el GPS y la brújula. Sobre esta base, el trabajo se orienta a integrar información AIS, con el fin de mostrarla al usuario a través de marcadores visuales geoespaciales dentro de una escena inmersiva.

La motivación del trabajo no se limita, por tanto, a trasladar una aplicación previa a un visor de tipo Cardboard, sino a analizar de forma experimental cuál es la configuración de visualización más adecuada para conseguir una experiencia suficientemente inmersiva, comprensible y funcional dentro de las

limitaciones ópticas y geométricas de este tipo de dispositivos. Asimismo, se pretende comprobar hasta qué punto es posible desarrollar una solución operativa sin recurrir a plataformas avanzadas de realidad aumentada como ARCore, apoyándose en una lógica propia de posicionamiento y representación implementada en Unity. De este modo, el TFG responde a una doble necesidad: por un lado, investigar una alternativa accesible para la visualización inmersiva de datos geoespaciales marítimos y, por otro, profundizar en los condicionantes técnicos que influyen en su viabilidad real. Todo ello se enmarca, además, en una línea de trabajo previa centrada en la captura, el procesamiento y la representación de datos AIS en entorno marítimo.

1.2 Objetivos

El objetivo general de este Trabajo de Fin de Grado es desarrollar un prototipo de bajo coste, mediante el motor Unity, capaz de representar de forma inmersiva información geoespacial asociada a embarcaciones, utilizando un visor tipo Cardboard y un teléfono móvil Android, con el fin de estudiar la viabilidad técnica de esta solución y analizar qué configuración de visualización resulta más adecuada para su uso práctico.

Para alcanzar este objetivo general, se plantean los siguientes objetivos específicos:

1. Desarrollar y evaluar distintas estrategias de visualización inmersiva en visores de bajo coste tipo Cardboard, analizando su comportamiento en términos de fusión binocular, nitidez, distorsión, latencia y confort visual para determinar la configuración más adecuada del sistema.
2. Implementar el acceso a los sensores del teléfono móvil, en particular GPS y brújula, para obtener la posición y orientación necesarias para el anclaje geoespacial de los marcadores.
3. Desarrollar una lógica propia de posicionamiento geoespacial, basada en coordenadas GPS, rumbo y cálculos geométricos, sin depender de plataformas externas como ARCore o técnicas de SLAM.
4. Desarrollar una cadena completa de adquisición, procesamiento y representación de datos AIS reales, integrando su recepción mediante AIS Catcher y un receptor RTL-SDR, su tratamiento en formato NMEA y su visualización mediante marcadores geoespaciales informativos dentro de la aplicación.
5. Validar experimentalmente el comportamiento del prototipo, identificando sus limitaciones y valorando sus posibilidades de mejora futura.

1.3 Estructura de la memoria

La presente memoria se organiza en varios capítulos que recogen de forma ordenada el contexto, desarrollo y resultados del trabajo realizado.

En el Capítulo 1 se presenta la introducción general del TFG, contextualizando el problema abordado, justificando la motivación del proyecto y definiendo los objetivos perseguidos.

En el Capítulo 2 se desarrolla el estado del arte, donde se revisan los conceptos y tecnologías que sirven de base al trabajo. En particular, se abordan la realidad aumentada, la visualización geoespacial, los sensores implicados, los visores de tipo Cardboard y el sistema AIS.

En el Capítulo 3 se describe el desarrollo del prototipo. En este capítulo se expone la arquitectura del sistema, el uso de Unity como entorno de desarrollo, las distintas alternativas de visualización inmersiva ensayadas, la creación de marcadores geoespaciales, el acceso a GPS y brújula, y la integración de información AIS mediante un receptor RTL-SDR.

En el Capítulo 4 se presentan los resultados obtenidos y la validación del sistema, analizando el comportamiento de las distintas soluciones implementadas.

Finalmente, en el Capítulo 5 se recogen las conclusiones del trabajo y se proponen posibles líneas futuras de mejora y ampliación del prototipo.

2 ESTADO DEL ARTE

2.1 Realidad Aumentada (AR)

2.1.1 Definición

La realidad aumentada es una tecnología que integra elementos virtuales generados digitalmente en el entorno físico percibido por el usuario, superponiéndolos en tiempo real y manteniendo una correspondencia espacial consistente entre ambos. El usuario continúa observando su entorno físico, una calle, un puerto, un buque o cualquier otro elemento real, pero sobre esa imagen aparecen datos virtuales como textos, iconos, modelos tridimensionales o indicadores que no existen físicamente, aunque parecen formar parte del entorno. Un ejemplo cotidiano sería utilizar el teléfono móvil para enfocar un edificio y que en la pantalla aparezca su nombre o información histórica superpuesta sobre la imagen real. En todos estos casos, el mundo real no desaparece, sino que se complementa con una capa digital que aporta información adicional.

Desde un punto de vista técnico, la realidad aumentada requiere que los elementos virtuales estén correctamente alineados con el entorno físico. No basta con mostrar información en pantalla: el sistema debe ser capaz de calcular la posición y orientación del dispositivo para colocar los objetos digitales en el lugar adecuado y mantenerlos estables cuando el usuario se mueve. Una de las definiciones más aceptadas en la literatura científica es la propuesta por Ronald T. Azuma en su artículo “*A Survey of Augmented Reality*” [7], donde establece que un sistema de AR (Augmented Reality) debe combinar elementos reales y virtuales, ser interactivo en tiempo real y mantener una correcta alineación espacial en tres dimensiones.

Para comprender mejor la AR, resulta útil compararla con la realidad virtual (VR). En la realidad virtual, el usuario deja de ver completamente el mundo real y se introduce en un entorno generado íntegramente por ordenador, donde todos los elementos son digitales. En cambio, en la realidad aumentada el entorno físico sigue siendo visible y los elementos virtuales se integran sobre él. Mientras que la VR sustituye la realidad para crear una experiencia totalmente inmersiva, la AR la enriquece añadiendo información contextual. Esta diferencia conceptual es clave, ya que determina tanto el diseño de los dispositivos como sus aplicaciones prácticas.

Paul Milgram y Fumio Kishino, en su modelo del “continuum de la realidad mixta” [8], sitúan la realidad aumentada en un punto intermedio entre el entorno real puro y la realidad virtual completa, mostrando que ambas tecnologías forman parte de un mismo espectro. En el contexto de este Trabajo de Fin de Grado, la AR no se emplea para crear un entorno artificial, sino para superponer información geoespacial, como los datos AIS de los buques directamente sobre el entorno marítimo observado por el

usuario. De este modo, la información deja de representarse únicamente en mapas bidimensionales y pasa a integrarse espacialmente en el campo visual, facilitando una comprensión más intuitiva y directa.

2.1.2 AR Geoespacial

Este proyecto se centra en la realidad aumentada geoespacial que se basa en la superposición de información digital directamente vinculada a coordenadas geográficas reales, de modo que los elementos virtuales mantienen una correspondencia estable con el mundo físico a gran escala. A diferencia de la realidad aumentada local, en la que los contenidos virtuales se sitúan exclusivamente con respecto a un sistema de referencia local generado a partir del entorno inmediato del usuario, la realidad aumentada geoespacial se basa en un sistema de referencia global vinculado a coordenadas geográficas reales (Latitud y Longitud). En ambos casos pueden emplearse técnicas de localización y mapeo simultáneos (SLAM, Simultaneous Localization and Mapping), que será explicado posteriormente, para estimar la pose del dispositivo.

Sin embargo, mientras que en la AR local el mapa generado mediante SLAM establece el propio marco de referencia, en la AR geoespacial los objetos virtuales se colocan en un sistema de coordenadas global, de modo que su posición corresponde a una ubicación real sobre la Tierra. En este contexto, el seguimiento visual-inercial no define la referencia principal, sino que actúa como mecanismo de apoyo para mejorar la orientación y la estabilidad del contenido, refinando la estimación de la pose y manteniendo la alineación con la referencia absoluta aportada por los sensores geoespaciales. Esta característica resulta especialmente relevante en aplicaciones donde la localización absoluta es fundamental, como la navegación, la monitorización de infraestructuras o la visualización de información distribuida en amplias áreas geográficas.

Al basarse en coordenadas geográficas, el contenido virtual deja de depender exclusivamente del “mapa” local construido a partir de lo que ve la cámara en ese momento, y pasa a quedar vinculado a ubicaciones reales del entorno. Esto permite representar puntos de interés situados fuera del campo de visión o incluso a distancias elevadas, manteniendo la coherencia espacial con su posición en el mundo físico. Además, al utilizar una referencia global, los anclajes pueden mantenerse de forma consistente a lo largo del tiempo y, en determinados casos, ser interpretables por distintos usuarios en la misma zona. Esta capacidad es especialmente valiosa en escenarios exteriores (puertos, costa o espacios abiertos), donde suele haber menos referencias visuales cercanas para construir un marco local estable y donde la información relevante se distribuye sobre áreas extensas. No obstante, la precisión final depende de la calidad de los sensores geoespaciales y de la fusión con el seguimiento visual-inercial, ya que errores en GNSS o en el rumbo pueden traducirse en desplazamientos apreciables del contenido aumentado.

Gracias a esta capacidad, la realidad aumentada geoespacial se ha consolidado como una tecnología adecuada para aplicaciones como la navegación asistida, la visualización de infraestructuras, la monitorización del territorio o la representación de objetos móviles cuya posición varía en el tiempo. En estos casos, la posibilidad de integrar la información directamente en el espacio físico facilita una interpretación más intuitiva de los datos, reduciendo la carga cognitiva del usuario.

Para lograr esta correspondencia entre el mundo real y el virtual, los sistemas de realidad aumentada geoespacial se apoyan en la combinación de distintos sensores integrados en el dispositivo, como sensores inerciales, sensores de orientación, sistemas de posicionamiento global y cámaras. Estos sensores permiten estimar la posición y orientación del usuario y mantener la alineación entre los elementos virtuales y el entorno físico. El funcionamiento y el papel de cada uno de estos sensores se detallan en el apartado 2.1.4.

2.1.3 Hardware

Los cascos de realidad aumentada, o *Head-Mounted Displays* (HMD) (véase la Figura 2-1), constituyen el elemento hardware central en los sistemas de AR avanzados, al permitir la integración directa y continua de información digital en el campo visual del usuario de forma inmersiva y manos libres. Desde un punto de vista técnico, estos dispositivos no actúan únicamente como sistemas de visualización, sino como plataformas integradas que combinan subsistemas ópticos de proyección, sensores de seguimiento de posición y orientación, unidades de procesamiento en tiempo real y mecanismos de interacción. La correcta sincronización entre estos componentes resulta esencial para garantizar una alineación precisa y estable entre el entorno físico y los contenidos virtuales, así como para minimizar la latencia perceptible y preservar la coherencia espacial, aspectos críticos en aplicaciones de realidad aumentada.



Figura 2-1 Visión HMD utilizada en aviación militar [9]

En línea con los HMD avanzados descritos anteriormente, hay dispositivos diseñados específicamente para realidad aumentada que integran varios subsistemas en un único equipo. Existen alternativas mucho más sencillas cuyo propósito es abaratar el acceso a una experiencia “tipo gafas” utilizando componentes de consumo. Dentro de estas soluciones destaca Google Cardboard (véase la Figura 2-1), un visor extremadamente económico que se popularizó por permitir que un teléfono móvil se convierta, de forma muy simple, en un sistema de visualización montado en la cabeza. Cardboard fue presentado por Google en 2014 con una idea clara: reducir al mínimo la barrera de entrada a la realidad virtual móvil y facilitar que cualquier persona, centro educativo o desarrollador pudiera probar contenidos inmersivos sin necesidad de adquirir hardware especializado.

Para ello se propuso un diseño deliberadamente básico (inicialmente de cartón), acompañado de dos lentes y una forma de sujeción que permite colocar el smartphone frente a los ojos. Aunque este tipo de visor fue concebido originalmente para aplicaciones de realidad virtual, su simplicidad también lo ha convertido en una base accesible para experimentar con propuestas inmersivas cercanas a la realidad aumentada cuando se emplea la cámara del teléfono como fuente de imagen. El resultado es una experiencia estereoscópica: la pantalla del móvil se divide en dos mitades, cada ojo observa una imagen ligeramente distinta a través de las lentes y el usuario percibe sensación de profundidad. Esta simplicidad explica su éxito como herramienta de divulgación y prototipado, ya que permitió extender rápidamente el uso de aplicaciones inmersivas y generar un estándar “de facto” para pruebas rápidas con smartphone. Sin embargo, precisamente por ser un visor pasivo, Cardboard no pretende competir con las gafas de AR dedicadas, y su calidad final depende en gran medida del propio teléfono y del modelo de visor utilizado. En la práctica, suele ofrecer menos comodidad que un HMD profesional, presenta ajustes ópticos limitados en muchos modelos (por ejemplo, para adaptarse a distintas distancias interpupilares) y la estabilidad del conjunto varía según la rigidez y el diseño de la carcasa. Aun así, su relevancia en

proyectos como este es notable, ya que proporciona un punto de partida realista para estudiar cómo cambia la experiencia de usuario cuando una aplicación móvil se traslada a un formato “head-mounted”, y sirve como antecedente directo de visores comerciales de bajo coste más robustos (fabricados en plástico y con mejores ajustes) que mantienen el mismo concepto de “smartphone insertado en el visor”.

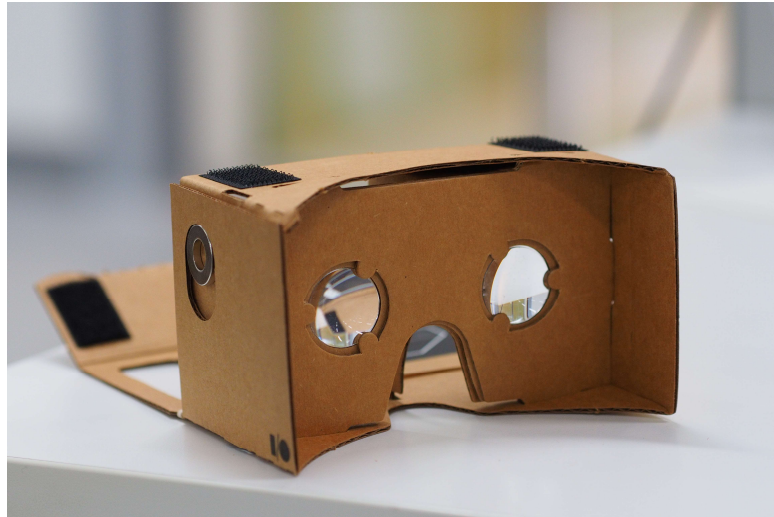


Figura 2-2 Google Cardboard [10]

Siguiendo esta misma filosofía de soluciones accesibles iniciada por Google Cardboard, en este trabajo se utilizará un visor comercial de bajo coste tipo VR BOX (véase la Figura 2-2), perteneciente a la familia de cascos “smartphone-based” que se han popularizado en el mercado por su amplia disponibilidad y facilidad de adquisición. Frente a las Cardboard más básicas, estos modelos suelen ofrecer una carcasa más robusta y un ajuste más cómodo, manteniendo un precio reducido, lo que los convierte en una opción práctica para realizar las pruebas del proyecto sin depender de hardware especializado.



Figura 2-3 VR Box [11]

2.1.4 Sensores

El funcionamiento de cualquier sistema de realidad aumentada depende de su capacidad para estimar de forma continua la posición y orientación del dispositivo en el espacio. Esta estimación, conocida como *pose*, se describe normalmente mediante seis grados de libertad (6DoF): tres correspondientes a traslaciones (movimiento en los ejes X, Y y Z) y tres correspondientes a rotaciones (inclinación, giro y balanceo). La precisión con la que se determine esta pose es fundamental, ya que de ella depende que los elementos virtuales permanezcan correctamente alineados con el entorno real. Para lograrlo, los sistemas de AR combinan distintos sensores y algoritmos de procesamiento que trabajan de manera coordinada.

- La Unidad de Medición Inercial (Inertial Measurement Unit, IMU), es uno de los componentes esenciales en cualquier dispositivo de realidad aumentada, ya que proporciona información continua sobre el movimiento y la orientación del sistema. Generalmente está formada por acelerómetros y giróscopos, y en algunos casos incorpora también un magnetómetro integrado en el mismo módulo físico. El acelerómetro mide aceleraciones lineales en los tres ejes espaciales, incluyendo la componente debida a la gravedad, mientras que el giróscopo mide las velocidades angulares, es decir, la rapidez con la que el dispositivo rota alrededor de cada eje. La principal ventaja de la IMU es su alta frecuencia de actualización, lo que permite obtener estimaciones muy rápidas y suaves del movimiento del dispositivo. Esto resulta especialmente importante en AR, donde incluso pequeños retrasos pueden provocar inestabilidad visual o desalineaciones perceptibles. Sin embargo, la IMU presenta una limitación inherente: al estimar la posición mediante integración temporal de aceleraciones y velocidades angulares, los errores se acumulan progresivamente. Este fenómeno, conocido como *drift*, provoca que la estimación pierda precisión con el tiempo si no se corrige mediante otras fuentes de información externas. Este comportamiento es ampliamente estudiado en el ámbito de la navegación inercial, donde se analizan los efectos de la integración acumulativa de errores y las técnicas de corrección mediante sensores externos [12], [13].
- El SLAM (*Simultaneous Localization and Mapping*), se utiliza para compensar las limitaciones de los sensores inerciales. Su objetivo es doble: por un lado, construir un modelo del entorno físico a partir de la información captada por las cámaras y, por otro, estimar la posición del dispositivo dentro de ese entorno mientras se construye dicho mapa. En los dispositivos actuales, el enfoque más extendido es el SLAM visual-inercial. En este método, las cámaras identifican características visuales estables del entorno, como esquinas, bordes o puntos con alto contraste, y siguen su evolución a lo largo del tiempo. Al comparar la posición relativa de estas características entre distintas imágenes, el sistema puede calcular el movimiento del dispositivo y corregir la deriva acumulada por la IMU. De esta manera, la información visual aporta estabilidad a largo plazo, mientras que la IMU proporciona rapidez y suavidad a corto plazo. Esta combinación constituye la base tecnológica de plataformas como ARCore [14], así como de numerosos visores comerciales de AR y realidad mixta.
- El magnetómetro, presente en algunos dispositivos además de la IMU y de las cámaras utilizadas en SLAM, aporta información complementaria que puede mejorar la estimación de la pose al permitir calcular el rumbo absoluto a partir del campo magnético terrestre. Esta medida proporciona una referencia global de orientación especialmente útil cuando se requiere alinear el sistema con un sistema de coordenadas geográficas, aunque su precisión puede verse afectada por interferencias electromagnéticas en determinados entornos.

- Por otro lado, los receptores GNSS (Global Navigation Satellite System), como GPS o Galileo, proporcionan información de posición absoluta en coordenadas geográficas. A diferencia del SLAM, que opera en un marco de referencia local generado dinámicamente a partir del entorno inmediato, el GNSS permite situar el dispositivo dentro de un sistema de referencia global vinculado a la superficie terrestre. La integración entre GNSS y sensores inerciales constituye una técnica habitual en sistemas de navegación multisensorial, ya que combina la estabilidad global de la posición absoluta con la alta frecuencia de actualización de la IMU [13]. Sin embargo, su precisión puede verse condicionada por la cobertura satelital o la presencia de obstáculos físicos.

2.1.5 Visualización

La visualización en sistemas de realidad aumentada determina la forma en que el usuario percibe el entorno físico y los elementos virtuales superpuestos. Existen principalmente dos enfoques tecnológicos: *optical see-through* y *video see-through*, en el ámbito comercial, esta última arquitectura suele denominarse *passthrough*. Cada uno de estos sistemas presenta características técnicas diferenciadas que afectan a la latencia, la percepción visual, la integración sensorial y la calidad de la experiencia del usuario[7].

En los sistemas *optical see-through*, (véase la Figura 2-5) el usuario observa directamente el mundo real a través de un elemento óptico transparente como lentes o guías de onda sobre el que se proyectan los objetos virtuales, (véase la Figura 2-4). Este enfoque permite mantener la resolución y el rango dinámico naturales del entorno físico, ya que la imagen real no depende de cámaras ni de procesamiento digital previo.



Figura 2-4 Holo Lens 2 [15]

Una ventaja fundamental de esta tecnología es la baja latencia perceptiva, dado que la escena real llega directamente al ojo sin intermediación electrónica. Sin embargo, presenta limitaciones relevantes. Una de las más importantes es la dificultad para representar oclusiones correctas entre objetos virtuales y reales, debido a la imposibilidad física de bloquear la luz procedente del entorno. Asimismo, los elementos virtuales pueden presentar bajo contraste en condiciones de alta iluminación ambiental, reduciendo su visibilidad. Estas limitaciones ópticas y perceptivas han sido ampliamente estudiadas en la literatura especializada [16].

Por otro lado, en los sistemas *video see-through* o *passthrough*, el entorno real se captura mediante cámaras y se muestra en pantallas internas del visor, donde se compone digitalmente con los elementos virtuales (véase la Figura 2-5). La principal implicación de esta arquitectura es que la imagen final

percibida por el usuario queda completamente generada y controlada por el sistema, lo que permite aplicar correcciones geométricas, ajustes dinámicos de brillo y contraste, compensaciones de color y, cuando se dispone de sensores de profundidad, implementar técnicas avanzadas de oclusión coherente entre objetos reales y virtuales, mejorando la consistencia de las experiencias de realidad mixta[17].

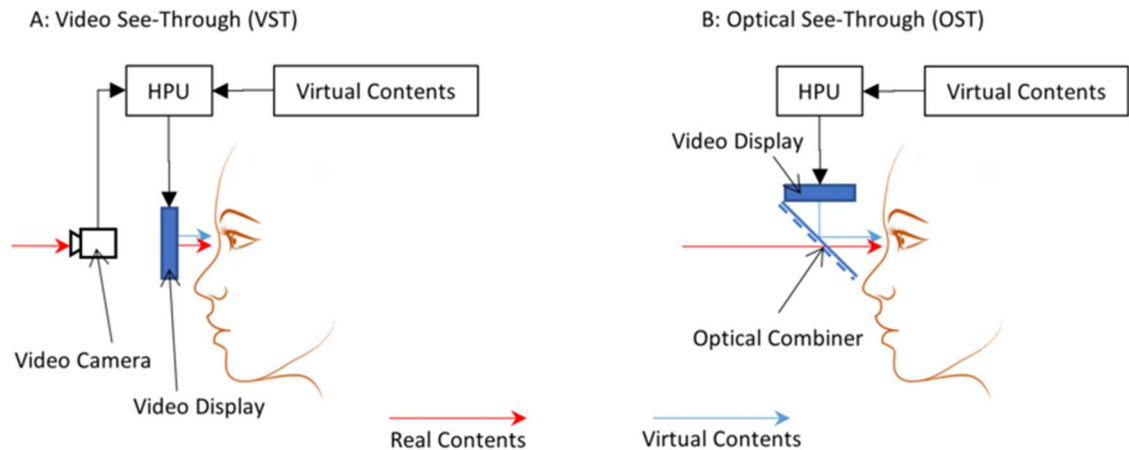


Figura 2-5 Video See-Through vs Optical See-Through [18]

No obstante, este enfoque introduce una latencia inherente asociada a la captura, el procesamiento y el renderizado de la imagen. Si dicha latencia no se mantiene dentro de márgenes reducidos, pueden aparecer desalineaciones perceptibles entre el movimiento real del usuario y la actualización de la escena mostrada, generando inestabilidad visual o incomodidad. Asimismo, la calidad de la experiencia depende de la resolución, el rango dinámico y la calibración de las cámaras, así como de la correcta sincronización con los sensores de movimiento; deficiencias en estos parámetros pueden provocar artefactos como distorsión geométrica, retrasos en movimientos rápidos o discrepancias en la percepción de profundidad, afectando a la coherencia espacial del contenido aumentado [7].

2.1.6 Dispositivos comerciales

El ecosistema XR evoluciona hacia dispositivos cada vez más integrados, con mayor capacidad de percepción del entorno y experiencias más cercanas a la realidad mixta, mientras que el mercado se segmenta con claridad entre soluciones de consumo y plataformas de orientación profesional. En este contexto, los fabricantes están consolidando líneas de producto que combinan avances en visualización y seguimiento con distintos compromisos en coste, ergonomía y nivel de inmersión, marcando la dirección tecnológica predominante en los próximos años.

1. Apple Vision Pro (XR/MR de gama alta)

Apple Vision Pro es un visor XR de realidad mixta basado en passthrough, es decir, el entorno se capta mediante cámaras y se muestra en pantallas internas en lugar de observarse directamente. Incorpora pantallas micro-OLED, múltiples cámaras de seguimiento, cámaras para eye-tracking, sensor LiDAR y unidades inerciales, configurándose como una de las plataformas XR más avanzadas del mercado. Su precio oficial de entrada se sitúa en torno a 3.499 €, lo que lo encuadra en la gama premium. En las especificaciones oficiales no se destaca la integración de GPS, por lo que, aunque ofrece un alto nivel de percepción espacial, no está planteado específicamente como solución autónoma para AR geoespacial basada en posicionamiento global.[19].

2. Meta Quest 3 (VR/MR de consumo, “calidad-precio”)

Meta Quest 3 es un visor orientado principalmente a realidad virtual que añade capacidades de realidad mixta mediante passthrough a color. El fabricante destaca el seguimiento inside-out, junto con cámaras RGB y sensores de profundidad para mejorar la interacción con el entorno. Por precio y disponibilidad se sitúa claramente en el segmento de consumo, con un valor de referencia en Europa cercano a 550 €. Aunque constituye una de las opciones con mejor relación entre prestaciones y coste en el ámbito XR, su enfoque principal continúa siendo la inmersión visual y el seguimiento local del usuario, y no incluye geolocalización mediante GPS [20].

3. Meta Quest Pro (VR/MR “prosumer”)

Meta Quest Pro se sitúa por encima de Quest 3 como visor orientado a un perfil prosumer o profesional, incorporando seguimiento ocular y seguimiento facial, lo que amplía sus posibilidades en interacción avanzada, colaboración y representación de avatares. Su precio de lanzamiento fue notablemente superior al de la gama Quest de consumo, situándose en torno a 1.800 €. Aun así, su planteamiento sigue centrado en la realidad mixta y la interacción avanzada, más que en aplicaciones geoespaciales apoyadas en sensores globales integrados.

4. Microsoft HoloLens 2 (AR óptica, enfoque empresarial)

HoloLens 2 constituye una referencia de AR óptica al emplear un enfoque *optical see-through*, en el que el usuario observa directamente el entorno real a través de lentes holográficas transparentes sobre las que se proyecta el contenido digital. El dispositivo integra cámaras de seguimiento, cámaras infrarrojas para *eye-tracking*, sensor de profundidad ToF (Time of Flight) y una IMU con acelerómetro, giroscopio y magnetómetro, aunque no incorpora GPS. Su orientación ha sido históricamente profesional, con un precio en torno a 3.500 €. Frente a otros visores, destaca precisamente por su mayor integración sensorial, aunque sigue tratándose de una plataforma de coste elevado y de carácter claramente empresarial. Además, se trata de un dispositivo ya discontinuado por Microsoft, si bien mantiene soporte oficial hasta finales de 2027 [21].

5. HTC VIVE XR Elite (XR/MR de gama media-alta)

Dentro del ecosistema HTC, VIVE XR Elite representa una referencia relevante en visores XR compactos. Se trata de un dispositivo todo en uno con seguimiento inside-out, cámaras de seguimiento, cámara RGB en color y sensor de profundidad, además de capacidades de passthrough para experiencias mixtas. En la tienda oficial europea su precio se sitúa alrededor de 569 €, lo que lo posiciona en una gama media-alta, por encima de soluciones de consumo generalistas, pero por debajo de los dispositivos empresariales más avanzados. No obstante, al igual que ocurre con otros visores de este tipo, su diseño prioriza el seguimiento espacial local y la interacción inmersiva, sin destacar integración de GPS en sus especificaciones oficiales [22].

6. XREAL One / gafas AR ligeras

Como ejemplo de una línea distinta dentro del mercado, las gafas XREAL representan una aproximación más ligera y portátil a la visualización aumentada. Modelos recientes como XREAL One se presentan como gafas AR con pantalla integrada y capacidades espaciales propias, con un precio oficial en Europa de alrededor de 499 €. Su propuesta resulta atractiva desde el punto de vista de portabilidad y consumo multimedia, aunque su enfoque difiere del de los HMD más completos, al priorizar formato ligero y visualización personal. En la información oficial consultada se detallan

aspectos como resolución, campo de visión y modos de seguimiento espacial, pero no se destaca la incorporación de GPS como sensor integrado del dispositivo [23].

2.1.7 Teléfono móvil + Visor Tipo Cardboard

A la vista de los dispositivos comerciales descritos, puede observarse que el mercado XR avanza hacia soluciones cada vez más completas en visualización, percepción del entorno e interacción, aunque no siempre orientadas a necesidades de posicionamiento geoespacial. En este sentido, el smartphone presenta una ventaja relevante para el tipo de aplicación abordada en este Trabajo de Fin de Grado, ya que constituye una plataforma integrada de bajo coste que reúne en un único dispositivo capacidades de sensorización, procesamiento y visualización.

Además, la incorporación habitual de sensores como GPS y magnetómetro lo hace especialmente adecuado para la representación de datos geoespaciales, al permitir estimar tanto la posición como la orientación del usuario. Frente a soluciones basadas en hardware dedicado o en infraestructuras externas más complejas, esta aproximación permite aprovechar una tecnología madura, accesible y ampliamente extendida. Por ello, el teléfono móvil actúa en este trabajo como núcleo funcional del sistema, tanto por su capacidad de captura y procesamiento como por su integración física en el visor tipo Cardboard.

2.2 AIS

2.2.1 Contexto

El Sistema de Identificación Automática (Automatic Identification System, AIS) surge en un momento de profunda transformación del transporte marítimo internacional. A finales del siglo XX, el crecimiento sostenido del comercio marítimo provocó un aumento significativo tanto del número de buques como de su tamaño y complejidad operativa, especialmente en rutas principales, estrechos y zonas portuarias. Esta evolución incrementó la congestión del tráfico y redujo los márgenes de seguridad durante las maniobras, haciendo más frecuentes las situaciones de riesgo y los incidentes relacionados con la navegación [24].

En este contexto, los sistemas tradicionales de ayuda a la navegación, como el radar, comenzaron a mostrar limitaciones relevantes. Aunque el radar permite detectar la presencia y distancia de otros buques, no proporciona información directa sobre su identidad, intenciones de navegación ni características básicas (véase en la Figura 2-6), lo que obliga a los operadores a interpretar manualmente la situación a partir de ecos y trayectorias. Estas limitaciones resultan especialmente críticas en

escenarios de baja visibilidad, tráfico intenso o navegación nocturna, donde la carga cognitiva del operador aumenta considerablemente.

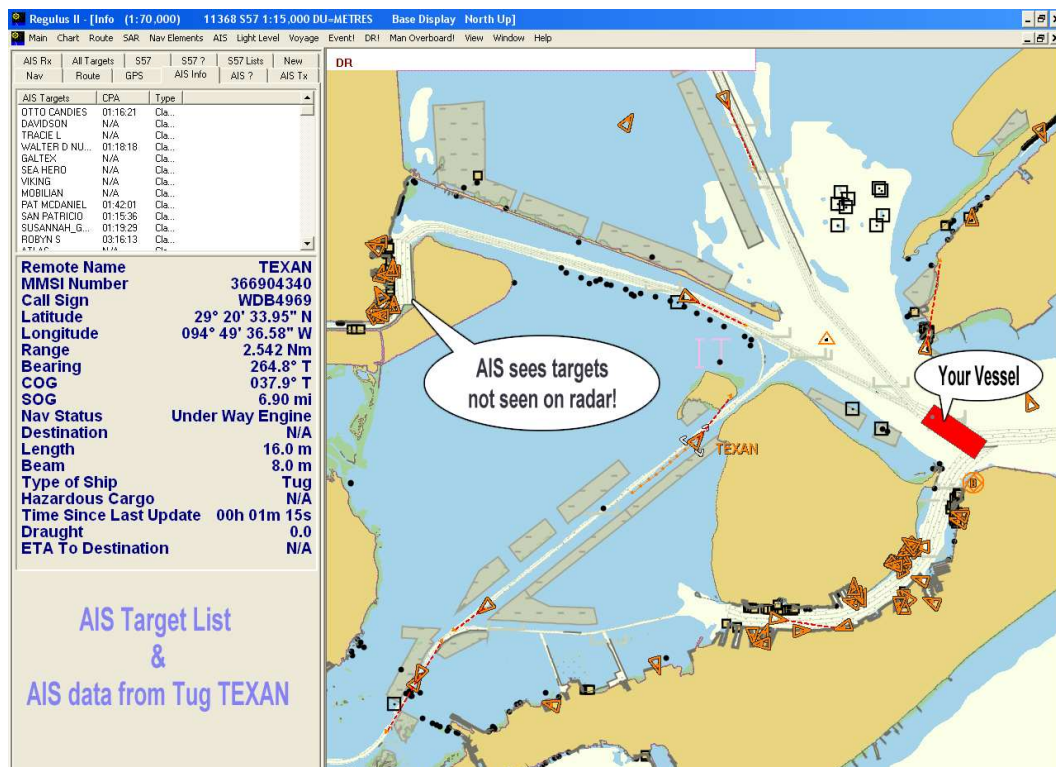


Figura 2-6 Representación AIS [25]

Ante esta necesidad, la Organización Marítima Internacional (IMO) promovió el desarrollo de un sistema cooperativo que complementara a los sensores existentes y mejorara la conciencia situacional tanto a bordo como en los centros de control en tierra. El objetivo del AIS no era sustituir a los sistemas tradicionales, sino aportar una fuente adicional de información estandarizada, capaz de identificar inequívocamente a los buques y describir su comportamiento de navegación de forma automática y compartida [26]. Este planteamiento sentó las bases de un sistema orientado a la prevención de colisiones y a la gestión eficiente del tráfico marítimo.

La incorporación del AIS al marco normativo del Convenio SOLAS (Safety of Life at Sea) supuso un paso decisivo para su adopción global. Al establecer su obligatoriedad para determinadas categorías de buques, la IMO garantizó una implantación amplia y homogénea del sistema, permitiendo la creación de una red cooperativa de intercambio de información a escala internacional [24]. Con el paso del tiempo, el AIS ha evolucionado desde su concepción inicial como herramienta de seguridad hasta convertirse en una infraestructura de información marítima fundamental, utilizada actualmente en la vigilancia costera, la gestión portuaria y el análisis del tráfico marítimo.

2.2.2 Funcionamiento

Desde un punto de vista técnico, el AIS es un sistema de comunicación automática por radio que opera en la banda VHF y permite el intercambio continuo de información entre buques y estaciones costeras. Su funcionamiento se basa en un esquema de acceso múltiple por división temporal o TDMA (*Time Division Multiple Access*), mediante el cual múltiples estaciones comparten el canal de forma organizada, transmitiendo en intervalos temporales asignados para reducir colisiones e interferencias. La OMI describe el AIS como un sistema capaz de proporcionar automáticamente información de

posición, identificación y otros datos relevantes del buque tanto a otras embarcaciones como a autoridades costeras [1].

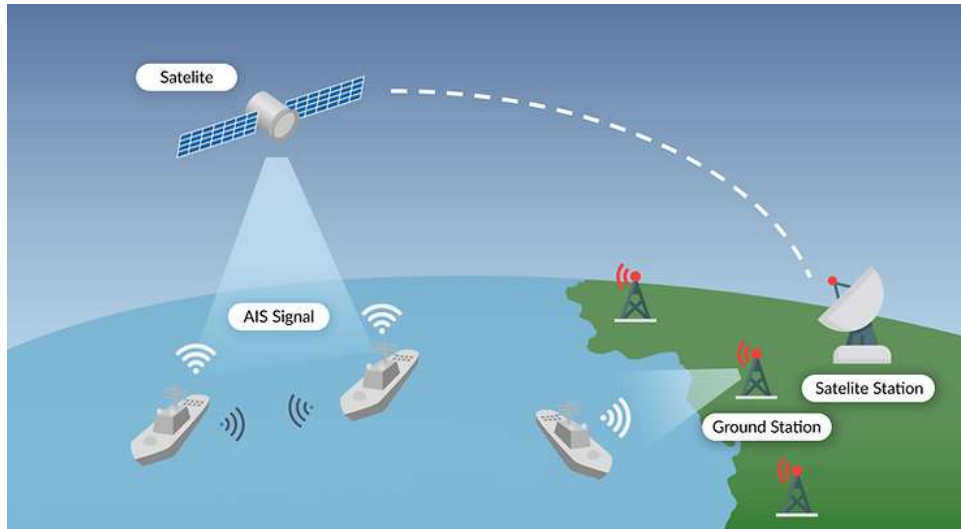


Figura 2-7 AIS signal [27]

Como se muestra en la Figura 2-7, cada buque equipado con un transpondedor AIS emite de forma periódica mensajes estructurados que son recibidos simultáneamente por otros buques cercanos y por estaciones en tierra. Este esquema cooperativo mejora la conciencia situacional en entornos de tráfico intenso, ya que no solo informa de la presencia de otros contactos, sino también de su identidad y de su comportamiento de navegación. Frente al radar, que ofrece principalmente detección física y seguimiento de ecos, el AIS añade una capa de información digital asociada directamente a cada embarcación.

El sistema AIS distingue principalmente entre estaciones de Clase A y Clase B, diseñadas para responder a diferentes requisitos operativos y normativos. De forma general, los equipos de Clase A, obligatorios en determinados buques sujetos al convenio SOLAS, transmiten con mayor potencia y con una gama más amplia de información, mientras que los equipos de Clase B están orientados a embarcaciones con menores exigencias y ofrecen prestaciones más limitadas. Esta diferenciación resulta relevante para interpretar tanto el contenido como la frecuencia de transmisión de algunos mensajes AIS.

Los mensajes AIS contienen varios grupos de campos. Por un lado, los datos dinámicos incluyen, entre otros, la posición geográfica del buque (latitud y longitud), la velocidad sobre el fondo (SOG, *Speed Over Ground*), el rumbo sobre el fondo (COG, *Course Over Ground*), el rumbo de proa o *heading*, la marca temporal y el estado de navegación. Estos datos se actualizan con mayor frecuencia, especialmente cuando la embarcación maniobra o navega a mayor velocidad. Por otro lado, los datos estáticos y relacionados con el viaje incluyen campos como el MMSI (*Maritime Mobile Service Identity*), el nombre del buque, el distintivo de llamada, el tipo de embarcación, sus dimensiones, el destino y la hora estimada de llegada (ETA). En los equipos de Clase A, este tipo de información se transmite de forma periódica mediante mensajes específicos, como el mensaje 5 de AIS, utilizado para datos estáticos y de viaje[28].

Esta estructura de información es especialmente relevante en aplicaciones de supervisión y representación geoespacial, ya que permite no solo situar cada buque en el espacio, sino también

identificarlo y mostrar atributos útiles para interpretar su comportamiento. En otras palabras, el AIS no se limita a indicar “dónde está” una embarcación, sino que también aporta información sobre “quién es” y “cómo se está moviendo”, lo que lo convierte en una fuente especialmente valiosa frente a sistemas puramente sensoriales.

Desde el punto de vista de intercambio de datos, la información AIS suele transportarse y compartirse mediante sentencias en formato NMEA 0183. Este estándar define el protocolo de transmisión serie y la estructura textual de las sentencias utilizadas en electrónica marina. En el caso del AIS, es habitual que los mensajes aparezcan encapsulados en sentencias del tipo ! AIVDM o ! AIVDO, que contienen en formato ASCII la carga útil codificada del mensaje AIS. Esta representación resulta especialmente útil en aplicaciones software, ya que permite recibir, reenviar y decodificar la información mediante cadenas de texto estructuradas, como ocurre en herramientas de captura y procesado tales como AIS Catcher [29].

3 DESARROLLO DEL TFG

3.1 Fases de desarrollo

Este capítulo recoge el desarrollo e implementación del sistema propuesto. En primer lugar, se describe la instalación y configuración de Unity como entorno de trabajo del proyecto. Después, se aborda la integración de los sensores del teléfono móvil y la lógica de posicionamiento empleada para situar la información en el espacio. A continuación, se presenta el proceso de exploración y evaluación de distintos modos de visualización inmersiva, junto con el desarrollo de los marcadores visuales asociados. Finalmente, se explica la incorporación de los datos AIS recibidos mediante AIS Catcher y una antena SDR, así como su procesamiento y representación dentro de la aplicación.

3.2 Entorno de desarrollo: Unity

Unity es un motor de desarrollo en tiempo real ampliamente utilizado para la creación de aplicaciones y experiencias interactivas en 2D y 3D, ofreciendo dentro de un mismo entorno herramientas para la gestión de escenas, objetos, cámaras, interfaces y scripts. La documentación oficial de Unity define su editor como una plataforma orientada al desarrollo de juegos, aplicaciones y experiencias interactivas multiplataforma, lo que justifica su idoneidad para un proyecto como el presente, en el que resulta necesario integrar elementos visuales tridimensionales, lógica de programación y despliegue sobre dispositivo móvil. En este trabajo se ha optado por Unity por su capacidad para centralizar en un único entorno el diseño de la escena, la implementación de scripts en C#, la organización jerárquica de objetos y la preparación de la aplicación para Android, facilitando así un desarrollo progresivo, visual y fácilmente modificable durante las distintas fases del prototipo [30].

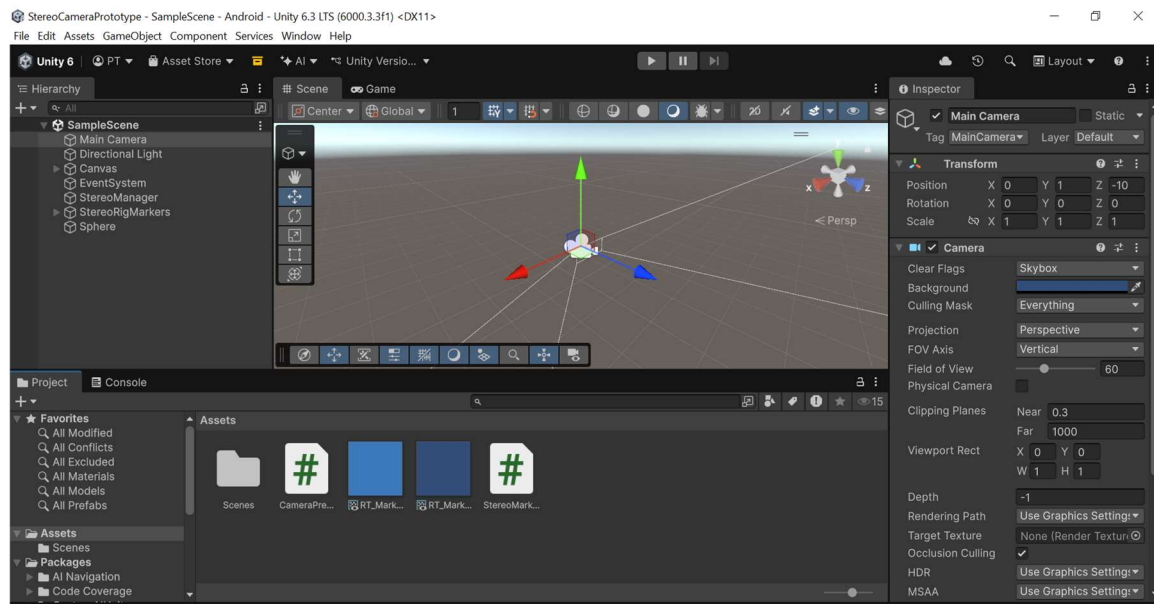


Figura 3-1 Vista general de la interfaz de Unity (Elaboración propia)

La Figura 3-1 muestra una vista general del entorno de trabajo de Unity, utilizado como motor de desarrollo del prototipo. De forma simplificada, en la parte central se encuentra la escena, que constituye el espacio de trabajo principal en el que se colocan, organizan y visualizan los objetos que formarán parte de la aplicación. En la zona izquierda se sitúa la jerarquía, donde aparecen listados todos los elementos presentes en la escena, permitiendo su selección y organización estructurada. A la derecha se localiza el inspector, panel desde el que pueden consultarse y modificarse las propiedades del objeto seleccionado, como su posición, rotación, escala o los distintos componentes asociados. Entre estos componentes pueden encontrarse también scripts, que permiten definir el comportamiento del objeto y modificar, durante la ejecución, las propiedades de otros componentes del propio objeto o de otros elementos de la escena. Finalmente, en la parte inferior se muestra el área de proyecto, en la que se almacenan los recursos utilizados durante el desarrollo, como escenas, scripts, materiales o texturas. En conjunto, estos paneles constituyen los elementos más habituales en el uso básico de Unity y permiten comprender de manera general la dinámica de trabajo seguida durante la implementación del prototipo.

Por otro lado, Unity dispone de distintas versiones del editor (véase la Figura 3-2), organizadas en ramas que pueden diferir en antigüedad, compatibilidad y periodo de soporte. En términos generales, las versiones LTS (*Long-Term Support*) están orientadas a entornos de desarrollo que requieren mayor estabilidad, mantenimiento prolongado y recepción continuada de correcciones, mientras que otras versiones pueden priorizar la incorporación más rápida de novedades o cambios funcionales. En el caso de este trabajo, se ha optado por emplear Unity 6.3 LTS, concretamente la versión 6000.3.3f1, al tratarse de una rama estable y adecuada para un desarrollo progresivo como el del prototipo planteado [6]. Esta elección permite trabajar sobre una base más robusta y menos expuesta a cambios inesperados, algo especialmente conveniente en un proyecto experimental en el que resulta prioritario mantener la compatibilidad del entorno, facilitar las pruebas sucesivas y reducir la probabilidad de incidencias derivadas del propio motor de desarrollo. Unity indica oficialmente que las versiones LTS están pensadas para producciones que buscan estabilidad y soporte de larga duración, y que Unity 6.3 LTS dispone de dos años de soporte, hasta diciembre de 2027.

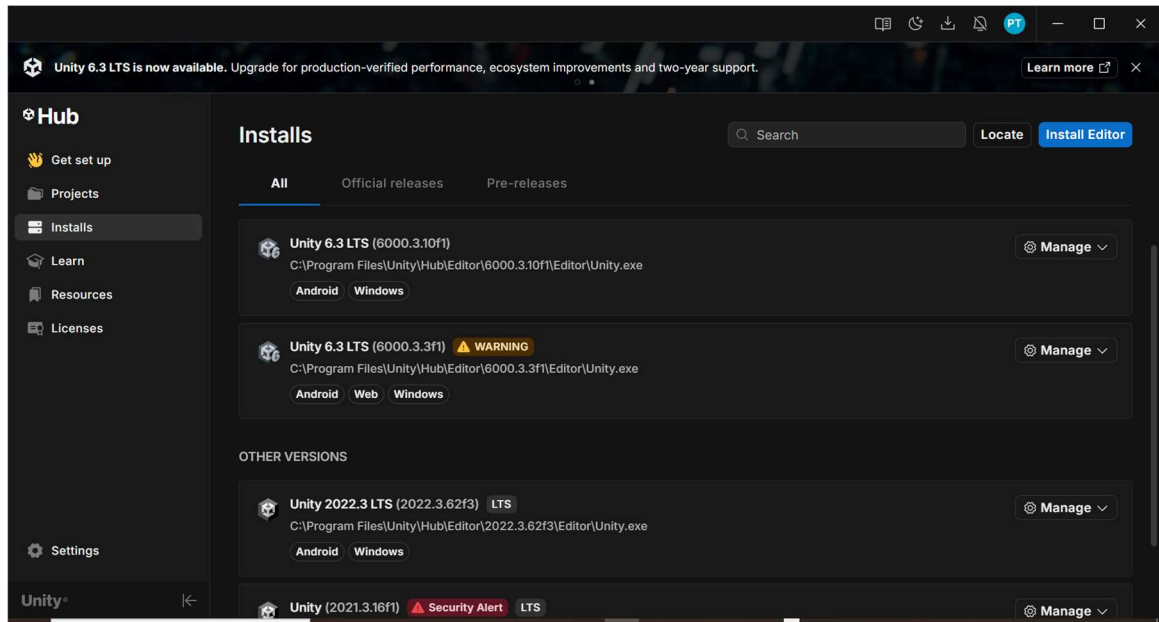


Figura 3-2 Versiones del editor Unity disponibles en Unity Hub (Elaboración propia)

Por otro lado, para poder desplegar el prototipo en un teléfono móvil, fue necesario configurar el proyecto de Unity para la plataforma Android mediante la opción Switch Platform, disponible dentro de los ajustes de compilación. Este cambio adapta el proyecto al entorno de destino y habilita las herramientas específicas necesarias para la generación de aplicaciones Android. Sin embargo, la simple selección de esta plataforma no resulta suficiente por sí sola, ya que Unity requiere la instalación previa de varios módulos y dependencias asociadas al proceso de compilación. Además del editor de Unity, fue preciso instalar los siguientes componentes mencionados a continuación a través de Unity Hub (véase la Figura 3-3), garantizando así la compatibilidad del entorno y la correcta preparación del sistema para generar la aplicación final sobre Android.

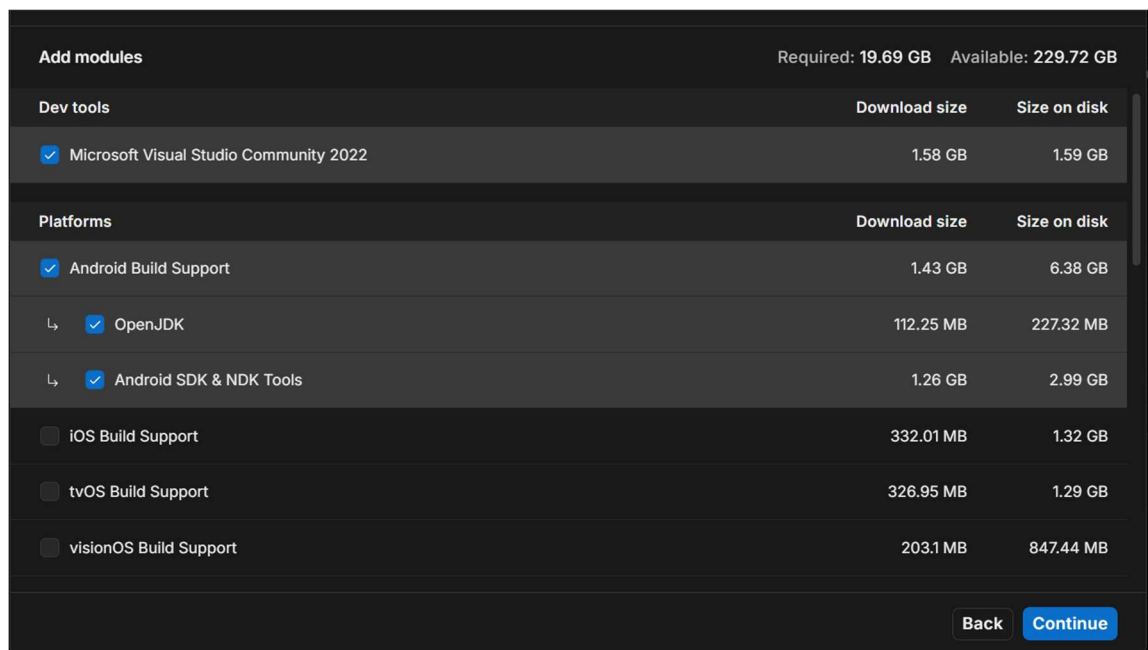


Figura 3-3 Unity installs (Elaboración propia)

- Microsoft Visual Studio Community 2022: este programa se utiliza como entorno de edición y programación del proyecto. Aunque la escena y la mayoría de elementos visuales se gestionan desde Unity, los scripts que controlan el comportamiento de cámaras, marcadores, interfaz y lógica interna se escriben normalmente en C#, y para ello Visual Studio facilita tareas como editar código, detectar errores, autocompletar instrucciones y depurar el funcionamiento de los scripts durante el desarrollo.
- Android Build Support: este módulo es el que permite a Unity preparar el proyecto para la plataforma Android. En la práctica, es el soporte que habilita al editor para cambiar el destino de compilación desde otras plataformas hacia Android mediante Switch Platform, y constituye la base necesaria para que el proyecto pueda terminar convirtiéndose en una aplicación instalable en un teléfono móvil. Unity lo incluye entre las dependencias necesarias para configurar un proyecto con destino Android.
- OpenJDK: este componente proporciona el entorno Java que Unity necesita para ejecutar determinadas fases internas del proceso de compilación. Aunque el desarrollo principal del prototipo se realiza desde Unity y mediante scripts en C#, la generación de una aplicación Android depende también de herramientas basadas en Java, por lo que disponer de OpenJDK es necesario para completar correctamente la construcción del archivo final de la aplicación. Unity lo incluye como uno de los módulos requeridos para el desarrollo Android.
- Android SDK & NDK Tools: este bloque reúne las herramientas específicas que Unity necesita para construir la aplicación Android. Por un lado, el SDK aporta los recursos principales para compilar, empaquetar y preparar la aplicación para su instalación en el dispositivo. Por otro, el NDK añade herramientas de más bajo nivel que algunos procesos internos del motor pueden requerir durante la compilación. En conjunto, estos módulos permiten que Unity genere correctamente la aplicación final y, por ello, forman parte de las dependencias oficiales que deben instalarse desde Unity Hub al preparar el entorno Android.

3.3 Visualización inmersiva

3.3.1 Visualización inmersiva con visor tipo Cardboard

Un visor tipo Cardboard es un soporte con lentes en el que se inserta un teléfono móvil en posición horizontal, de forma que cada ojo observa una mitad distinta de la pantalla. Para conseguir la sensación de inmersión, el dispositivo móvil debe generar una imagen dividida en dos partes (véase la Figura 3-4), una para el ojo izquierdo y otra para el derecho. Las lentes del visor permiten que cada ojo vea su correspondiente imagen a una distancia aparente mayor, facilitando así la percepción estereoscópica. Este planteamiento se utiliza habitualmente en aplicaciones de realidad virtual, donde el contenido mostrado en pantalla ha sido generado completamente por el motor gráfico.

En este trabajo, sin embargo, el objetivo no era representar únicamente un entorno virtual, sino superponer información digital sobre la escena real observada por el usuario. Para adaptar un visor tipo Cardboard a este propósito, fue necesario incorporar a la aplicación una configuración de tipo *passthrough*, en la que la imagen del mundo real es captada por la cámara del teléfono móvil y utilizada como fondo visual de la escena en Unity. Sobre esa imagen se representan después los elementos virtuales, en este caso los marcadores asociados a datos AIS.

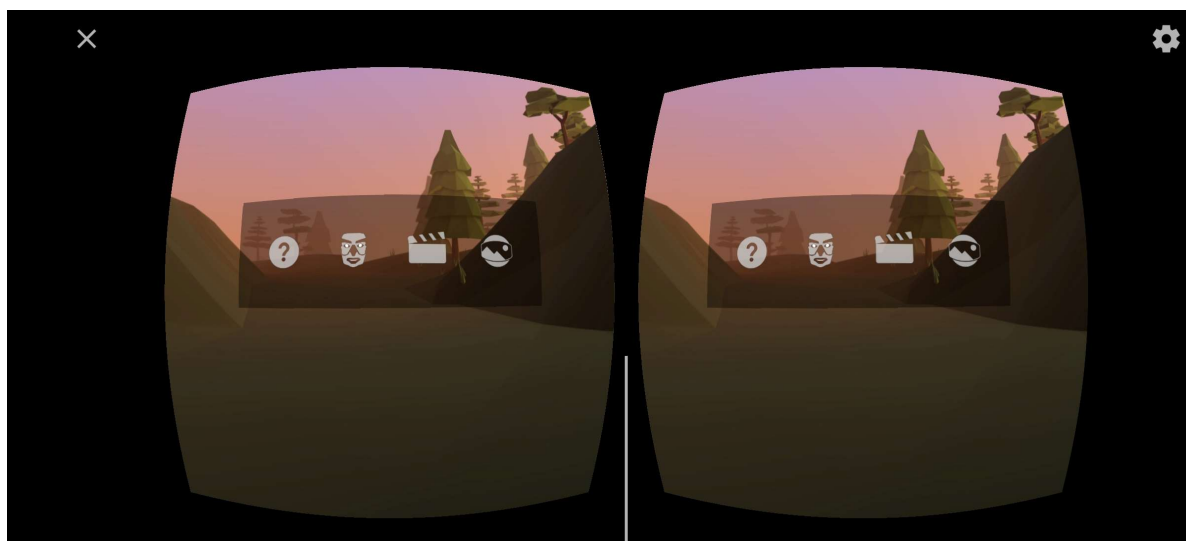


Figura 3-4 Vista estereoscópica mostrada en la pantalla del teléfono móvil mediante la aplicación Cardboard, ilustrando la disposición binocular empleada para su observación con visor (Elaboración propia).

Desde el punto de vista de la implementación, el proceso general seguido en las distintas soluciones evaluadas fue el siguiente: en primer lugar, la cámara del teléfono móvil, o en algunas pruebas una cámara externa, captura la imagen del entorno. Esa imagen se introduce en Unity y se proyecta sobre un elemento de la escena, ya sea un objeto bidimensional o tridimensional, según la solución considerada. A continuación, dos cámaras virtuales, una para cada ojo, observan esa escena y generan las dos imágenes que se muestran en pantalla de forma dividida. Finalmente, al insertar el teléfono móvil en el visor, cada ojo percibe su mitad correspondiente a través de la lente asociada.

La correcta percepción visual depende de que todos estos elementos queden adecuadamente ajustados. Pequeñas variaciones en parámetros como el campo de visión, la distancia entre cámaras, la escala del plano donde se proyecta la imagen o la colocación física del móvil en el visor pueden provocar problemas como visión doble, falta de fusión binocular, distorsión o pérdida de comodidad visual. Por este motivo, fue necesario estudiar distintas configuraciones de visualización inmersiva antes de llegar a una solución funcional.

3.3.2 Visualización mediante proyección sobre pantalla virtual

Como primera aproximación para lograr una visualización inmersiva mediante gafas de tipo Cardboard, se planteó una solución basada en el uso de dos cámaras virtuales en Unity y un único plano central de representación, (véase la Figura 3-5) en la que se puede apreciar las dos cámaras y el plano en color naranja. La idea de esta configuración consistía en utilizar un plano dentro de la escena de Unity que simulase una pantalla plana situada a cierta distancia frente al usuario. Sobre esa superficie se proyectaba la imagen captada por la cámara del teléfono móvil, de manera que actuase como fondo visual de la escena. A su vez, dos cámaras virtuales, una para cada ojo y separadas aproximadamente según la distancia interpupilar habitual, observaban ese plano desde posiciones ligeramente distintas y generaban una imagen con una perspectiva diferente para cada ojo, (véase la Figura 3-6). Con ello se buscaba que, al visualizar el contenido a través del visor, el usuario percibiese una especie de pantalla virtual frontal con la imagen del entorno real situada a una cierta distancia aparente.

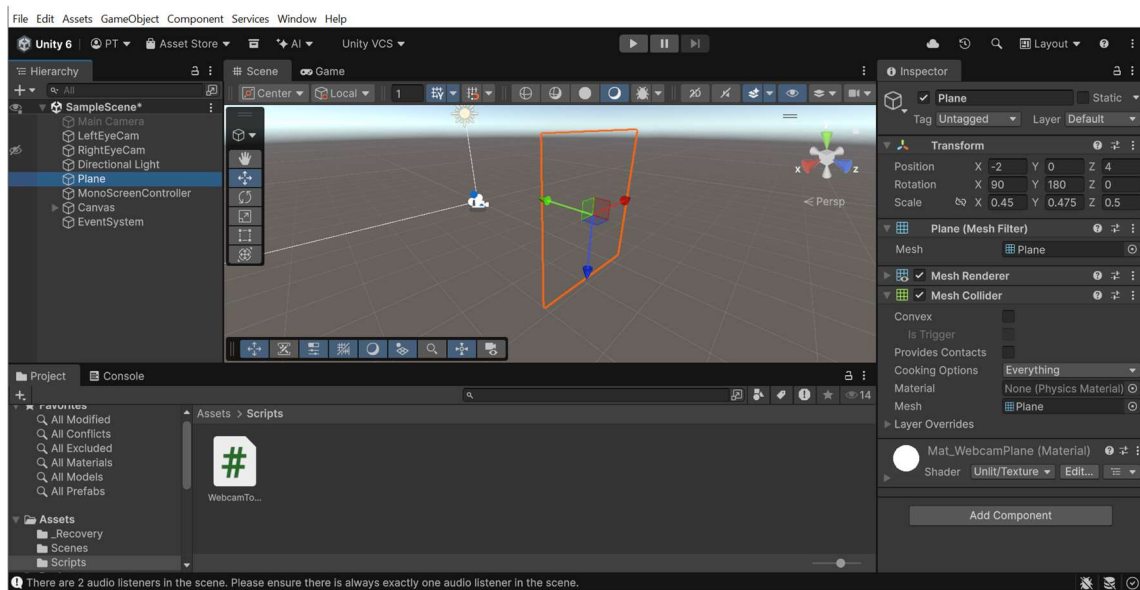


Figura 3-5 Interfaz dos cámaras y un plano (Elaboración propia)

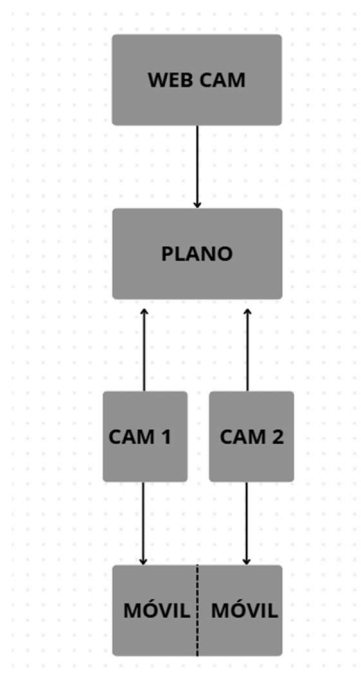


Figura 3-6 Esquema 1 (Elaboración propia)

Desde el punto de vista técnico, la implementación se realizó asignando la WebCamTexture del dispositivo a un material aplicado sobre un plano 3D situado frente a las cámaras virtuales. Posteriormente, cada una de ellas se configuró para mostrar solo su mitad correspondiente mediante el ajuste del Viewport Rect (véase la Figura 3-7) que define qué parte de la pantalla ocupa una cámara al renderizar dentro de Unity, de forma que una renderiza la parte izquierda de la pantalla y la otra la parte derecha, y el FOV ajustando la amplitud angular de la escena que esa cámara es capaz de captar. Esta

disposición permitía construir una visualización binocular sencilla a partir de una única fuente de imagen, manteniendo además una estructura relativamente simple dentro de Unity.

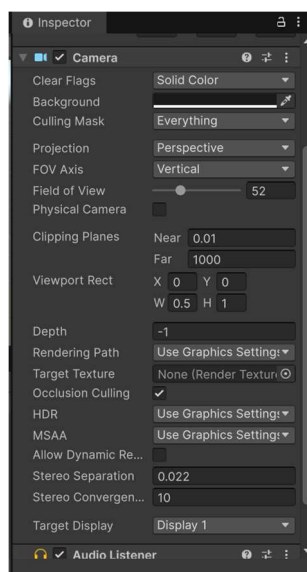


Figura 3-7 Inspector Cámara izq (Elaboración propia)

Esta primera aproximación resultó especialmente útil para identificar varios aspectos clave del sistema, entre ellos la influencia de la separación entre cámaras virtuales, la posición y escala del plano de representación, y la necesidad de adaptar con mayor precisión la imagen a las características ópticas del visor. Su desarrollo tuvo un marcado carácter experimental, basado en la modificación progresiva de parámetros de visión tanto de las cámaras como del propio plano, con el fin de observar cómo afectaban al resultado visual dentro de las gafas. A lo largo de estas pruebas se fueron ajustando de forma sucesiva valores relacionados con la separación entre vistas, el campo de visión, la distancia de observación, el tamaño aparente de la imagen y su grado de solape, buscando mejorar aspectos como la percepción de profundidad, la continuidad entre ambas mitades y el nivel de distorsión necesario para una visualización más natural. Asimismo, este proceso permitió comprobar que la simple división binocular de la imagen no garantiza por sí sola una experiencia inmersiva satisfactoria, ya que intervienen también factores como la alineación de las lentes del visor Cardboard, la distancia interpupilar del usuario y la separación física entre las lentes, especialmente en las VR BOX permiten cierto ajuste. En conjunto, esta fase de pruebas puso de manifiesto que el confort visual y la sensación de inmersión dependen de una calibración fina entre la configuración virtual de Unity y las condiciones físicas reales del sistema de visualización.

Aun así, esta configuración seguía presentando limitaciones asociadas a la dificultad de conseguir una fusión visual completamente cómoda al usar las gafas CardBoard, ya que pequeños desajustes en la separación entre cámaras, en la geometría de representación o en la alineación física del visor podían traducirse en visión doble, pérdida de nitidez o una sensación de inmersión insuficiente.

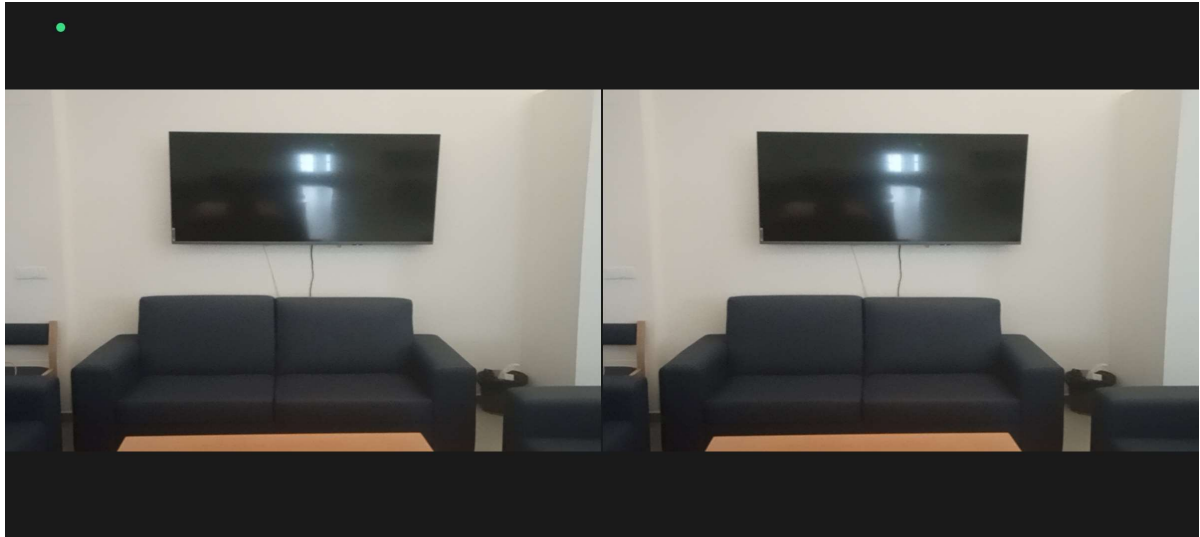


Figura 3-8 Salida visual de la configuración basada en dos cámaras virtuales y un único plano de representación. (Elaboración propia)

La Figura 3-8 ilustra la salida visual correspondiente a la solución implementada con dos cámaras virtuales y un único plano de representación. En ella puede apreciarse que ambas mitades de la pantalla contienen una misma escena, aunque con ligeras diferencias en la posición relativa de los objetos, introducidas por la separación entre las cámaras que simulan la visión de cada ojo. Esta diferencia controlada entre ambas imágenes constituye la base de la percepción estereoscópica. Cuando el dispositivo se coloca en un visor tipo Cardboard, cada ojo observa su mitad de pantalla a través de una lente independiente, de manera que la imagen izquierda queda asociada al ojo izquierdo y la derecha al ojo derecho. La combinación de ambas vistas por parte del sistema visual humano permite reconstruir una única escena con sensación de profundidad.

3.3.3 Visualización mediante proyección monoscópica directa

Como segunda alternativa de visualización, se planteó una configuración orientada a mostrar la misma imagen al ojo izquierdo y al ojo derecho, con el objetivo de simplificar la presentación visual dentro del visor tipo Cardboard y disponer de un mayor margen de ajuste sobre la proyección en pantalla, (véase la Figura 3-10). A diferencia de la solución anterior, que trataba de generar una percepción estereoscópica a partir de un único plano observado desde dos puntos de vista ligeramente distintos, en este caso se buscó que ambas mitades de la pantalla mostrasen esencialmente la misma escena, de forma que la imagen recibida por ambos ojos fuese equivalente.

Para ello, se emplearon dos superficies 3D independientes de tipo *quad* (véase la Figura 3-9), una asociada a cada mitad de la pantalla. La elección de estos elementos permitió trabajar de forma separada con cada imagen mostrada en el dispositivo, facilitando ajustes individuales sobre su tamaño, posición y alineación. Aunque tanto el *plane* como el *quad* pueden utilizarse como soporte para representar texturas en Unity, el *quad* presenta una geometría más simple y ligera, lo que lo hacía especialmente adecuado para actuar como pantalla de imagen en esta fase del desarrollo.

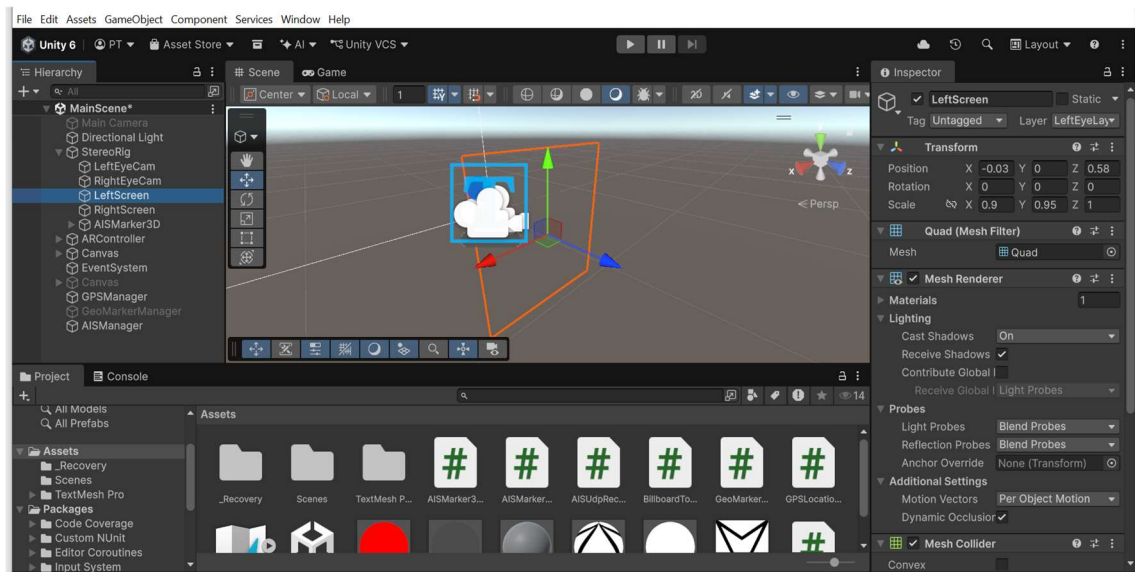


Figura 3-9 Interfaz dos cámaras y dos planos (Plano izq seleccionado)

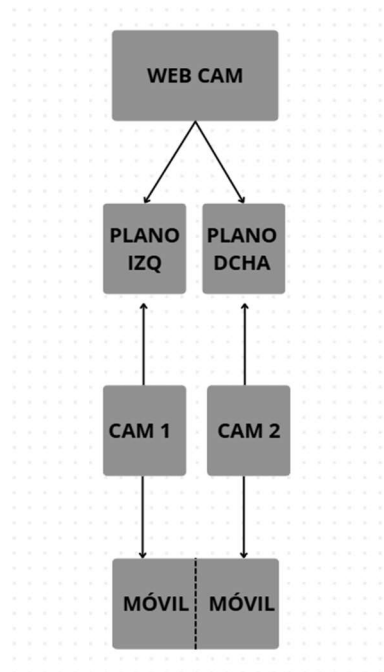


Figura 3-10 Esquema 2 (Elaboración propia)

Desde el punto de vista de implementación, esta propuesta se construyó mediante dos cámaras virtuales, LeftEyeCam y RightEyeCam, (véase la parte de la jerarquía de la Figura 3-9), configuradas para renderizar, respectivamente, la mitad izquierda y la mitad derecha de la pantalla mediante el ajuste del *Viewport Rect*. No obstante, a diferencia de una configuración estereoscópica estricta, ambas cámaras se utilizaron aquí con la misma lógica de proyección, de modo que el objetivo no era generar dos perspectivas diferentes de una misma escena, sino duplicar la imagen de entrada en ambas mitades y disponer al mismo tiempo de un mecanismo flexible para reajustar su presentación visual dentro del visor.

La imagen captada por la cámara del teléfono móvil se proyectó sobre dos objetos 3D de tipo *quad*, denominados LeftScreen y RightScreen, colocados frente a las cámaras virtuales. A cada uno de ellos se le asignó un material basado en la textura procedente de la cámara del dispositivo. De este modo, cada *quad* actuaba como una superficie de visualización independiente, pero mostrando en esencia la misma imagen exterior. Esta organización permitía desplazar o reajustar cada soporte por separado para adaptar mejor el resultado a la geometría del visor y al confort visual del usuario, manteniendo al mismo tiempo una entrada monoscópica común.

La finalidad de esta configuración no era, por tanto, generar relieve binocular real, sino facilitar una visualización más controlada de una misma escena en ambos ojos. Desde el punto de vista perceptivo, la sensación esperable en este tipo de solución es la de observar una imagen proyectada a gran distancia aparente, próxima al infinito visual, más que la de percibir una profundidad estereoscópica marcada en la escena. Precisamente por ello, esta alternativa se planteó como una vía intermedia entre la proyección sobre pantalla virtual y una solución binocular más compleja, buscando mejorar la comodidad de observación y reducir algunos problemas de solape.

Con el fin de facilitar la calibración de esta configuración, se implementó en Unity un sistema de ajustes independientes accesible desde el inspector del script empleado (véase en la Figura 3-11). Este sistema permitía modificar parámetros como el zoom, los desplazamientos horizontales y verticales de cada imagen, así como su posible inversión. Aunque la imagen de partida era la misma para ambos ojos, esta capacidad de ajuste fino resultó especialmente útil para compensar pequeños desajustes del visor, adaptar la posición aparente de cada mitad de la pantalla y mejorar la comodidad visual durante las pruebas.

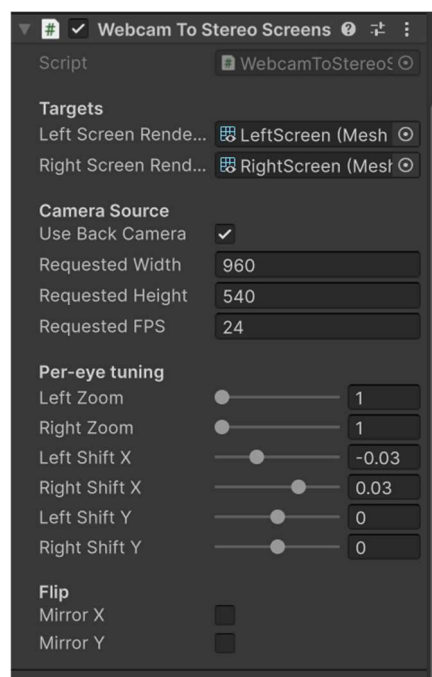


Figura 3-11 Ajuste independiente de cada ojo. (Elaboración propia)



**Figura 3-12 Salida visual de la configuración basada en dos cámaras virtuales y dos planos de representación.
(Elaboración propia)**

La Figura 3-12 muestra la salida visual correspondiente a esta configuración. En ella puede apreciarse que ambas mitades de la pantalla presentan esencialmente la misma escena, sin buscar una diferencia geométrica significativa entre la imagen izquierda y la derecha. Esta característica distingue claramente esta alternativa de la primera solución evaluada, ya que aquí el interés principal no residía en generar profundidad a partir de dos perspectivas diferentes, sino en conseguir una presentación binocular más estable de una imagen común, susceptible de ser reajustada para adaptarse mejor al uso con gafas Cardboard.

3.3.4 Visualización mediante proyección estereoscópica

Como tercera alternativa de visualización, se planteó el uso de una cámara estereoscópica externa, formada por dos cámaras físicas (véase la Figura 3-13), en sustitución de la cámara convencional del teléfono móvil. Esta opción surgió al constatar que, en las configuraciones anteriores, el uso de una única fuente de imagen para ambos ojos impedía, desde el propio planteamiento, la obtención de una percepción estereoscópica real. Al recibir ambos ojos la misma escena o el mismo punto de vista, no podía generarse profundidad binocular, lo que limitaba claramente el nivel de inmersión alcanzable. Por este motivo, se decidió explorar como última opción una cámara estéreo de pequeño tamaño, capaz de proporcionar dos capturas diferenciadas de una misma escena, una para cada ojo, con el objetivo de aproximarse de forma más natural al principio de visión binocular y mejorar la adaptación de la imagen al uso con gafas de tipo Cardboard.

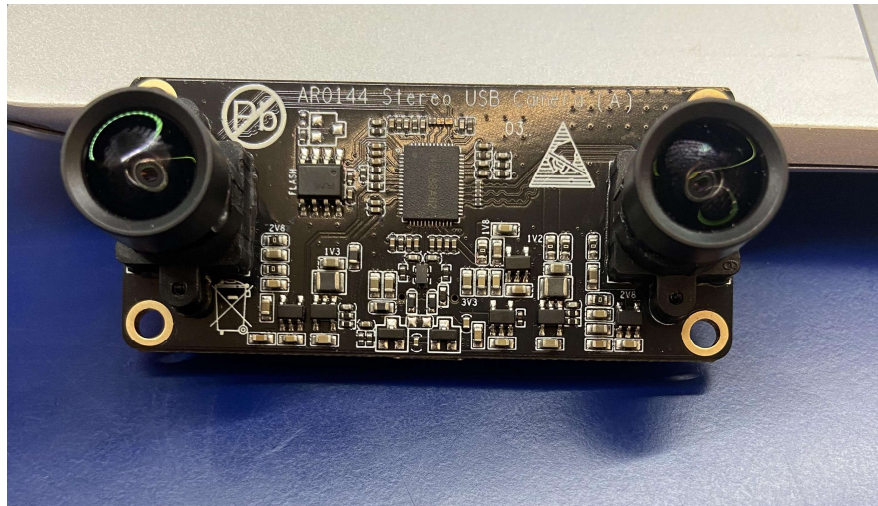


Figura 3-13 Cámara estéreo (Elaboración propia)

Desde el punto de vista de implementación, esta tercera propuesta mantuvo una estructura similar a la empleada en la alternativa anterior dentro de Unity, basada en dos vistas diferenciadas y en su representación independiente dentro de la escena. La diferencia principal residió en que, en este caso, la separación entre imagen izquierda e imagen derecha no se generaba artificialmente mediante scripts a partir de una única cámara del teléfono, sino que venía dada directamente por la propia cámara estéreo, capaz de proporcionar dos capturas simultáneas de una misma escena desde puntos de vista físicamente distintos (véase la Figura 3-14). De este modo, la imagen de entrada llegaba ya dividida en dos vistas, lo que permitía trabajar en Unity con una base visual más próxima al principio real de visión binocular. A partir de ahí, se siguió un procedimiento similar al de las configuraciones anteriores, manteniendo la lógica de representación para cada ojo y realizando en el entorno de Unity los mismos procesos de ajuste, colocación y calibración orientados a adaptar la visualización al uso con visor tipo Cardboard.

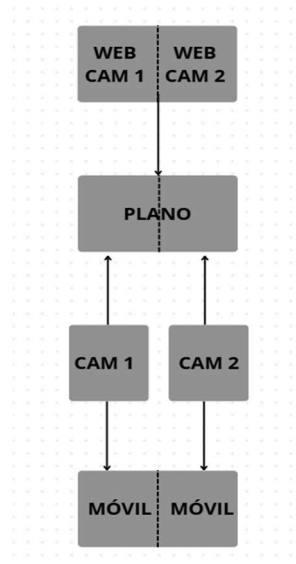


Figura 3-14 Esquema 3 (Elaboración propia)

No obstante, esta alternativa no llegó a integrarse de forma funcional dentro de la aplicación desarrollada en Unity. La principal dificultad radicó en que la cámara estéreo utilizada trabajaba mediante su propia aplicación externa, lo que impedía incorporarla de forma directa al flujo de visualización implementado en el proyecto (véase en la Figura 3-15). Por ello, la evaluación de esta solución no se realizó sobre una salida generada íntegramente desde Unity, sino a partir de una prueba

externa orientada a comprobar, de forma preliminar, si el uso de una cámara con dos puntos de vista reales podía mejorar la sensación de inmersión frente a las configuraciones anteriores.

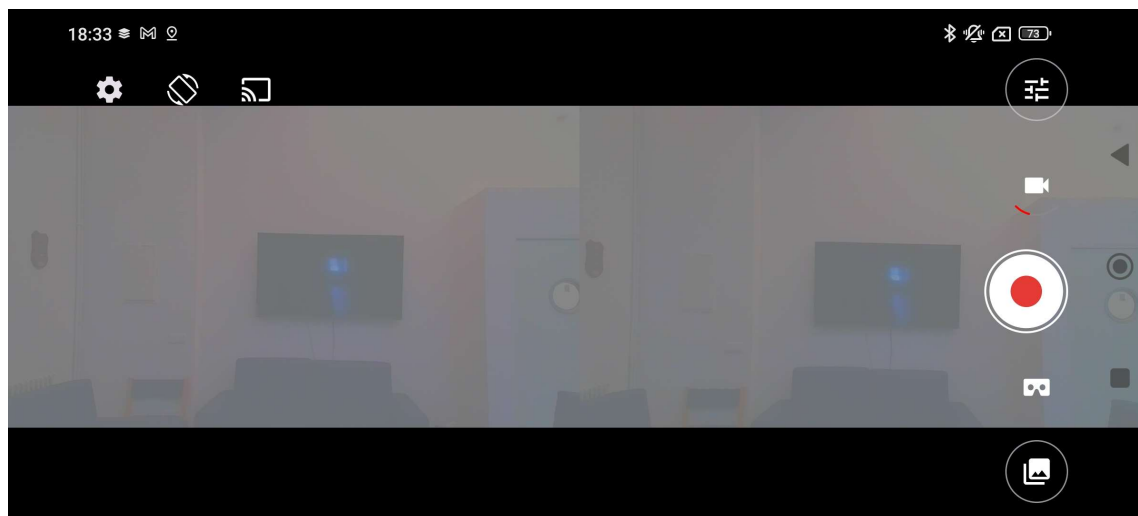


Figura 3-15 Visualización Cámara estéreo (Elaboración propia)

Los resultados de esta prueba pusieron de manifiesto varias limitaciones, tanto del propio dispositivo como de su integración física en el visor. Por un lado, la imagen presentaba problemas apreciables de colorimetría y luminosidad, lo que reducía la calidad visual de la escena. Por otro, la colocación de la cámara en la parte frontal del visor no reproducía con suficiente precisión la geometría de visión del usuario, ya que la posición de sus lentes de captura no coincidía adecuadamente con la separación y orientación esperables para los ojos dentro de las gafas. Como consecuencia, al observar la imagen en el visor tipo Cardboard, no se logró una fusión binocular cómoda ni estable, de modo que la solución no pudo considerarse válida como alternativa práctica dentro del sistema desarrollado.

3.4 Creación de marcadores geoespaciales

3.4.1 Marcador 3D geoespacial

Como primera prueba para el desarrollo de los marcadores geoespaciales, se creó en Unity un marcador 3D sencillo asociado a una posición GPS fija. El objetivo de esta prueba era comprobar, antes de integrar datos AIS reales, si era posible mostrar sobre la escena un marcador con información básica de forma clara y en una posición espacial coherente. Para ello, se diseñó un marcador capaz de presentar en pantalla varios datos AIS definidos manualmente (véase la Figura 3-16) y (Anexo III: Primera Versión Código GeoMarkerText), y se configuraron en el inspector las variables necesarias para controlar su contenido y su comportamiento en la escena (véase la Figura 3-17).

Esta primera implementación permitió validar dos aspectos básicos del sistema. Por un lado, permitió comprobar que la información podía mostrarse de forma legible mediante un marcador 3D dentro de Unity. Por otro lado, permitió verificar que dicho marcador podía asociarse de forma preliminar a una posición fija de referencia, lo que resultaba necesario como paso previo al desarrollo posterior del anclaje geoespacial basado en los sensores del teléfono móvil. Además, desde esta fase inicial se tuvo en cuenta su visualización en gafas tipo Cardboard, de manera que tanto el diseño del marcador como su ubicación se plantearon pensando en su correcta percepción dentro del entorno inmersivo.



Figura 3-16 Primer marcador geoespacial (Elaboración propia)

```

11      [Header("Objetivo GPS")]
12      public double targetLatitude = 42.40783;
13      public double targetLongitude = -8.70757;
14
15      [Header("Datos AIS")]
16      public string vesselName = "SANTA MARIA";
17      public string vesselNationality = "ESP";
18      public string vesselMMSI = "224123456";
19      public float vesselSOG = 12.4f;
20      public float vesselCOG = 245f;
21

```

Figura 3-17 Variables configurables del marcador geoespacial en Unity para la asignación manual de datos de posición e identificación. (Elaboración propia)

3.4.2 Visualización de marcadores

Una vez definido un marcador 3D válido para la escena, se plantearon dos formas iniciales de presentarlo en pantalla con el objetivo de estudiar cuál podía adaptarse mejor a una visualización inmersiva mediante gafas de tipo Cardboard. En una primera prueba (véase la Figura 3-16), el marcador se representó de forma equivalente para ambos ojos, es decir, sin introducir una diferencia geométrica apreciable entre la vista izquierda y la derecha. En términos perceptivos, esta disposición hace que el elemento tienda a visualizarse como proyectado a gran distancia aparente, próxima al infinito, en lugar de situarse a una distancia concreta del usuario mediante disparidad binocular. Con esta configuración se pretendía comprobar si una presentación visual simétrica del marcador podía resultar suficientemente clara y estable dentro del entorno inmersivo.

Como segunda prueba, se ensayó una configuración alternativa en la que la información textual asociada al marcador se mostraba solo en la parte izquierda de la pantalla (véase en la Figura 3-18). En este caso, el objetivo no era tanto obtener una representación binocular completa del texto, sino explorar de forma experimental si una disposición parcial de la información podía resultar más cómoda o más legible dentro del visor. Esta prueba permitió valorar, al menos de forma preliminar, cómo influía la

distribución del texto entre ambas mitades de la imagen y qué planteamiento podía resultar más adecuado de cara a futuros desarrollos del sistema orientados al uso de marcadores informativos en gafas Cardboard.



Figura 3-18 Información visual mostrada solo parte izquierda (Elaboración propia)

3.4.3 Anclaje geoespacial

A diferencia de la prueba anterior, en la que el marcador se vinculaba de forma estática a una posición de referencia, en este apartado se abordó el desarrollo del anclaje geoespacial dinámico del sistema. Para ello, fue necesario acceder desde la aplicación a los sensores de posicionamiento y orientación del teléfono móvil, en concreto al GPS y a la brújula digital integrados en el dispositivo (véase la Figura 3-19). Esta funcionalidad se implementó en Unity mediante el uso de las clases `Input.location` e `Input.compass`, encargadas de activar respectivamente el servicio de localización y la lectura del rumbo del terminal (véase en Anexo V: Código AIS MarkerWorldAnchor).

Desde el punto de vista sensorial, esta información de orientación se relaciona con el funcionamiento del conjunto de sensores internos del dispositivo, especialmente con la IMU (*Inertial Measurement Unit*), ya que, en este caso, la orientación empleada por el sistema se obtiene directamente a partir del magnetómetro del dispositivo, sin aplicar una fusión adicional con otros sensores. En una primera fase se desarrolló el script `GPSCompassDebug` (véase en Anexo IV: Código GPS Compass Debug), cuya función principal fue comprobar el acceso correcto a estos sensores, mostrando en pantalla valores como la latitud, la longitud, el rumbo magnético, el rumbo verdadero y la precisión de la brújula (véase la Figura 3-20). Esta etapa inicial permitió verificar que la aplicación podía obtener de forma continua la información necesaria para relacionar la posición del usuario con la orientación real del dispositivo.

```

11  |         if (!Input.location.isEnabledByUser)
12  |         {
13  |             gpsText.text = "GPS desactivado";
14  |             yield break;
15  |         }
16  |
17  |         Input.compass.enabled = true;
18  |         Input.location.Start();
19  |
20  |         int maxWait = 20;
21  |         while (Input.location.status == LocationServiceStatus.Initializing && maxWait > 0)
22  |         {
23  |             gpsText.text = "Iniciando GPS...";
24  |             yield return new WaitForSeconds(1f);
25  |             maxWait--;
26  |         }
27  |

```

Figura 3-19 Fragmento del script utilizado para activar la brújula e iniciar el servicio de localización del dispositivo. (Elaboración propia)

```

33  |
34  |         while (true)
35  |         {
36  |             LocationInfo data = Input.location.lastData;
37  |
38  |             gpsText.text =
39  |                 "Lat: " + data.latitude.ToString("F6") + "\n" +
40  |                 "Lon: " + data.longitude.ToString("F6") + "\n" +
41  |                 "Compass enabled: " + Input.compass.enabled + "\n" +
42  |                 "Mag: " + Input.compass.magneticHeading.ToString("F1") + "\n" +
43  |                 "True: " + Input.compass.trueHeading.ToString("F1") + "\n" +
44  |                 "Acc: " + Input.compass.headingAccuracy.ToString("F1") + "\n" +
45  |                 "Time: " + Input.compass.timestamp.ToString("F2");
46  |
47  |             yield return new WaitForSeconds(0.2f);
48  |         }
49  |

```

Figura 3-20 Lectura de latitud, longitud, rumbo magnético, rumbo verdadero y precisión de la brújula en el script. (Elaboración propia)

Una vez comprobado el acceso a los sensores, se desarrolló la lógica necesaria para trasladar esta información al posicionamiento visual del marcador. Para ello, a partir de la posición GPS del usuario y de una posición objetivo definida mediante latitud y longitud, el sistema calculaba la distancia al punto de interés y el bearing o rumbo geográfico hacia dicho objetivo. Este valor se comparaba posteriormente con la orientación instantánea del dispositivo, obtenida desde la brújula y corregida mediante los offsets de calibración, permitiendo determinar la posición relativa del marcador dentro del campo visual. A partir de esta diferencia angular, el sistema determinaba la posición horizontal del marcador dentro de la escena, proyectándolo en la dirección correspondiente al rumbo del objetivo respecto a la orientación del usuario. De este modo, la representación en pantalla quedaba vinculada tanto a la información geográfica como al rumbo medido por el dispositivo.

Durante las pruebas realizadas se observó que la lectura de orientación proporcionada por la brújula no coincidía de forma suficientemente precisa con la dirección real de observación dentro del visor, debido a esto, antes del inicio del funcionamiento de la aplicación, la implementación realizada requería una calibración previa de la brújula del dispositivo para mejorar la fiabilidad de la orientación obtenida a partir de los sensores internos del teléfono.

Esta corrección se sumó a otros ajustes de orientación ya presentes en la lógica de visualización, con el objetivo de alinear de forma más adecuada la referencia del norte con la posición esperada de los marcadores dentro de la escena. Este comportamiento puso de manifiesto que la orientación final no depende únicamente de la brújula del teléfono, sino también de factores como la colocación física del móvil en el visor, la orientación de la cámara respecto al eje visual y las pequeñas desviaciones introducidas por el propio hardware (véase la Figura 3-21). De forma indirecta, esto refleja una limitación típica de los sistemas basados en sensores inerciales y magnéticos cuando no se dispone de mecanismos más avanzados de corrección espacial. Esta corrección permitió alinear de forma más adecuada la referencia del norte con la posición esperada de los marcadores dentro de la escena.

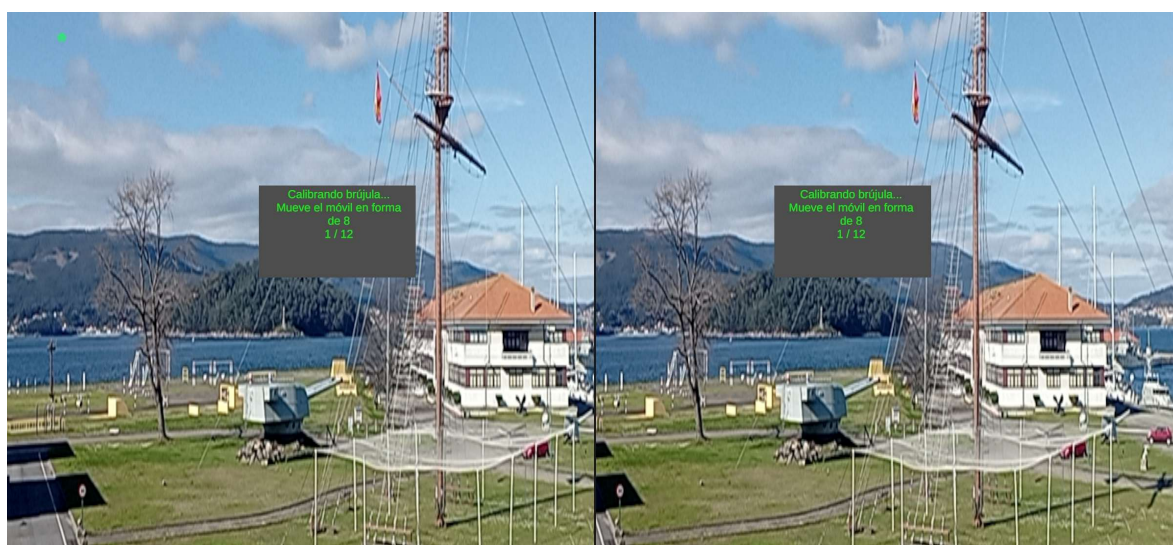


Figura 3-21 Presentación calibración brújula (Elaboración propia).

Todo este proceso de anclaje espacial de los marcadores se realizó sin emplear ARCore ni sistemas equivalentes de realidad aumentada basados en SLAM. Esta decisión fue deliberada, ya que ese tipo de soluciones ya había sido explorado en otros proyectos previos, mientras que en este trabajo se buscó analizar la viabilidad de una implementación más directa, basada únicamente en el uso de GPS, brújula y cálculos geométricos realizados dentro de Unity.

3.5 Captación e integración de información AIS

Una vez resuelta la parte relativa a la visualización y al posicionamiento geoespacial de los marcadores dentro de la escena, el siguiente paso del desarrollo consistió en incorporar una fuente real de datos AIS que permitiese alimentar la aplicación con información procedente de embarcaciones cercanas. Para ello, se recurrió al uso de la aplicación AIS Catcher instalada en el teléfono móvil y a un receptor RTL-SDR conectado mediante la correspondiente antena, configurando así una cadena de adquisición capaz de captar, decodificar y reenviar los mensajes AIS recibidos. Siguiendo una arquitectura similar a la empleada en el trabajo anterior [4], AIS Catcher se utilizó como elemento intermedio entre la recepción radioeléctrica y la aplicación desarrollada en Unity, ya que permite trabajar con receptores RTL-SDR, visualizar los mensajes AIS en formato NMEA y reenviarlos mediante UDP hacia otro proceso en el propio dispositivo. De este modo, la aplicación de visualización pudo utilizar la

información captada por AIS Catcher como entrada para actualizar los marcadores y asociarles los datos AIS correspondientes.

3.5.1 Sintonizador RTL-SDR

El sistema de recepción empleado estuvo formado por una antena externa conectada a un receptor RTL-SDR basado en los chips RTL2832U y R820T2, (véase la Figura 3-22). En este conjunto, la antena se encarga de captar la señal radioeléctrica presente en el entorno, mientras que el dongle actúa como un receptor de radio definida por software (SDR), realizando la sintonización y digitalización de la señal para su posterior tratamiento en el dispositivo móvil. El modelo utilizado presenta un rango de funcionamiento aproximado de 500 kHz a 1,766 GHz y un ancho de banda estable de hasta 2,4 MHz, por lo que resulta adecuado para la recepción de señales AIS, cuyos canales principales se sitúan en 161,975 MHz y 162,025 MHz dentro de la banda marina de VHF.[31],[32].

Dentro de la configuración del receptor, uno de los parámetros más relevantes fue el ajuste de PPM (*parts per million*) en nuestro caso ajustado a 1 PPM, utilizado para corregir pequeñas desviaciones de frecuencia debidas al oscilador interno del dongle. En la práctica, esta corrección permite mejorar la precisión de sintonización y favorecer una decodificación más estable de la señal recibida. De este modo, el conjunto antena–RTL-SDR constituyó la base hardware de captación de información AIS en el sistema, permitiendo que posteriormente la aplicación AIS Catcher procesase y reenviase los mensajes a la aplicación desarrollada en Unity.[32],[33].



Figura 3-22 Sintonizador RTL-SDR con chips RTL2832U y R820T2 (Elaboración propia)

3.5.2 AIS Catcher

La aplicación AIS Catcher se empleó en este trabajo como herramienta intermedia entre la recepción radioeléctrica de las señales AIS, realizada mediante la antena y el receptor RTL-SDR, y su posterior utilización en la aplicación desarrollada en Unity. Su función principal fue verificar la correcta recepción de los mensajes y facilitar su uso posterior en la actualización de los marcadores geoespaciales.

Como puede observarse en la Figura 3-23, la aplicación presenta una interfaz en la que se muestran distintas vistas de trabajo, entre ellas el mapa, el resumen de recepción, la lista de buques detectados y la sección de mensajes NMEA. En la captura correspondiente al mapa se aprecia la representación geográfica de los contactos AIS recibidos en el entorno, así como información básica asociada a uno de ellos, como su identificador y rumbo. Esta vista resultó útil durante las pruebas, ya que permitía verificar

de forma rápida que la aplicación estaba recibiendo señales válidas y situando los objetivos dentro de una referencia cartográfica coherente con la zona de trabajo.

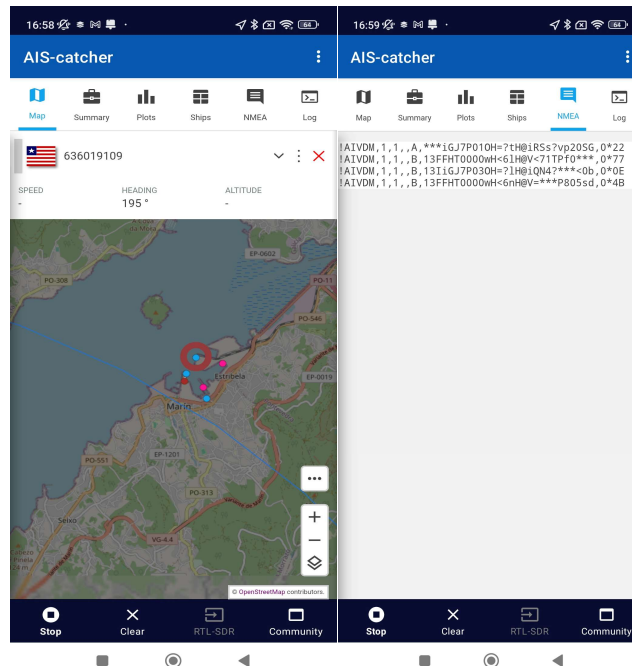


Figura 3-23 Interfaz principal de AIS Catcher mostrando la representación cartográfica de contactos AIS detectados (izquierda).

Figura 3-24 Vista de mensajes NMEA en AIS Catcher, donde se muestran las tramas AIS recibidas en formato textual estructurado (derecha).

Además de la visualización sobre mapa, AIS Catcher permite mostrar los mensajes recibidos en formato NMEA, que fue el formato empleado como base para transferir la información hacia la aplicación desarrollada. Tal como se aprecia en la Figura 3-24, estos mensajes aparecen en forma de cadenas de texto estructuradas, ¡en este caso mediante sentencias del tipo! AIVDM, utilizadas para encapsular información AIS. En ellas se codifican datos relevantes del buque, como su identificación, posición, rumbo o velocidad, que posteriormente pueden ser interpretados por la lógica implementada en Unity.

3.5.3 Procesamiento de la información AIS en Unity

Una vez establecida la recepción de mensajes AIS mediante AIS Catcher, fue necesario desarrollar en Unity una lógica capaz de recibir, interpretar y utilizar esa información para actualizar la representación de las embarcaciones dentro de la escena. Este paso resultaba fundamental, ya que la aplicación no trabajaba directamente con señales de radio, sino con los mensajes ya decodificados y enviados en un formato textual estructurado.

Para ello, la aplicación implementada en Unity se configuró para recibir las sentencias AIS reenviadas por AIS Catcher a través de red local mediante protocolo UDP (véase la Figura 3-25). De este modo, cada vez que AIS Catcher detectaba y procesaba un mensaje válido, este podía ser enviado a la aplicación en forma de sentencia NMEA, permitiendo su lectura desde el entorno de Unity en tiempo casi real.

```

75 |
76 | private void UDPListen()
77 | {
78 |     IPEndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, listenPort);
79 |
80 |     while (isRunning)
81 |     {
82 |         try
83 |         {
84 |             byte[] data = udpClient.Receive(ref remoteEndPoint);
85 |             string msg = Encoding.UTF8.GetString(data).Trim();
86 |
87 |             if (string.IsNullOrEmpty(msg))
88 |                 continue;
89 |
90 |             string[] lines = msg.Split(new[] { "\r\n", "\n", "\r" }, StringSplitOptions.RemoveEmptyEntries);
91 |
92 |             foreach (string line in lines)
93 |             {
94 |                 string cleanLine = line.Trim();
95 |
96 |                 if (cleanLine.StartsWith("!AIVDM") || cleanLine.StartsWith("!AIVDO"))
97 |                 {
98 |                     messageQueue.Enqueue(cleanLine);
99 |                 }
100 |             }

```

Figura 3-25 Recepción en Unity de sentencias AIS reenviadas por UDP y filtrado de mensajes NMEA.
(Elaboración propia)

Una vez recibidas, estas sentencias debían ser decodificadas para extraer los campos de interés necesarios para la representación geoespacial. Entre ellos destacaban principalmente el identificador MMSI de cada embarcación, su latitud, longitud, rumbo, velocidad y, cuando estaba disponible, información adicional como el nombre o el tipo de buque. Esta fase de procesamiento permitió transformar una cadena de texto en datos estructurados utilizables por la lógica interna de la aplicación.

Esquema de decodificación de sentencias AIS en Unity

Basado en la función TryDecodeAisPosition(...) del script AISUdpReceiver.cs

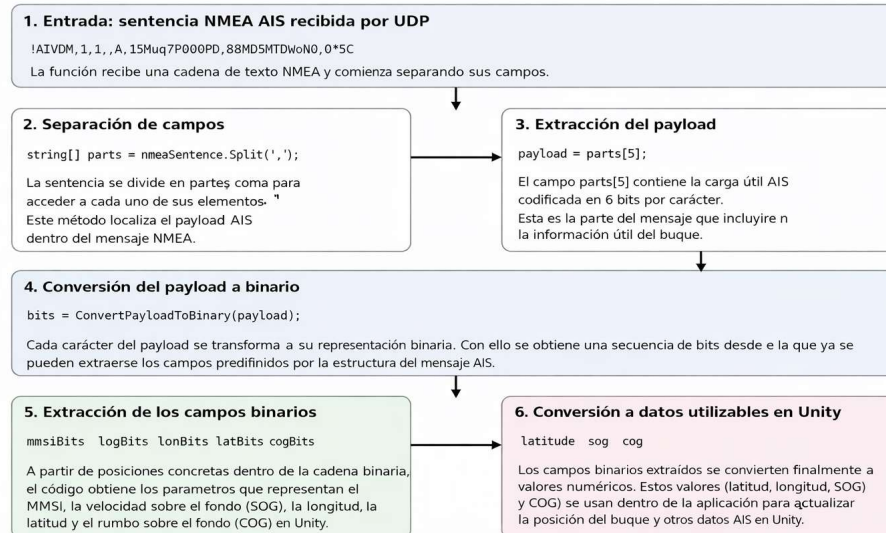


Figura que muestra los métodos descritos en el código que se utiliza para transformar una sentencia !AIVDM en datos AIS utilizables.

Figura 3-26 Esquema del proceso de decodificación de sentencias AIS (Elaborado con IA)

A partir de esa información decodificada (véase la Figura 3-26), se desarrolló un mecanismo de gestión de marcadores dentro de la escena. Cuando se recibía un MMSI que todavía no estaba representado, el sistema instanciaba un nuevo marcador asociado a esa embarcación. Por el contrario, si

el MMSI ya existía en la escena, el marcador correspondiente no se volvía a crear, sino que se actualizaban sus datos y su posición en función de la nueva información recibida. Esta estrategia permitía mantener una correspondencia estable entre cada buque detectado y su representación visual dentro del entorno inmersivo (véase en Anexo VI: Código AIS UDP Receiver).

La actualización de la posición de cada marcador se realizaba a partir de las coordenadas geográficas recibidas en el mensaje AIS y de la posición del usuario obtenida mediante el GPS del teléfono móvil. A partir de ambos valores, la aplicación calculaba la posición relativa del objetivo respecto al usuario y la proyectaba en la escena de Unity según la orientación medida por el dispositivo. De este modo, los marcadores no se mostraban de forma arbitraria, sino vinculados a una dirección coherente con la localización real de la embarcación.

Además de la posición, el sistema actualizaba también la información visual mostrada por cada marcador. Así, cuando el mensaje recibido contenía nuevos valores de rumbo, velocidad, identificación u otros campos relevantes, estos se reflejaban en el contenido asociado al marcador correspondiente. De esta forma, la representación no solo indicaba dónde se encontraba cada embarcación, sino también información útil para su identificación e interpretación dentro del entorno marítimo.

En conjunto, este procesamiento en Unity permitió cerrar el flujo completo de funcionamiento del sistema: desde la captura de la señal AIS y su decodificación inicial en AIS Catcher hasta su conversión en marcadores geospaciales dinámicos dentro de la escena inmersiva. Gracias a ello, fue posible integrar información marítima real en tiempo casi real dentro de la aplicación desarrollada.

4 RESULTADOS Y VALIDACIÓN

En este capítulo se presentan y analizan los resultados obtenidos durante el desarrollo del prototipo, centrándose especialmente en la validación de las distintas soluciones de visualización inmersiva ensayadas. Para ello, se comparan las tres configuraciones implementadas en función de criterios como la fusión binocular, la presencia de visión doble, la nitidez de la imagen, la latencia y la comodidad de uso en visores de tipo Cardboard.

A partir de esta comparación, se valora qué alternativa ofreció el comportamiento más adecuado para el objetivo del trabajo. Asimismo, se identifican las principales limitaciones observadas y su influencia en la viabilidad del sistema desarrollado. Conviene señalar, no obstante, que la percepción final experimentada por el usuario al utilizar las gafas Cardboard no puede recogerse de forma completamente fiel mediante capturas o grabaciones convencionales, ya que la visualización binocular depende de la observación directa a través del visor y de la fusión realizada por el propio sistema visual humano.

4.1 Resultados de visualización

Con el fin de comparar de forma ordenada las distintas soluciones de visualización desarrolladas, se definieron varios criterios de evaluación directamente vinculados al objetivo principal del trabajo: lograr una representación inmersiva funcional y visualmente cómoda en un visor de tipo Cardboard. Dado que el prototipo combina software, sensores del dispositivo móvil y limitaciones físicas propias del visor, la evaluación no se centró únicamente en la calidad gráfica, sino también en la percepción final del usuario y en la viabilidad práctica de cada configuración. Asimismo, aunque el núcleo del TFG se centra en el estudio de la visualización inmersiva, también se valoró de forma complementaria el comportamiento de los marcadores desarrollados, comprobando si estos se percibían de manera coherente con la referencia geoespacial buscada y si permitían mostrar correctamente la información AIS asociada.

4.1.1 Criterios de evaluación

Uno de los criterios principales fue la capacidad de fusión binocular, entendida como la correcta integración de las imágenes presentadas a cada ojo en una única percepción visual. Asociado a ello, se consideró también la presencia de visión doble, especialmente en la zona central de la imagen, por su influencia directa sobre la inmersión y el confort visual.

También se evaluaron la nitidez de la imagen exterior y el grado de distorsión visual, ya que ambos aspectos condicionan la claridad con la que el usuario percibe el entorno y la información geoespacial representada. Del mismo modo, se tuvo en cuenta la latencia del sistema, por su efecto sobre la

naturalidad de la experiencia, así como la ergonomía e integración física de cada solución en el visor, considerando su facilidad de montaje, estabilidad y adecuación al uso real.

A partir de estos criterios, fue posible realizar una comparación cualitativa entre las alternativas ensayadas y determinar cuál de ellas ofrecía el comportamiento más adecuado para los fines del proyecto.

4.1.2 Resultados visualización mediante proyección sobre pantalla virtual

La primera configuración evaluada correspondió a la solución basada en dos cámaras virtuales y un único plano central de representación. Desde el punto de vista conceptual, esta alternativa permitía construir una visualización binocular sencilla a partir de una única fuente de imagen, manteniendo una estructura relativamente simple dentro de Unity. Sin embargo, los resultados obtenidos durante las pruebas mostraron que esta configuración no ofrecía un comportamiento adecuado para el objetivo de inmersión planteado en el trabajo.

Los primeros resultados percibidos con esta configuración pusieron de manifiesto que pequeñas variaciones en los parámetros de visualización influían de forma muy notable en el resultado final. Las siguientes dos imágenes permiten apreciar este comportamiento comparando dos capturas de la misma solución basada en dos cámaras virtuales y un único plano central. En la Figura 4-1 se muestra una imagen sin ajustar, mientras que en la Figura 4-2 se presenta una versión en la que se modificaron algunos parámetros explicados previamente en el capítulo de desarrollo, entre ellos el FOV (pasando de 60° a 46° respectivamente), la distancia entre el plano y las cámaras virtuales (a una distancia de aproximadamente 2 unidades de Unity, equivalentes en este contexto a unos 2 m) y la resolución de la imagen (se elevó la resolución de la imagen hasta 1920 × 1080 píxeles). Como resultado, en la imagen ajustada la escena adquiere una presencia visual algo más adecuada, con un tamaño aparente más favorable y una percepción ligeramente mejor de algunos elementos del entorno.

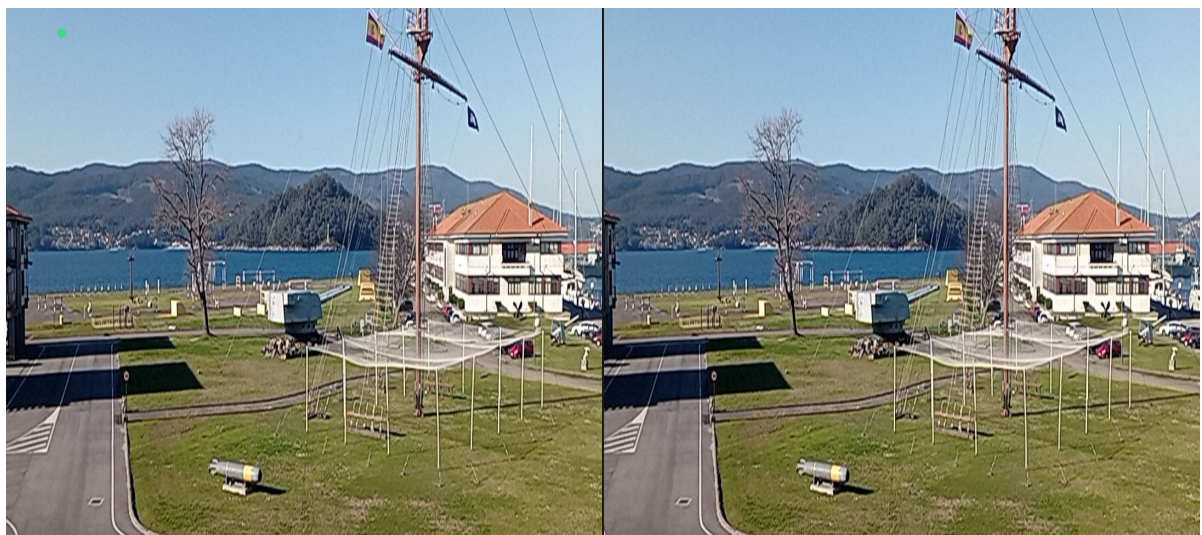


Figura 4-1 Solución mediante proyección sobre pantalla virtual sin ajustar. (Elaboración propia)

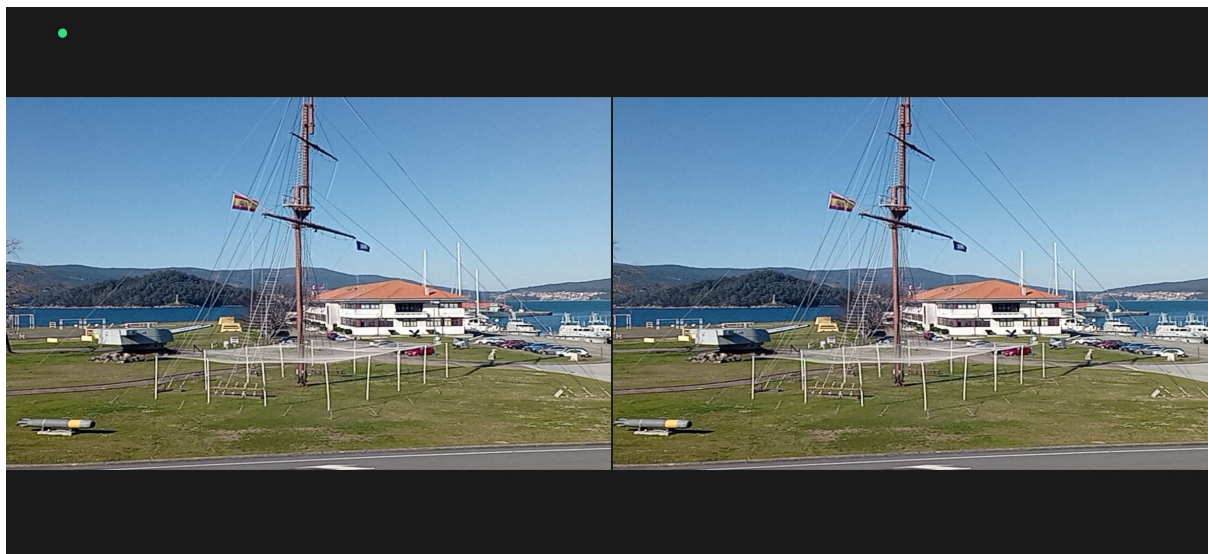


Figura 4-2 Solución mediante proyección sobre pantalla virtual ajustada. (Elaboración propia)

Esta comparación permitió comprobar que el comportamiento del sistema era muy sensible a cambios incluso pequeños en la calibración, lo que confirma la importancia del ajuste fino en una visualización inmersiva de este tipo. Sin embargo, aunque estas modificaciones mejoraban parcialmente el aspecto general de la imagen, la solución seguía sin resultar válida para su uso en gafas de tipo Cardboard. En la práctica, no se conseguía una fusión binocular estable, manteniéndose una percepción de visión doble que obligaba al usuario a forzar excesivamente la vista para intentar unir ambas imágenes.

Desde un punto de vista técnico, esto puede explicarse porque las dos vistas seguían generándose a partir de una única imagen proyectada sobre un solo plano, de modo que la escena completa quedaba reducida a una sola profundidad geométrica. Por ello, aunque el ajuste de parámetros modificaba la escala y la proyección de la imagen, no resolvía la limitación estructural de esta arquitectura, que no proporcionaba una correspondencia binocular real entre objetos situados a distintas distancias de la escena

4.1.3 Resultados visualización mediante proyección monoscópica directa

La segunda configuración evaluada, basada en dos cámaras virtuales y dos superficies independientes de tipo quad, fue la que ofreció los resultados más favorables dentro del conjunto de alternativas ensayadas. A diferencia de la solución anterior, en este caso la correspondencia entre la imagen izquierda y la derecha resultó considerablemente más adecuada para su observación en gafas de tipo Cardboard. Como consecuencia, se consiguió una fusión binocular más estable, reduciéndose de

forma notable la percepción de visión doble en la zona central y mejorando la integración general de la escena.



Figura 4-3 Solución mediante proyección monoscópica directa ajustada (Elaboración propia)

Aunque esta configuración fue la que ofreció los mejores resultados en términos de visualización, no puede afirmarse que también fuese la más favorable desde el punto de vista de la resolución y la calidad de imagen (véase la Figura 4-3). En este sentido, se alcanzó como límite práctico la resolución máxima que podía implementarse de forma estable en el dispositivo, coincidiendo además con la especificada en la ficha técnica para la grabación de vídeo, 1920×1080 a 30 fps [34].

Sin embargo, aun trabajando con este valor máximo, la imagen presentaba una mayor distorsión y una menor calidad aparente que en otras soluciones evaluadas. Parte de esta pérdida de calidad se debió también a la disposición de los quads, que tuvieron que situarse a una distancia relativamente cercana para favorecer la fusión binocular, mejorar la comodidad visual y aumentar la sensación de inmersión en el visor. Esta proximidad hacía que la imagen se percibiese más cercana al observador, lo que acentuaba la distorsión y reducía la nitidez, dando lugar a una escena más borrosa.

Asimismo, se comprobó que cualquier intento de aumentar la resolución por encima de dicho umbral provocaba un incremento notable de la latencia, lo que afectaba directamente al funcionamiento en tiempo real de la aplicación. Esta limitación repercutía especialmente en la representación de los marcadores, que en determinadas ocasiones quedaban bloqueados o no se actualizaban con la fluidez necesaria. Por tanto, aunque esta alternativa resultó ser la más adecuada para favorecer la fusión visual y mejorar la comodidad de uso en Cardboard, también puso de manifiesto la necesidad de sacrificar parcialmente la calidad de imagen para mantener una experiencia estable, así como de encontrar un equilibrio entre resolución, rendimiento e inmersión. No obstante, una vez fijada una configuración de funcionamiento estable, pudo comprobarse que tanto el retardo como la resolución resultaban suficientes para el propósito del prototipo. En estas condiciones, la principal limitación de la visualización no se asoció tanto a la latencia o a la falta de resolución, sino sobre todo a la distorsión geométrica de la imagen y a las dificultades de calibración entre la cámara, el visor y la configuración binocular.

4.1.4 Resultados visualización mediante proyección estereoscópica

La solución basada en una cámara estéreo externa parecía, en principio, una alternativa adecuada para mejorar la inmersión, ya que permitía obtener dos imágenes reales de una misma escena, una para cada ojo. Sin embargo, en las pruebas realizadas no se consiguió una visualización cómoda ni una fusión binocular estable. Aunque la cámara proporcionaba dos vistas diferenciadas, la sensación de profundidad seguía siendo limitada y la experiencia no resultaba plenamente inmersiva.

Una de las principales razones fue que la integración entre la cámara, el visor y el smartphone no era lo suficientemente precisa. La colocación física de la cámara en las gafas condicionaba mucho el resultado final, y pequeños desajustes dificultaban la correcta unión de ambas imágenes. A ello se sumaban las limitaciones propias del dispositivo utilizado, tanto por su calidad de imagen como por la necesidad de emplear una aplicación externa para poder hacerlo funcionar, lo que reducía las posibilidades de ajuste y hacía menos práctico el conjunto.

Por tanto, esta alternativa no llegó a ofrecer un resultado satisfactorio en el prototipo desarrollado. No obstante, puede considerarse una línea prometedora de mejora futura, ya que una mejor integración física de la cámara en el visor y el uso de un dispositivo stereo de mayor calidad, capaz de funcionar de forma más directa y con mayores opciones de configuración, podrían convertir esta solución en una opción más válida para lograr una experiencia inmersiva.

Finalmente, en la Tabla 4-1 se presenta un resumen comparativo de las tres soluciones evaluadas, con el fin de sintetizar de forma clara los principales resultados obtenidos.

Solución	Fusión Binocular	Calidad visual	Rendimiento	Valoración final
Proyección sobre pantalla virtual	Baja; persistía visión doble y era necesario forzar la vista	Mejoraba tras el ajuste de parámetros, pero seguía siendo insuficiente	Aceptable	No resultó válida para el uso con visor Cardboard
Proyección monoscópica directa	La más estable de las tres soluciones	Aceptable, aunque con mayor distorsión y menor nitidez aparente	Correcto hasta 1920×1080 ; resoluciones superiores aumentaban la latencia	Fue la solución que ofreció los mejores resultados globales
Proyección estereoscópica	Insuficiente; no se logró una fusión cómoda ni estable	Limitada por problemas de colorimetría, luminosidad e integración física	Menos práctico por la necesidad de aplicación externa y menor margen de ajuste	No ofreció un resultado satisfactorio en el prototipo, aunque se considera una posible línea futura

Tabla 4-1 Comparación resumida de las soluciones de visualización evaluadas. (Elaboración propia).

4.2 Resultados geospaciales

Los resultados obtenidos en la parte geoespacial pueden considerarse satisfactorios. La aplicación fue capaz de recibir datos AIS reales captadas por el receptor a través de AIS Catcher y representarlos mediante marcadores sobre la escena observada con el móvil y las gafas Cardboard. En general, los marcadores se situaban cerca de los buques reales, lo que permitía relacionar de forma clara la información mostrada con la embarcación observada.

La colocación de los marcadores no siempre fue exacta desde el primer momento (véase en la Figura 4-4). Al inicio de la aplicación se apreciaban ciertos desajustes, probablemente debidos a la calibración de la brújula, a la estabilización del GPS y a pequeñas variaciones en la orientación del teléfono móvil. Sin embargo, tras unos minutos de uso, el sistema tendía a estabilizarse y los marcadores se colocaban de forma más próxima a los barcos, con la estrategia de posicionamiento desarrollada, los marcadores se representaban de forma aproximada sobre la zona superior de la embarcación, ofreciendo una sensación espacial más convincente.

En cuanto a la información representada, los resultados también fueron positivos. La aplicación consiguió mostrar correctamente el MMSI del buque, así como datos como la velocidad sobre el fondo, el rumbo y la distancia aproximada. Esto demuestra que la cadena completa de recepción y procesamiento AIS funcionó de forma correcta. No obstante, no fue posible mostrar el nombre real de la embarcación, ya que esa información pertenece a mensajes AIS estáticos cuya decodificación no llegó a integrarse completamente en la versión final. Por este motivo, el sistema utiliza el MMSI como identificador principal del barco.

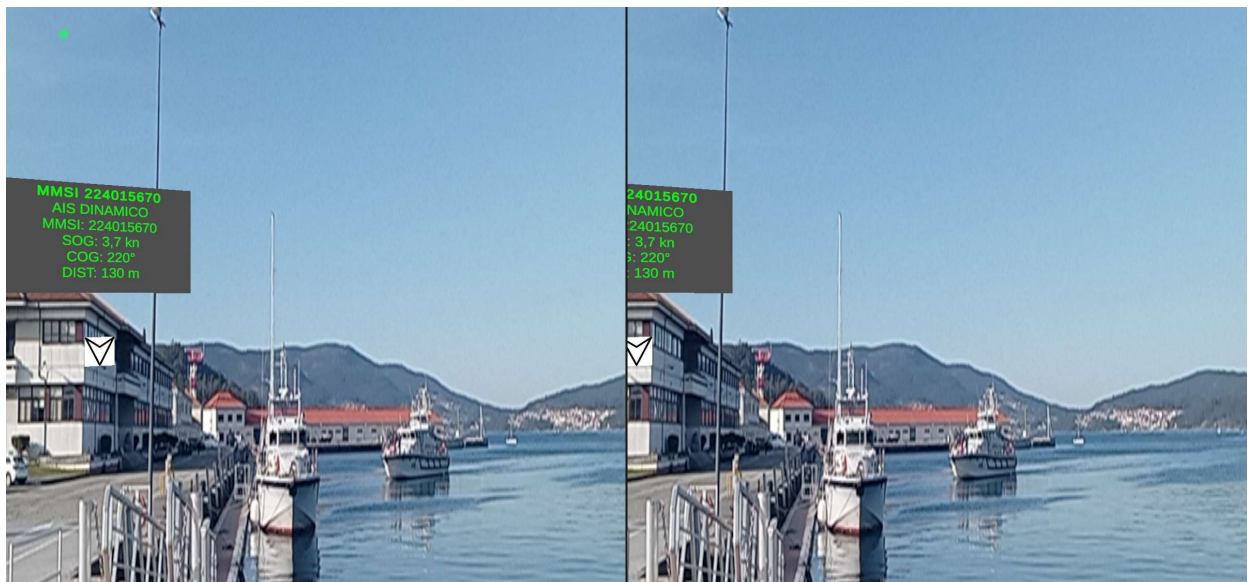


Figura 4-4 Lancha de Instrucción Chereguini entrando en puerto. (Elaboración propia)

La velocidad y el rumbo mostrados tampoco deben interpretarse como valores totalmente exactos en todo momento (véase en la Figura 4-5), en la que el marcador se encuentra más centrado pero la velocidad presenta algún ligero fallo de precisión. Esto se debe principalmente a las imprecisiones del magnetómetro y de la brújula del teléfono móvil, que condicionaban la orientación del sistema y, por tanto, la posición aparente de los marcadores sobre la escena real. De forma secundaria, también influían la recepción periódica de los mensajes AIS y el procesamiento simplificado adoptado en el prototipo.

Aun así, estos valores resultan suficientemente útiles para la finalidad del sistema, que no es ofrecer una herramienta de navegación de precisión, sino una visualización funcional e inmersiva de contactos marítimos reales.

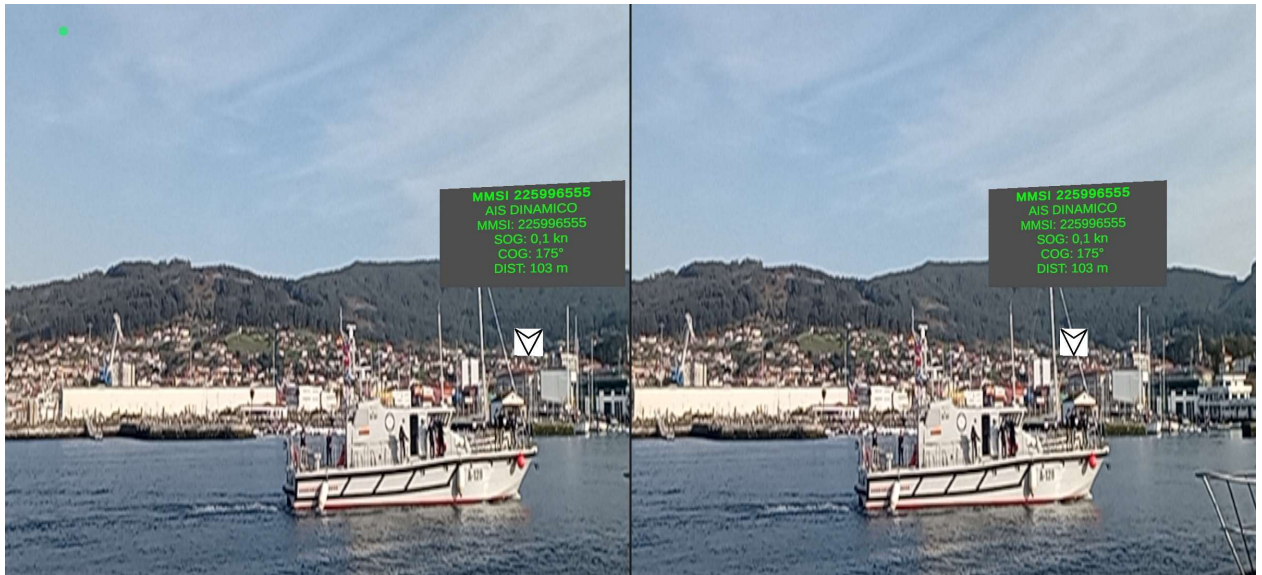


Figura 4-5 Lancha de Instrucción Godinez entrando en puerto. (Elaboración propia)

En conjunto, el resultado final puede valorarse de forma positiva. En las pruebas con gafas Cardboard, los marcadores se veían con buena nitidez y resultaban fáciles de interpretar, incluso aunque la imagen captada por la cámara presentase algo más de distorsión o menor calidad visual que el propio elemento aumentado. Por tanto, puede concluirse que el sistema logró cumplir su objetivo principal: representar de forma inmersiva y comprensible información AIS real sobre el entorno observado.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones y revisión de los objetivos

De acuerdo con los objetivos fijados al inicio del trabajo, puede concluirse que el objetivo general se ha alcanzado de forma satisfactoria. Se ha desarrollado un prototipo capaz de representar de forma inmersiva información geoespacial asociada a embarcaciones, utilizando un visor tipo Cardboard, un teléfono móvil y el motor Unity, e integrando además datos AIS reales recibidos mediante una antena y un receptor RTL-SDR.

En primer lugar, se han desarrollado y evaluado distintas estrategias de visualización inmersiva en visores de bajo coste tipo Cardboard, lo que ha permitido analizar su comportamiento en términos de fusión binocular, nitidez, distorsión, latencia y confort visual. Esta comparación ha hecho posible identificar la configuración más adecuada dentro de las limitaciones del sistema y valorar de forma práctica la viabilidad de este tipo de solución.

En relación con la visualización inmersiva, se desarrollaron y evaluaron distintas estrategias de representación para su uso con visor tipo Cardboard. Las pruebas realizadas permitieron comprobar que parámetros como el campo de visión, la disposición de las cámaras virtuales, la geometría de proyección o la colocación física del dispositivo influyen de forma muy significativa en el resultado visual. Asimismo, se constató que no todas las soluciones estudiadas eran igualmente válidas para este tipo de visor, lo que permitió comparar alternativas y determinar cuál ofrecía el comportamiento más adecuado dentro de las limitaciones del sistema. En este contexto, también pudo concluirse que, una vez fijada una configuración de funcionamiento estable, ni la latencia ni la resolución constituyeron la principal barrera del prototipo, ya que ambas se mantuvieron en niveles funcionales para la visualización ensayada. La limitación más relevante se situó, sobre todo, en la distorsión de la imagen y en la dificultad de lograr una calibración visual suficientemente precisa y cómoda en un visor de bajo coste.

También se cumplió el objetivo de integrar los sensores del teléfono móvil y desarrollar una lógica propia de posicionamiento geoespacial. A partir del uso del GPS y del magnetómetro, fue posible estimar la posición y la orientación del usuario y proyectar los marcadores en la dirección correspondiente al objetivo. Esto permitió validar que una aproximación basada en sensores integrados del teléfono puede resultar funcional para una representación geoespacial básica, aunque las pruebas también pusieron de manifiesto limitaciones en precisión y estabilidad.

Otro de los objetivos alcanzados fue la implementación de una cadena de adquisición, recepción y procesamiento de datos AIS reales. La utilización de AIS Catcher junto con la antena y el receptor RTL-SDR permitió recibir sentencias NMEA y reenviarlas a la aplicación desarrollada en Unity, donde pudieron ser procesadas para extraer información relevante, como el MMSI, la latitud, la longitud, la velocidad y el rumbo de las embarcaciones. Esto permitió trabajar con información marítima real y emplearla en la actualización de los marcadores geoespaciales dentro de la escena.

En cuanto al desarrollo de los marcadores, también puede afirmarse que el objetivo se ha cumplido satisfactoriamente. Se diseñó una solución visual capaz de mostrar información útil de cada embarcación de forma comprensible dentro de la escena, y se comprobó que dichos marcadores podían anclarse espacialmente a una posición coherente con la información recibida.

No obstante, el trabajo también ha puesto de manifiesto algunas limitaciones relevantes. La principal se encuentra en la precisión del anclaje geoespacial y en la estabilidad de la orientación, condicionadas por el uso del GPS y, especialmente, del magnetómetro del teléfono móvil. Del mismo modo, la visualización inmersiva con visor tipo Cardboard, aunque funcional, depende en gran medida de ajustes finos y de las limitaciones físicas del propio visor. Además, no se logró integrar de forma completa cierta información AIS estática, como el nombre de las embarcaciones.

En conjunto, la principal conclusión que se extrae de este Trabajo de Fin de Grado es que resulta viable desarrollar una solución de bajo coste para la representación inmersiva de datos AIS en un entorno geoespacial. Aunque el sistema obtenido no puede considerarse una solución final lista para uso operativo, sí constituye una prueba de concepto válida y funcional, capaz de integrar recepción de datos AIS reales, procesamiento en Unity, anclaje geoespacial y visualización inmersiva en una misma aplicación. En este sentido, el trabajo ha permitido cumplir de forma satisfactoria los objetivos planteados al inicio y establecer una base útil para futuros desarrollos.

5.2 Líneas futuras

Una vez desarrollado un prototipo capaz de recibir datos AIS reales, procesarlos en Unity y representarlos mediante marcadores geoespaciales en un entorno inmersivo de bajo coste, se proponen las siguientes líneas futuras orientadas a mejorar tanto la calidad de la visualización como la precisión y utilidad del sistema:

- En primer lugar, resulta de especial interés seguir profundizando en la mejora de la visualización inmersiva, ya que este ha sido uno de los aspectos centrales del trabajo. A lo largo del desarrollo se comprobó que la calidad de la experiencia visual depende de forma muy significativa de factores como la colocación física del móvil, la geometría del visor, el ajuste de las cámaras virtuales, el campo de visión, la resolución de la imagen y la calidad óptica de las lentes. Por ello, una línea futura especialmente relevante consistiría en continuar optimizando esta parte del sistema, evaluando visores de mayor calidad e incorporando técnicas de corrección de la distorsión que permitan deformar previamente la imagen para compensar los efectos introducidos por la cámara y por el visor. En Unity, esta mejora podría implementarse mediante shaders o correcciones aplicadas directamente sobre la textura.
- Otra línea futura de interés consistiría en explorar el uso de gafas de realidad virtual con modo *passthrough*, como alternativa a los visores tipo Cardboard empleados en este trabajo. Este enfoque podría facilitar una sensación de inmersión más sólida y una visualización más estable, al apoyarse en un sistema óptico y de visualización más avanzado. No obstante, su aplicación en un contexto geoespacial como el de este proyecto requeriría previsiblemente integrar sensores externos o

adicionales, en particular un sistema de posicionamiento GPS y una referencia fiable de orientación basada en magnetómetro o brújula digital.

- Perfeccionar el anclaje geoespacial de los marcadores. Aunque la solución desarrollada ha demostrado ser funcional, la precisión obtenida depende en gran medida del comportamiento del GPS y, espacialmente, del magnetómetro del teléfono móvil, así como de la calibración previa del dispositivo. Por ello, una continuación natural del trabajo consistiría en estudiar métodos que permitan reducir el error angular y mejorar la estabilidad de la representación espacial.
- Ampliar el procesamiento de los mensajes AIS. En la versión desarrollada se consiguió integrar correctamente la información dinámica más relevante, como el MMSI, la posición, la velocidad y el rumbo, pero no se llegó a incorporar de forma completa cierta información estática, como el nombre del buque. Una posible mejora futura consistiría en ampliar la decodificación de mensajes AIS para enriquecer el contenido de los marcadores y ofrecer una representación más completa de cada embarcación.
- Mejorar la gestión visual de los propios marcadores. Sería posible trabajar en una presentación más elaborada de la información, incorporando filtros por distancia, prioridad o tipo de contacto, así como mecanismos que mejoren la estabilidad temporal de los elementos mostrados cuando los mensajes AIS se reciben de forma intermitente. Esto permitiría aumentar la claridad de la escena y facilitar la interpretación de los datos por parte del usuario.
- Ampliar la fase experimental del trabajo. La realización de nuevas pruebas en diferentes condiciones de luz, visibilidad, tráfico marítimo y dispositivos permitiría valorar con mayor precisión el comportamiento del prototipo y determinar mejor su alcance real. En conjunto, todas estas mejoras permitirían evolucionar el sistema desarrollado hacia una solución más robusta, precisa y útil para futuras aplicaciones de visualización geoespacial marítima.

6 BIBLIOGRAFÍA

- [1] «AIS transponders». Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://www.imo.org/en/ourwork/safety/pages/ais.aspx>
- [2] «M.1371 : Technical characteristics for VHF automatic identification system using time division multiple access in the maritime mobile service». Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://www.itu.int/rec/R-REC-M.1371>
- [3] U. Technologies, «Unity - Manual: AR development in Unity». Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://docs.unity3d.com/6000.3/Documentation/Manual/AROverview.html>
- [4] P. D. García y M. R. Lacruz, «Desarrollo de una aplicación de Realidad Aumentada para representación de datos AIS en tiempo real».
- [5] «SDK de código abierto para Cardboard», Google for Developers. Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://developers.google.com/cardboard?hl=es-419>
- [6] «Unity 6.3 LTS ya está disponible», Unity. Accedido: 14 de marzo de 2026. [En línea]. Disponible en: <https://unity.com/blog/unity-6-3-lts-is-now-available>
- [7] R. T. Azuma, «A Survey of Augmented Reality», *Presence: Teleoperators and Virtual Environments*, vol. 6, n.º 4, pp. 355-385, 1997.
- [8] P. Milgram y F. Kishino, «A Taxonomy of Mixed Reality Visual Displays», *IEICE Transactions on Information and Systems*, vol. E77-D, n.º 12, pp. 1321-1329, 1994.
- [9] «Helmet Mounted Displays (HMD)», Elbit Systems UK. Accedido: 4 de febrero de 2026. [En línea]. Disponible en: <https://www.elbitsystems-uk.com/what-we-do/air-space/aircraft-systems/helmet-mounted-displays>
- [10] «Realidad virtual con las Google Cardboard». Accedido: 25 de marzo de 2026. [En línea]. Disponible en: <https://codigo21.educacion.navarra.es/autoaprendizaje/realidad-virtual-con-las-google-cardboard/>
- [11] «RevolutionLine - Gafas de realidad virtual 3D VR BOX | Para Smartphone IOS de Apple y Android desde 4,7 a 6 pulgadas | Películas en 3D y todo tipo de juegos | Distancia de pupila y objetivo ajustables : Amazon.es: Electrónica». Accedido: 25 de marzo de 2026. [En línea]. Disponible en: <https://www.amazon.es/RevolutionLine-Smartphone-Pel%C3%ADculas-Distancia-ajustables/dp/B0BKXS1RQ1>
- [12] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, y P. Furgale, «Keyframe-Based Visual-Inertial Odometry Using Nonlinear Optimization», *Int. J. Robot. Res.*, vol. 34, feb. 2014, doi: 10.1177/0278364914554813.
- [13] «Swiftnav». Accedido: 12 de febrero de 2026. [En línea]. Disponible en: <https://www.swiftnav.com/>

- [14] «Conceptos fundamentales | ARCore», Google for Developers. Accedido: 5 de febrero de 2026. [En línea]. Disponible en: <https://developers.google.com/ar/develop/fundamentals?hl=es-419>
- [15] AleLeo y REDACCION, «Microsoft trae a España su versátil dispositivo de realidad mixta HoloLens 2», Digital AV Magazine. Accedido: 12 de febrero de 2026. [En línea]. Disponible en: <https://www.digitalavmagazine.com/2020/09/25/microsoft-trae-espana-versatil-dispositivo-realidad-mixta-hololens-2/>
- [16] E. Kruijff, J. Swan, y S. Feiner, «Perceptual issues in augmented reality revisited», presentado en 9th IEEE International Symposium on Mixed and Augmented Reality 2010: Science and Technology, ISMAR 2010 - Proceedings, nov. 2010, pp. 3-12. doi: 10.1109/ISMAR.2010.5643530.
- [17] J. P. Rolland y H. Fuchs, «Optical Versus Video See-Through Head-Mounted Displays in Medical Visualization», *Presence Teleoperators Virtual Environ.*, vol. 9, n.º 3, pp. 287-309, jun. 2000, doi: 10.1162/105474600566808.
- [18] «Figure 2. The configuration of Video See-Through (A) and Optical...», ResearchGate. Accedido: 12 de febrero de 2026. [En línea]. Disponible en: https://www.researchgate.net/figure/The-configuration-of-Video-See-Through-A-and-Optical-See-Through-B-as-applied-in_fig9_360119167
- [19] «Apple Vision Pro», Apple. Accedido: 4 de febrero de 2026. [En línea]. Disponible en: <https://www.apple.com/apple-vision-pro/>
- [20] «Meet Meta Quest 3, Our Mixed Reality Headset Starting at \$499.99». Accedido: 4 de febrero de 2026. [En línea]. Disponible en: <https://about.fb.com/news/2023/09/meet-meta-quest-3-mixed-reality-headset/>
- [21] «HoloLens 2 es oficial, características, precio y ficha técnica». Accedido: 4 de febrero de 2026. [En línea]. Disponible en: <https://www.xataka.com/accesorios/hololens-2-caracteristicas-precio-ficha-tecnica>
- [22] «VIVE XR Elite - Base Station-Free PC VR in a Standalone Headset | VIVE European Union». Accedido: 20 de marzo de 2026. [En línea]. Disponible en: <https://www.vive.com/eu/product/vive-xr-elite/overview/>
- [23] «XREAL One», XREAL EU shop. Accedido: 20 de marzo de 2026. [En línea]. Disponible en: <https://eu.shop.xreal.com/es/products/xreal-one>
- [24] «AIS transponders». Accedido: 9 de febrero de 2026. [En línea]. Disponible en: <https://www.imo.org/en/ourwork/safety/pages/ais.aspx>
- [25] «Home | Navigation Center». Accedido: 25 de marzo de 2026. [En línea]. Disponible en: <https://www.navcen.uscg.gov/>
- [26] M. Public, «AIS Complete Guide: How It Works & Maritime Regulations», Marine Public. Accedido: 9 de febrero de 2026. [En línea]. Disponible en: <https://www.marinepublic.com/blogs/training/872993-ais-complete-guide-how-it-works-maritime-regulations>
- [27] G. Rodriguez, «Seguimiento AIS por satélite con un AIS Clase B SOTDMA», Digital Yacht Spanish Blog. Accedido: 10 de febrero de 2026. [En línea]. Disponible en: <https://digitalyacht.es/blog/2025/07/15/seguimiento-ais-por-satelite-en-transpondedores-ais-clase-b-sotdma/>
- [28] «AIS Class A Ship Static and Voyage Related Data (Message 5) | Navigation Center». Accedido: 21 de marzo de 2026. [En línea]. Disponible en: <https://www.navcen.uscg.gov/ais-class-a-static-voyage-message-5?>
- [29] «NMEA 0183», National Marine Electronics Association (NMEA) - Marine Electronics Standards, Training & Certification. Accedido: 21 de marzo de 2026. [En línea]. Disponible en: <https://www.nmea.org/nmea-0183.html>
- [30] «Unity - Manual: Manual de Unity». Accedido: 13 de marzo de 2026. [En línea]. Disponible en: <https://docs.unity3d.com/es/530/Manual/UnityManual.html>

- [31] «New w-i-d-e-b-a-n-d RTL-SDR modules - May 2020 - Silicon Chip Online». Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://www.siliconchip.com.au/Issue/2020/May/New+w-i-d-e-b-a-n-d+RTL-SDR+modules>
- [32] «RTL-SDR Blog V3 Datasheet», rtl-sdr.com. Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://www.rtl-sdr.com/rtl-sdr-blog-v3-datasheet/>
- [33] Jasper, *jdde-github/AIS-catcher-for-Android*. (15 de marzo de 2026). Java. Accedido: 15 de marzo de 2026. [En línea]. Disponible en: <https://github.com/jdde-github/AIS-catcher-for-Android>
- [34] «Redmi 10 Specs», Mi Global Home. Accedido: 18 de marzo de 2026. [En línea]. Disponible en: <https://www.mi.com/global/product/redmi-10/specs/>

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

El desarrollo de soluciones inmersivas de bajo coste para la representación de información marítima en tiempo real presenta un interés económico y social relevante, especialmente en contextos donde la conciencia situacional y la interpretación rápida del entorno resultan factores importantes. La posibilidad de utilizar un teléfono móvil, un visor tipo Cardboard y datos AIS reales para representar embarcaciones cercanas mediante marcadores geoespaciales demuestra que es posible plantear herramientas de apoyo visual sin recurrir a plataformas de alto coste. Esta aproximación puede resultar especialmente útil en ámbitos de formación, supervisión y prototipado, donde disponer de soluciones accesibles y fácilmente reproducibles supone una ventaja clara desde el punto de vista económico e industrial. Además, el uso de arquitecturas basadas en hardware comercial y de amplia disponibilidad reduce la barrera de entrada para futuros desarrollos y favorece la escalabilidad de este tipo de tecnologías.

Desde la perspectiva de la seguridad y del impacto ambiental, este tipo de sistemas puede contribuir a una mejor interpretación del tráfico marítimo y del posicionamiento relativo de las embarcaciones, lo que abre la puerta a aplicaciones futuras de apoyo a la vigilancia, la supervisión portuaria o la navegación asistida. No obstante, el trabajo también pone de manifiesto que, en su estado actual, el prototipo presenta limitaciones en precisión, calibración y estabilidad, por lo que no debe entenderse como un sistema crítico de navegación, sino como una herramienta experimental y de apoyo. En términos ambientales, el uso de dispositivos ya existentes y de pequeño tamaño constituye una alternativa más razonable que otras soluciones basadas en hardware específico de mayor complejidad, aunque, como en todo desarrollo apoyado en electrónica de consumo, deben considerarse aspectos como el consumo energético, la vida útil de los equipos y la generación de residuos electrónicos. En conjunto, el trabajo refleja una conciencia adecuada de las implicaciones económicas, de seguridad y ambientales asociadas al desarrollo de este tipo de tecnología.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El trabajo presentado se enmarca en el desarrollo de una solución inmersiva de bajo coste para la representación de información geoespacial marítima mediante datos AIS reales, utilizando un teléfono móvil, un visor tipo Cardboard y el motor Unity. Aunque se trata de un prototipo con fines académicos y experimentales, su desarrollo plantea una reflexión ética relacionada con el uso de tecnologías de visualización aplicadas a la observación del entorno y a la interpretación de información de tráfico marítimo. En este sentido, resulta importante considerar que herramientas de este tipo podrían emplearse en contextos sensibles, donde la representación aumentada de datos espaciales podría derivar no solo en usos orientados a la seguridad y la supervisión, sino también en aplicaciones de vigilancia intensiva o control no justificado. Por ello, cualquier evolución futura de este tipo de sistemas debería situarse dentro de marcos técnicos, legales y éticos adecuados, especialmente cuando se plantee su uso en entornos reales o en ámbitos de especial sensibilidad operativa.

Desde una perspectiva social, el trabajo tiene un enfoque claramente positivo, ya que explora una tecnología que puede contribuir a una comprensión más intuitiva de la información marítima y a una mejora de la conciencia situacional del usuario. La posibilidad de representar visualmente la posición y el movimiento de embarcaciones cercanas puede resultar útil en tareas de apoyo a la navegación, formación o supervisión del tráfico marítimo. No obstante, también debe señalarse que una confianza excesiva en este tipo de sistemas podría generar una falsa sensación de precisión o seguridad, especialmente cuando el propio trabajo ha puesto de manifiesto limitaciones en la estabilidad del anclaje geoespacial, en la orientación del dispositivo y en la calidad de la visualización inmersiva. Por ello, desde un punto de vista ético y social, estas tecnologías deben entenderse como herramientas complementarias de apoyo a la interpretación y a la toma de decisiones, y no como sustitutos del criterio humano ni de los sistemas profesionales certificados.

ANEXO III: PRIMERA VERSIÓN CÓDIGO GEOMARKERTEXT

```
using System.Collections;
using UnityEngine;
using TPro;

public class GeoMarkerTest : MonoBehaviour
{
    [Header("UI")]
    public RectTransform markerRoot;    // Marker_Root
    public TMP_Text markerText;        // Marker_Text

    [Header("Objetivo GPS")]
    public double targetLatitude = 42.40783;
    public double targetLongitude = -8.70757;

    [Header("Datos AIS de prueba")]
    public string vesselName = "SANTA MARIA";
    public string vesselNationality = "España";
    public float vesselSOG = 12.4f;
    public float vesselCOG = 245f;

    [Header("Brújula")]
    public bool useTrueHeading = false;
    public float headingOffset = 270f;
    public float northCorrectionOffset = -50f;
    public bool invertDelta = false;

    [Header("Ajustes")]
    public float updateInterval = 0.2f;
    public float visibleAngle = 60f;
    public bool alwaysShowDebug = false;

    [Header("Posición UI")]
    public float leftX = 220f;
    public float centerX = 320f;
    public float rightX = 420f;
    public float posY = -250f;

    IEnumerator Start()
    {
        markerText.text = "Iniciando marcador AIS...";
        markerRoot.gameObject.SetActive(true);

        if (!Input.location.isEnabledByUser)
        {
            markerText.text = "GPS desactivado";
            yield break;
        }

        Input.compass.enabled = true;
        Input.location.Start();

        int maxWait = 20;
        while (Input.location.status == LocationServiceStatus.Initializing && maxWait >
0)
        {
            markerText.text = "Iniciando GPS...";
            yield return new WaitForSeconds(1f);
            maxWait--;
        }
    }
}
```

```
if (maxWait <= 0)
{
    markerText.text = "Timeout GPS";
    yield break;
}

if (Input.location.status == LocationServiceStatus.Failed)
{
    markerText.text = "GPS falló";
    yield break;
}

while (true)
{
    LocationInfo data = Input.location.lastData;

    double userLat = data.latitude;
    double userLon = data.longitude;

    float rawHeading = useTrueHeading ? Input.compass.trueHeading :
Input.compass.magneticHeading;
    if (rawHeading < 0f) rawHeading += 360f;

    float correctedHeading = (rawHeading + headingOffset + northCorrectionOffset
+ 360f) % 360f;

    double distance = HaversineDistance(userLat, userLon, targetLatitude,
targetLongitude);
    float bearingToTarget = CalculateBearing(userLat, userLon, targetLatitude,
targetLongitude);

    float delta = Mathf.DeltaAngle(correctedHeading, bearingToTarget);
    if (invertDelta) delta = -delta;

    string side = GetSide(delta);
    bool isVisible = Mathf.Abs(delta) <= visibleAngle;

    if (alwaysShowDebug || isVisible)
    {
        markerRoot.gameObject.SetActive(true);

        if (alwaysShowDebug)
        {
            markerText.text =
                vesselName + "\n" +
                vesselNationality + "\n" +
                "SOG: " + vesselSOG.ToString("F1") + " kn\n" +
                "COG: " + vesselCOG.ToString("F0") + " °\n\n" +
                "Dist: " + distance.ToString("F0") + " m\n" +
                "Raw: " + rawHeading.ToString("F1") + " °\n" +
                "Corr: " + correctedHeading.ToString("F1") + " °\n" +
                "Bear: " + bearingToTarget.ToString("F1") + " °\n" +
                "Delta: " + delta.ToString("F1") + " °";
        }
        else
        {
            markerText.text =
                vesselName + "\n" +
                vesselNationality + "\n" +
                "SOG: " + vesselSOG.ToString("F1") + " kn\n" +
                "COG: " + vesselCOG.ToString("F0") + " °";
        }
    }
}
```

```

        UpdateMarkerPosition(side);
    }
    else
    {
        markerRoot.gameObject.SetActive(false);
    }

    yield return new WaitForSeconds(updateInterval);
}
}

void UpdateMarkerPosition(string side)
{
    Vector2 pos = markerRoot.anchoredPosition;

    if (side == "IZQUIERDA")
        pos.x = leftX;
    else if (side == "DERECHA")
        pos.x = rightX;
    else
        pos.x = centerX;

    pos.y = posY;
    markerRoot.anchoredPosition = pos;
}

string GetSide(float delta)
{
    if (delta < -15f) return "IZQUIERDA";
    if (delta > 15f) return "DERECHA";
    return "CENTRO";
}

double HaversineDistance(double lat1, double lon1, double lat2, double lon2)
{
    double R = 6371000.0;
    double dLat = Mathf.Deg2Rad * (float)(lat2 - lat1);
    double dLon = Mathf.Deg2Rad * (float)(lon2 - lon1);

    double a =
        System.Math.Sin(dLat / 2) * System.Math.Sin(dLat / 2) +
        System.Math.Cos(Mathf.Deg2Rad * (float)lat1) *
        System.Math.Cos(Mathf.Deg2Rad * (float)lat2) *
        System.Math.Sin(dLon / 2) * System.Math.Sin(dLon / 2);

    double c = 2.0 * System.Math.Atan2(System.Math.Sqrt(a), System.Math.Sqrt(1.0 -
a));
    return R * c;
}

float CalculateBearing(double lat1, double lon1, double lat2, double lon2)
{
    float lat1Rad = Mathf.Deg2Rad * (float)lat1;
    float lat2Rad = Mathf.Deg2Rad * (float)lat2;
    float dLonRad = Mathf.Deg2Rad * (float)(lon2 - lon1);

    float y = Mathf.Sin(dLonRad) * Mathf.Cos(lat2Rad);
    float x = Mathf.Cos(lat1Rad) * Mathf.Sin(lat2Rad) -
        Mathf.Sin(lat1Rad) * Mathf.Cos(lat2Rad) * Mathf.Cos(dLonRad);

    float bearing = Mathf.Atan2(y, x) * Mathf.Rad2Deg;
    return (bearing + 360f) % 360f;
}

```

```
private void OnDestroy()
{
    Input.location.Stop();
    Input.compass.enabled = false;
}
```

ANEXO IV: CÓDIGO GPS COMPASS DEBUG

```
using System.Collections;
using UnityEngine;
using TPro;

public class GPSCompassDebug : MonoBehaviour
{
    public TMP_Text gpsText;

    IEnumerator Start()
    {
        if (!Input.location.isEnabledByUser)
        {
            gpsText.text = "GPS desactivado";
            yield break;
        }

        Input.compass.enabled = true;
        Input.location.Start();

        int maxWait = 20;
        while (Input.location.status == LocationServiceStatus.Initializing && maxWait >
0)
        {
            gpsText.text = "Iniciando GPS...";
            yield return new WaitForSeconds(1f);
            maxWait--;
        }

        if (Input.location.status == LocationServiceStatus.Failed)
        {
            gpsText.text = "GPS fallo";
            yield break;
        }

        while (true)
        {
            LocationInfo data = Input.location.lastData;

            gpsText.text =
                "Lat: " + data.latitude.ToString("F6") + "\n" +
                "Lon: " + data.longitude.ToString("F6") + "\n" +
                "Compass enabled: " + Input.compass.enabled + "\n" +
                "Mag: " + Input.compass.magneticHeading.ToString("F1") + "\n" +
                "True: " + Input.compass.trueHeading.ToString("F1") + "\n" +
                "Acc: " + Input.compass.headingAccuracy.ToString("F1") + "\n" +
                "Time: " + Input.compass.timestamp.ToString("F2");

            yield return new WaitForSeconds(0.2f);
        }
    }

    private void OnDestroy()
    {
        Input.location.Stop();
        Input.compass.enabled = false;
    }
}
```


ANEXO V: CÓDIGO AIS MARKERWORLDANCHOR

```
using System.Collections;
using UnityEngine;
using TMPro;

public class AISMarkerWorldAnchor : MonoBehaviour
{
    [Header("Referencias")]
    public Transform stereoRig;
    public TMP_Text markerText;

    [Header("Objetivo GPS")]
    public double targetLatitude = 42.40783;
    public double targetLongitude = -8.70757;

    [Header("Suavizado GPS AIS")]
    public bool smoothGpsTarget = true;
    [Range(0.5f, 20f)]
    public float gpsTargetSmoothSpeed = 4f;

    [Header("Datos AIS")]
    public string vesselName = "SIN DATOS AIS";
    public string vesselNationality = "-----";
    public string vesselMMSI = "-----";
    public float vesselSOG = 0f;
    public float vesselCOG = 0f;

    [Header("Brújula")]
    public bool useTrueHeading = false;
    public float headingOffset = 270f;
    public float northCorrectionOffset = -70f;
    public bool invertDelta = false;

    [Header("Calibración brújula")]
    public bool requireCalibrationAtStart = true;
    public int calibrationMinRotations = 12;
    public float calibrationThreshold = 8f;
    public float calibrationTimeout = 12f;

    [Header("Anclaje 3D")]
    public float worldRadius = 0.6f;
    public float heightOffset = 0.00f;
    public float maxVisibleAngle = 60f;
    public float smoothSpeed = 8f;

    [Header("Actualización")]
    public float updateInterval = 0.05f;

    [Header("Visualización")]
    public bool debugMode = false;
    public bool alwaysShowMarker = true;

    private Vector3 currentLocalPos;
    private Transform markerBody;
    private Transform markerPointer;

    [Header("Debug AIS recibido")]
    [TextArea(2, 6)]
    public string lastRawNmea = "Sin mensajes AIS todavía";
```

```
private double smoothedTargetLatitude;
private double smoothedTargetLongitude;
private bool hasSmoothedTarget = false;

IEnumerator Start()
{
    markerBody = transform.Find("MarkerBody");
    markerPointer = transform.Find("MarkerPointer");

    currentLocalPos = new Vector3(0f, heightOffset, worldRadius);
    transform.localPosition = currentLocalPos;

    smoothedTargetLatitude = targetLatitude;
    smoothedTargetLongitude = targetLongitude;
    hasSmoothedTarget = true;

    if (markerBody != null)
        markerBody.gameObject.SetActive(true);

    if (markerPointer != null)
        markerPointer.gameObject.SetActive(true);

    if (markerText != null)
    {
        markerText.gameObject.SetActive(true);
        markerText.text = "Iniciando AIS...";
    }

    if (!Input.location.isEnabledByUser)
    {
        if (markerText != null)
            markerText.text = "GPS desactivado";
        yield break;
    }

    Input.compass.enabled = true;
    Input.location.Start();

    int maxWait = 20;
    while (Input.location.status == LocationServiceStatus.Initializing && maxWait >
0)
    {
        if (markerText != null)
            markerText.text = "Iniciando GPS...";
        yield return new WaitForSeconds(1f);
        maxWait--;
    }

    if (maxWait <= 0)
    {
        if (markerText != null)
            markerText.text = "Timeout GPS";
        yield break;
    }

    if (Input.location.status == LocationServiceStatus.Failed)
    {
        if (markerText != null)
            markerText.text = "GPS falló";
        yield break;
    }

    yield return new WaitForSeconds(1f);
}
```

```

if (requireCalibrationAtStart)
{
    yield return StartCoroutine(CalibrateCompassRoutine());
}

while (true)
{
    LocationInfo data = Input.location.lastData;

    double userLat = data.latitude;
    double userLon = data.longitude;

    float rawHeading = GetRawHeading();

    float correctedHeading =
        (rawHeading + headingOffset + northCorrectionOffset + 180f + 360f) %
360f;

    UpdateSmoothedTarget();

    double calcLat = smoothGpsTarget ? smoothedTargetLatitude : targetLatitude;
    double calcLon = smoothGpsTarget ? smoothedTargetLongitude : targetLongitude;

    double distance = HaversineDistance(userLat, userLon, calcLat, calcLon);
    float bearingToTarget = CalculateBearing(userLat, userLon, calcLat, calcLon);

    float delta = Mathf.DeltaAngle(correctedHeading, bearingToTarget);
    if (invertDelta) delta = -delta;

    bool isVisible = Mathf.Abs(delta) <= maxVisibleAngle;
    bool shouldShow = alwaysShowMarker || debugMode || isVisible;

    if (shouldShow)
    {
        if (markerBody != null)
            markerBody.gameObject.SetActive(true);

        if (markerPointer != null)
            markerPointer.gameObject.SetActive(true);

        if (markerText != null)
            markerText.gameObject.SetActive(true);

        float deltaRad = delta * Mathf.Deg2Rad;

        float localX = Mathf.Sin(deltaRad) * worldRadius;
        float localZ = Mathf.Cos(deltaRad) * worldRadius;

        Vector3 targetLocalPos = new Vector3(localX, heightOffset, localZ);
        currentLocalPos = Vector3.Lerp(currentLocalPos, targetLocalPos,
smoothSpeed * Time.deltaTime * 10f);
        transform.localPosition = currentLocalPos;

        if (markerText != null)
        {
            if (debugMode)
            {
                markerText.text =
                    "<b>" + vesselName + "</b>\n" +
                    vesselNationality + "\n" +
                    "MMSI: " + vesselMMSI + "\n" +
                    "SOG: " + vesselSOG.ToString("F1") + " kn\n" +
                    "COG: " + vesselCOG.ToString("F0") + " °\n" +

```

```

        "DIST: " + distance.ToString("F0") + " m\n\n" +
        "RAW: " + rawHeading.ToString("F1") + "°\n" +
        "CORR: " + correctedHeading.ToString("F1") + "°\n" +
        "BEAR: " + bearingToTarget.ToString("F1") + "°\n" +
        "DELTA: " + delta.ToString("F1") + "°\n\n" +
        "NMEA:\n" + lastRawNmea;
    }
    else
    {
        markerText.text =
            "<b>" + vesselName + "</b>\n" +
            vesselNationality + "\n" +
            "MMSI: " + vesselMMSI + "\n" +
            "SOG: " + vesselSOG.ToString("F1") + " kn\n" +
            "COG: " + vesselCOG.ToString("F0") + "°\n" +
            "DIST: " + distance.ToString("F0") + " m";
    }
}
}
else
{
    if (markerBody != null)
        markerBody.gameObject.SetActive(false);

    if (markerPointer != null)
        markerPointer.gameObject.SetActive(false);

    if (markerText != null)
        markerText.gameObject.SetActive(false);
}

yield return new WaitForSeconds(updateInterval);
}
}

private void UpdateSmoothedTarget()
{
    if (!hasSmoothedTarget)
    {
        smoothedTargetLatitude = targetLatitude;
        smoothedTargetLongitude = targetLongitude;
        hasSmoothedTarget = true;
        return;
    }

    if (!smoothGpsTarget)
    {
        smoothedTargetLatitude = targetLatitude;
        smoothedTargetLongitude = targetLongitude;
        return;
    }

    float t = Mathf.Clamp01(gpsTargetSmoothSpeed * updateInterval);

    smoothedTargetLatitude = LerpDouble(smoothedTargetLatitude, targetLatitude, t);
    smoothedTargetLongitude = LerpDouble(smoothedTargetLongitude, targetLongitude,
t);
}

private double LerpDouble(double a, double b, float t)
{
    return a + (b - a) * t;
}

```

```
IEnumerator CalibrateCompassRoutine()
{
    int rotationCount = 0;
    float elapsed = 0f;
    float previousHeading = GetRawHeading();

    if (markerText != null)
    {
        markerText.gameObject.SetActive(true);
        markerText.text = "Calibrando brújula...\nMueve el móvil en forma de 8";
    }

    if (markerBody != null)
        markerBody.gameObject.SetActive(true);

    if (markerPointer != null)
        markerPointer.gameObject.SetActive(false);

    while (elapsed < calibrationTimeout && rotationCount < calibrationMinRotations)
    {
        float currentHeading = GetRawHeading();
        float deltaHeading = Mathf.Abs(Mathf.DeltaAngle(previousHeading,
currentHeading));

        if (deltaHeading >= calibrationThreshold)
        {
            rotationCount++;
            previousHeading = currentHeading;

            if (markerText != null)
            {
                markerText.text =
                    "Calibrando brújula...\n" +
                    "Mueve el móvil en forma de 8\n" +
                    rotationCount + " / " + calibrationMinRotations;
            }
        }

        elapsed += Time.deltaTime;
        yield return null;
    }

    if (markerText != null)
    {
        if (rotationCount >= calibrationMinRotations)
            markerText.text = "Brújula calibrada";
        else
            markerText.text = "Calibración incompleta\nUsando brújula igualmente";
    }

    yield return new WaitForSeconds(1.2f);

    if (markerPointer != null)
        markerPointer.gameObject.SetActive(true);
}

float GetRawHeading()
{
    float heading = useTrueHeading ? Input.compass.trueHeading :
Input.compass.magneticHeading;
    if (heading < 0f) heading += 360f;
    return heading;
}
```

```
double HaversineDistance(double lat1, double lon1, double lat2, double lon2)
{
    double R = 6371000.0;
    double dLat = Mathf.Deg2Rad * (float)(lat2 - lat1);
    double dLon = Mathf.Deg2Rad * (float)(lon2 - lon1);

    double a =
        System.Math.Sin(dLat / 2) * System.Math.Sin(dLat / 2) +
        System.Math.Cos(Mathf.Deg2Rad * (float)lat1) *
        System.Math.Cos(Mathf.Deg2Rad * (float)lat2) *
        System.Math.Sin(dLon / 2) * System.Math.Sin(dLon / 2);

    double c = 2.0 * System.Math.Atan2(System.Math.Sqrt(a), System.Math.Sqrt(1.0 -
a));
    return R * c;
}

float CalculateBearing(double lat1, double lon1, double lat2, double lon2)
{
    float lat1Rad = Mathf.Deg2Rad * (float)lat1;
    float lat2Rad = Mathf.Deg2Rad * (float)lat2;
    float dLonRad = Mathf.Deg2Rad * (float)(lon2 - lon1);

    float y = Mathf.Sin(dLonRad) * Mathf.Cos(lat2Rad);
    float x = Mathf.Cos(lat1Rad) * Mathf.Sin(lat2Rad) -
        Mathf.Sin(lat1Rad) * Mathf.Cos(lat2Rad) * Mathf.Cos(dLonRad);

    float bearing = Mathf.Atan2(y, x) * Mathf.Rad2Deg;
    return (bearing + 360f) % 360f;
}

public void UpdateAISData(string newName, string newNationality, string newMMSI,
float newSOG, float newCOG, double newLat, double newLon)
{
    vesselName = newName;
    vesselNationality = newNationality;
    vesselMMSI = newMMSI;
    vesselSOG = newSOG;
    vesselCOG = newCOG;
    targetLatitude = newLat;
    targetLongitude = newLon;

    if (!hasSmoothedTarget)
    {
        smoothedTargetLatitude = newLat;
        smoothedTargetLongitude = newLon;
        hasSmoothedTarget = true;
    }
}

public void UpdateRawNmea(string rawMessage)
{
    lastRawNmea = rawMessage;
}

private void OnDestroy()
{
    Input.location.Stop();
    Input.compass.enabled = false;
}
}
```

ANEXO VI: CÓDIGO AIS UDP RECEIVER

```
using System;
using System.Collections.Concurrent;
using System.Globalization;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using UnityEngine;

public class AISUdpReceiver : MonoBehaviour
{
    [Header("UDP")]
    public int listenPort = 9000;
    public bool showRawMessages = true;

    [Header("Marker único")]
    public AISMarkerWorldAnchor markerController;

    [Header("Fallback")]
    public double fallbackLatitude = 42.40783;
    public double fallbackLongitude = -8.70757;

    [Header("Bloqueo de barco")]
    public bool lockFirstValidVessel = true;
    public string lockedMmsi = "";

    private UdpClient udpClient;
    private Thread udpThread;
    private volatile bool isRunning = false;

    private readonly ConcurrentQueue<string> messageQueue = new
    ConcurrentQueue<string>();
    private int receivedMessageCount = 0;

    private struct AisDecodedData
    {
        public string Mmsi;
        public double Latitude;
        public double Longitude;
        public double Sog;
        public double Cog;
    }

    void Start()
    {
        Debug.Log("[AIS UDP] Start ejecutado");

        if (markerController == null)
        {
            Debug.LogError("[AIS UDP] markerController NO asignado.");
        }

        StartListening();
    }

    public void StartListening()
    {
        if (isRunning) return;

        try
        {
```

```
        udpClient = new UdpClient(listenPort);
        isRunning = true;

        udpThread = new Thread(UDPListen);
        udpThread.IsBackground = true;
        udpThread.Start();

        Debug.Log("[AIS UDP] Escuchando en puerto " + listenPort);
    }
    catch (Exception e)
    {
        Debug.LogError("[AIS UDP] Error al iniciar: " + e.Message);
    }
}

private void UDPListen()
{
    IPEndPoint remoteEndPoint = new IPEndPoint(IPAddress.Any, listenPort);

    while (isRunning)
    {
        try
        {
            byte[] data = udpClient.Receive(ref remoteEndPoint);
            string msg = Encoding.UTF8.GetString(data).Trim();

            if (string.IsNullOrEmpty(msg))
                continue;

            string[] lines = msg.Split(new[] { "\r\n", "\n", "\r" },
StringSplitOptions.RemoveEmptyEntries);

            foreach (string line in lines)
            {
                string cleanLine = line.Trim();

                if (cleanLine.StartsWith("!AIVDM") || cleanLine.StartsWith("!AIVDO"))
                {
                    messageQueue.Enqueue(cleanLine);
                }
            }
        }
        catch (SocketException)
        {
            if (isRunning)
                Debug.LogWarning("[AIS UDP] Socket cerrado o interrumpido.");
        }
        catch (Exception e)
        {
            if (isRunning)
                Debug.LogError("[AIS UDP] Error en escucha: " + e.Message);
        }
    }
}

void Update()
{
    while (messageQueue.TryDequeue(out string message))
    {
        receivedMessageCount++;

        if (showRawMessages)
            Debug.Log("[AIS RAW] " + message);
    }
}
```

```
if (markerController == null)
    continue;

AisDecodedData decoded;
bool ok = TryDecodeAisPosition(message, out decoded);

if (!ok)
    continue;

if (lockFirstValidVessel)
{
    if (string.IsNullOrEmpty(lockedMmsi))
    {
        lockedMmsi = decoded.Mmsi;
        Debug.Log("[AIS UDP] Primer barco fijado: MMSI " + lockedMmsi);
    }

    if (decoded.Mmsi != lockedMmsi)
        continue;
}

markerController.UpdateRawNmea("MSG #" + receivedMessageCount + "\n" +
message);

markerController.UpdateAISData(
    "MMSI " + decoded.Mmsi,
    "AIS DINAMICO",
    decoded.Mmsi,
    (float)decoded.Sog,
    (float)decoded.Cog,
    decoded.Latitude,
    decoded.Longitude
);

Debug.Log(
    "[AIS DECODED] MMSI=" + decoded.Mmsi +
    " LAT=" + decoded.Latitude.ToString("F6", CultureInfo.InvariantCulture) +
    " LON=" + decoded.Longitude.ToString("F6", CultureInfo.InvariantCulture)
+
    " SOG=" + decoded.Sog.ToString("F1", CultureInfo.InvariantCulture) +
    " COG=" + decoded.Cog.ToString("F1", CultureInfo.InvariantCulture)
);
}

private bool TryDecodeAisPosition(string nmeaSentence, out AisDecodedData data)
{
    data = default;

    try
    {
        string[] parts = nmeaSentence.Split(',');
        if (parts.Length < 6)
            return false;

        string payload = parts[5];
        if (string.IsNullOrEmpty(payload))
            return false;

        string bits = ConvertPayloadToBinary(payload);
        if (bits.Length < 128)
            return false;
    }
}
```

```
int messageType = Convert.ToInt32(bits.Substring(0, 6), 2);

if (messageType < 1 || messageType > 3)
    return false;

string mmsiBits = bits.Substring(8, 30);
string sogBits = bits.Substring(50, 10);
string lonBits = bits.Substring(61, 28);
string latBits = bits.Substring(89, 27);
string cogBits = bits.Substring(116, 12);

int mmsiInt = Convert.ToInt32(mmsiBits, 2);
int rawLon = TwosComplementToInt(lonBits);
int rawLat = TwosComplementToInt(latBits);

double longitude = rawLon / 600000.0;
double latitude = rawLat / 600000.0;
double sog = DecodeSpeed(sogBits);
double cog = DecodeCourse(cogBits);

if (Math.Abs(latitude) > 90 || Math.Abs(longitude) > 180)
    return false;

data = new AisDecodedData
{
    Mmsi = mmsiInt.ToString(),
    Latitude = latitude,
    Longitude = longitude,
    Sog = sog,
    Cog = cog
};

return true;
}
catch (Exception e)
{
    Debug.LogWarning("[AIS UDP] Fallo al decodificar: " + e.Message);
    return false;
}
}

private string ConvertPayloadToBinary(string payload)
{
    StringBuilder bitString = new StringBuilder();

    foreach (char c in payload)
    {
        int sixBit = c - 48;
        if (sixBit > 40)
            sixBit -= 8;

        bitString.Append(Convert.ToString(sixBit, 2).PadLeft(6, '0'));
    }

    return bitString.ToString();
}

private int TwosComplementToInt(string bits)
{
    if (string.IsNullOrEmpty(bits))
        return 0;

    if (bits[0] == '1')
    {
```

```
        char[] inverted = bits.ToCharArray();
        for (int i = 0; i < inverted.Length; i++)
            inverted[i] = inverted[i] == '1' ? '0' : '1';

        int magnitude = Convert.ToInt32(new string(inverted), 2) + 1;
        return -magnitude;
    }

    return Convert.ToInt32(bits, 2);
}

private double DecodeSpeed(string sogBits)
{
    int sogRaw = Convert.ToInt32(sogBits, 2);

    if (sogRaw == 1023)
        return 0.0;

    return sogRaw / 10.0;
}

private double DecodeCourse(string cogBits)
{
    int cogRaw = Convert.ToInt32(cogBits, 2);

    if (cogRaw == 3600)
        return 0.0;

    return cogRaw / 10.0;
}

public void StopListening()
{
    isRunning = false;

    try
    {
        udpClient?.Close();
        udpClient = null;
    }
    catch { }

    try
    {
        if (udpThread != null && udpThread.IsAlive)
            udpThread.Join(200);
    }
    catch { }
}

private void OnDestroy()
{
    StopListening();
}

private void OnApplicationQuit()
{
    StopListening();
}
}
```


ANEXO VII: CÓDIGO CÁMARA SOLUCIÓN MONOSCÓPICA DIRECTA

```
using System.Linq;
using System.Collections;
using UnityEngine;

#if UNITY_ANDROID
using UnityEngine.Android;
#endif

public class WebcamToStereoScreens : MonoBehaviour
{
    [Header("Targets")]
    public Renderer leftScreenRenderer;
    public Renderer rightScreenRenderer;

    [Header("Camera Source")]
    public bool useBackCamera = false;
    public int requestedWidth = 1920;
    public int requestedHeight = 1080;
    public int requestedFPS = 30;

    [Header("Per-eye tuning")]
    [Range(1.0f, 1.4f)]
    public float leftZoom = 1.12f;

    [Range(1.0f, 1.4f)]
    public float rightZoom = 1.12f;

    [Range(-0.08f, 0.08f)]
    public float leftShiftX = -0.030f;

    [Range(-0.08f, 0.08f)]
    public float rightShiftX = 0.030f;

    [Range(-0.08f, 0.08f)]
    public float leftShiftY = 0f;

    [Range(-0.08f, 0.08f)]
    public float rightShiftY = 0f;

    [Header("Flip")]
    public bool mirrorX = false;
    public bool mirrorY = false;

    private WebCamTexture webcam;
    private Material leftMatInstance;
    private Material rightMatInstance;

    private IEnumerator Start()
    {
#if UNITY_ANDROID
        if (!Permission.HasUserAuthorizedPermission(Permission.Camera))
        {
            Permission.RequestUserPermission(Permission.Camera);
            while (!Permission.HasUserAuthorizedPermission(Permission.Camera))
                yield return null;
        }
#endif

        leftMatInstance = leftScreenRenderer.material;
```

```
        rightMatInstance = rightScreenRenderer.material;

        yield return StartCoroutine(StartCameraRoutine());
    }

    private IEnumerator StartCameraRoutine()
    {
        var devices = WebCamTexture.devices;
        if (devices.Length == 0)
        {
            Debug.LogError("No se encontró ninguna cámara.");
            yield break;
        }

        string selectedCamera = devices[0].name;

        if (useBackCamera)
        {
            var backCam = devices.FirstOrDefault(d => !d.isFrontFacing);
            if (!string.IsNullOrEmpty(backCam.name))
                selectedCamera = backCam.name;
        }

        if (webcam != null)
        {
            if (webcam.isPlaying) webcam.Stop();
            webcam = null;
        }

        webcam = new WebCamTexture(selectedCamera, requestedWidth, requestedHeight,
requestedFPS);
        webcam.Play();

        webcam.filterMode = FilterMode.Trilinear;
        webcam.wrapMode = TextureWrapMode.Clamp;

        float timeout = 5f;
        while (timeout > 0f && (webcam.width <= 16 || !webcam.didUpdateThisFrame))
        {
            timeout -= Time.deltaTime;
            yield return null;
        }

        Debug.Log("Resolucion real webcam: " + webcam.width + "x" + webcam.height);

        leftMatInstance.SetTexture("_MainTex", webcam);
        rightMatInstance.SetTexture("_MainTex", webcam);

        ApplyPerEyeTuning();
    }

    private void Update()
    {
        if (webcam == null || !webcam.isPlaying) return;
        ApplyPerEyeTuning();
    }

    private void ApplyPerEyeTuning()
    {
        ApplyEye(leftMatInstance, leftZoom, leftShiftX, leftShiftY);
        ApplyEye(rightMatInstance, rightZoom, rightShiftX, rightShiftY);
    }

    private void ApplyEye(Material mat, float zoom, float shiftX, float shiftY)
```

```
{
    if (mat == null) return;

    float size = 1f / zoom;
    float scaleX = mirrorX ? -size : size;
    float scaleY = mirrorY ? -size : size;

    mat.mainTextureScale = new Vector2(scaleX, scaleY);

    float baseOffsetX = (1f - size) * 0.5f;
    float baseOffsetY = (1f - size) * 0.5f;

    mat.mainTextureOffset = new Vector2(baseOffsetX + shiftX, baseOffsetY + shiftY);
}

private void OnApplicationPause(bool pauseStatus)
{
    if (webcam == null) return;

    if (pauseStatus)
    {
        if (webcam.isPlaying) webcam.Stop();
    }
    else
    {
        StartCoroutine(RestartCamera());
    }
}

private IEnumerator RestartCamera()
{
    yield return new WaitForSeconds(0.5f);
    yield return StartCoroutine(StartCameraRoutine());
}

private void OnDestroy()
{
    if (webcam != null && webcam.isPlaying)
        webcam.Stop();
}
}
```


ANEXO VIII: USO INTELIGENCIA ARTIFICIAL

En el desarrollo del presente Trabajo de Fin de Grado se ha recurrido al uso de herramientas de inteligencia artificial generativa, como recurso de apoyo en determinadas fases del proyecto. Su utilización se ha entendido en todo momento como una asistencia complementaria de carácter técnico y metodológico, subordinada siempre al criterio, la supervisión y la responsabilidad del autor.

En relación con la elaboración de la memoria, esta herramienta se ha empleado como ayuda en la revisión del lenguaje, la corrección de aspectos gramaticales y la mejora de la claridad expositiva, tomando siempre como base las ideas, la estructura y los contenidos definidos por el propio alumno. Del mismo modo, en la parte de desarrollo software, se ha utilizado como apoyo para la propuesta de fragmentos de código, la detección de errores y la agilización de ciertas tareas de programación, práctica que en entornos informales suele denominarse *vibe coding*. No obstante, tanto el diseño de la solución como la integración del código, su comprobación y la validación final del sistema han sido realizados directamente por el autor.

En consecuencia, se deja constancia de que el uso de esta herramienta no ha sustituido en ningún caso el trabajo personal, el aprendizaje ni la autoría del proyecto, correspondiendo exclusivamente al alumno la responsabilidad sobre el contenido de la memoria, el desarrollo técnico realizado y las conclusiones alcanzadas.