uc3m | Universidad Carlos III de Madrid
Escuela Politécnica Superior

ESCUELA POLITÉCNICA SUPERIOR
MASTER EN ROBÓTICA Y AUTOMATIZACIÓN

TRABAJO DE INVESTIGACIÓN TUTELADO

Navegación autónoma de un drone mediante visión 2D.

Realizado por
André Jose Lorenzo Torres
Tutorizado por
David Martin ...
Departamento
Laboraatorio de sistemas inteligentes

UNIVERSIDAD CARLOS III
MADRID, 04 de Junio de 2025

Índice de contenidos

CAPÍTULO 1

Introducción

Este proyecto dispone del código fuente, ejemplos y procedimientos de instalación en un repositorio público de GitHub.

Uno de los grandes problemas actualmente en la navegación de UAV, es la evitación de obstáculos. La mayoría de los drones no tienen la capacidad de disponer de un sistema lidar, cámaras 3D (RGB-D) o incluso sonar, debido claro está a al peso, el coste o la capacidad de embarcar a su vez un ordenador que maneje los datos, como si que es el caso de un vehículo autonomo terrestre. Todavía existen más problemas cuando hablamos de drones de bajo coste, los cuales están muy de moda a día de hoy para la creación de enjambres o "swarms".

1.1. Motivación

La idea de conseguir un sistema en tiempo real que sea capaz de maniobrar en un espacio obstaculizado sin necesidad de costosos sensores y complejos algoritmos suena interesante, por ello este trabajo de investigación trata de dar solución a este claro problema contando exclusivamente con una cámara de baja resolución RGB. En concreto, se plantea el reto de identificar obstáculos y generar rutas alternativas en tiempo real empleando solo una cámara RGB, sin información explícita de profundidad ni sensores adicionales.

Esta aproximación no solo abre la puerta a soluciones más económicas y ligeras, sino que también plantea nuevas posibilidades para aplicaciones donde el tamaño, el peso o la disponibilidad energética del dron son factores críticos. Desde misiones de exploración en interiores hasta operaciones de reconocimiento en entornos no estructurados, un sistema de navegación visual basado en una única cámara puede ofrecer ventajas significativas si se logra extraer información suficiente del entorno a partir de las imágenes.

Entre los distintos estimadores de profundidad monocular disponibles, el modelo MiDaS destaca especialmente por su equilibrio entre precisión y velocidad. Este estimador se basa en arquitecturas modernas como Vision Transformers (ViTs), que permiten capturar relaciones espaciales globales dentro de la imagen con gran eficiencia. En particular, las versiones ligeras de MiDaS, como MiDaS v2.1 small o las variantes con ViT-S, han sido optimizadas para funcionar en tiempo real incluso en sistemas con recursos limitados, lo que las convierte en candidatas ideales para su uso embarcado o en aplicaciones donde la latencia es crítica.

Gracias a esta arquitectura, MiDaS puede generar mapas de profundidad coherentes y densos a partir de una única imagen RGB, permitiendo extraer información estructural del entorno sin necesidad de sensores físicos adicionales. Esta capacidad, unida a su rendimiento en escenarios no estructurados y su generalización a distintas condiciones de iluminación o texturas, lo convierte en una herramienta esencial dentro de este trabajo, facilitando la navegación visual autónoma en drones ligeros y de bajo coste.

1.2. Herramientas utilizadas

En este proyecto, la principal preocupación ha sido la velocidad de procesamiento de imágenes, tanto en el envío de estas al ordenador como en las fases de inferencia y procesamiento previas y posteriores a dicha inferencia. En este apartado se describen las herramientas utilizadas, incluyendo los lenguajes, plataformas, librerías y versiones correspondientes.

El código se ha desarrollado en C++ (estándar C++20) y se ejecuta en un sistema Linux Ubuntu 22.04 LTS. Para la inferencia de modelos se ha empleado la versión 2.7 de LibTorch (la versión nativa de PyTorch en C++), con aceleración por CUDA 12.1. Además, se ha utilizado OpenCV 4.12, compilado específicamente para soportar aceleración mediante cuDNN y CUDA.

El hardware utilizado incluye un procesador Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz con 8 núcleos, junto con una GPU GeForce MX110.

En el lado del dron, se emplea una cámara ESP32-CAM con una resolución de 640×480 px (VGA), la cual dispone de conectividad Wi-Fi para la transmisión de datos al ordenador de procesamiento en tierra.

CAPÍTULO 2

Desarrollo

En el presente capítulo se expone el desarrollo de las distintas implementaciones evaluadas. Dada la naturaleza investigadora de este proyecto, la arquitectura —al igual que el resto de componentes del trabajo— ha sido desarrollada en múltiples variantes tanto a nivel de software como de hardware, con el objetivo de probar distintas aproximaciones y determinar cuál ofrece la mejor solución al problema planteado.

2.1. Arquitectura de la comunicación

La arquitectura del sistema se divide en 2 secciones, la comunicación entre la cámara (espnow) y el ordenador central, y por otro lado, la arquitectura software de procesamiento de las imágenes tanto en el código de la cámara como en el ordenador principal.

2.1.1. Protocolos y Hardware

La comunicación es uno de los aspectos clave en este proyecto por dos razones principales. En primer lugar, un mal planteamiento en esta etapa puede generar un cuello de botella que limite la velocidad a la que es posible procesar los fotogramas. En segundo lugar, se trata de la primera etapa de toda la implementación, y por tanto constituye la base sobre la que se apoyará el resto del sistema.

Por esta razón, se han desarrollado y probado dos métodos de comunicación.

El primero se basa en la incorporación de un tercer dispositivo (ESP32 master) que actúa como bridge entre la cámara y el ordenador. Esta estrategia se eligió porque el protocolo **espnow** ofrece una distancia máxima de hasta 480 m, lo que favorece la comunicación con el dron desde tierra. Además, este protocolo destaca por su baja latencia, lo que resulta especialmente ventajoso en aplicaciones en tiempo real.

El esquema de esta arquitectura se muestra en la **figura 2.1**.

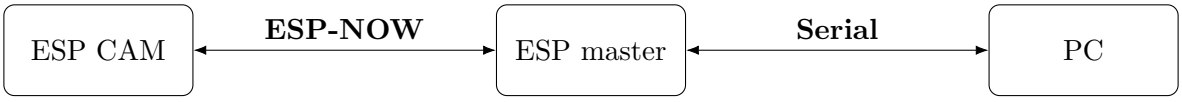

Figura 2.1: Esquema de comunicación entre módulos: ESP CAM, ESP master y PC mediante protocolo espnow.

El segundo esquema de comunicación es más sencillo y se compone únicamente de la cámara y el ordenador. En este caso, la comunicación se realiza mediante Wi-Fi, utilizando el protocolo TCP para garantizar la correcta recepción de los datos. Esta configuración se muestra en la **figura 2.2**.

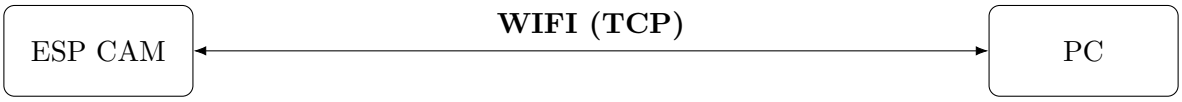

Figura 2.2: Esquema de comunicación entre módulos: ESP CAM y PC mediante Wifi 2.4 Ghz.

En la **tabla 2.1** se muestran las diferencias entre los dos protocolos, cuales son sus ventajas y desventajas. Como se puede observar, el protocolo espnow parace ser perfecto para la tarea, sobre todo por el rango y la baja latencía, por ello este fue la primera arquitectura que se probó.

Especificación	**ESPNOW**	**WIFI (TCP)**
Frecuencia	2.4 GHz	2.4 GHz
Velocidad	200 - 300 Kbps	1 - 10 Mbps
Latencia	2 - 5 ms	10 - 100 ms
Consumo	Bajo	Medio
Fiabilidad	Nula	Muy Alta
Compatibilidad	solo ESP32	Muy alta
Rango	$<$ 480 m	$<$ 300 m
Tamaño del paquete	250 bytes	2500 bytes

Tabla 2.1: Comparación de especificaciones entre protocolos de comunicación.

2.1.2. Software

El flujo de operación para la arquitectura nº 1 (ESP-NOW) se basa en dos bucles de comunicación infinitos. En el primero, la cámara y el ESP32 master intercambian los fotogramas capturados por la cámara a la máxima velocidad que esta permite. El proceso consiste en capturar la imagen, dividirla en n paquetes de 200 bytes cada uno (reservando 50 bytes para un posible envío de telemetría), y enviarlos mediante ESP-NOW. A cada imagen se le añaden los siguientes campos: un identificador de paquete, el número total de paquetes y el tamaño total de la imagen en bytes, con el siguiente formato:

$$[id][n][size][img]$$

En el lado receptor, el ESP32 master va recibiendo los paquetes hasta completar la imagen. A continuación, verifica si se ha perdido alguno; en caso afirmativo, solicita a la cámara el reenvío del paquete faltante, indicando su identificador correspondiente.

Para esta arquitectura, la cámara captura imágenes con una resolución de 320x240x3 px y las comprime en JPEG con una calidad de 10. El tamaño medio resultante es de aproximadamente 8000 bytes, lo que equivale a unos 40 paquetes por imagen. Estas imágenes se almacenan temporalmente en la memoria flash (SPIFFS) del ESP32 master, que actúa como búfer intermedio.

Posteriormente, el segundo bucle de comunicación se establece entre el ESP32 master y el ordenador, donde la imagen JPEG se serializa y se transmite por puerto serie a la máxima velocidad soportada por el sistema, en este caso 460800 baudios.

Los resultados obtenidos para la mejor versión de este algoritmo (sin realizar ninguna operación matemática en el ordenador, únicamente mostrando la imagen con OpenCV) alcanzan una velocidad de entre 0.5 y 1.5 fps. No obstante, una proporción significativa de los fotogramas llega corrupta, a pesar de haberse implementado mecanismos de aseguramiento de entrega de paquetes.

En el caso de la arquitectura nº 2, el procedimiento es considerablemente más sencillo. La cámara genera una red Wi-Fi a la que el ordenador se conecta directamente, estableciendo una comunicación punto a punto fiable. Gracias a la mayor velocidad de transmisión que ofrece esta interfaz, los resultados son significativamente más prometedores, alcanzando una velocidad de recepción de entre 150 y 200 fps, con imágenes de 640x480x3 px comprimidas en JPEG con una calidad de 15 y sin pérdida de datos, gracias al uso del protocolo TCP.

A causa de la baja velocidad de transmisión en la comunicación por espnow, se escogió la arquitectura Nº 2 como base para el resto del proyecto.

2.2. Modelos para inferencia de datos

AL llegar a esta parte del proyecto se desarrollo una investigación sobre las principales implementaciones de estimación de profundidad, entre muchos modelos destaca MiDaS v2.1 por su rapidez y depth anything v2 por su estabilidad y precisión. Los dataset con los que han sido entrenados también son un punto extremadamente importante a la hora de escoger el modelo, en este caso destacan **KITTY y cityscapes**, 2 datasets de todo tipo de sensores 3D en exteriores. Otro punto importante que se ha buscado en todo momento es que el modelo exista o se pueda crear en formato onnx (Open Neural Network Exchange), este es un formato estandarizado que solo cuenta con funciones simples y que promete hacer más rápida la inferencia de los datos, por otro lado, esto también limita la búsqueda a causa de que no todos los modelos son convertibles a onnx y la conversión para los que si lo son no es trivial.

En la **tabla** se muestra un pequeño análisis de "mercado" sobre los modelos que existen junto con sus puntos fuertes y su tamaño, en la **tabla** se muestran los modelos que se pudieron transformar o conseguir en onnx y por tanto ser probados para así, comprobar su tiempo de inferencia en el código.

Al realizar las pruebas se obtuvo una conclusión clara, el único modelo viable para el

proyecto es el MiDaS small v2.1 y el modelo depth Anything v2.0 como opción B.

2.3. Aceleración y optimización

2.4. Inferencia de la dirección de movimiento

2.5. Interpolación de distancias reales

2.6. Fine tuning del detector de objetos

CAPÍTULO 3

Pruebas y resultados

3.1. Pruebas de arquitecturas de comunicación

3.2. Validación de la velocidad de inferencia y benchmarking del algoritmo

3.3. Estabilización de la inferencia de profundidad

3.4. Pruebas de interpolación de distancias

3.5. Prueba en escenario real

CAPÍTULO 4

Conclusiones y lineas futuras

Apéndices

APÉNDICE A

Algoritmos desarrollados