Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina 2026

RBVM
Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina

Telepresencia monocular: de 2D a 3D en tiempo real sin sensores dedicados

Cangas, D.[a,b,c,*], Herrera, P.J.[a], Piedra, J.A.[b,c]

[a] *Dpto. de Ingeniería de Software y Sistemas Informáticos, Universidad Nacional de Educación a Distancia, C/ Juan del Rosal, 16, 28040, Madrid, España*
[b] *Departamento de Informática., Universidad de Almería, Ctra. Sacramento s/n, 04120, La Cañada de San Urbano, Almería, España.*
[c] *Virtual Dor S.L. Spin-off de la Universidad de Almería, Ctra. Sacramento s/n, 04120, La Cañada de San Urbano, Almería, España.*

Resumen

Este trabajo evalúa la viabilidad de la reconstrucción 3D en tiempo real mediante un sistema monocular, planteándolo como una alternativa de software económica frente a los costosos sistemas de comunicación volumétrica basados en hardware especializado (LiDAR o multicámara). El objetivo central es demostrar que una arquitectura optimizada permite democratizar la telepresencia inmersiva en hardware de consumo. Se ha desarrollado un prototipo basado en microservicios que desacopla el procesamiento (*Backend* en Python/Flask) de la visualización (*Frontend* en Unity 3D). El sistema integra YOLO para segmentación y el modelo Intel MiDaS para la estimación de profundidad en tiempo real sobre GPUs NVIDIA RTX. Para garantizar la calidad, se emplea una red FSRCNN personalizada que recupera detalles geométricos finos. El *Frontend* utiliza VFX Graph para delegar el renderizado íntegramente a la GPU, alcanzando tasas estables de 30 a 60 FPS. Los resultados confirman que la tecnología es técnicamente viable para aplicaciones de consumo masivo, sentando las bases para futuras implementaciones móviles (APK).

Palabras clave: Modelado, Identificación y procesamiento de señales, Sistemas Humano Máquina, IA y control, Telemática: Control a través de Redes de Comunicación, Control basado en Datos.

Monocular Telepresence: from 2D to 3D in real time without dedicated sensors

Abstract

This work evaluates the feasibility of real-time 3D reconstruction using a monocular system, proposing it as a cost-effective software alternative to expensive volumetric communication systems based on specialized hardware (LiDAR or multi-camera). The central objective is to demonstrate that an optimized architecture can democratize immersive telepresence on consumer hardware. A microservices-based prototype has been developed that decouples processing (Backend in Python/Flask) from visualization (Frontend in Unity 3D). The system integrates YOLO for segmentation and the Intel MiDaS model for real-time depth estimation on NVIDIA RTX GPUs. To ensure quality, a custom FSRCNN network is used to recover fine geometric details. The Frontend uses VFX Graph to delegate rendering entirely to the GPU, achieving stable frame rates of 30 to 60 FPS. The results confirm that the technology is technically viable for mass consumer applications, laying the foundation for future mobile implementations (APK).

Keywords: Modelling, Identification and Signal Processing, Human Machine Systems, AI and Control, Telematics: Control via Communication Networks, Data-driven Control.

1. Introducción

Las videollamadas se han convertido en una herramienta esencial para la comunicación remota, especialmente en el ámbito personal, educativo y profesional. Según Fernández (2024) en 2023 casi el 70% de la población europea hizo uso de esta tecnología. Plataformas como Skype, Zoom o Google Meet han dominado este campo durante más de dos décadas, permitiendo interacciones "cara a cara" en un entorno bidimensional. Sin embargo, estas soluciones, aunque eficaces, no logran replicar la sensación de cercanía y presencia que se experimenta en una conversación física y, en sesiones prolongadas incluso son capaces de generar cierto nivel de fatiga en el usuario (Webb, 2021).

Con el desarrollo de tecnologías como la realidad virtual o los nuevos *wearables* inteligentes con pantalla, surge una oportunidad para evolucionar hacia un modelo de comunicación más inmersivo y cercano: las videollamadas volumétricas.

*Autor para correspondencia: dcangas1@alumno.uned.es; diegocangas6@gmail.com

Las videollamadas volumétricas buscan superar las limitaciones de las plataformas bidimensionales tradicionales al representar a los interlocutores en 3D, facilitando una interacción más natural y cercana. Recientemente, se han desarrollado sistemas que generan avatares tridimensionales en tiempo real a partir de cámaras web monoculares, combinando nubes de puntos con representaciones faciales de alta resolución para mejorar la expresividad y la interacción en entornos virtuales (Deng *et al*., 2023). Este enfoque permite representar a los interlocutores de manera tridimensional mediante nubes de puntos o mallas 3D generadas a partir de imágenes capturadas por una o varias cámaras, y visualizarlas dentro de un entorno inmersivo (Realidad Virtual) o integrarlas dentro del mundo real a través de pantallas holográficas o dispositivos de Realidad Mixta.

Al revisar las soluciones actuales, pueden encontrarse dos enfoques principalmente, por un lado, existen soluciones empresariales como Microsoft Holoportation (Orts-Escolano *et al*., 2016) o Google Beam (Nartker, 2025) que requieren de una gran cantidad de sensores dedicados (sensores de profundidad o sensores LiDAR) que realizan la tarea de escanear al usuario. Este enfoque requiere, además de una alta capacidad de cómputo para procesar en tiempo real toda la información de los sensores, una sólida infraestructura de red (hasta 10 Gbps en el caso de Google Beam) para trasmitir dichos datos. Esto da como resultado un entorno tecnológico de alto coste inviable para entornos domésticos.

El segundo enfoque, de carácter más doméstico, representado por Meta Codec Avatar (Ma *et al*., 2021), Apple Persona o Soapbox, utiliza una arquitectura de dos etapas: primero se realiza una fase de escaneo inicial (mediante sensores 3D en Apple y Meta, o un sistema de 48 cámaras en Soapbox); una vez realizado el escaneo, se puede proceder a la etapa de retransmisión, donde se envía una versión simplificada del modelo que debe descargarse antes de visualizarse (Soapbox) o datos limitados, como la "información semántica" (Cheng *et al*., 2024), que incluyen expresiones faciales, posición y orientación de la cabeza, etc. (Meta y Apple) para poder aplicarse en tiempo real sobre la malla generada originalmente.

Como puede observarse, ambos enfoques necesitan de tecnología especifica (sensores de profundidad o sistemas multicámaras) y requieren o bien de una amplia infraestructura tecnológica o bien de un proceso de dos etapas que está lejos de la facilidad de uso de una videollamada convencional.

El objetivo de este trabajo, enmarcado en una investigación más amplia sobre reconstrucción volumétrica (Cangas, 2026), consiste en estudiar la viabilidad de obtener resultados equiparables a los sistemas empresariales en un entorno doméstico, prescindiendo del hardware específico que requieren las soluciones actuales. Es por ello por lo que se propone un enfoque basado íntegramente en software, apoyado en algoritmos de visión por computador, como la estimación monocular de profundidad (Zollhöfer *et al*., 2018), la detección de regiones de interés (ROI) y la superresolución. El fin último es alcanzar el procesamiento en tiempo real sobre hardware convencional. Para validar esta propuesta, se ha desarrollado un prototipo que permitirá contrastar la hipótesis inicial.

Frente al uso estándar o de caja negra de los modelos preentrenados, la principal contribución de este trabajo radica en la arquitectura de integración, la paralelización y la adaptación algorítmica necesaria para viabilizar la telepresencia volumétrica en tiempo real sobre hardware de consumo. Concretamente, estas aportaciones se condensan en tres puntos:

I. Arquitectura de retención en VRAM y pipeline asíncrono: Se propone y evalúa un modelo de ejecución que evita el cuello de botella del bus de transferencia entre RAM y VRAM (PCIe). Al mantener y procesar el flujo tensorial íntegramente (desde la normalización hasta la inferencia paralela de YOLO y MiDaS), se logra un tiempo de viable para su ejecución en tiempo real.
II. Adaptación topológica y normalización dinámica en FSRCNN: Se presenta una reingeniería de la red FSRCNN (originalmente concebida para imágenes RGB) para operar sobre tensores de profundidad de un solo canal. El punto clave radica en la implementación de una normalización / desnormalización dinámica relativa que estabiliza la variabilidad de la salida de MiDaS, logrando restaurar bordes geométricos en menos de 3 ms sin requerir de redes generativas de alta latencia.
III. Validación del paradigma de telepresencia basado en "*Software-Only*": Se demuestra empíricamente que la orquestación optimizada de algoritmos de visión por computador puede suplir la información espacial que tradicionalmente aportan los sensores dedicados (LiDAR o multicámara), democratizando así la captura volumétrica inmersiva.

El resto del documento se organiza de la siguiente manera. La segunda sección describe la propuesta tecnológica basada en un *pipeline* desacoplado. La tercera sección presenta las decisiones técnicas y las estrategias de optimización adoptadas para materializar el sistema software propuesto. La cuarta sección muestra los resultados de la validación técnica y experimental del sistema. Finalmente, la quinta sección presenta las principales conclusiones y las líneas de investigación futuras.

2. Propuesta tecnológica: pipeline desacoplado

Para dar respuesta a los desafíos planteados, se propone el diseño de un sistema basado en un *pipeline* desacoplado. Esta estrategia busca separar la lógica de procesamiento pesado de la lógica de la representación visual. A continuación, se detalla la arquitectura del sistema y el flujo de datos que permite alcanzar el tiempo real en hardware doméstico.

2.1. Arquitectura

La arquitectura del sistema se basa en un *pipeline* desacoplado, con dos partes diferenciadas entre sí, por un lado, un *Frontend* basado en Unity3D, que se encarga de realizar las peticiones y mostrar el resultado en forma de nubes de puntos 3D. Por otro lado, un *Backend* basado en Python que se encarga de procesar la imagen capturada y devolver como resultado la estructura 3D en forma de mapa de profundidad al *Frontend*. La Figura 1 ejemplifica visualmente este funcionamiento.

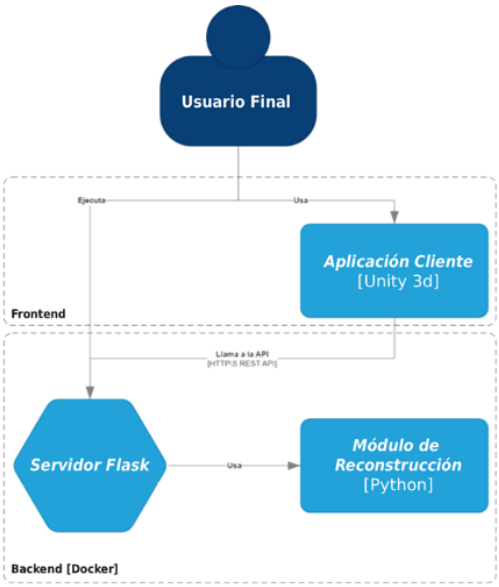

Figura 1: Diagrama C4 - N2 (Simplificado).

2.2. *Flujo de datos*

Dado que el *Backend* debe ejecutarse en tiempo real, se ha optado por una estructura de hilos paralelos intercomunicados que realicen las distintas tareas, de esta manera se evitan los "semáforos" y los tiempos de espera de la CPU y la GPU. Además, se ha tenido en cuenta sobre qué recurso hardware (CPU o GPU) se ejecuta cada módulo para evitar cuellos de botella provenientes del uso del bus PCIE para el intercambio de datos entre la RAM (CPU) y la VRAM (GPU), dado que esta es una operación que, dentro de los márgenes de trabajo (donde el procesamiento completo no debe de durar más de 66ms), es bastante costosa si se repite varias veces por ciclo de trabajo.

Teniendo en cuenta lo anterior, el *Backend* se divide en cuatro procesos principales:

I. Módulo de captura: se encarga de conectarse al servidor RTMP (la entrada de vídeo desde OBS). Abre un flujo de datos, consume las imágenes y las elimina de la pila. Este proceso debe ser independiente porque, si el tiempo de procesamiento de este bloque supera a la velocidad de llenado del *buffer* de imágenes, puede producirse un *buffer Overflow*.

II. Módulo de YOLO: identifica al usuario y genera la caja que lo contiene para reducir la carga al módulo principal (módulo de procesamiento).

III. Módulo de procesamiento: módulo principal que se encarga de generar y procesar la profundidad. Recibe la imagen del *buffer* del módulo anterior y la carga, y la procesa directamente en la GPU, ahorrando así el intercambio de datos en la CPU. Como salida, guarda en un segundo *buffer* la imagen original (recortada) del módulo de captura, junto con los datos enviados a la CPU de la profundidad.

IV. Módulo de compresión: último módulo que, a diferencia de los anteriores, no funciona en bucle sino bajo petición HTTP. Obtiene los datos del módulo anterior, los comprime en una imagen PNG y los expone al *Frontend*.

La Figura 2 muestra el flujo de datos del *Backend*.

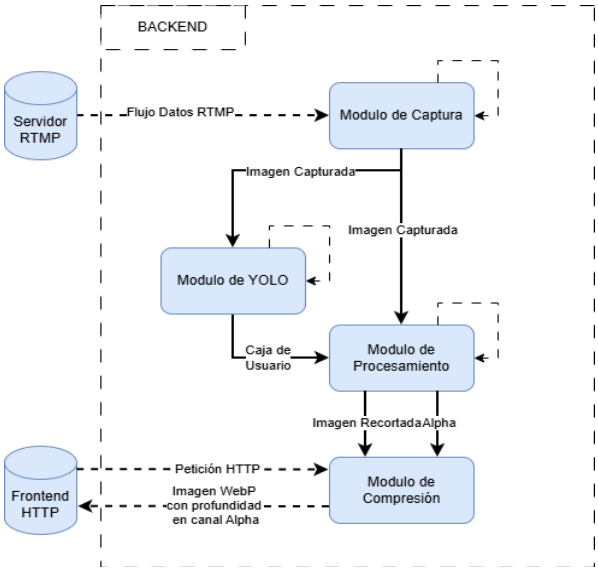

Figura 2: Diagrama Flujo Datos del *Backend.*

3. Implementación y optimización

En esta sección se detallan las decisiones técnicas y las estrategias de optimización adoptadas para materializar el sistema software propuesto. El desarrollo se ha centrado en minimizar la latencia de procesamiento para garantizar una ejecución fluida.

3.1. *Detección de la zona de interés (ROI)*

Para la etapa de atención selectiva y de segmentación del usuario se ha hecho uso del modelo YOLOv8 en su variante Nano (YOLOv8n). Esta elección responde a la necesidad de reducir la huella de memoria y maximizar la velocidad de inferencia en paralelo al proceso principal de profundidad. (Redmon *et al*., 2016). Esta versión representa una evolución respecto a sus predecesores al usar una arquitectura *anchor-free* (es decir, sin anclajes predefinidos), lo que reduce la cantidad de predicciones de cajas y acelera el proceso de inferencia.

Desde el punto de vista de la implementación del sistema, se ha hecho uso de la librería *ultralytics* bajo un esquema de ejecución asíncrona de hilos. Los parámetros de configuración que se han utilizado han sido los siguientes:

I. Precisión: se ha usado FP16 media precisión para reducir el consumo de VRAM a la mitad y acelerar la inferencia en GPUs compatibles con núcleos Tensor, sin una pérdida significativa de precisión.

II. Filtro de Clases: se ha configurado para filtrar exclusivamente la clase 0 (Personas) descartando todo lo demás para así evitar interferencias en el cálculo de la profundidad.

III. Intervalo de Ejecución: a diferencia de la estimación de profundidad, que ocurre fotograma a fotograma, la detección de YOLO se ejecuta solo cada medio segundo; esta estrategia asume que el movimiento del usuario, dado el contexto del sistema, será bajo. Esto permite liberar recursos de la GPU para el modelo de profundidad (MiDaS).

3.2. *Estimación de la profundidad*

MiDaS (Ranftl *et al*., 2020) es sin duda el núcleo de la reconstrucción volumétrica del sistema. Se seleccionó MiDaS v2.1 Small por su balance velocidad-precisión, delegando la recuperación de detalles a la etapa de superresolución. A diferencia de una implementación estándar, el modelo ha sido sometido a un proceso de optimización para garantizar su ejecución dentro de los márgenes de tiempo real (<50ms por fotograma) que los requisitos de este trabajo precisan. Para ello, se han hecho uso de tres estrategias diferenciadas:

En primer lugar, se ha integrado la compilación del modelo mediante *torch.compile*, lo que posibilita una compilación *Just-In-Time* (JIT) que permite fusionar operaciones de capas consecutivas y optimizar la ejecución del modelo a la GPU objetivo, reduciendo la sobrecarga del intérprete de Python, a costa de un periodo inicial de compilación (en las pruebas de entre 60-120 segundos) al arrancar la ejecución.

Por otro lado, se ha forzado al modelo a operar en precisión media (FP16), lo que además de reducir el consumo de memoria VRAM, permite aprovechar los *Tensor Cores* de las tarjetas gráficas modernas para acelerar las operaciones matriciales sin degradar la calidad percibida del mapa de profundidad.

Finalmente, uno de los cuellos de botella identificados durante el desarrollo fue la normalización de imágenes. Por lo que, en la implementación final, se hizo uso de un *pipeline* híbrido de preprocesamiento en el que, por un lado se hizo uso de OpenCV para ajustar la imagen de entrada al tamaño nativo de la red (256×256) y, por el otro, y a diferencia de las implementaciones normales que normalizan la imagen en la CPU, se optó por la carga completa de la imagen en la GPU para posteriormente aplicar la normalización estándar de ImageNet directamente sobre los tensores en VRAM, eliminando así, en concordancia con la filosofía general de este trabajo, ciclos innecesarios de transferencia de datos entre la RAM y la VRAM del sistema.

3.3. *Superresolución modificada (FSRCNN)*

Para abordar la limitación de los mapas de profundidad generados por MiDaS (limitados a 256×256), y tras descartar arquitecturas generativas complejas como ESRGAN debido a su alta latencia (>100ms), se ha implementado una red FSRCNN (*Fast Super-Resolution Convolutional Neural Network*) personalizada. Esta red ha sido diseñada específicamente para operar directamente sobre el tensor de profundidad normalizado de un solo canal, con un factor de escala de ×3, elevando la resolución efectiva y restaurando los bordes geométricos de la imagen.

A diferencia de la implementación original de FSRCNN diseñada para imágenes RGB (Dong *et al*., 2016), en este trabajo se ha adaptado su uso para procesar tensores de un único canal, lo que reduce la carga computacional de la primera y la última capa (al tener que procesar solo un tercio de los datos).

Uno de los desafíos técnicos identificados durante el desarrollo fue la variabilidad en el rango de valores de profundidad generados por MiDaS. Esto generaba un problema con FSRCNN que requería una entrada normalizada para funcionar correctamente. Es por ello que se realizó un proceso de normalización de los datos de manera dinámica al rango [0,1], acotando el resultado de manera relativa entre sus valores máximos y mínimos. Una vez obtenidos los resultados del reescalado, estos datos se devolvían a su rango inicial realizando un proceso de desnormalización inverso al anteriormente descrito.

Siguiendo la estrategia de MiDaS, FSRCNN se optimizó con FP16 y compilación JIT, logrando un tiempo de ejecución despreciable (<3ms) que valida su integración en el *pipeline*.

4. Análisis y evaluación de los resultados

Esta sección presenta los resultados de la validación técnica y experimental del sistema propuesto con el objetivo de verificar la hipótesis de la viabilidad de la telepresencia basada exclusivamente en software en hardware de consumo. Las pruebas se han llevado a cabo en diversos escenarios de red, haciendo uso de un sistema principal representativo del hardware de consumo actual orientado al ocio digital (AMD Ryzen 9 5900X y NVIDIA RTX 3060Ti). Cabe destacar que la clasificación de este equipo como "entorno doméstico" se apoya en encuestas de hardware de adopción masiva (como el Steam Hardware & Software Survey), las cuales sitúan consistentemente a la RTX 3060 y RTX 4060 como las dos tarjetas gráficas más usadas a nivel global en los hogares. Siendo estas de la misma gama de rendimiento que la 3060Ti (estando esta última en el décimo puesto de la encuesta de Steam de marzo de 2026). Adicionalmente, para validar la viabilidad de la arquitectura en escenarios de menores prestaciones, se ha empleado un equipo secundario más liviano (Intel i5 11300H, sin GPU dedicada) para la ejecución del *Frontend*.

4.1. *Métricas técnicas*

A continuación, se presentan las métricas que cuantifican el rendimiento del sistema dentro de un entorno de ejecución controlado. A través de estas métricas técnicas se busca identificar tanto la capacidad de cálculo del motor de IA como los posibles cuellos de botella derivados de la arquitectura propuesta. Los datos descritos en la Figura 3 muestran el equilibrio logrado entre la latencia, el uso de recursos y el rendimiento bruto del sistema.

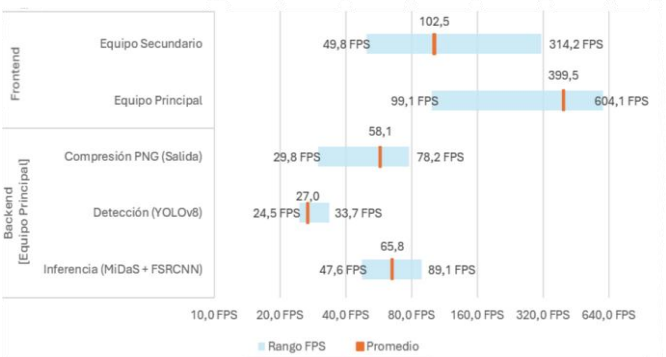

Figura 3: Rendimiento bruto del sistema.

El *Backend*, tal y como se muestra en la Figura 3, ejecutado en el equipo principal, demostró una alta capacidad de procesamiento gracias a la ejecución integral en la GPU y en el uso de precisión media (FP16). El subsistema de inferencia alcanza picos de hasta 89,1 FPS, lo que confirma que el modelo de IA no constituye un límite o un cuello de botella crítico del

sistema. Sin embargo, el rendimiento real de salida se ve condicionado por la etapa de compresión PNG que promedia 58,1 FPS.

Para contextualizar las ventajas de la arquitectura propuesta, se comparó el rendimiento del núcleo de estimación frente a una implementación estándar de MiDaS Base. Tal y como se puede observar en la Tabla 1, mientras que la versión base registra una latencia de 17,13 ms (58,38 FPS), la solución optimizada de este trabajo logra reducir el tiempo de proceso a 8,93 ms (112 FPS). Este margen de rendimiento es el que permite integrar posteriormente los módulos de YOLO y FSRCNN sin comprometer el tiempo real.

Tabla 1: Comparativa MiDaS base VS Optimizado.

Método	*Latencia*	*FPS*
MiDaS Base	17,13 ms	58,38 FPS
MiDaS Optimizado	8,93 ms	112,00 FPS

Estos resultados confirman que la optimización mediante compilación JIT y gestión de tensores en VRAM no solo es viable, sino que supone una mejora crítica frente a las implementaciones convencionales.

Por otro lado, el *Frontend* (basado en Unity), muestra un resultado muy relevante al obtener un rendimiento bruto medio de 399,5 FPS en el equipo principal y de 102,5 FPS en el equipo secundario, que no disponía de una tarjeta gráfica dedicada.

Respecto al consumo de recursos del *Backend*, tal y como se aprecia en la Figura 4, el sistema mantiene una huella de memoria muy reducida, consumiendo un promedio de 1,72 GB de RAM y reservando apenas 0,41GB de VRAM para el uso del sistema. Respecto al consumo de los recursos de procesamiento, la CPU registra picos de uso de un 116%, un uso comedido (1,16 núcleos físicos) considerando los estándares de hardware actuales. Por otro lado, la GPU opera en un rango del 90-96% de uso, lo que implica que el programa está aprovechando todos los recursos de la tarjeta gráfica para su funcionamiento.

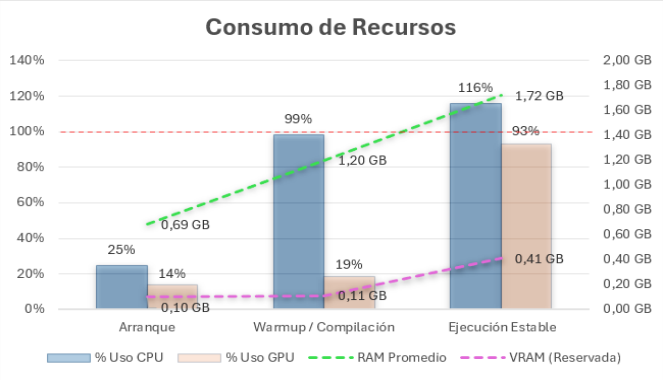

Figura 4: Consumo de Recursos del *Backend.*

Respecto al estudio de la latencia, los datos obtenidos muestran que el ancho de banda y la estabilidad de la red son factores críticos a la hora mantener la experiencia en tiempo real (Tabla 2). Por un lado, la conexión por cable Ethernet se erige como la más estable, teniendo una latencia de 31,3 ms. Esto supone un rendimiento de 22,9 FPS reales, superando el requisito de fluidez mínima de 20 FPS establecido en los requisitos.

Tabla 2: Métricas de latencia y anchos de banda.

480p c0	*Ancho de Banda (Mbps)*	*Latencia Transmisión (ms)*	*Latencia Total (ms)*	*FPS Reales*
Localhost	218,4	21,4	33,8	29,5
Cable Ethernet	186,4	31,3	43,7	22,9
Wi-FI 5 GHz	137,6	46,2	58,6	17,1
Wi-FI 2,4 GHz	91,5	76,8	89,2	11,2

Las conexiones Wi-Fi, por otro lado, muestran un peor desempeño. Mientras que la conexión de Wi-Fi de 5GHz se posiciona como una alternativa viable (46,2 ms\17,1 FPS), la conexión de 2,4 GHz se descarta debido a su alta latencia de transmisión, que sube hasta los 76,8ms, ofreciendo un rendimiento de tan solo 11,2 FPS.

4.2. Calidad perceptual

Más allá de las métricas cuantitativas, la naturaleza interactiva del sistema requiere de una evaluación de la calidad perceptual de la reconstrucción. La Figura 5 presenta un análisis comparativo entre el *ground truth* (capturado con Kinect V2), la reconstrucción inicial (MiDaS Small con interpolación bicúbica) y la solución propuesta, permitiendo observar detalles de textura y definición de bordes que las métricas globales suelen omitir.

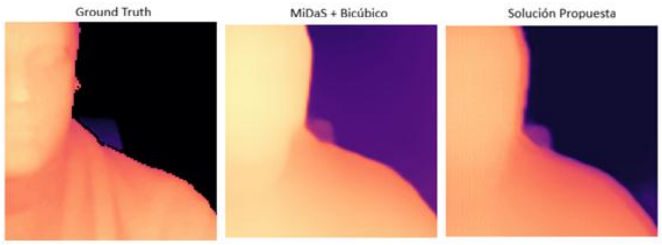

Figura 5: Comparativa de la reconstrucción de profundidad.

El modelo propuesto muestra una capacidad superior a la hora de interpretar la geometría del sujeto. Mientras que en MiDaS Base se aprecia una degradación en la zona de la mejilla (mostrando una curvatura hacia dentro del rostro), el algoritmo desarrollado logra mantener la forma y los bordes de una manera más fidedigna. Esto se traduce en transiciones más suaves y coherentes en regiones críticas como el hombro y el perfil facial. Por otro lado, la propuesta también presenta un ligero "efecto rejilla". Este fenómeno se atribuye a la transferencia de la distribución de ruido durante el entrenamiento de la red FSRCNN personalizada; al emplear datos reales capturados con el sensor Kinect V2 como *ground truth*, el modelo ha aprendido y replicado el ruido estructurado y las imperfecciones sistemáticas inherentes al hardware de captura, proyectándolas durante el proceso de superresolución espacial.

Como se observa en la Figura 6, al trasladar estos resultados al *Frontend* de Unity, el beneficio de tener bordes más definidos compensa la presencia de ruido técnico, resultando en una representación volumétrica más integrada y menos aplanada que con el método tradicional. Aunque la evaluación actual se centra en la percepción cualitativa (Figura 5), pruebas preliminares de similitud estructural apuntan a una mejora

geométrica objetiva que será evaluada en futuras iteraciones del proyecto.

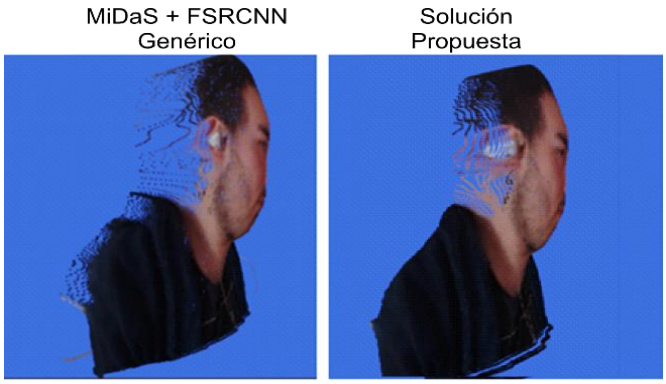

Figura 6: Vista lateral de la reconstrucción.

5. Conclusiones

Los resultados obtenidos validan la viabilidad del sistema propuesto, logrando una reconstrucción 3D fluida (22,9 FPS) sin requerir hardware especializado. La optimización mediante FSRCNN permite democratizar el acceso a la telepresencia inmersiva sobre hardware de consumo. Como trabajo futuro se plantea la migración de los protocolos de transporte a sistemas más eficientes como WebRTC con el fin de minimizar la latencia en redes locales y no locales, así como la optimización de la inferencia para su despliegue en dispositivos de menores recursos. Adicionalmente, dado el margen de cómputo remanente en la inferencia (picos de 89,1 FPS), se plantea aprovechar estos recursos para aumentar la resolución base, incrementar la frecuencia de YOLO o integrar formatos de compresión más eficientes.

Agradecimientos

Este trabajo ha sido realizado gracias al apoyo del proyecto GO-HYBRID (PID2024-156427OB-I00), financiado por MICIU/AEI/10.13039/501100011033/ FEDER, UE.

Referencias

Cangas, D. 2026. Telepresencia Monocular: Del 2D al 3D en tiempo real mediante modelos de IA sin sensores dedicados. Trabajo de Fin de Máster en Investigación en Ingeniería de Software y Sistemas Informáticos, Universidad Nacional de Educación a Distancia, Madrid.

Cheng, R., Wu, N., Varvello, M., Chai, E. Chen, S., Han, B. 2024. A First Look at Immersive Telepresence on Apple Vision Pro. Proceedings of the 2024 ACM on Internet Measurement Conference. 555–562. DOI: 10.1145/3646547.3689006

Deng, H., Zhang, Q., Jin, H., Chang-Hun, K., 2023. Real-Time Interaction for 3D Pixel Human in Virtual Environment. Applied Sciences 13, 966. DOI: 10.3390/app13020966.

Dong, C., Change, C., Tang, X., 2016. Accelerating the Super-Resolution Convolutional Neural Network. DOI: 10.48550/arXiv.1608.00367.

Fernández, R., 2024. Porcentaje de personas que hizo llamadas o videollamadas online por país en la unión europea. Statista.

Ma, H., Simon, T., Saragih, J., Wang, D., Li, Y., De La Torre, F., Sheikh, Y., 2021. Pixel Codec Avatars.

Nartker, A., 2025. Google Beam: Our AI-first 3D video communication platform.

Orts-Escolano, S., Fanello, S., Chang, W., Kowdle, A., Degtyarev, Y., 2016. Holoportation: Virtual 3D Teleportation in Real-time. Proceedings of the 29th Annual Symposium on User Interface Software and Technology, 741–754. DOI: 10.1145/2984511.2984517.

Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., & Koltun, V. 2020. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. arXiv (preprint). DOI: 10.48550/arXiv.1907.01341

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. 2016. You Only Look Once: Unified, Real-Time Object Detection. DOI: 10.48550/arXiv.1506.02640

Webb, M., 2021. Zoom Fatigue and How to Prevent It. J Registry Manag. 48, 181–182.

Zollhöfer, M., Thies, J., Garrido, P., Bradley, D., Beeler, T., Perez, P., & Stamminger, M. 2018. State of the Art on Monocular 3D Face Reconstruction, Tracking, and Applications. Computer Graphics Forum, 37(2), 523-550. DOI: 10.1111/cgf.133