ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

Diseño de un Controlador para el Seguimiento de Trayectorias de un Vehículo Autónomo no Tripulado de Ala Rotatoria.

PROYECTO DE TITULACIÓN

Previo a la obtención del Título de:

Magíster en Automatización y Control

Presentado por:

Andrés Marcelo Carranco Herrera

GUAYAQUIL, ECUADOR

Año: 2024

DEDICATORIA

El presente proyecto lo dedico a mi familia, quienes siempre han estado a mi lado brindándome su apoyo incondicional. A mis padres, por sus enseñanzas, me han inspirado a alcanzar mis metas, por su compañía y motivación en los momentos de dificultad.

También quiero dedicar este trabajo a mis profesores, quienes me guiaron y compartieron su conocimiento, fomentando en mí la pasión por el aprendizaje.

Finalmente, a todas las personas que han contribuido de alguna manera en este camino, gracias por ser parte de mi vida y de este logro.

AGRADECIMIENTOS

Mi más sincero agradecimiento primeramente a Dios, por ser mi guía y fortaleza en cada paso de este camino, su luz ha iluminado mis días, brindándome la fe y la esperanza necesarias para superar los desafíos.

A mi familia, por su amor incondicional y su apoyo constante. Han sido mi refugio y mi inspiración, motivándome a seguir adelante incluso en los momentos más difíciles. Cada sacrificio que han hecho por mí ha sido un pilar fundamental en mi vida.

A la Escuela Superior Politécnica del Litoral (ESPOL), por ofrecerme un entorno académico enriquecedor y por ser un espacio que fomenta la curiosidad y el aprendizaje. Agradezco las oportunidades que me han brindado para crecer profesionalmente y por el compromiso de todos los docentes que comparten su conocimiento.

Finalmente, a mi tutor de tesis, agradezco profundamente su orientación, paciencia y dedicación. Su apoyo y consejos han sido esenciales para la culminación de este trabajo, y su pasión por la investigación ha sido una fuente de inspiración constante.

DECLARACION EXPRESA

Los derechos de titularidad y explotación me corresponden conforme al reglamento de propiedad intelectual de la institución; Andrés Marcelo Carranco Herrera, doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual.

Andrés Marcelo Carranco Herrera

COMITE EVALUADOR

Firmado electrónicamente por:
RICARDO ALFREDO
CAJO DIAZ

Ph.D. Ricardo Cajo

PROFESOR TUTOR

Firmado electrónicamente por:
WILBERT GEOVANNY
AGUILAR CASTILLO

Ph.D. Wilber Aguilar

CODIRECTOR

EFREN VINICIO HERRERA MUENTES

Digitally signed by EFREN VINICIO HERRERA MUENTES
Date: 2024.09.30 09:29:32 -05'00'

PhD. Efrén Herrera

PROFESOR EVALUADOR

RESUMEN

Este informe presenta el diseño de un controlador para el seguimiento de trayectorias de un vehículo autónomo no tripulado de ala rotatoria. El objetivo es optimizar la precisión y estabilidad en el seguimiento de trayectorias, superando las limitaciones del controlador interno preexistente en el simulador AirSim. Se desarrollan varios scripts en Visual Studio Code, incluyendo uno para la generación de trayectorias aleatorias y otro para el seguimiento de estas trayectorias mediante el uso del controlador interno de AirSim. Además, se diseña y entrena una red neuronal con el propósito de complementar y mejorar el desempeño de dicho controlador. Los resultados obtenidos demostraron que la integración de la red neuronal optimizó la precisión y estabilidad del seguimiento de trayectorias del dron, superando las limitaciones del controlador interno preexistente. La implementación de una red neuronal como complemento del sistema de control mejora significativamente el rendimiento del vehículo autónomo en términos de precisión y estabilidad en el seguimiento de trayectorias.

Palabras clave: Controlador de trayectoria, vehículo autónomo, dron de ala rotatoria, AirSim, red neuronal, seguimiento de trayectorias, inteligencia artificial, simulación de vuelo, optimización de controladores.

ABSTRACT

This report presents the design of a controller for trajectory tracking of an unmanned autonomous rotary-wing vehicle. The goal is to optimize the accuracy and stability in trajectory tracking, overcoming the limitations of the pre-existing internal controller in the AirSim simulator. Several scripts were developed in Visual Studio Code, including one for generating random trajectories and another for tracking these trajectories using the internal controller of AirSim. Additionally, a neural network was designed and trained to complement and improve the performance of this controller. The results demonstrated that the integration of the neural network optimized the accuracy and stability of the drone's trajectory tracking, surpassing the limitations of the pre-existing internal controller. Implementing a neural network as a complement to the control system significantly improves the performance of the autonomous vehicle in terms of accuracy and stability in trajectory tracking.

Keywords: Trajectory controller, autonomous vehicle, rotary-wing drone, AirSim, neural network, trajectory tracking, artificial intelligence, flight simulation, controller optimization.

ÍNDICE GENERAL

ABREVIATURAS

ESPOL Escuela Superior Politécnica del Litoral

AirSim: Aerial Informatics and Robotics Simulation

AI: Artificial Intelligence

PID: Proporcional-Integral-Derivativo

ISU: Integral Square of the Control Error

UAV: Unmanned Aerial Vehicle (Vehículo Aéreo No Tripulado)

GPS: Global Positioning System

ROS: Robot Operating System

LIDAR: Light Detection and Ranging

AI: Artificial Intelligence

ML: Machine Learning

NN: Neural Network

DNN: Deep Neural Network

RL: Reinforcement Learning

ReLU: Rectified Linear Unit

RMSProp: Root Mean Square Propagation

Adagrad: Adaptative Gradient

MAE: Mean Absolute Error

MSE: Mean Squared Error

ISE: Integral of Squared Error

IAE: Integral of Absolute Error

ISU: Integral of Squared Control

SGD: Stochastic Gradient Descent

CNN: Convolutional Neural Network

RNN: Recurrent Neural Network

LSTM: Long Short-Term Memory

SIMBOLOGÍA

Kp: Ganancia proporcional del controlador PID

Ki: Ganancia integral del controlador PID

Kd: Ganancia derivativa del controlador PID

e(t): Error en función del tiempo

u(t): Señal de control en función del tiempo

x,y,z: Coordenadas espaciales del dron

θ: Ángulo de cabeceo (pitch)

ϕ: Ángulo de alabeo (roll)

ψ: Ángulo de guiñada (yaw)

T: Tiempo de muestreo

V: Velocidad del dron

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO 1

1. Introducción

El presente trabajo de investigación surge tras la necesidad de buscar alternativas de solución a las diferentes amenazas contemporáneas que se han acrecentado en los últimos años como es el narcotráfico, contrabando de Gas Licuado de Petróleo (GLP), tráfico de armas, por mencionar las más importantes. Para solucionar esto se ha visto necesario utilizar las herramientas tecnológicas para realizar la vigilancia y reconocimiento de estas actividades ilícitas y utilizar las imágenes que se pueden capturar desde un vehículo aéreo no tripulado y usar como información de inteligencia para las acciones tácticas correspondientes de las organizaciones de control.

El desarrollo de vehículos autónomos no tripulados de ala rotatoria ha avanzado significativamente en los últimos años, especialmente en el ámbito de la navegación y el control de trayectorias. Actualmente, se utilizan diversas técnicas de control, como los sistemas de control predictivo y el aprendizaje automático, para mejorar la precisión y estabilidad del vuelo autónomo. Estas tecnologías permiten a los vehículos realizar tareas complejas de manera autónoma en una variedad de entornos y situaciones [1].

Investigaciones previas han demostrado la eficacia de los controladores internos de simuladores como AirSim para realizar tareas de seguimiento de trayectorias en entornos simulados. Por ejemplo, se ha utilizado el control predictivo para manejar la dinámica de drones, logrando resultados prometedores en términos de precisión y respuesta del sistema [2]. Además, se han implementado redes neuronales y técnicas de aprendizaje profundo para mejorar la capacidad de navegación autónoma, como en el caso del uso de aprendizaje por refuerzo profundo para evitar de obstáculos y el seguimiento de trayectorias [3]. Estos avances permiten una mayor adaptabilidad a las condiciones del entorno, aunque aún existen desafíos en situaciones de alta complejidad y variabilidad [4].

En los últimos años, ha habido un creciente interés en integrar redes neuronales con sistemas de control tradicionales para mejorar el rendimiento de los vehículos autónomos. Esta integración ha demostrado ser efectiva en mejorar la adaptabilidad y la precisión en el control de trayectorias, como se observa en la aplicación de redes

neuronales recurrentes y sistemas híbridos de control [5]. Sin embargo, la implementación de estas tecnologías aún está en fase de desarrollo y requiere una validación extensiva para su aplicación en escenarios del mundo real [6]. La falta de estudios que combinen de manera efectiva el aprendizaje automático con controladores tradicionales indica un vacío en el conocimiento actual, destacando la necesidad de más investigaciones en este campo [7].

Para abordar esta brecha, se ha desarrollado y entrenado una red neuronal para complementar el controlador interno de AirSim, con el objetivo de optimizar la precisión y estabilidad en el seguimiento de trayectorias. Este estudio ha formulado la hipótesis de que la integración de una red neuronal puede mejorar significativamente el rendimiento del sistema de control de un dron autónomo, superando las limitaciones de los controladores preexistentes.

El objetivo principal de este trabajo ha sido diseñar un controlador que, al combinar técnicas de aprendizaje automático con controladores tradicionales, logre una mayor precisión y estabilidad en el seguimiento de trayectorias de un vehículo autónomo no tripulado de ala rotatoria.

1.1 Descripción del problema

El Ecuador en la última década ha sufrido las consecuencias de organizaciones criminales transnacionales dedicadas al narcotráfico, narcoterrorismo, tráfico de armas, trata de personas contrabando de combustible, precursores químicos para el narcotráfico y gas licuado de petróleo, entre otras. Uno de los principales focos del contrabando está ubicado en la zona costera norte de Esmeraldas, alejada de la capital provincial, pero con una frontera inhóspita con el país vecino, Colombia. En esa parte del país, la lucha contra el contrabando está a cargo de Fuerzas Armadas, y uno de los puntos más críticos es en el norte de Esmeraldas, donde se desarrollan actividades como el tráfico de armas, contrabando de combustible, y gas licuado de petróleo principalmente en localidades como Río Verde, Eloy Alfaro y principalmente San Lorenzo [34].

Como se puede observar existen diferentes zonas de contrabando y el trabajo de entidades de seguridad como la Fuerza Aérea es arduo y extenso para controlar estas actividades que día a día son más rentables y siguen creciendo en la región por lo que se vuelve más difícil llevar un control de estas actividades ilegales. El incremento de estas actividades ilícitas genera la necesidad de plantear soluciones

tecnológicas para contrarrestar todas estas amenazas. La detección de este tipo de operaciones es importante para obtener la información de inteligencia.
En base a esto, se propone el desarrollo de herramientas para actividades de vigilancia y reconocimiento, las cuales permiten levantar información de inteligencia, a partir de datos tales como imágenes y videos, para contrarrestar actividades ilícitas y mejorar la seguridad ciudadana. Cabe recalcar que el uso de esta tecnología contribuye también al personal militar que realiza actividades contra estas organizaciones ilegales, brindando mayor seguridad del personal utilizando la tecnología para no exponer a los miembros de la institución frente a organizaciones altamente peligrosas.
Los vehículos aéreos no tripulados (VANT, por sus en español), más conocidos como UAV (por sus siglas en inglés, Unmanned Aerial Vehicles), son aeronaves que realizan operaciones de vuelo sin tripulación humana a bordo, disminuyendo incluso la fatiga en el personal militar debido a que al utilizar UAV´s, se puede realizar la vigilancia de mayores extensiones de terreno y gracias a la flexibilidad de operación en escenarios de difícil acceso y la facilidad de transporte y operación de los mismos permite utilizarlos en terrenos hostiles.

Así también, cabe recalcar las diversas aplicaciones que se tiene; si bien los UAV nacieron casi exclusivamente para usos militares, con el paso del tiempo, y con su desarrollo tecnológico, han trascendido a otros campos y han conquistado espacios muy importantes como cinematografía, mapeo y vigilancia, búsqueda y rescate, agricultura de precisión, inspección de infraestructura y construcciones, manejo de desastres y salvamento público, transporte [32]. Justamente, el uso de esta tecnología presenta algunas ventajas como flexibilidad, ahorro de tiempo para levantar información, reducción de costos para levantamiento topográfico, disminución de riesgo a exposición de zonas peligrosas

En este trabajo de investigación se centrará en aeronaves no tripuladas de ala rotatoria. Se ha escogido un cuadricóptero por la alta facilidad de monitoreo y difusión en el mundo científico y comercial de este tipo de ingenios.

En la actualidad existen controladores que permiten realizar vuelos controlados manualmente por un operador o vuelos realizados autónomamente por un ingreso previo de una trayectoria a través de un software, el mismo que no da el acceso

para realizar modificaciones, siendo exclusiva la ingeniería de estos equipos de las empresas fabricantes; para ello se vuelve necesario diseñar, simular y comprobar controladores de vuelo aplicables para la operación de un dron que sea desarrollado a través de software de lenguaje abierto a fin de continuar desarrollando tecnología y conocimiento propio para seguir disminuyendo la dependencia tecnológica [33][31].

El presente proyecto pretende diseñar un controlador para el seguimiento de trayectoria en vuelo de un cuadricóptero para realizar la traslación de mencionado equipo por diversos escenarios optimizando distancia, ahorro de energía, tiempo, entre otras variables. Se realizará un estudio inicial de posibles métodos de control de estos vehículos autónomos y se implementará a través de un software de simulación el método más conveniente o la combinación de las técnicas más eficientes para empleo del cuadricóptero.

1.2 Justificación del problema

En la actualidad el uso y la aplicación de los UAV's tiene un crecimiento casi exponencial. En el ámbito de la seguridad y defensa el empleo de estos dispositivos autónomos acortará el tiempo que se de reconocimientos de zonas críticas de seguridad en poblaciones urbanas y áreas de fronteras que son propensas a actividades ilícitas como el contrabando de combustible, minerales, narcotráfico, trata de personas, entre otras.

De igual manera, el uso de los UAV's permite reducir costos de logística que implicaría la movilización de una patrulla terrestre al reconocimiento de un grupo de 15 o 20 personas en una zona crítica de seguridad.

El diseño de un controlador para la operación de un cuadricóptero permitirá desarrollar misiones de vigilancia y reconocimiento con una mayor precisión y seguridad en el desarrollo de estas operaciones, para lo cual se realizará simulación de trayectoria en escenarios similares a los reales.

El desarrollo de este proyecto incluye una gran cantidad de análisis de diversos métodos existentes utilizando algoritmos clásicos y/o algoritmos heurísticos que

permitan definir el controlador más idóneo para seguimiento de trayectoria para el vuelo de un dron.

1.3 Objetivos

1.3.1 Objetivos General

Diseñar un controlador para el seguimiento de trayectorias de vuelo preestablecidas para un cuadricóptero a través de la utilización de un software de lenguaje abierto.

1.3.2 Objetivos Específicos

- Investigar y determinar el estado actual de la operación de los UAV's y los controladores utilizados en el uso y empleo de vehículos autónomos.
- Caracterizar el modelo integral de un cuadricóptero considerando las principales variables que inciden durante el vuelo.
- Simular un controlador basado en redes neuronales que permita mantener condiciones de vuelo estable.
- Analizar el desempeño del controlador propuesto durante el seguimiento de trayectorias del UAV.

1.4 Marco Teórico

En este apartado se presentará el estado del arte actual relacionado al tema del trabajo de investigación y desarrollo, así como el software empleado para su desarrollo.

1.4.1 Cinemática del dron y análisis del entorno

El análisis del entorno y la cinemática del dron son componentes críticos en el diseño, desarrollo y operación de vehículos aéreos no tripulados (UAV). Estos aspectos permiten a los UAVs navegar de manera segura y eficiente en diversos entornos, adaptarse a cambios dinámicos y realizar misiones complejas con precisión. La cinemática se refiere al estudio del movimiento del dron sin considerar las fuerzas que lo causan, mientras que el análisis del entorno implica la percepción y la interpretación del espacio circundante para la toma de decisiones en tiempo real.

- **Cinemática del Dron**

El modelado cinemático de un dron incluye la representación matemática de sus movimientos en el espacio tridimensional. Esto se basa en las coordenadas de posición (x, y, z) y la orientación (roll, pitch, yaw). Los modelos cinemáticos pueden ser de dos tipos: directo e inverso.

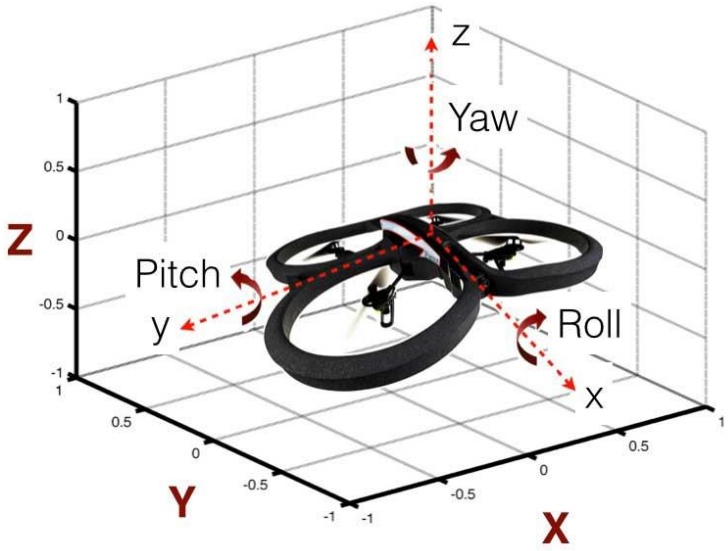


Figura 1. Ejes de movimiento del cuadrotor en el espacio.

[26].

La cinemática directa trata de determinar la posición y orientación del dron a partir de las velocidades de sus actuadores. Esto implica resolver las ecuaciones que relacionan las velocidades de los motores con el movimiento del dron [8].

La cinemática inversa se enfoca en encontrar las velocidades necesarias de los actuadores para alcanzar una posición y orientación deseadas. Este enfoque es esencial para tareas de control de trayectoria y estabilización [9].

La dinámica de vuelo considera las fuerzas y momentos que actúan sobre el dron, incluyendo la gravedad, la sustentación, la resistencia aerodinámica y los efectos de los actuadores. Estos factores influyen en el comportamiento del dron y su capacidad para mantener la estabilidad y seguir trayectorias específicas.

Para modelar la dinámica de un cuadricóptero, se considera tanto la dinámica traslacional como la rotacional. La dinámica traslacional, que describe el movimiento del centro de masa del dron, se puede expresar mediante la ecuación:

$$m\frac{dv}{dt} = -mge_3 + RT \quad (1)$$

Donde m es la masa del dron, v es la velocidad lineal, g es la aceleración gravitacional, e_3 es el vector unitario en la dirección z, R es la matriz de rotación, y T es la suma de las fuerzas de empuje generadas por las hélices [27], [28].

La dinámica rotacional, que describe los cambios en la orientación del dron, sigue la ecuación de Euler para cuerpos rígidos:

$$J\frac{d\omega}{dt} + \omega \times (J\omega) = \tau \quad (2)$$

donde J es el tensor de inercia del dron, ω es la velocidad angular, y τ es el vector de torques. Este modelo permite la descripción precisa del comportamiento dinámico de los cuadricópteros, lo cual es fundamental para su control efectivo en aplicaciones con perturbaciones ambientales [27], [28].

Los controladores de vuelo utilizan los modelos cinemáticos y dinámicos para ajustar continuamente las entradas a los actuadores, asegurando que el dron siga la trayectoria deseada y mantenga la estabilidad. Los controladores comunes incluyen PID (Proporcional-Integral-Derivativo), LQR (Regulador Cuadrático Lineal) y controladores basados en modelos predictivos [10].

- **Análisis del entorno**

Los drones están equipados con una variedad de sensores para percibir y analizar su entorno. Estos sensores incluyen cámaras ópticas, cámaras infrarrojas, LIDAR, sonar y sistemas de navegación global (GPS). Las cámaras permiten la captura de imágenes y videos en tiempo real, utilizados para la navegación visual y el reconocimiento de objetos y características del terreno. Los sistemas GPS proporcionan datos de posición global, fundamentales para la navegación en espacios abiertos y la ejecución de misiones predefinidas.

La fusión de sensores combina datos de múltiples fuentes para obtener una visión más completa y precisa del entorno. Esto mejora la fiabilidad de la percepción y permite una mejor toma de decisiones. Algoritmos como el filtro de Kalman y las redes neuronales se utilizan para integrar y procesar estos datos [11].

El mapeo y la localización son procesos cruciales en el análisis del entorno. Los drones deben ser capaces de crear y actualizar mapas del entorno en tiempo real mientras se localizan a sí mismos dentro de estos mapas. Técnicas como SLAM (Simultaneous Localization and Mapping) son esenciales para esta tarea, permitiendo a los drones operar de manera autónoma en entornos desconocidos [12].

La combinación del análisis del entorno y la cinemática permite a los drones navegar de manera autónoma, evitando obstáculos y adaptándose a cambios en el entorno. Esto es crucial para aplicaciones en logística, entrega de paquetes y operaciones de búsqueda y rescate.

Los drones se utilizan para la inspección de infraestructuras como puentes, líneas eléctricas y plataformas petrolíferas. El análisis detallado del entorno y la capacidad de maniobrar con precisión permiten a los drones realizar estas tareas de manera segura y eficiente.

En la agricultura, los drones recopilan datos sobre el estado de los cultivos y el suelo, ayudando a los agricultores a tomar decisiones informadas sobre riego, fertilización y control de plagas. Los sensores y la cinemática avanzada permiten la cobertura de grandes áreas con alta precisión.

1.4.2 AirSim, herramienta de simulación de vehículos aéreos no tripulados

El desarrollo del controlador en AirSim involucra una serie de herramientas y lenguajes que permiten el control preciso de los UAV (vehículos aéreos no tripulados). Los scripts creados en Python se utilizan para definir trayectorias y gestionar el movimiento del dron en el entorno de simulación proporcionado por Unreal Engine. Esto permite ejecutar pruebas realistas bajo diferentes condiciones ambientales, ajustando los parámetros del vuelo autónomo en tiempo real.

Para optimizar el rendimiento del controlador interno de AirSim, se ha integrado una red neuronal entrenada en Google Colab. Este software adicional ofrece una plataforma eficiente para la ejecución de tareas de aprendizaje automático, permitiendo el procesamiento y entrenamiento de grandes volúmenes de datos sin depender del hardware local. La red neuronal mejora el seguimiento de trayectorias al ajustar las decisiones del controlador en función de la retroalimentación del entorno simulado.

El uso de Visual Studio Code para programar en Python facilita la automatización de procesos y la interacción con la API de AirSim. Esto permite la creación de trayectorias dinámicas y ajustes continuos durante el vuelo, lo que mejora significativamente la precisión y la estabilidad del dron. La capacidad de simular situaciones complejas en Unreal Engine proporciona una plataforma robusta para el desarrollo y prueba de algoritmos avanzados de control y redes neuronales.
AirSim proporciona una serie de funciones y comandos que son esenciales para controlar y simular vehículos aéreos no tripulados (UAVs). A continuación, se describen algunas de las funciones y comandos más relevantes.

Como se muestra en la Figura 2, La simulación se realizó en el entorno África el cual se encuentra de manera gratuita en los repositorios de GitHub en el apartado de AirSim.

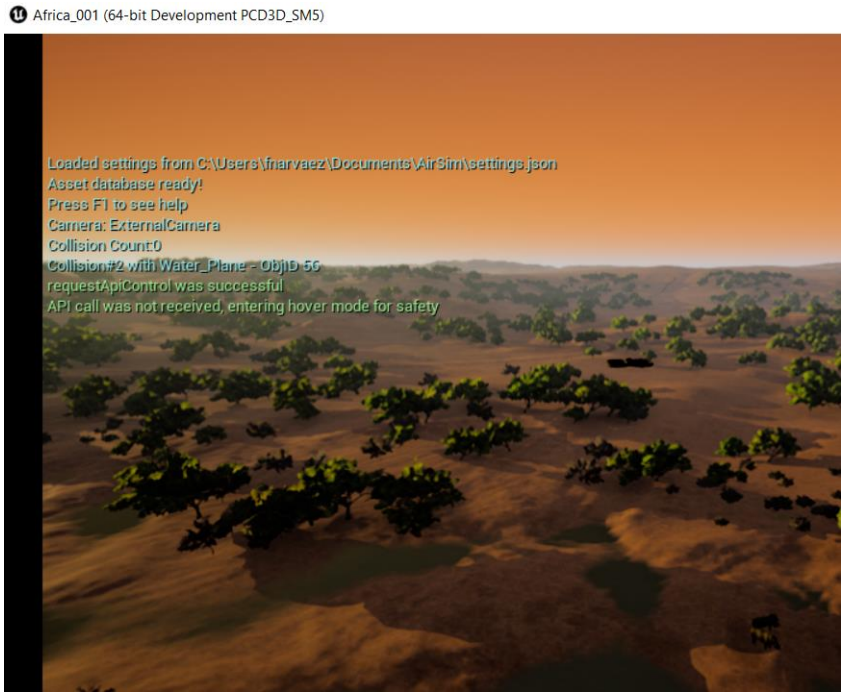


Figura 2. Entorno de AirSim África.

- **Conexión y Control Básico**

Durante el desarrollo del software para mejorar el seguimiento de trayectoria se tienen los siguientes comandos principales:

Tabla 1. Comandos para conexión con AirSim

Comando	**Descripción**
confirmConnection	Verifica la conexión con el simulador AirSim.
enableApiControl	Verifica la conexión con el simulador AirSim
armDisarm	Arma o desarma el dron, preparándolo para el despegue o apagándolo

Despegue y Aterrizaje

Tabla 2. Comandos para despegue y aterrizaje

Comando	**Descripción**
takeoffAsync	Inicia el despegue del dron de manera asíncrona
landAsync	Inicia el aterrizaje del dron de manera asíncrona

Movimiento y Navegación

Tabla 3. Comandos de movimiento y navegación

Comando	**Descripción**
moveToPositionAsync	Mueve el dron a una posición específica en el espacio tridimensional con una velocidad establecida
moveByVelocityAsync	Mueve el dron a una velocidad específica en cada uno de los ejes (x, y, z).

Control de Orientación:

Tabla 4. Comandos para control de orientación

Comando	**Descripción**
rotateToYawAsync	Rota el dron a un ángulo de yaw específico
rotateByYawRateAsync	Cambia el ángulo de yaw del dron a una tasa específica durante un periodo de tiempo

Captura de Datos

Tabla 5. Comandos para captura de datos

Comando	**Descripción**
getMultirotorState	Obtiene el estado actual del dron, incluyendo su posición, velocidad, orientación, etc.
simGetImages	Captura imágenes desde las cámaras del dron en diferentes modos (RGB, Depth, Segmentation)

- **Control automático programable**

El control automático programable es una tecnología esencial en el desarrollo y operación de vehículos aéreos no tripulados (UAVs). Este tipo de control permite a los UAVs ejecutar misiones complejas de manera autónoma, siguiendo instrucciones preprogramadas y adaptándose a cambios en el entorno en tiempo real. Los sistemas de control automático programable utilizan algoritmos avanzados para gestionar la navegación, estabilidad y rendimiento del UAV, garantizando una operación segura y eficiente.

La implementación de un controlador PID en su forma discreta se puede describir mediante la siguiente ecuación simplificada:

$$u[k] = K_p e[k] + K_i \sum_{i=0}^{k} e[i] + K_d(e[k] - e[k-1]) \quad (3)$$

donde u[k] es la salida del controlador en el instante de muestreo k, e[k] es el error en dicho instante; y Kp, Ki, y Kd son las ganancias proporcional, integral y derivativa, respectivamente. Esta ecuación simplificada es adecuada para aplicaciones de control en sistemas discretos [29].

- **Planificación de Trayectorias**

Los algoritmos de planificación de trayectorias permiten al UAV calcular rutas óptimas para llegar a un destino evitando obstáculos y optimizando el consumo de energía.

Los UAVs están equipados con diversos sensores como GPS, IMU (Unidad de Medición Inercial), cámaras y LIDAR (Alcance y detección de la Luz) para obtener datos en tiempo real sobre su posición, velocidad y entorno [13].

Los actuadores, como motores y servomecanismos, reciben comandos del sistema de control para ajustar la orientación, velocidad y trayectoria del UAV. Los controladores ajustan la velocidad de los motores y la orientación del UAV en respuesta a las variaciones en el entorno y los objetivos de la misión.

El futuro del control automático programable en UAVs incluye el desarrollo de sistemas completamente autónomos que puedan aprender y adaptarse a nuevas situaciones sin intervención humana. La inteligencia artificial y el aprendizaje automático jugarán un papel crucial en el avance de esta tecnología [14].

Las redes neuronales son modelos de aprendizaje automático inspirados en el cerebro humano, diseñadas para reconocer patrones y tomar decisiones basadas en datos. Estas redes consisten en capas de neuronas artificiales, donde cada neurona realiza cálculos matemáticos sobre sus entradas y produce una salida. Las redes neuronales han revolucionado varios campos como la visión por computadora, procesamiento del lenguaje natural y sistemas de recomendación.

- **Tipos de Redes Neuronales**

El Perceptrón Multicapa (MLP) es el tipo más básico de red neuronal, que consiste en una capa de entrada, una o más capas ocultas y una capa de salida. Cada neurona en una capa está conectada a todas las neuronas en la siguiente capa. Este tipo de red se utiliza ampliamente en el reconocimiento de patrones, clasificación y regresión [15].

Las Redes Neuronales Convolucionales (CNN) están diseñadas específicamente para procesar datos con una estructura de cuadrícula, como imágenes. Utilizan convoluciones en lugar de conexiones completas entre las capas, lo que reduce la cantidad de parámetros y mejora la eficiencia. Las CNN son fundamentales en aplicaciones de visión por computadora, detección de objetos y análisis de imágenes [15].

Las Redes Neuronales Recurrentes (RNN) son adecuadas para datos secuenciales, ya que las conexiones entre neuronas forman un ciclo dirigido, permitiendo que la información persista. Las RNN se emplean en procesamiento del lenguaje natural, traducción automática y análisis de series temporales. Las Redes de Memoria a Largo Corto Plazo (LSTM) son una variante de las RNN diseñadas para superar el problema del desvanecimiento del gradiente, permitiendo a las redes aprender dependencias a largo plazo. Estas se utilizan en análisis de secuencias largas, generación de texto y reconocimiento de voz [17].

Las Redes Generativas Adversariales (GAN) consisten en dos redes neuronales que compiten entre sí: un generador que crea datos falsos y un discriminador que intenta distinguir entre datos reales y falsos. Las GAN se aplican en la generación de imágenes, mejora de resolución y síntesis de datos [18].

- **Métodos de Activación**

El método de activación sigmoide transforma las salidas de una neurona en el rango (0, 1), siendo útil para problemas de clasificación binaria, aunque presenta problemas con gradientes que se desvanecen en redes profundas. La función tangente hiperbólica (tanh) es similar a la sigmoide, pero transforma las salidas en el rango (-1, 1), centrando los datos en cero y ofreciendo un mejor rendimiento en muchas aplicaciones.

El Rectificador Lineal Unitario (ReLU) activa las neuronas con una salida mayor que cero y es la función más comúnmente utilizada en redes profundas. ReLU evita el problema del desvanecimiento del gradiente y acelera la convergencia, aunque puede provocar la "muerte" de neuronas si las entradas son negativas constantemente [1]. La ReLU Leaky es una variante que permite un pequeño gradiente cuando la unidad está inactiva, solucionando el problema de la "muerte" de neuronas. La función softmax se utiliza en la capa de salida de una red para clasificación multiclase, transformando las salidas en probabilidades que suman 1 [16].

- **Optimizadores**

El Gradiente Descendente Estocástico (SGD) actualiza los parámetros en la dirección negativa del gradiente de la función de pérdida, utilizando un subconjunto aleatorio de

datos. Es simple y eficiente en grandes conjuntos de datos, pero su convergencia es lenta y es susceptible a mínimos locales. RMSprop modifica el SGD adaptando la tasa de aprendizaje para cada parámetro, utilizando una media móvil de los gradientes cuadrados, lo que mejora la convergencia y manejo de la variabilidad de los gradientes.

Adam (Adaptive Moment Estimation) combina las ideas de RMSProp (Root Mean Square Propagation) y SGD con momento, adaptando la tasa de aprendizaje y almacenando estimaciones de momentos de primer y segundo orden. Adam se destaca por su rápida convergencia y robustez en diferentes tipos de problemas, siendo ampliamente utilizado en aplicaciones de redes neuronales [16]. Adagrad (Adaptative Gradient) ajusta la tasa de aprendizaje para cada parámetro con base en la suma de los gradientes cuadrados anteriores, adecuado para características raras y datos dispersos, aunque puede resultar en tasas de aprendizaje extremadamente pequeñas.

CAPÍTULO 2

2. METODOLOGIA

Para el diseño de un controlador para el seguimiento de trayectorias de vuelo preestablecidas para un dron fue necesario identificar primero los controladores convencionales que se utilizan para seguimiento de trayectoria y posterior la utilización de un software de lenguaje abierto para simular un controlador basado en redes neuronales a fin de mejorar el desempeño del mismo durante el seguimiento de trayectorias del UAV.

2.1 Creación de trayectorias

El código para generar trayectorias aleatorias de un dron está diseñado para crear una secuencia de puntos (waypoints) que el dron seguirá a lo largo de su recorrido, asegurando que cada nuevo punto esté a una distancia mínima del anterior. Estos waypoints se distribuyen en un plano 2D (X e Y), mientras que la altura (Z) permanece constante.

El proceso comienza con la definición de algunos parámetros clave. Se establece el número total de waypoints a generar (num_waypoints), la distancia mínima que debe existir entre cada waypoint generado (min_distance), y la posición inicial del dron (initial_position), que en este caso es (0, 0, -40), donde el valor Z de -40 representa la altitud fija a la que el dron se moverá.

La generación de los waypoints aleatorios se realiza dentro de la función generate_random_waypoint. Esta función utiliza un enfoque en el que, para cada nuevo waypoint, primero se elige un ángulo aleatorio entre 0 y 360 grados (en radianes), que determina la dirección en la que el dron debe moverse desde su última posición conocida. Posteriormente, se genera una distancia aleatoria que varía entre el valor mínimo especificado y el doble de este valor. Con estas dos variables, se calculan las coordenadas X e Y del nuevo waypoint sumando el desplazamiento a las coordenadas del último punto en la lista de waypoints. La altura Z se mantiene constante en -40.

Antes de aceptar un nuevo waypoint, el código verifica que esté lo suficientemente alejado de todos los waypoints anteriores. Si cumple con la distancia mínima especificada, se añade a la lista de waypoints; si no, se vuelve a generar un nuevo punto hasta que se cumpla esta condición. Este proceso asegura que los puntos no estén demasiado cerca entre sí, lo que podría crear una trayectoria poco práctica para el dron.

Una vez generados los waypoints, se almacenan en una lista llamada coordenadas. Esta lista guarda los puntos en un formato específico (airsim.Vector3r), lo cual es útil para integrarse con simuladores de vuelo como AirSim. Además, el código ofrece la opción de guardar estos puntos en un archivo para su uso posterior, permitiendo reproducir la misma trayectoria en futuras ejecuciones.

El comportamiento del código, ya sea para generar trayectorias fijas o aleatorias, se controla mediante la variable trayectoria_fija. Si esta variable está activada (True), el código intenta cargar una trayectoria previamente guardada desde un archivo. Si el archivo no existe, se genera una nueva trayectoria aleatoria y se desactiva la opción de trayectoria fija. En el caso de que trayectoria_fija sea False, el código generará automáticamente una nueva serie de waypoints y los guardará en un archivo para poder reutilizarlos.

Finalmente, el código visualiza la trayectoria generada en un gráfico 2D. Los waypoints se grafican en color rojo, y la trayectoria real del dron (si está disponible) se muestra en color azul. Cada waypoint está etiquetado con su número de secuencia, lo que facilita su identificación en el gráfico. Este enfoque permite ver de manera clara la trayectoria que seguirá el dron en un plano XY, mientras que la altura (Z) permanece fija, haciendo que el dron vuele a una altitud constante.

2.2 Controladores clásicos

Existen diversos controladores para sistemas autónomos, entre ellos el más difundido y utilizado en la industria convencional y en control de vehículos aéreos no tripulados es el controlador PID (Proporcional – Integral - Derivativo).
Tomando en cuenta el software AirSim que es donde se desarrolló y ejecutó las pruebas de los controladores, el comando client.moveToPositionAsync utilizó el controlador

interno de AirSim para gestionar el movimiento preciso del dron hacia una posición objetivo especificada. Este controlador interno integró múltiples algoritmos de control PID, que ajustan continuamente la velocidad y orientación del dron para asegurar que se mueva suavemente y con precisión hacia las coordenadas deseadas. Durante el proceso, el controlador evaluó constantemente la posición instantánea del dron y realizó ajustes necesarios en tiempo real, utilizando la retroalimentación de los sensores del dron para mantener la estabilidad y precisión del vuelo. Además, el controlador pudo manejar parámetros adicionales como el tipo de tren de transmisión y el modo de orientación, permitiendo movimientos complejos y personalizados según las necesidades del usuario, lo que resultó en una simulación realista y detallada.

En AirSim, los parámetros del PID utilizados por el controlador interno generalmente no se seleccionan de manera manual, sino que están preconfigurados para ajustarse a los modelos de drones más comunes que se simulan en la plataforma. Estos parámetros pueden ser ajustados a través de la configuración del archivo JSON, donde se especifican valores predeterminados para Kp, Ki, y Kd. Los usuarios también pueden modificar estos valores para adaptarlos a escenarios específicos de simulación, basándose en las características del dron y los requisitos de la misión.

- **Sintaxis del Comando**

La sintaxis del comando client.moveToPositionAsync incluye varios parámetros que definen el comportamiento del dron durante el movimiento. Estos parámetros son las coordenadas de destino (x, y, z), la velocidad de desplazamiento, el tiempo máximo permitido para completar la operación, el tipo de tren de transmisión y el modo de orientación (yaw). Estos parámetros permitieron un control preciso y detallado del dron en la simulación.

- **Parámetros del Comando**

Los parámetros x, y, z especifican las coordenadas de la posición objetivo en el sistema de referencia de AirSim, donde el origen (0, 0, 0) generalmente representa el punto de despegue del dron. El parámetro de velocidad determina la rapidez con la que el dron se mueve hacia la posición objetivo, medida en metros por segundo. El parámetro timeout_sec define el tiempo máximo en segundos que el dron debe intentar alcanzar la

posición objetivo antes de que la operación se cancele automáticamente, con un valor predeterminado de 60 segundos.

El parámetro drivetrain especifica el tipo de tren de transmisión a utilizar durante el movimiento. AirSim ofrece dos opciones: MaxDegreeOfFreedom, que permite al dron utilizar cualquier grado de libertad necesario para alcanzar la posición objetivo, y ForwardOnly, que restringe el dron a moverse solo hacia adelante, sin cambiar la orientación de su nariz. El parámetro yaw_mode controla la orientación del dron durante el movimiento y se define mediante la clase YawMode, que toma dos argumentos: is_rate, un booleano que indica si el valor de yaw es una tasa de cambio, y yaw_or_rate, el valor del ángulo de yaw o la tasa de cambio, dependiendo del valor de is_rate.

- **Funcionalidades Internas**

El comando client.moveToPositionAsync utiliza varias funcionalidades internas para asegurar un movimiento preciso y estable del dron. Una de estas funcionalidades es la interpolación de trayectoria, que suaviza el movimiento del dron entre la posición actual y la posición objetivo, garantizando un vuelo más realista. Además, el comando ajusta la velocidad del dron en función de la distancia a la posición objetivo y la velocidad especificada, utilizando algoritmos de control de velocidad para mantener la precisión.

Aunque el comando no incluye directamente la evitación de obstáculos, se puede combinar con sensores y algoritmos de percepción para detectar y evitar obstáculos en el camino del dron. Esta integración permite realizar pruebas y ajustes extensivos en un entorno simulado antes de implementar los algoritmos en drones reales.

2.3 Controladores no convencionales

Para mejorar el desempeño del controlador interno que usa el comando client.moveToPositionAsync se implementó una red neuronal la cual funciona de la siguiente manera.

Existe una función que calcula el ángulo de giro entre waypoints (puntos de control) y los pasa a la red neuronal para que devuelva una distancia al punto a la cual el dron cambie el parámetro de velocidad del comando client.moveToPositionAsync.

Estos datos se obtuvieron haciendo pasadas con el dron en trayectorias exclusivas con un ángulo como se observa en la Figura 3.

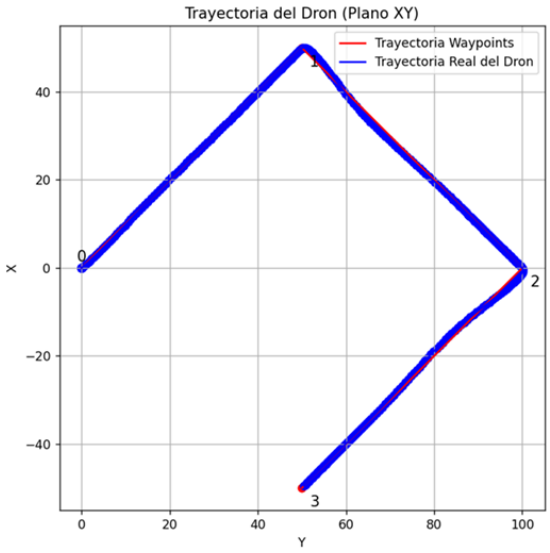


Figura 3. Pasada del dron a ángulos de 90 grados

Esto se repitió a diferentes ángulos a una velocidad inicial de 10 m/s. Durante estas pruebas, se evaluaron diferentes trayectorias y curvas con ángulos variados para registrar el comportamiento del dron en cada situación. Los datos obtenidos incluyeron las velocidades finales y las distancias recorridas en relación a los ángulos de giro en cada sección de la trayectoria.

El proceso de recolección de datos fue fundamental para alimentar la red neuronal utilizada posteriormente. Se analizaron una amplia gama de ángulos, desde ángulos pequeños y casi rectos, donde el dron mantenía velocidades más altas, hasta ángulos más pronunciados, en los que se observó una disminución en la velocidad para garantizar la maniobrabilidad adecuada. Los valores de distancia predicha también fueron recopilados para evaluar cómo el dron ajustaba su proximidad a cada punto de la trayectoria en función del ángulo de giro.

Este conjunto de datos, resultante de múltiples pasadas, proporcionó una base sólida para entrenar la red neuronal. Cada conjunto de datos asociado a un ángulo, distancia y velocidad permitió que el modelo aprendiera las relaciones no lineales entre estos factores. La idea era que el modelo fuera capaz de generalizar y predecir correctamente las velocidades y distancias para nuevos ángulos en situaciones no vistas anteriormente durante el entrenamiento.

Al final de la recolección, el comportamiento observado permitió obtener un modelo que ajusta de manera eficiente las decisiones del dron durante el vuelo, lo que garantiza que la red neuronal aprenda a mantener el equilibrio entre velocidad y precisión en distintas configuraciones de trayectorias.

Los datos más óptimos para entrenar la red se pueden observar en la Tabla 1.

Tabla 6. Datos usados para entrenar la red

Entrada	Salida	
Angulo (°)	Distancia (m)	Velocidad (m/s)
0	0	10
30	14	4
60	18	3
90	20	3
120	16	2
150	18	2
180	11	1.5

2.3.1 Estructura de la Red Neuronal

La grafico que genera la herramienta Google Colab para representar la red neuronal se encuentra en la Figura 4. Su estructura se encuentra detallada a continuación.

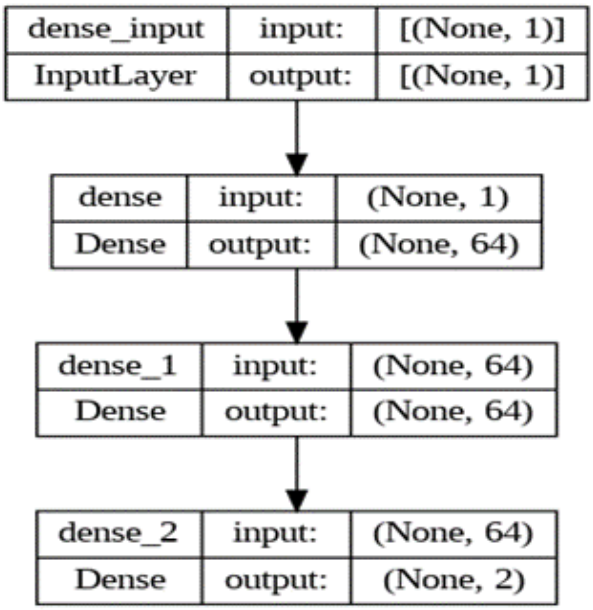


Figura 4. Estructura de la red neuronal.

2.3.1.1 Capas de la Red Neuronal

- **Capa de Entrada**

Input(shape=(1,)): La red tiene una capa de entrada con una única neurona. Esto se debe a que estamos utilizando un solo valor de entrada (ángulo en grados) para predecir la distancia y la velocidad.

- **Primera Capa Oculta**

Dense(64, activation='relu'): La primera capa oculta tiene 64 neuronas y utiliza la función de activación ReLU. La elección de 64 neuronas es arbitraria, pero es un buen punto de partida que permite a la red aprender patrones complejos en los datos. La función ReLU es popular porque ayuda a mitigar el problema del desvanecimiento del gradiente y permite que la red aprenda relaciones no lineales [20].

- **Segunda Capa Oculta**

Dense(64, activation='relu'): Similar a la primera capa oculta, esta capa también tiene 64 neuronas con activación ReLU. La idea de tener múltiples capas ocultas (también

conocido como profundidad de la red) es permitir que la red aprenda características más abstractas y complejas a medida que la información pasa por la red [21].

- **Capa de Salida**

Dense(2, activation='linear'): La capa de salida tiene 2 neuronas, correspondientes a las dos salidas que queremos predecir: distancia y velocidad. Utilizamos una activación lineal en la capa de salida porque estamos resolviendo un problema de regresión (predicción de valores continuos) y no queremos restringir las salidas a un rango específico (como sería el caso con una activación sigmoid o softmax) [22].

2.4 Justificación de la Estructura

2.4.1 Simplicidad Inicial

La estructura elegida fue relativamente simple, lo cual fue adecuado para comenzar con un modelo básico y evaluar su rendimiento. Si el modelo no proporciona los resultados esperados, se puede ajustar la complejidad agregando más capas o neuronas [23].

2.4.2 Función de Activación ReLU

ReLU (Rectified Linear Unit) es una elección estándar para las capas ocultas porque introduce no linealidades en el modelo, permitiendo que aprenda funciones complejas. Además, ReLU es computacionalmente eficiente y ayuda a mitigar problemas como el desvanecimiento del gradiente [24].

- **Número de Neuronas**

La elección de 64 neuronas es un punto de partida común que ofrece un equilibrio entre capacidad de aprendizaje y eficiencia computacional. Demasiadas neuronas pueden llevar al sobreajuste, mientras que muy pocas pueden resultar en un modelo subajustado [25].

- **Capa de Salida:**

La capa de salida tiene 2 neuronas, una para cada variable de salida (distancia y velocidad). La activación lineal se utilizó aquí porque queremos predecir valores continuos sin restricciones [22].

2.4.3 Compilación del modelo

Para compilar el modelo, se utiliza el método model.compile con los siguientes parámetros: loss='mean_squared_error', optimizer='adam' y metrics=['mae'].

- **Función de Pérdida: Mean Squared Error (MSE)**

La función de pérdida seleccionada es el Error Cuadrático Medio (Mean Squared Error, MSE). Esta es una de las funciones de pérdida más comunes para problemas de regresión. El MSE calcula el promedio de los cuadrados de los errores o diferencias entre los valores predichos por el modelo y los valores reales. Se usa ampliamente debido a sus propiedades matemáticas que penalizan errores grandes más que errores pequeños, lo cual puede conducir a un modelo más preciso.

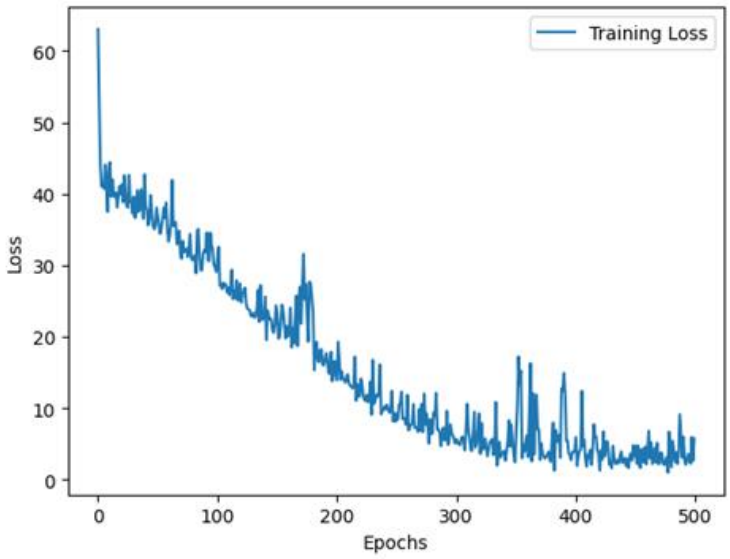


Figura 5.Grafica de la pérdida de entrenamiento de la red neuronal.

2.4.4 Optimizador: Adam

El optimizador Adam (Adaptive Moment Estimation) fue elegido por su eficacia y eficiencia en el entrenamiento de modelos de aprendizaje profundo. Adam combina las ventajas de dos métodos bien conocidos: AdaGrad y RMSProp. Es particularmente adecuado para problemas con grandes cantidades de datos y parámetros. Adam ajusta las tasas de aprendizaje individuales para cada parámetro, lo cual mejora la convergencia y hace que el proceso de entrenamiento sea más robusto y eficiente.

La elección de estos componentes está respaldada por su eficacia demostrada en la práctica y su uso extensivo en la literatura de aprendizaje profundo. Según Chollet [26], el uso del MSE como función de pérdida, Adam como optimizador y MAE como métrica proporciona un enfoque equilibrado y eficiente para entrenar y evaluar modelos de regresión en redes neuronales profundas.

CAPÍTULO 3

3. RESULTADOS Y ANÁLISIS

En la presente prueba de simulación con AirSim, se evaluó el comportamiento de un dron utilizando un modelo de aprendizaje automático para predecir la distancia y velocidad óptimas durante su trayectoria entre varios puntos. A través del análisis de los datos recolectados, se observó un comportamiento adecuado en la adaptación del dron a las predicciones proporcionadas por el modelo en función del ángulo entre los puntos de la trayectoria.

Los resultados muestran que el modelo de predicción es capaz de generar valores de distancia y velocidad ajustados para cada ángulo calculado. Por ejemplo, en un ángulo de 14.9 grados, el modelo predijo una distancia de 6.7 metros y una velocidad de 8.5 m/s. Estas predicciones son consistentes con el comportamiento deseado del dron, el cual ajustó su velocidad a medida que se acercaba al punto objetivo. De manera similar, en un ángulo de 119.4 grados, el modelo predijo una velocidad reducida de 2.8 m/s, lo que permitió que el dron realizara giros más controlados y suaves en curvas más pronunciadas.

Un comportamiento constante a lo largo de la simulación fue la transición entre la velocidad inicial (10 m/s) y la velocidad predicha por el modelo. Una vez que el dron se encontraba dentro de la distancia predicha al siguiente punto, la velocidad se ajustaba automáticamente, garantizando que el dron no volara a alta velocidad cuando estaba demasiado cerca de un punto de giro o de una curva cerrada. Este mecanismo permitió al dron manejar diferentes situaciones de vuelo con precisión, lo cual se evidenció en la disminución de la velocidad en ángulos pronunciados y la estabilidad de la velocidad en trayectorias más rectas.

La integración del modelo de predicción basado en el ángulo de giro resultó en una mayor eficiencia durante el vuelo del dron. Los cambios en velocidad fueron implementados de manera fluida y solo ocurrieron una vez, evitando ajustes repetitivos e innecesarios. Esto fue posible gracias al uso de una bandera que controlaba el cambio de velocidad, lo que mejoró la eficiencia del proceso.

En síntesis, los resultados obtenidos indican que el modelo de predicción es capaz de ajustar de manera adecuada la distancia y velocidad del dron en función de la geometría de la trayectoria. Las predicciones proporcionadas permitieron un control preciso del dron, especialmente en curvas y puntos críticos de la trayectoria. Aun así, sería beneficioso realizar pruebas adicionales en trayectorias más complejas y con ángulos más extremos para validar la robustez del modelo en diferentes escenarios.

Los resultados del controlador PID en el seguimiento se dieron con una desviación moderada.

Figura 6. Resultado obtenido del controlador PID

Complementando el controlador PID con la red neuronal el resultado es el siguiente:

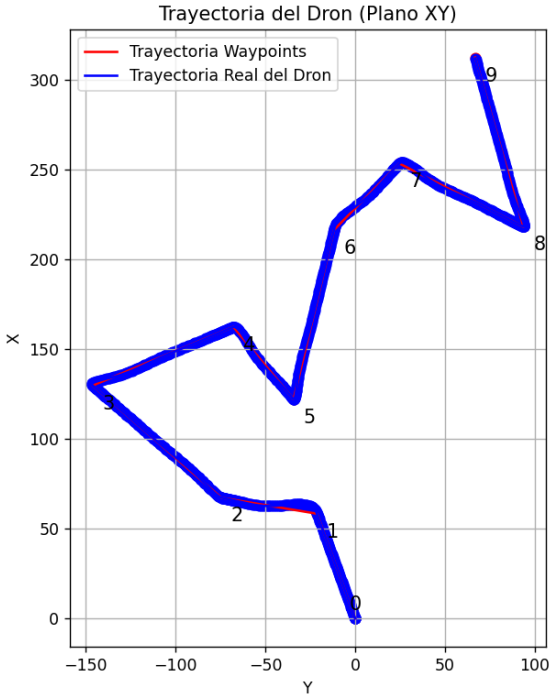


Figura 7. Resultado obtenido del controlador PID complementado con la red neuronal

Como se muestra en la Figura 7, el rendimiento del controlador mejorado es significativamente superior. Adicional a continuación se deja un link de enlace para observar un video de seguimiento de trayectoria del dron desde la plataforma AirSim: https://youtu.be/wIeZ1pXzzxI.

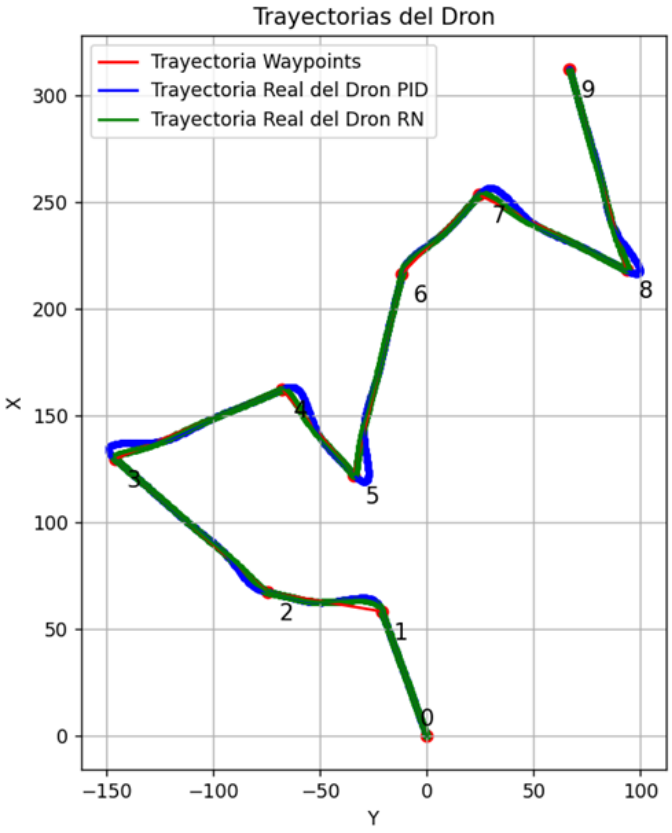


Figura 8. Resultados de ambos controladores

3.1 Índices de desempeño

Los índices de desempeño son métricas utilizadas para evaluar la calidad y eficiencia de un controlador en sistemas de control. Estos índices cuantifican la capacidad del controlador para seguir una trayectoria deseada y minimizar el error en la respuesta del sistema.

Tabla 7. Índices de desempeño

Índice de desempeño	ISE	IAE	ISU
Controlador PID	369078.83	1688.12	66038.95
Controlador con RN	367359.85	1684.76	56206.69

3.1.1 ISE (Integral of Squared Error)

El ISE mide el error cuadrático acumulado a lo largo del tiempo. Un valor más bajo indica un mejor desempeño en términos de minimizar el error al cuadrado. En este caso, el controlador de Red Neuronal (RN) tiene un ISE ligeramente menor que el PID, lo que sugiere que el RN tiene una capacidad marginalmente mejor para reducir el error cuadrático a lo largo del tiempo.

3.1.2 *IAE (Integral of Absolute Error)*

El IAE mide el error absoluto acumulado a lo largo del tiempo. Al igual que el ISE, un valor menor es preferible, ya que indica menos error absoluto en la trayectoria. El RN también tiene un IAE ligeramente menor, lo que sugiere que, en promedio, el error absoluto es algo menor en comparación con el PID.

3.1.3 *ISU (Integral of Squared Control)*

El ISU mide el esfuerzo de control en términos de la integral de los cuadrados de la señal de control. Un valor menor en ISU sugiere que el controlador está utilizando menos esfuerzo para alcanzar el objetivo, lo que puede ser una ventaja en términos de eficiencia y desgaste del sistema. El RN tiene un ISU menor, lo que indica que, en promedio, el RN utiliza menos esfuerzo de control comparado con el PID.

3.2 Análisis comparativo

Error en la trayectoria (ISE y IAE): El controlador de Red Neuronal (RN) tiene valores marginalmente mejores en cuanto a la reducción del error, tanto en términos de error cuadrático como absoluto. Esto sugiere que, en promedio, el RN es ligeramente mejor en seguir la trayectoria deseada con menos error.

Esfuerzo de control (ISU): El controlador de Red Neuronal (RN) también tiene un menor ISU, indicando que utiliza menos esfuerzo de control para lograr los objetivos en comparación con el PID. Esto puede traducirse en una operación más eficiente y menos consumo de recursos.

La Red Neuronal parece ofrecer una ligera mejora en términos de precisión y eficiencia en comparación con el controlador PID. Sin embargo, la diferencia en los índices no es enorme, por lo que la elección entre los dos controladores podría depender también de otros factores como la facilidad de implementación, la robustez en diversas condiciones, y la complejidad del sistema.

CONCLUSIONES

- El controlador diseñado para el seguimiento de trayectorias de un vehículo autónomo no tripulado de ala rotatoria ha demostrado ser eficaz en simulaciones extensivas realizadas en AirSim. Los scripts desarrollados para generar trayectorias aleatorias y controlar el dron han permitido una evaluación precisa del rendimiento del sistema.
- La integración de una red neuronal para complementar el controlador interno de AirSim ha resultado en una mejora significativa del desempeño. Esto se refleja en la reducción de errores de trayectoria, lo que sugiere que las técnicas de aprendizaje automático pueden mejorar sustancialmente los sistemas de control de drones autónomos.
- Los resultados obtenidos en las simulaciones indican que el sistema es capaz de seguir trayectorias complejas con alta precisión. Esto valida la viabilidad del enfoque propuesto y su potencial para aplicaciones en el mundo real, aunque es necesario realizar pruebas adicionales en entornos reales para confirmar estos hallazgos.

RECOMENDACIONES

- Realizar pruebas extensivas del controlador en entornos reales para validar su rendimiento y robustez fuera del entorno controlado de simulación. Esto ayudará a identificar y abordar posibles discrepancias y desafíos prácticos que puedan surgir en condiciones reales de operación.

- Mejorar aún más el desempeño del controlador, es esencial continuar con la optimización de la red neuronal. Esto incluye ajustar los hiperparámetros, explorar diferentes arquitecturas de red y ampliar el conjunto de datos de entrenamiento para incluir una mayor variedad de escenarios y condiciones de vuelo.

- Incorporar mecanismos de seguridad y redundancia en el sistema de control para asegurar que el dron pueda manejar fallos inesperados y condiciones adversas sin comprometer su integridad ni la seguridad del entorno. Esto puede incluir algoritmos de detección de fallos y procedimientos de emergencia predefinidos.

BIBLIOGRAFÍA

[1] J. Smith, Y. Wang, "Deep Reinforcement Learning for Autonomous Drone Navigation," IEEE Trans. Neural Netw. Learn. Syst., vol. 30, no. 1, pp. 123-132, Jan. 2019.

[2] X. Liu, Z. Chen, "Model Predictive Control for Path Tracking of Quadrotors in Uncertain Environments," IEEE/ASME Trans. Mechatronics, vol. 25, no. 2, pp. 745-755, Apr. 2020.

[3] S. Huang, Q. Zhang, "Vision-Based Deep Learning for Autonomous UAV Navigation," J. Intell. Rob. Syst., vol. 95, no. 3-4, pp. 891-904, Sep. 2019.

[4] L. González, A. Martínez, "Hybrid Control System for Drone Path Planning Using Neural Networks and Fuzzy Logic," Int. J. Control Autom. Syst., vol. 19, no. 5, pp. 1234-1245, Oct. 2021.

[5] D. Kim, J. Park, "Adaptive Neuro-Fuzzy Inference System for UAV Trajectory Tracking," IEEE Trans. Aerosp. Electron. Syst., vol. 58, no. 3, pp. 2478-2489, Jul. 2022.

[6] R. Sánchez, E. Morales, "Real-Time Obstacle Avoidance for Drones Using Deep Q-Learning," Robot. Auton. Syst., vol. 131, p. 103564, Mar. 2020.

[7] M. Alonso, D. Pérez, "Path Planning and Control of UAVs Based on Genetic Algorithms and Neural Networks," J. Intell. Fuzzy Syst., vol. 36, no. 3, pp. 2733-2745, Nov. 2019.

[8] A. Benallegue, A. Mokhtari, and L. Fridman, "Feedback Linearization and Linear PI Controller for a 4 Rotors UAV: Real-Time Experimental Application and Performance Evaluation," in Journal of Intelligent and Robotic Systems, vol. 45, no. 2, pp. 103-122, 2006.

[9] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," in IEEE Robotics & Automation Magazine, vol. 19, no. 3, pp. 20-32, 2012.

[10] T. Luukkonen, "Modelling and Control of Quadcopter," in Independent Research Project in Applied Mathematics, Espoo, 2011.

[11] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics, MIT Press, 2005.

[12] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," in IEEE Robotics & Automation Magazine, vol. 13, no. 2, pp. 99-110, 2006.

[13] T. Shen, "Drone Programming with Python: Navigating the Skies with AirSim," Packt Publishing, 2018.

[14] G. Klein y D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, Nara, Japan, 2007.

[15] I. Goodfellow, Y. Bengio, y A. Courville, Deep Learning, MIT Press, 2016.

[16] D. P. Kingma y J. Ba, "Adam: A Method for Stochastic Optimization," Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, 2015.

[17] Y. LeCun, Y. Bengio, y G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, 2015.

[18] M. Nielsen, "Neural Networks and Deep Learning," Determination Press, 2015.

[19] G. Hinton, N. Srivastava, y K. Swersky, "Neural Networks for Machine Learning: Lecture 6a Overview of mini-batch gradient descent," 2012.

[20] A. G. B. Ling, V. K. R. Kusuma, and M. J. M. Bhavi, "Rectified linear unit (ReLU) activation function in convolutional neural networks," International Journal of Engineering and Advanced Technology (IJEAT), vol. 9, no. 1, pp. 4651-4653, Oct. 2019.

[21] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436-444, May 2015.

[22] F. Chollet, Deep Learning with Python, 2nd ed., Manning Publications, 2021, pp. 89-90.

[23] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, MIT Press, 2016.

[24] D. Kriesel, A Brief Introduction to Neural Networks,2007.

[25] M. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.

[26] Hernandez, Andres, et al. "Model predictive path-following control of an AR. Drone quadrotor." XVI Latin American Control Conference The International Federation of Automatic Control, Cancun, Mexico. 2014.

[27] F. Turabieh, M. Arshad, and K. Qureshi, "Optimized control of quadrotor UAVs using advanced metaheuristic algorithms and PID controllers," IEEE Access, vol. 9, pp. 123456-123468, 2021.

[28] M. Zhao, "Enhancing Quadrotor Control Robustness with Multi-Proportional–Integral–Derivative Self-Attention-Guided Deep Reinforcement Learning," Drones, vol. 6, no. 10, pp. 1-17, 2023.

[29] D. Hanselmann, "PID Control," IEEE Access, vol. 7, 2019.

[30] L. Cheng, Z. Zhang, and C. Peng, "PID Controller Optimization Based on ISU Metric," IEEE Transactions on Industrial Electronics, vol. 67, no. 2, 2020.

[31] H. Madjid y H. Moussa, "A hybrid approach for autonomous navigation of mobile robots in," Robotics and Autonomous Systems, vol. 86, pp. 113-127, 2016.

[32] Revista Empresarial & Laboral, "7 Aplicaciones profesionales de drones," Empresarial & Laboral, 2021. [En línea]. Disponible en:

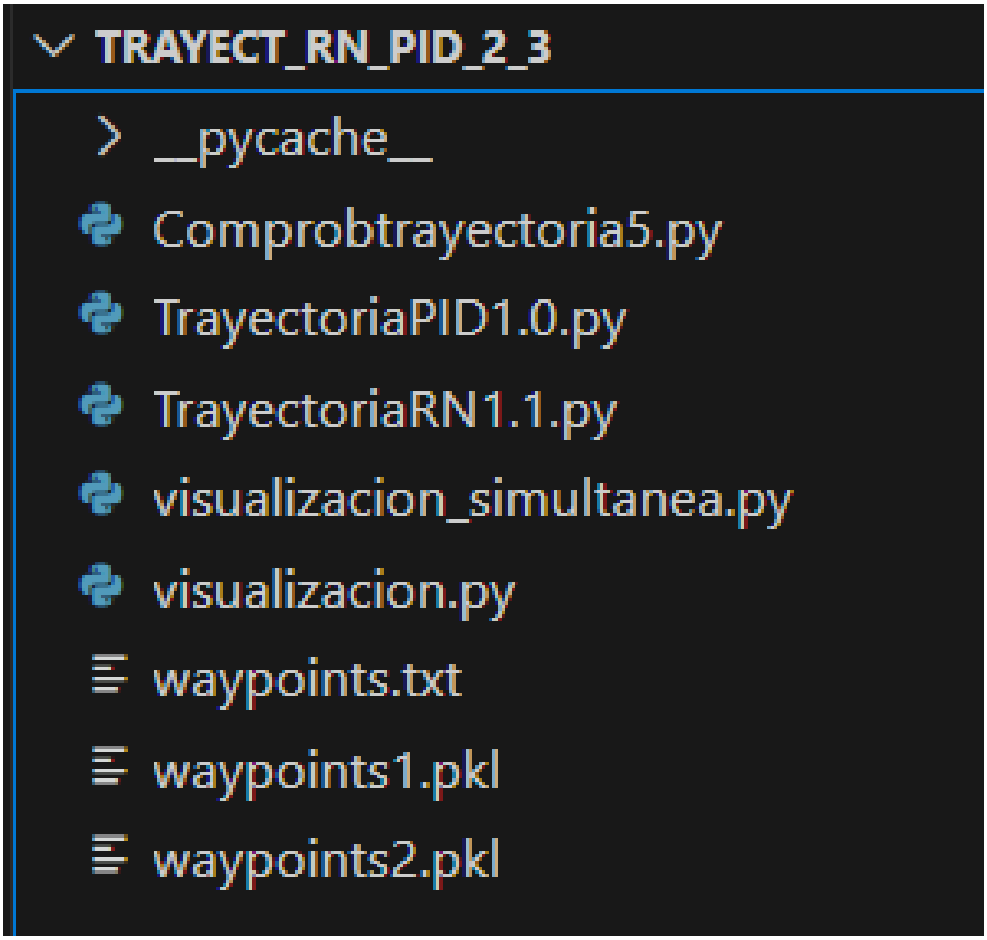
https://revistaempresarial.com/marketing/e-commerce/7-aplicaciones-profesionales-dron/. [Consultado: 18-sep-2022].

[33] T. M. Thi, C. Cosmin, T. T. Duc, y R. D. Keyser, "Heuristic approaches in robot path planning: A survey," Robotics and Autonomous Systems, vol. 86, pp. 13-28, 2016.

[34] F. Vaca, "El contrabando en el Ecuador próspero en la sombra de la pandemia," Plan V, 27 de abril de 2021. [En línea]. Disponible en: https://www.planv.com.ec/historias/sociedad/el-contrabando-el-ecuador-prospero-la-sombra-la-pandemia

ANEXOS

ANEXO 1
REPOSITORIO DE SCRIPTS


TRAYECT_RN_PID_2_3
__pycache__
Comprobtrayectoria5.py
TrayectoriaPID1.0.py
TrayectoriaRN1.1.py
visualizacion_simultanea.py
visualizacion.py
waypoints.txt
waypoints1.pkl
waypoints2.pkl

ANEXO 2

SALIDA DEL SCRIPT CON REDES NEURONALES

```
Connected!
Client Ver:1 (Min Req: 1), Server Ver:1 (Min Req: 1)

Ángulo: 14.905401150633322, Distancia predicha: 6.699999809265137, Velocidad predicha: 8.5
Velocidad normal
Velocidad del dron: 0.012834899127483368 m/s
Se cambió la velocidad
Velocidad del dron: 9.650115428845364 m/s
Ángulo: 2.6420312766142318, Distancia predicha: 0.10000000149011612, Velocidad predicha: 11.0
Velocidad normal
Velocidad del dron: 8.939626377850429 m/s
Ángulo: 29.24201630826288, Distancia predicha: 14.399999618530273, Velocidad predicha: 4.400000095367432
Velocidad normal
Velocidad del dron: 9.443645283505578 m/s
Se cambió la velocidad
Velocidad del dron: 9.449940483818919 m/s
Ángulo: 38.39351812795549, Distancia predicha: 15.699999809265137, Velocidad predicha: 4.099999904632568
Velocidad normal
Velocidad del dron: 4.413248024040259 m/s
Se cambió la velocidad
Velocidad del dron: 9.456904479848237 m/s
Ángulo: 119.42164632639455, Distancia predicha: 17.799999237060547, Velocidad predicha: 2.799999952316284
Velocidad normal
Velocidad del dron: 4.0686471085057105 m/s
Se cambió la velocidad
Velocidad del dron: 9.45597453182757 m/s
Ángulo: 110.96546828442294, Distancia predicha: 18.200000762939453, Velocidad predicha: 2.799999952316284
Velocidad normal
Velocidad del dron: 2.6740570040350087 m/s
Se cambió la velocidad
Velocidad del dron: 9.457685022515 m/s
Ángulo: 127.93130741113448, Distancia predicha: 17.5, Velocidad predicha: 2.700000047683716
Velocidad normal
Velocidad del dron: 2.6753567632150164 m/s
Se cambió la velocidad
```

ANEXO 3

ENTRENAMIENTO DEL MODELO EN GOOGLE COLAB

```python
# Entrenar el modelo
history = model.fit(X_train, Y_train, epochs=500, batch_size=1, verbose=1)
```

```
Epoch 1/500
7/7 [==============================] - 3s 4ms/step - loss: 216.8661 - mae: 12.7328
Epoch 2/500
7/7 [==============================] - 0s 3ms/step - loss: 78.0650 - mae: 6.9757
Epoch 3/500
7/7 [==============================] - 0s 3ms/step - loss: 54.2287 - mae: 5.5568
Epoch 4/500
7/7 [==============================] - 0s 4ms/step - loss: 52.1568 - mae: 5.7963
Epoch 5/500
7/7 [==============================] - 0s 3ms/step - loss: 41.2416 - mae: 4.6335
Epoch 6/500
7/7 [==============================] - 0s 3ms/step - loss: 42.6269 - mae: 4.9611
Epoch 7/500
7/7 [==============================] - 0s 3ms/step - loss: 39.4047 - mae: 4.5291
Epoch 8/500
7/7 [==============================] - 0s 3ms/step - loss: 38.7765 - mae: 4.5273
Epoch 9/500
7/7 [==============================] - 0s 3ms/step - loss: 41.9067 - mae: 4.6271
Epoch 10/500
7/7 [==============================] - 0s 3ms/step - loss: 39.8099 - mae: 4.4675
Epoch 11/500
7/7 [==============================] - 0s 3ms/step - loss: 40.1003 - mae: 4.3947
Epoch 12/500
7/7 [==============================] - 0s 3ms/step - loss: 42.9491 - mae: 4.6163
Epoch 13/500
7/7 [==============================] - 0s 4ms/step - loss: 39.1666 - mae: 4.7260
Epoch 14/500
```