CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

FlyChams: plataforma open-source escalable para simulación aérea fotorrealista

López Ruiz, J.F.[a], Mañas Álvarez, F.J.[b], Vargas, M.[a,*]

[a]*Dpto. Ing. de Sistemas y Automática. Universidad de Sevilla. Avda. de los Descubrimientos s/n. 41092, Sevilla, España.*
[b]*Dpto. Informática y Automática. Escuela Técnica Superior de Ingeniería Informática. Universidad Nacional de Educación a Distancia (UNED). C/ Juan del Rosal, 16, 28040, Madrid, España.*

To cite this article: López Ruiz, J.F., Mañas-Álvarez, F.J., Vargas, M. 2026. FlyChams: a scalable open-source platform for photorealistic aerial simulation. Jornadas de Automática, 47
https://doi.org/10.17979/ja-cea.2026.47.13824

Resumen

La simulación fotorrealista es fundamental para la investigación en robótica aérea, especialmente para tareas de percepción que involucran múltiples vehículos. Aun así, el paradigma de simuladores de código abierto resulta ser insuficiente, ya que las plataformas de referencia han sido descontinuadas o han pasado a modelos cerrados como *AirSim* —actual *AirGen*—. Además, otras alternativas más novedosas basadas en *Isaac Sim* todavía se encuentran inmaduras para flotas aéreas. Este trabajo introduce *FlyChams*, una plataforma de simulación de código abierto, alojada localmente y de alto rendimiento. Se construye sobre *Unreal Engine 5* y un *fork* propio de *AirSim*, es gobernada por el autopiloto *PX4* e integrada con *ROS 2*. Su diseño soporta escenarios exigentes —múltiples agentes, gran número de transmisiones de vídeo simultáneas y alta resolución— gracias a una capa de *streaming* acelerada por hardware. Finalmente, se propone una misión de búsqueda y rescate para ilustrar el funcionamiento de la plataforma y estudiar su escalabilidad.

Palabras clave: Robots aéreos, Trabajo en entornos reales y virtuales, Soporte al operador humano, Percepción y detección, Sistemas multi-vehículo, Redes de robots y sensores inteligentes.

FlyChams: an open-source platform for large-scale photorealistic aerial simulation.

Abstract

Photorealistic simulation is essential for aerial robotics research, particularly for perception tasks involving multi-vehicle systems. However, the current open-source simulation paradigm falls short; benchmark platforms have either been discontinued or transitioned to closed-source models, such as *AirSim* —now *AirGen*—. Meanwhile, newer alternatives based on *Isaac Sim* remain immature for aerial fleets. This work introduces *FlyChams*, a high-performance, self-hosted, open-source simulation platform. Built on *Unreal Engine 5* and a custom fork of *AirSim*, it is governed by the *PX4* autopilot and integrated with *ROS 2*. Its design accommodates demanding scenarios —including multiple agents, numerous simultaneous video streams, and high resolutions— thanks to a hardware-accelerated streaming layer. Finally, we present a search and rescue mission to demonstrate the platform's capabilities and evaluate its scalability.

Keywords: Flying robots, Work in real and virtual environments, Human operator support, Perception and sensing, Multi-vehicle systems, Networks of robots and intelligent sensors.

1. Introducción

En la última década, los vehículos aéreos no tripulados (UAV) se han consolidado como herramientas clave en una gran variedad de misiones civiles, como la inspección de infraestructuras (Jordan et al., 2018; Salahat et al., 2019), las misiones de búsqueda y rescate (Sun et al., 2016; Burke et al., 2019; Lyu et al., 2023) o el seguimiento visual de objetivos móviles (Ahmed et al., 2021; Sun et al., 2023; Chen et al.,

*Autor para correspondencia: mvargas@us.es.

2024). En muchas de estas aplicaciones, la cámara embarcada —a menudo orientable a través de un balancín motorizado (*gimbal*)— es el sensor principal, lo que lleva a la percepción visual activa a ser el centro del problema. Dicha percepción ya no se limita a la detección de objetos en la imagen: requiere decidir dónde posicionar el robot, cómo orientar los sensores y con qué nivel de detalle observar cada región de interés, sobre todo cuando la misión involucra muchos objetivos, multitud de vehículos o restricciones temporales severas.

En este contexto cobran especial relevancia estrategias que cierran el bucle entre percepción y control, desde el seguimiento activo con cámaras orientables (Hansen and de Figueiredo, 2024; Dionigi et al., 2024) hasta arquitecturas foveadas o de *zoom* digital sobre sensores de muy alta resolución (Zang et al., 2025; Zhao et al., 2025). Estas líneas complementan trabajos recientes sobre agentes aéreos multicámara con *zoom* óptico (Vargas et al., 2022, 2024) y apuntan hacia flotas en las que conviven vehículos especializados, con multitud de sensores y sistemas de control. Esta evolución desplaza el foco desde el diseño de un único agente hacia la validación de sistemas complejos, que incluyen: coordinación de flota, asignación de recursos visuales, continuidad de las transmisiones y supervisión humana de múltiples flujos de vídeo.

El desarrollo y validación de este tipo de sistemas dependen de las herramientas de simulación, ya que posibilitan pruebas de concepto y ensayos preliminares de forma segura, ágil y económica. Ahora bien, cuando el objetivo es investigar algoritmos de percepción visual activa y operación multiagente, la simulación debe satisfacer tres requisitos: un renderizado *fotorrealista*, una dinámica de vuelo *físicamente realista* y la capacidad de *escalar* a escenarios con muchos agentes, multitud de cámaras y transmisiones de vídeo de alta resolución.

Sin embargo, el panorama actual de simuladores de código abierto no cubre lo suficientemente bien esta demanda. Las plataformas que se suelen integrar con *ROS* proporcionan un renderizado mediocre, mientras que las opciones fotorrealistas o bien son cerradas y dependientes de la nube, o bien resultan todavía inmaduras para flotas complejas, como se analiza en la Sección 2. Por tanto, existe la necesidad de una plataforma de simulación aérea que sea, a la vez, fotorrealista, abierta y de alto rendimiento.

Este trabajo aborda esa necesidad mediante *FlyChams*: una plataforma de simulación de código abierto y alojada localmente construida sobre *Unreal Engine 5* (*UE5*) y un *fork* propio del simulador *Cosys-AirSim* (Jansen et al., 2023). El código fuente de la plataforma está disponible en tres repositorios públicos: la pila *ROS 2* y el control de la flota (López Ruiz et al., 2026b), el *fork* de *Cosys-AirSim* (López Ruiz et al., 2026a) y el escenario de *UE5* (López Ruiz et al., 2026c). Esta propuesta supone la evolución natural de un trabajo previo (López Ruiz et al., 2024), en el que se exploró *UE5* como una herramienta de simulación cinemática y gráfica. Esta nueva plataforma añade física realista gobernada por el autopiloto *PX4*, middleware estándar *ROS 2* (Macenski et al., 2022), operación reproducible y una capa de *streaming* de vídeo acelerada por hardware. El principal objetivo es aportar la infraestructura necesaria para ejecutar, visualizar, registrar y evaluar misiones aéreas fotorrealistas con flotas operando según dos enfoques de seguimiento multiobjetivo: *multicámara* —varias cámaras orientables con *zoom* óptico independiente— y *multiventana* —recortes digitales sobre una única cámara de muy alta resolución—. Las contribuciones principales son:

(C1) Una plataforma de simulación aérea *open-source* de alto rendimiento, que trata de suplir las carencias observadas en entornos de simulación alternativos, bien sean cerrados, como es el caso de *AirGen*, bien sean abiertos pero no totalmente adaptados aún al contexto de interés del presente trabajo, como es el caso de *Isaac Sim*/*Pegasus*.

(C2) Una integración modular *UE5–AirSim–ROS 2–PX4*, fotorrealista y físicamente realista.

(C3) Una capa de *streaming* de vídeo acelerada por hardware, que habilita la escalabilidad del sistema para escenarios complejos, en número de agentes y variedad de configuraciones.

(C4) Un flujo de trabajo reproducible, validado en un caso de uso de búsqueda y rescate marítimo multiagente.

La estructura del artículo es la siguiente. La Sección 2 repasa los entornos de simulación existentes y argumenta la oportunidad de este trabajo. La Sección 3 describe la arquitectura de la plataforma *FlyChams*. La Sección 4 ilustra su utilidad mediante un caso de uso de seguimiento multiobjetivo en una misión de salvamento y rescate (SAR) marítimo. Finalmente, la Sección 5 recoge las conclusiones y líneas de trabajo futuro.

2. Simulación para robótica aérea

La investigación en robótica aérea se condiciona de forma decisiva, en muchos casos, por la elección del simulador. A continuación se repasan las opciones más relevantes y se evalúan según cubran —o no— los tres requisitos propuestos: fotorrealismo, fidelidad física y escalabilidad.

Entre las herramientas *open-source* consolidadas, *Gazebo* (Koenig and Howard, 2004) es el simulador multi-robot más usado junto a *ROS*. Cuenta con amplio soporte de modelos UAV y controladores, pero tiene una capacidad de renderizado muy limitada. *Webots* (Michel, 2004) ofrece igualmente un catálogo amplio de robots y sensores, pero carece de renderizado fotorrealista. Ninguno de los dos satisface el requisito de fidelidad visual necesario para investigar percepción. El ecosistema *Matlab/Simulink* con el *UAV Toolbox* (The MathWorks, 2024) ofrece simulación sobre *Unreal Engine*. Aun así, su dependencia de *Matlab* no permite una transición sencilla a sistemas reales y supone un coste de licencia.

En el contexto de fotorrealismo, la referencia histórica ha sido *AirSim* (Shah et al., 2018), desarrollado por *Microsoft* sobre *Unreal Engine 4* y específicamente orientado a vehículos aéreos. Sin embargo, *AirSim* fue descontinuado y su evolución, *AirGen* (General Robotics, 2024), se ha transformado en una plataforma cerrada y en la nube, abandonando el modelo abierto que previamente poseía. Como reacción, la comunidad mantiene *forks* activos como *Cosys-AirSim* (Jansen et al., 2023), que extienden el proyecto original a versiones más recientes de *Unreal Engine*. Otras opciones basadas en motores de videojuego incluyen *Flightmare* (Song et al., 2021), que combina el renderizado de *Unity* con un motor de física flexible. Más recientemente, *NVIDIA Isaac Sim* (NVIDIA, 2024) ofrece renderizado de gran calidad, aunque su *plugin*

de robótica aérea *Pegasus* (Jacinto et al., 2024) aún es inmaduro para flotas y conlleva fuertes dependencias de hardware y licencia.

De este análisis surge una oportunidad clara: no existe una plataforma *open-source*, alojada localmente y de alto rendimiento que combine fotorrealismo, física realista y escalabilidad. Para cubrirlo, en este trabajo se construye sobre *Unreal Engine 5* (*UE5*, Epic Games (2024)) —gratuito para desarrolladores, respaldado por una amplia comunidad y con renderizado fotorrealista de carga configurable— y sobre el *fork Cosys-AirSim*, integrando además el autopiloto *PX4* y el middleware *ROS 2*, tal y como se detalla en la siguiente sección.

3. Arquitectura de la plataforma

FlyChams sigue una arquitectura modular por capas, conectadas entre sí por protocolos estándar. Estas capas dividen la simulación gráfica y física, el control de los agentes y la operación de la misión (véase la Figura 1). Esta separación permite escalar el sistema a flotas numerosas sin acoplamiento entre los distintos subsistemas. La plataforma se constituye de cuatro componentes de código abierto: el *simulador* (*UE5 + AirSim*), la pila de *control y coordinación* (*ROS 2*), el autopiloto (*PX4 + Micro XRCE-DDS*) y la interfaz gráfica (*Foxglove*). A continuación se describe cada capa.

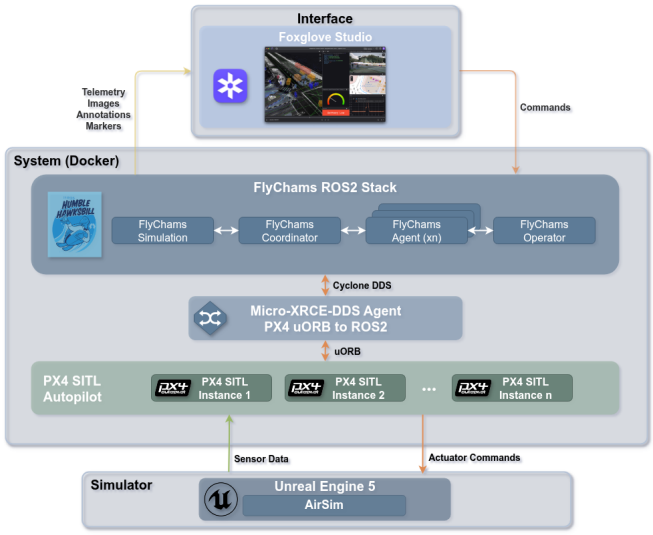


Figura 1: Arquitectura modular de *FlyChams*: la interfaz de operador (*Foxglove*), el sistema de control en contenedores *Docker* (pila *ROS 2* y autopilotos *PX4 SITL*, uno por agente) y el simulador (*UE5+AirSim*), con el flujo de datos entre ellas.

3.1. Simulador: Unreal Engine 5 y AirSim

La simulación se construye sobre *UE5* y un *fork* propio de *Cosys-AirSim*. Este último aporta el motor de física de vuelo y la simulación de sensores (IMU, GPS y barómetro con modelos de ruido configurables). La dinámica de cada agente es gobernada por el autopiloto *PX4* (v1.16) mediante *MAVLink*, en modo *Software-* o *Hardware-In-The-Loop* (SITL/HITL), lo que aporta realismo físico al comportamiento de la flota. Se han configurado dos modelos de vehículo realistas: un cuadricóptero basado en el *Holybro X500 v2* y un hexacóptero inspirado en el *DJI S900*, con parámetros físicos detallados (masa, empuje, arrastre, etc.). Por otro lado, se ha emulado una cámara/*gimbal* basada en el *DJI Zenmuse X5S*. El *gimbal* es físicamente realista, con rango angular limitado, velocidad angular máxima, tiempo de subida no despreciable y amortiguamiento no ideal. La Figura 2 muestra uno de los modelos de agente empleados.

Figura 2: Modelo de agente: cuadricóptero *X500 v2* con cámara/*gimbal Zenmuse X5S*.

Sobre este simulador se ha recreado un entorno marítimo fotorrealista (véase la Figura 3) y un conjunto de objetivos humanos. Los objetivos se han generado mediante IA generativa de modelos 3D (*Hyper3D Rodin Gen-2.5*) a partir de imágenes sintéticas. El resultado se puede observar en la Figura 4.

Figura 3: Entorno marítimo fotorrealista recreado en *UE5*: buque portacontenedores accidentado con costa desértica al fondo.

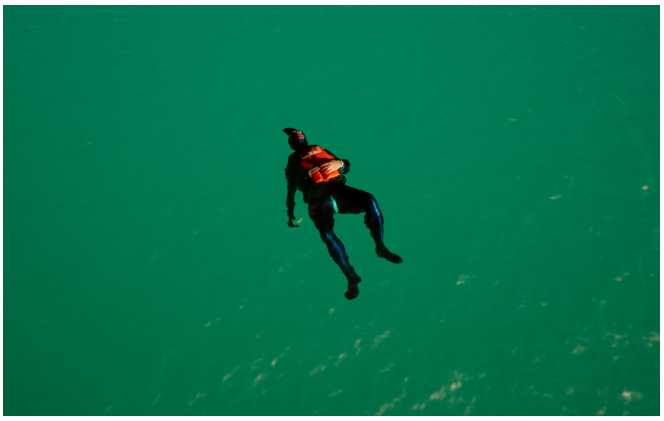


Figura 4: Ejemplo de objetivo generado mediante IA generativa de modelos 3D.

3.2. *Streaming de alto rendimiento*

Uno de los elementos diferenciales de la plataforma, directamente vinculado al rendimiento es su capa de *streaming* de vídeo, integrada en el simulador a partir de un *plugin* de *GStreamer* fuertemente modificado. Cada cámara simulada se publica como un flujo *RTSP* (servidor compartido en el puerto 8554) codificado en *H.265*, seleccionando automáticamente, según el hardware disponible, el proceso de codificación más eficiente: *NVENC* (`nvh265enc`), *VA-API* (`vah265enc`) o, como último recurso, software (`x265enc`). La tasa de bits se ajusta dinámicamente por resolución y la cola de captura se limita a dos *buffers* para minimizar el uso de VRAM. Esta capa es la que habilita escenarios con multitud de agentes y un gran número de transmisiones simultáneas, que se convertirían en un cuello de botella con una captura de imagen convencional.

3.3. *Control y coordinación en ROS 2*

La lógica del sistema se distribuye en 6 paquetes de *ROS 2 Humble* con responsabilidades bien delimitadas:

1. `flychams_api`: Definiciones de mensajes y servicios personalizados del sistema.
2. `flychams_common`: Biblioteca común que centraliza los algoritmos compartidos.
3. `flychams_coordinator`: Coordinador de la misión y los distintos agentes de la flota.
4. `flychams_agent`: Pila de control individual por cada agente del sistema.
5. `flychams_simulation`: Puente de conexión y comunicación con el entorno de simulación.
6. `flychams_operator`: Interfaz de usuario para el operador del sistema.

El coordinador es responsable de la asignación de los recursos visuales. Agrupa los objetivos en *clusters* —mediante una variante del algoritmo *K-Means* con coherencia temporal— y los asigna a los sensores disponibles con un *solver* combinatorial que pondera calidad de observación, distancia y coste de reasignación. Los detalles de este coordinador quedan fuera del alcance de este artículo; el lector puede consultar Vargas et al. (2024) y López Ruiz et al. (2024) para una descripción más detallada.

Por otra parte, cada agente ejecuta su propia pila de control en bucle cerrado. Primero, resuelve su posición óptima a partir de las posiciones de los *clusters*. Luego, calcula la orientación y el *zoom* de cada sensor para realizar un seguimiento adecuado de cada *cluster*. El control a bajo nivel se delega en el autopiloto *PX4*, con el que *ROS 2* se comunica a través de *Micro XRCE-DDS*. La integración de *PX4* exige una conversión entre los sistemas de referencia *NED* (convención de *PX4*) y *ENU* (convención de *ROS 2*).

3.4. *Operación, despliegue y reproducibilidad*

La operación de la misión se realiza desde una interfaz implementada sobre *Foxglove Studio* (vía *WebSocket*). Dicha interfaz proporciona una escena 3D, las transmisiones de vídeo con anotaciones de seguimiento y los controles de misión (Figura 5). La plataforma contempla dos enfoques de operación complementarios: un modo en tiempo real, en el que el operador supervisa de forma continua toda la información de la misión; y un modo de grabación y análisis post-misión, que registra en ficheros *rosbag* todos los datos generados, incluido cada fotograma comprimido de cada una de las cámaras. Los *rosbags* resultantes son voluminosos, pero de gran valor, ya que permiten reproducir y analizar *a posteriori* la misión completa en la interfaz de *Foxglove*.

En cuanto al despliegue, todo el sistema (exceptuando *UE5* y *Foxglove*) se ejecuta mediante contenedores *Docker* organizados en una jerarquía de imágenes *Docker* por capas. La configuración de cada misión se genera de forma declarativa a partir de una hoja de cálculo. De esta, se derivan automáticamente los ficheros de parámetros que utilizan *ROS 2*, *AirSim* y la interfaz de operación. Todo esto garantiza la reproducibilidad de la plataforma, además de facilitar la futura transición a hardware embarcado tipo *NVIDIA Jetson Orin*.

4. Caso de uso: seguimiento multiobjetivo en una misión SAR marítima

4.1. *Escenario y paradigmas de seguimiento*

Se ha recreado una misión de búsqueda y rescate (SAR) tras accidente marítimo para ilustrar las capacidades de la plataforma. El escenario reproduce un entorno costero fotorrealista, con un buque accidentado y un mar con texturas detalladas. Los objetivos de interés son náufragos, cuyas posiciones son agrupadas dinámicamente en *clusters*.

Sobre este escenario, se ensaya el despliegue de flotas con diferente número de agentes, aprovechando los dos paradigmas de seguimiento soportados por la plataforma. Por un lado, agentes *multicámara*, que orientan físicamente varias cámaras con *zoom* óptico independiente hacia cada *cluster*. Por el otro lado, agentes *multiventana*, que recortan digitalmente múltiples regiones de interés a partir de una única cámara de ultra-alta resolución. En ambos paradigmas, el agente dispone de una cámara central cenital que proporciona una vista global del área de operación. En el paradigma multicámara, dicha cámara central coexiste con varias cámaras de seguimiento, mientras que en el paradigma multiventana esa cámara central es la única presente, de la que se extraen digitalmente las regiones de interés. La Figura 5 muestra la interfaz del operador durante la misión, con la escena 3D de la flota y varias transmisiones de vídeo activas en tiempo real.

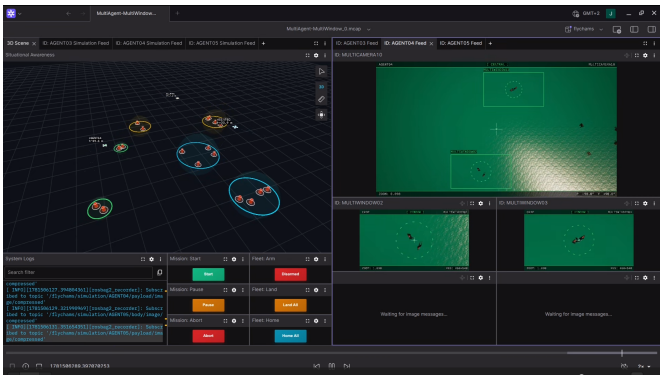
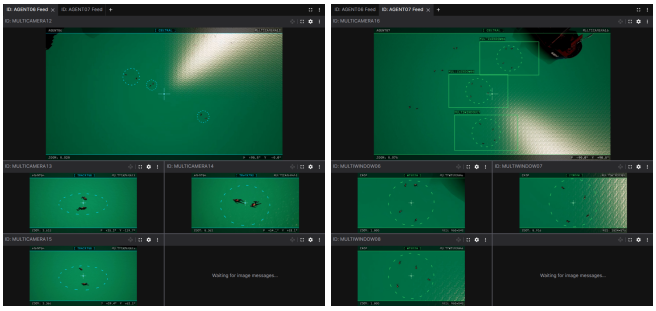


Figura 5: Interfaz del operador (*Foxglove*) durante la misión SAR: escena 3D de la flota (izquierda), registros y control de misión (inferior) y transmisiones de vídeo en tiempo real (derecha).

La Figura 6 compara las vistas de seguimiento de ambos paradigmas de seguimiento dada una configuración de *clusters* similar. A cada agente se le asignan automáticamente tantos *clusters* como cámaras/ventanas posee, y ajusta su orientación/posición y sus factores de *zoom* para encuadrarlos adecuadamente. Se proporcionan dos vídeos demostrativos del caso multiagente con tres agentes —uno en modo multicámara y otro en multiventana— en López Ruiz et al. (2026d).



(a) Agente multicámara (b) Agente multiventana

Figura 6: Vistas de seguimiento: la cámara central (arriba) proporciona una imagen general y las vistas inferiores muestran (a) el *zoom* óptico de cuatro cámaras físicas o (b) los recortes digitales de una cámara de ultra-alta resolución.

4.2. *Evaluación de la escalabilidad*

El objetivo principal de la plataforma es soportar escenarios complejos sin comprometer el rendimiento. Para evaluar esta capacidad, se ha incrementado progresivamente el número de agentes (de 1 a 3) en los dos paradigmas de seguimiento, que imponen cargas de cómputo de distinta naturaleza. El modo *multicámara* genera muchos flujos de resolución media-baja (una cámara central a 720p más dos cámaras de seguimiento a 540p por agente), mientras que el modo *multiventana* produce pocos flujos de muy alta resolución (una única cámara a 4K por agente, sobre la que se definen dos ventanas de seguimiento). Todos los flujos se codifican en *H.265* por hardware. Los ensayos se ejecutaron en un equipo portátil con GPU *NVIDIA GeForce RTX 5070 Laptop* (8 GB de VRAM), CPU *Intel Core i7-13620H* y 32 GB de RAM, con un preajuste de calidad gráfica intermedio. El estudio se acota a tres agentes por una limitación del hardware de prueba, no de la plataforma. Al tratarse de un único equipo portátil, su GPU de 8 GB de VRAM es el primer recurso en agotarse: con tres agentes el sistema ya trabaja al borde de ese límite (véase la Tabla 1). Sobre una estación de sobremesa con mayor VRAM y más núcleos de CPU cabría esperar flotas notablemente más numerosas.

# Agentes	CPU (%)	RAM (GB)	GPU (%)	VRAM (GB)
Multicámara				
1	23.8 / 53.5	6.3 / 6.5	56.0 / 73.0	3.2 / 3.2
2	44.3 / 94.2	7.8 / 8.4	67.7 / 87.0	4.9 / 5.3
3	64.1 / 91.8	8.6 / 9.3	58.0 / 94.0	6.5 / 7.2
Multiventana				
1	29.2 / 56.8	6.3 / 6.5	51.3 / 63.0	3.4 / 3.4
2	53.1 / 86.3	8.6 / 8.7	75.3 / 92.0	5.1 / 5.2
3	71.7 / 95.2	9.4 / 9.6	66.7 / 96.0	6.6 / 6.6

Tabla 1: Uso de recursos (*media*/*máximo*) en función del número de agentes y del modo de seguimiento.

La Tabla 1 resume los resultados. El uso de CPU escala de forma casi lineal con el número de agentes y constituye el primer cuello de botella, aproximándose a la saturación (picos del 95 %) con tres agentes en ambos modos. El modo *multiventana*, resulta algo más exigente en este aspecto. El uso de GPU, dominado por el renderizado de *AirSim* y la codificación de vídeo, alcanza picos del 96 % con tres agentes. Resulta especialmente relevante que el consumo de VRAM crece de forma aproximadamente lineal y se mantiene dentro del límite de 8 GB de la GPU portátil incluso con tres agentes (máximo de 7.2 GB), mientras que la RAM permanece holgadamente por debajo de los 32 GB del sistema. Esta contención de recursos es posible gracias a la codificación por hardware (*NVENC* o *VA-API*) y al sistema de calidad gráfica configurable, que permite desplazar el punto de saturación intercambiando fidelidad visual por capacidad de cómputo según las necesidades del experimento.

Conviene precisar qué entendemos por una simulación operativamente válida. Se considera válida mientras se ejecute en tiempo real y las transmisiones de vídeo se mantengan estables, sin pérdida de fotogramas ni degradación perceptible de la imagen. Este criterio se traduce, en términos de la Tabla 1, en que ningún recurso permanezca saturado de forma sostenida: los picos puntuales del 95–96 % en CPU y GPU con tres agentes son tolerables, pero marcan el umbral práctico de operación fiable en este equipo. Por debajo de él, la flota mantiene la frecuencia de simulación y la fluidez del vídeo; al superarlo, aparecen caídas de fotogramas y desincronización que invalidarían el ensayo.

5. Conclusiones

En este trabajo se ha presentado *FlyChams*, una plataforma de simulación aérea de código abierto, alojada localmente y de alto rendimiento. Esta plataforma es capaz de cubrir la necesidad que dejan las alternativas actuales: cerradas y en la nube, como *AirGen*, o aún inmaduras para flotas aéreas, como *Isaac Sim*/*Pegasus*. La plataforma integra el fotorrealismo de *Unreal Engine 5*, el control realista del autopiloto *PX4* y el middleware estándar *ROS 2*, todo ello integrado en una arquitectura modular por capas. Además, añade una capa

de *streaming* de vídeo acelerada por hardware que habilita la funcionalidad de la herramienta en escenarios complejos, con agentes dotados de múltiples cámaras de resolución variable. Su utilidad ha sido ilustrada a través de la recreación de una misión de búsqueda y rescate marítimo de múltiples objetivos utilizando distintos enfoques de seguimiento.

Como líneas de trabajo futuro se consideran la extensión de la plataforma con nuevos sensores y modalidades de simulación (*Hardware-In-The-Loop*) y la transición a hardware embarcado tipo *NVIDIA Jetson Orin*. Además, está previsto realizar simulaciones en una estación de mayor capacidad, permitiendo estudiar escenarios con más agentes, sensores y flujos de vídeo simultáneos.

Agradecimientos

Este trabajo fue financiado en parte por MICIU/AEI/10.13039/501100011033 y FEDER/UE a través de los proyectos PID2023-148456OB-C43 y PID2022-142946NA-I00, y por la Agencia Estatal de Investigación (AEI) a través de los proyectos PID2022-139187OB-I00 y PID2024-156427OB-I00

Referencias

Ahmed, I., Din, S., Jeon, G., Piccialli, F., Fortino, G., 2021. Towards collaborative robotics in top view surveillance: A framework for multiple object tracking by detection using deep learning. IEEE/CAA Journal of Automatica Sinica 8, 1253–1270. doi:10.1109/JAS.2020.1003453.

Burke, C., McWhirter, P.R., Veitch-Michaelis, J., McAree, O., Pointon, H.A., Wich, S., Longmore, S., 2019. Requirements and limitations of thermal drones for effective search and rescue in marine and coastal areas. Drones 3, 78. doi:10.3390/drones3040078.

Chen, Z., Zhao, J., Yang, M., Zhou, W., Li, H., 2024. Optimizing camera motion with MCTS and target motion modeling in multi-target active object tracking. ACM Transactions on Multimedia Computing, Communications and Applications 20, 1–19. doi:10.1145/3648369.

Dionigi, A., Felicioni, S., Leomanni, M., Costante, G., 2024. D-VAT: End-to-end visual active tracking for micro aerial vehicles. IEEE Robotics and Automation Letters 9, 5046–5053. doi:10.1109/LRA.2024.3385700.

Epic Games, 2024. The most powerful real-time 3d creation tool - unreal engine. https://www.unrealengine.com/en-us. Accessed: 2026-06-02.

General Robotics, 2024. AirGen: A high-fidelity generative simulation platform for robotics. https://www.generalrobotics.company/post/airgen. Accessed: 2026-05-30.

Hansen, J.G., de Figueiredo, R.P., 2024. Active object detection and tracking using gimbal mechanisms for autonomous drone applications. Drones 8, 55. doi:10.3390/drones8020055.

Jacinto, M., Pinto, J., Patrikar, J., Keller, J., Cunha, R., Scherer, S., Pascoal, A., 2024. Pegasus simulator: An isaac sim framework for multiple aerial vehicles simulation, in: 2024 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, Chania - Crete, Greece. pp. 917–922. doi:10.1109/ICUAS60882.2024.10556959.

Jansen, W., Verreycken, E., Schenck, A., Blanco-Claraco, J.L., Lenssen, S., Steckel, J., 2023. Cosys-AirSim: A real-time simulation framework expanded for complex industrial applications. https://github.com/Cosys-Lab/Cosys-AirSim. Accessed: 2026-05-30.

Jordan, S., Moore, J., Hovet, S., Box, J., Perry, J., Kirsche, K., Lewis, D., Tse, Z.T.H., 2018. State-of-the-art technologies for UAV inspections. IET Radar, Sonar & Navigation 12, 151–164. doi:10.1049/iet-rsn.2017.0251.

Koenig, N., Howard, A., 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator, in: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Sendai, Japan. pp. 2149–2154. doi:10.1109/IROS.2004.1389727.

López Ruiz, J.F., Mañas-Álvarez, F.J., Vargas, M., 2026a. FlyChams-Cosys-AirSim: Fork de Cosys-AirSim con extensiones propias de la plataforma FlyChams. https://github.com/JoseLopez36/FlyChams-Cosys-AirSim. Accessed: 2026-06-03.

López Ruiz, J.F., Mañas-Álvarez, F.J., Vargas, M., 2026b. FlyChams-ROS2: Repositorio principal de la plataforma FlyChams (ROS2, PX4, coordinación de flota y control de agentes). https://github.com/JoseLopez36/FlyChams-ROS2. Accessed: 2026-06-03.

López Ruiz, J.F., Mañas-Álvarez, F.J., Vargas, M., 2026c. FlyChams-Sim-UE5: Escenario de simulación fotorrealista en Unreal Engine 5 para la plataforma FlyChams. https://github.com/JoseLopez36/FlyChams-Sim-UE5. Accessed: 2026-06-03.

López Ruiz, J.F., Mañas, F.J., Vargas, M., 2026d. Vídeos demostrativos del caso multiagente (3 agentes): modos multicámara y multiventana. https://uses0-my.sharepoint.com/:f:/g/personal/josloprui6_alum_us_es/IgBYfLFGeKbYSIC--lnfk13BATYRBP79RKAvD6RYQ0uDCBc?e=VmxL8c. Accessed: 2026-06-04.

López Ruiz, J.F., Rico Ranea, J., Álamo Cantarero, T., Gil Ortega Linares, M., Vargas Villanueva, M., 2024. Monitorización mediante vehículos aéreos multicámara. caso de uso de unreal engine, in: Jornadas de Automática, Comité Español de Automática (CEA), Malaga, Spain. doi:10.17979/ja-cea.2024.45.10800.

Lyu, M., Zhao, Y., Huang, C., Huang, H., 2023. Unmanned aerial vehicles for search and rescue: A survey. Remote Sensing 15, 3266. doi:10.3390/rs15133266.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7, eabm6074. doi:10.1126/scirobotics.abm6074.

Michel, O., 2004. Cyberbotics ltd. webots™: professional mobile robot simulation. International Journal of Advanced Robotic Systems 1, 5. doi:10.5772/5618.

NVIDIA, 2024. NVIDIA Isaac Sim: Robotics simulation and synthetic data generation. https://developer.nvidia.com/isaac/sim. Accessed: 2026-05-30.

Salahat, E., Asselineau, C.A., Coventry, J., Mahony, R., 2019. Waypoint planning for autonomous aerial inspection of large-scale solar farms, in: IECON 2019-45th Annual Conference of the IEEE Industrial Electronics Society, IEEE, Lisbon, Portugal. pp. 763–769. doi:10.1109/IECON.2019.8927123.

Shah, S., Dey, D., Lovett, C., Kapoor, A., 2018. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, in: Field and Service Robotics: Results of the 11th International Conference, Springer. pp. 621–635. doi:10.1007/978-3-319-67361-5_40.

Song, Y., Naji, S., Kaufmann, E., Loquercio, A., Scaramuzza, D., 2021. Flightmare: A flexible quadrotor simulator, in: Conference on Robot Learning, PMLR. pp. 1147–1157.

Sun, J., Li, B., Jiang, Y., Wen, C., 2016. A camera-based target detection and positioning UAV system for search and rescue (SAR) purposes. Sensors 16, 1778. doi:10.3390/s16111778.

Sun, P., Li, S., Zhu, B., Zuo, Z., Xia, X., 2023. Vision-based fixed-time uncooperative aerial target tracking for UAV. IEEE/CAA Journal of Automatica Sinica 10, 1322–1324. doi:10.1109/JAS.2023.123510.

The MathWorks, I., 2024. Unreal engine simulation for unmanned aerial vehicles - matlab and simulink. https://es.mathworks.com/help/uav/ug/3d-simulation-for-unmanned-aerial-vehicles.html. Accessed: 2026-06-02.

Vargas, M., Vivas, C., Alamo, T., 2024. Optimal positioning strategy for multi-camera, zooming drones. IEEE/CAA J. Autom. Sinica 11, In press. doi:10.1109/JAS.2024.124455.

Vargas, M., Vivas, C., Rubio, F.R., Ortega, M.G., 2022. Flying chameleons: A new concept for minimum-deployment, multiple-target tracking drones. Sensors 22, 2359. doi:10.3390/s22062359.

Zang, Z., Kim, D.Y., Zeng, Y., Gao, L., 2025. Content-aware foveated camera for multi-target tracking. Optics Express 33, 43359–43368. doi:10.1364/OE.577673.

Zhao, Q., Dong, L., Chu, X., Liu, M., Kong, L., Zhao, Y., 2025. Particle filter tracking system based on digital zoom and regional image measure. Sensors 25, 880. doi:10.3390/s25030880.