UNIVERSIDAD SANTO TOMÁS

IMPLEMENTACIÓN DE CONTROLADOR DE VUELO PARA VEHÍCULOS AÉREOS NO TRIPULADOS MULTI-ROTOR BASADO EN TÉCNICAS DE APRENDIZAJE PROFUNDO

Realizado por

Javier Alexis Cárdenas Bohórquez

Uriel Eduardo Carrero Cuadrado

Proyecto de Grado presentado en cumplimiento del requisito
para optar por el grado de Ingeniería Electrónica

Grupo de Investigación GED (Grupo de Estudio y Desarrollo en Robótica)
Facultad de Ingeniería Electrónica
División de Ingenierías

Julio de 2022

IMPLEMENTACIÓN DE CONTROLADOR DE VUELO PARA VEHÍCULOS AÉREOS NO TRIPULADOS MULTI-ROTOR BASADO EN TÉCNICAS DE APRENDIZAJE PROFUNDO

Realizado por

Javier Alexis Cárdenas Bohórquez

Uriel Eduardo Carrero Cuadrado

Proyecto de Grado presentado en cumplimiento del requisito
para optar por el grado de Ingeniería Electrónica

Dirigido por

Ph.D Juan Manuel Calderón Chávez

Co-Dirigido por

M.Sc Edgar Camilo Camacho Poveda

Grupo de Investigación GED (Grupo de Estudio y Desarrollo en Robótica)
Facultad de Ingeniería Electrónica
División de Ingenierías

Julio de 2022

UNIVERSIDAD SANTO TOMÁS
Ingeniería Electrónica
Este proyecto de grado fue aceptado para la carrera de Ingeniero(a) Electrónico(a) en

Director: Juan Manuel Calderón Chávez
Doctorado Electric Engineering
University of South Florida
Magíster en Ingeniería Electrónica y de Computadores
Universidad de los Andes
Co-director: Edgar Camilo Camacho Poveda
Magíster en Ingeniería Electrónica
Universidad Nacional

Jurados: Carlos Andrés Torres Pinzón
Doctorado en Ingeniería Electrónica, Automática y Comunicaciones
Magíster Ingeniería Electrónica
Universitat Rovira i Virgili

Gerson David Cruz Capador
Candidato a Magíster en Ciencias de la Información y las Comunicaciones
Universidad Distrital Francisco José de Caldas

Sustentación proyecto de grado: 23 de Junio 2022, Bogotá D.C.

Javier Alexis Cárdenas Bohórquez _____

Uriel Eduardo Carrero Cuadrado _____

NOTA DE ACEPTACIÓN

El trabajo de grado titulado «**Implementación de controlador de vuelo para vehículos aéreos no tripulados multi-rotor basado en técnicas de aprendizaje profundo**» realizado por los estudiantes **Javier Alexis Cárdenas Bohórquez** y **Uriel Eduardo Carrero Cuadrado** cumple con los requisitos exigidos por la **Universidad Santo Tomás** para optar al título de **Ingeniero Electrónico.**

Juan Manuel Calderón Chávez _____

Edgar Camilo Camacho _____

Carlos Andrés Torres Pinzón _____

Gerson David Cruz Capador _____

DEDICATORIA

Dedico este proyecto a mi Madre que siempre estuvo pendiente de este proyecto hasta el día que falleció. Su curiosidad y anhelo por este proyecto me motivó a realizar un buen trabajo. Así mismo, dedico este proyecto a ti que estás leyendo esto, ya que de alguna manera espero poderte aportar algún conocimiento aquí adquirido.

Javier Cárdenas

Dedico este proyecto a Dios y a toda mi familia, en los cuales siempre encontré apoyo, sabiduría y motivación, para desempeñar mi vocación.

Uriel Carrero

AGRADECIMIENTOS

Agradezco a los mecanismos evolutivos que permitieron que el *Homo Sapiens* se asentara en comunidades, lo cual formó estructuras sociales que crecieron con el tiempo. Estas relaciones sociales permitieron que la humanidad sobreviviera ante los depredadores y aún perduran hoy en día. Este trabajo es la prueba fiel de que el ser humano es una especie interdependiente de otros individuos y que, sin ayuda de otras personas, no hubiese podido lograrlo. Agradezco a mi familia, amigos y compañeros que me apoyaron (se necesitaría un *Googleplex* de memoria para almacenar sus nombres). Así mismo, mención especial a mi compañero de trabajo y a mis directores. Finalmente, agradezco a todos mis profesores desde la infancia hasta la universidad (incluyendo a todas esas personas de las que pude aprender algo), ya que me permitieron pararme sobre sus hombros, sobre hombros de gigantes.

Javier Cárdenas

Agradezco a Dios por guiarme y permitirme llegar a la consecución de este sueño, a toda mi familia y amigos por acompañarme, ayudarme e inspirarme a afrontar este reto. Agradezco a todos mis profesores y compañeros, especialmente a los directores y a mi compañero de trabajo en este proyecto. A ellos les agradezco el siempre estar dispuestos a aportar conocimiento, tiempo y esfuerzo para el desarrollo de este proyecto.

Uriel Carrero

Índice general

Índice de Figuras

Índice de Cuadros

Índice de Abreviaturas

ANN	**A**rtificial **N**eural **N**etwork
CRAI	**C**entro de **R**ecursos para el **A**prendizaje y la **I**nvestigación
CNN	**C**onvolutional **N**eural **N**etwork
DBN	**D**eep **B**elief **N**etwork
DFCS	**D**igital **F**light **C**ontrol **S**ystem
DPG	**D**eterministic **P**olicy **G**radient
DQN	**D**eep **Q**-**N**etwork
FAC	**F**ully-**A**utonomous **C**ontrol
GED	**G**rupo de **E**studio y **D**esarrollo
GPS	**G**lobal **P**ositioning **S**ystem
GPU	**G**raphics **P**rocessing **U**nit
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineer
LQR	**L**inear **Q**uadratic **R**egulator
LSTM	**L**ong **S**hort-**T**erm **M**emory
MAE	**M**ean **A**bsolute **E**rror
MDP	**M**arkov **D**ecision **P**roblem
PID	Controlador **P**roporcional **I**ntegral y **D**erivativo
POMDP	**P**artially **O**bservable **M**arkov **D**ecision **P**rocess
PPO	**P**roximal **P**olicy **O**ptimization
RC	**R**adio **C**ontrolled
RL	**R**einforcement **L**earning
RNN	**R**ecurrent **N**eural **N**etwork
SAC	**S**emi-**A**utonomous **C**ontrol
SMC	**S**lidding **M**ode **C**ontrol
SPARC	**S**cholarly **P**ublishing and **A**cademic **R**esources **C**oalitionnmanned
TCM	**T**eoría de **C**antidad de **M**ovimiento
TEP	**T**eoría de **E**lemento de **P**ala

UAV **U**nmanned **A**erial **V**ehicle
USTA **U**niversidad **S**anto **T**omás
VLoS **V**isual **L**ine **o**f **S**ight
VTOL **V**ertical **T**ake-**O**ff and **L**anding

Resumen

Este proyecto de grado presenta el diseño e implementación de un controlador de posición para un UAV multi-rotor basado en redes neuronales profundas y entrenado mediante aprendizaje supervisado, tomando como referencia un controlador PID. Se detalla el proceso de selección del entorno de simulación, el controlador y el modelo seleccionado. Así mismo, se realizan evaluaciones de trayectorias de control para la construcción de un conjunto de datos que permita entrenar el modelo. Se entrenan distintas arquitecturas de redes neuronales, mediante el uso del algoritmo *Hyperband* para determinar los mejores hiperparámetros. Finalmente se evalúa el rendimiento del controlador entrenado con respecto al controlador base mediante la respuesta temporal con diferentes señales de control. Como producto final se presenta: el conjunto de datos del controlador de referencia, un repositorio con los programas realizados para el desarrollo y análisis, y el modelo de la red neuronal.

Abstract

This project presents the design and implementation of a position controller for a multi-rotor UAV based on deep neural networks and trained by supervised learning, taking as reference a PID controller. The process of selection of the simulation environment, the controller and the selected model is detailed. Likewise, control trajectory evaluations are carried out for the construction of a data set to train the model. Different neural network architectures are trained, using the *Hyperband* algorithm to determine the best hyperparameters. Finally, the performance of the trained controller with respect to the base controller is evaluated by means of the time response with different control signals. As a final product we present: the data set

of the reference controller, a repository with the programs made for the development and analysis, and the neural network model.

Capítulo 1

Introducción

Los inicios del concepto del vehículo aéreo no tripulado (UAV) se remontan hacia 1849, empleado principalmente en el ámbito militar. La investigación centrada en controlar dichos vehículos que ha involucrado modelos, técnicas y teorías a lo largo de la historia, se ha dado debido a que siempre han sido observados como una tecnología con el potencial de solucionar distintos problemas en múltiples áreas (ver capítulos 2 y 4). Algunas de esas posibles tareas previstas para UAVs están relacionadas con la mitigación del COVID 19, desarrollando tareas en el monitoreo aéreo de áreas de aislamiento, distribución rápida de medicinas o aspersión y desinfección aérea [1].

Los UAV de ala fija son utilizados en aplicaciones que requieran una mayor cobertura, mayor potencia de carga o mayor velocidad. Por otro lado, las capacidades especiales de los UAV tipo multi-rotor, como el despegue y aterrizaje vertical (VTOL), el vuelo omnidireccional y la maniobrabilidad en espacios reducidos permiten la aplicación tanto en interiores como en exteriores. Así mismo, la facilidad de vuelo (incluyendo despegue y aterrizaje) hace que el desarrollo de aeronaves autónomas sea más sencilla en los multi-rotores [2].

Para lograr desarrollar aeronaves autónomas, primero es necesario garantizar que los sistemas de control de bajo nivel respondan adecuadamente a los comandos de posición, orientación y velocidad dados. Sin embargo, el diseño de sistemas de control automáticos es complejo debido a las dinámica inestable, no lineal, acoplada y sub-actuada del multi-rotor. Igualmente, parámetros cambiantes como la temperatura y presión atmosféricas, dirección del viento o efectos aerodinámicos como el efecto suelo, la fricción con el aire y el flujo inducido con otros vehículos; dificultan aún más el proceso.

Este problema se ha abordado desde distintos enfoques (ver capítulo 3). No obstante, el rendimiento de estos controladores no garantizan estabilidad frente a maniobras agresivas o son altamente dependiente al modelo y no se adapta ante los cambios de factores ambientales. Los modelos inteligentes basados en redes neuronales cuentan con la capacidad de generalizar experiencia a partir de datos dados, por lo que se supone un gran potencial para el control y modelado de complejos sistemas dinámicos [3], como el control a bajo nivel de UAVs [4].

En el capítulo 3 se proponen distintos métodos de aprendizaje profundo, tanto *Online* como *Offline*. Los métodos *Online* van construyendo los datos al mismo tiempo que los va experimentando, por lo que el modelo se va adaptando conforme el entorno percibido cambie, aunque demandan gran tiempo y costo computacional. Por otro lado, los métodos *Offline* aprenden desde un conjunto de datos predefinido y son más rápidos y más eficientes que los *Online*, sin embargo, el rendimiento del modelo depende de la calidad y cantidad de los datos, así mismo, si el ambiente cambia, el agente debe volver a entrenarse con nuevos datos recolectados.

En [5], los autores proponen agilizar el aprendizaje del modelo utilizando una combinación de ambos métodos. El desarrollo de este trabajo va orientado a obtener un controlador basado en redes neuronales para un UAV multi-rotor utilizando aprendizaje *Offline* supervisado, a partir del planteamiento de un problema de regresión con un controlador de referencia, definiendo las características como las entradas al controlador y las etiquetas como las señales de control. La contribución del trabajo propuesto radica en la evaluación de diversas trayectorias para la elaboración del conjunto de datos y la comparación de distintas arquitecturas de redes neuronales para el control del UAV.

La estructura en la que se desarrolla este documento es la siguiente: el problema en la obtención de modelos para diseñar controladores de UAVs a bajo nivel se explica en detalle en el capítulo 2, la revisión de las investigaciones sobre el control de UAVs, enfocado principalmente al uso de Inteligencia Artificial para control se expone en el capítulo 3, las investigaciones y propuestas de diferentes entidades que motivan la implementación de controladores de UAVs se presentan en el capítulo 4, los objetivos del proyecto se plantean en el capítulo 5, posteriormente se explican los conceptos que involucran la abstracción matemático-física de los UAVs, el control de vuelo y los modelos de aprendizaje profundo contemplados para el desarrollo del proyecto, seguidamente se presenta la metodología seguida en el capítulo 8, finalmente los resultados y conclusiones son presentados en los capítulos 9 y 11, respectivamente.

Este proyecto se realizó gracias al apoyo del Grupo de investigación y Desarrollo en Robótica de la Universidad Santo Tomás (G.E.D.). Este grupo de investigación se enfoca principalmente en el desarrollo de proyectos de investigación basados en fútbol de robots, robótica de enjambre, robótica cooperativa, desarrollo de robots móviles, sistemas de aprendizaje automático y robótica educativa entre otros. Desde 2011 el grupo G.E.D. ha participado activamente en la iniciativa RoboCup [6] siendo parte de la liga Small Size [7]. En el área de robótica de enjambre se han realizado proyectos encaminados al desarrollo de algoritmos de navegación de enjambres [8], [9] y generación de comportamientos colectivos [10], [11]. Adicionalmente, en el área de robótica cooperativa se ha trabajado el problema de transporte de carga suspendida con vehículos aéreos no tripulados [12], [13]. En el campo de diseño e implementación de plataformas robótica se ha enfocado en el desarrollo de un laboratorio para la evaluacion de algoritmos para la navegación de enjambres de robots [14], ademas del diseño e implementación de robots de asistencia casera y desinfección [15], [16]. En al área de los sistemas de aprendizaje se han realizado proyectos encaminados al uso de sistemas de aprendizaje para la generación de comportamientos específicos en la navegación de robots móviles [17], [18], sistemas de recomendación de estilos de vestir [19] y Procesamiento de Lenguaje Natural para la automatización de consultorio jurídico [20].

Capítulo 2

Planteamiento del Problema

Durante la última década, el trabajo a nivel de desarrollo e investigación en drones ha aumentado vertiginosamente [21]. Las primeras modificaciones residían en el diseño mecánico, los métodos de fabricación y de mantenimiento, buscando crear vehículos ligeros, resistentes, versátiles, miniaturizados y de bajo costo [22]. La empresa alemana Microdrones GmbH, dedicada a la fabricación de drones industriales y especializados para investigación científica, aseguran que, gracias a los desarrollos tecnológicos actuales, ya se puede garantizar confiabilidad y rendimiento a bajo costo en estos dispositivos [23].

Pascual Campoy, catedrático de la Universidad Politécnica de Madrid (UPM), asegura que «el nuevo reto que tiene la comunidad científica es convertir los drones en robots autónomos», lo cual abre la posibilidad del uso de drones en gran cantidad de áreas [24]. Actualmente los vehículos aéreos no tripulados (UAVs) multi-rotor son utilizados en diferentes ámbitos, como captura de eventos artísticos y deportivos, reparto de comida, rescate marítimo, mitigación de incendios forestales, apoyo en búsqueda y rescate, detección de plagas y maleza, esparcimiento de pesticidas y fertilizantes en grandes zonas agrícolas, vigilancia de líneas eléctricas con cámaras térmicas, estudio de huracanes o volcanes, entre otros [25]. Verbigracia de esta última, se utilizaron en Fukushima para tomar muestras del reactor nuclear y realizar un plan de limpieza [25].

Dentro de las principales dificultades encontradas en cuanto al diseño de un controlador para un UAV se encuentran su naturaleza no lineal, la dependencia de movimientos traslacionales y rotacionales en todos los ejes, la inestabilidad del sistema ante perturbaciones en el ambiente, las limitaciones a variables climatológicas como la densidad del aire o la temperatura y la acción directa de los motores en dirección perpendicular al plano del UAV,

también llamado ángulo de actitud [26]. Con la intención de modelar el sistema teniendo en cuenta las anteriores variables, se encuentran métodos como el modelo no lineal, Euler-Newton, Euler-Lagrange y cuaterniones. Algunos enfoques adicionalmente tienen en cuenta incertidumbres o situaciones problémicas [27], tales como, ruido en las lecturas de los sensores, cambios repentinos de velocidad del viento, lluvia, carga o fallas en los rotores [28].

El diseño del controlador de vuelo depende de la aplicación, puesto que los requerimientos son distintos. Los sistemas de control lineal, como los controladores proporcionales-derivadores (PD) o los reguladores cuadráticos lineales (LQR), se utilizan ampliamente para mejorar las propiedades de estabilidad de un equilibrio en drones de propósito general [29-31]. Sin embargo, dichos controladores lineales no permiten realizar correctamente el seguimiento de una trayectoria compleja, como en el caso de drones de acrobacias, por lo que es necesario abordar el problema desde enfoques como control robusto [32] o control geométrico [33].

Debido a la versatilidad que se propone en la aplicación de los UAVs, se requieren así mismo controladores que permitan realizar los desplazamientos y maniobras necesarias para ejecutar efectivamente un plan de vuelo preestablecido, capaces de adaptarse a nuevas condiciones que se produzcan, puesto que las aeronaves se diseñan para un uso concreto y no tienen un buen rendimiento en otras tareas [34]. Dicha flexibilidad en el controlador de vuelo permitiría mejorar el rendimiento en el despegue y aterrizaje autónomo, la estabilización de posición o la ejecución de rutas complejas en tiempo real. El ideal es obtener un controlador que garantice la seguridad del vuelo independiente de la aplicación, que responda incluso en entornos peligrosos o con información ruidosa e incompleta.

No obstante, aunque algunos de los estudios nombrados en el capítulo 3 resuelven exitosamente el problema del control de vuelo de los UAVs, dada la naturaleza del método de control, se siguen teniendo problemas en cuanto a la alta dependencia al modelo matemático, ofreciendo baja versatilidad cuando se requieran hacer modificaciones en el controlador ya desarrollado, lo cual, para aplicaciones específicas como la navegación autónoma y evasión de obstáculos en ambientes específicos [35], no se logre el desempeño esperado en la implementación. Por lo tanto, se observa la necesidad de desarrollar un controlador de vuelo adaptable, flexible, moldeable.

Una herramienta con el potencial de satisfacer los requerimientos mencionados anteriormente (especialmente en el campo de la robótica móvil) son los algoritmos de aprendizaje profundo [36, 37]. Se plantea entonces la pregunta: *¿Cómo desarrollar un sistema de control de vuelo para vehículos aéreos no tripulados basado en técnicas de aprendizaje profundo?*

Capítulo 3

Estado del Arte

3.1. Breve Historia del Control de los UAVs

El concepto del UAV (también llamado dron) se remonta hacia 1849, empleado principalmente para uso militar [38]. Durante la Primera Guerra Mundial se intensificó su uso en vigilancia aérea y misiles. En la Segunda Guerra Mundial se incorporaron sistemas radio controlados y de control semiautomático (RC-SAC), que contaban con problemas de interferencia y requerían que la aeronave estuviese dentro de la línea visual de visión (VLoS) del piloto remoto [39]. A partir de la década de 1960, durante la Guerra de Vietnam, se desarrollaron principios técnicos, modelos estadísticos y sistemas electrónicos que permitieron el crecimiento de los UAVs, utilizados principalmente para tareas de reconocimiento [40]. En la última década han sido empleados principalmente por Estados Unidos en misiones de ataque con drones han culminado con la muerte de líderes de Al Qaeda en Pakistán, Yemen, Libia y Afganistán [41]. Si bien, los orígenes de estos dispositivos son en el ámbito militar, actualmente se desempeñan en áreas como cinematografía, arqueología, geología, topografía, meteorología, mitigación incendios o recreación [25, 42, 43].

Inicialmente, en los sistemas de control para UAVs, se utilizaban giroestabilizadores mecánicos a partir de giróscopos que eran implementados en aplicaciones marítimas, sin embargo su comportamiento era regular [39]. En 1911, Glenn Curtiss presentó un nuevo modelo más pequeño y permitía el control en los tres ejes mediante servomotores [44]. Hasta finales de la década de los 60 los sistemas eran principalmente de radio seguimiento basados en los controladores clásicos con los avances teóricos propuestos por Maxwell, Mason, Ziegler y Nichols hacia 1930 [45]. Posteriormente, hacia 1990, se diseñaron sistemas más sofisticados

para despegue y aterrizaje vertical (VTOL), utilizando nuevas tecnologías como el sistema de posicionamiento global (GPS), sistemas digitales de control de vuelo (DFCS), visión artificial y sistemas biomiméticos. Actualmente, los trabajos investigativos residen en la búsqueda de nuevos materiales, nuevos modelos, operaciones completamente autónomas, adaptación a nuevos ambientes, integración con otros sistemas y nuevas aplicaciones.

3.2. Modelado y Control

El problema principal de estos sistemas es su no linealidad y la inestabilidad, lo que dificulta el control [25, 46]. Se han propuesto diferentes modelos lineales mediante matrices de transformación [47], elementos aerodinámicos según teorías de cantidad de movimiento (TCM) y de elemento de pala o hélice (TEP) [48], o influencias en los motores dado el movimiento traslacional y rotacional [49, 50]. Estos modelos permiten desarrollar sistemas de control con métodos clásicos lineales y de control óptimo. Entre otros, [51] utiliza el método de Ziegler-Nichols en lazo cerrado para determinar los límites de estabilidad del sistema y posteriormente ajustar los coeficientes de un controlador proporcional, integral y derivativo (PID). Métodos lineales más complejos se establecen con controladores PID con reguladores lineales cuadráticos (PID+LQR) [52], LQ y H_∞ [53] o por realimentación de estados [49], los cuales han logrado adoptar las capacidades básicas para el control de vuelo de un UAV, incluso en ambientes con cambios de velocidad y dirección en el viento [53].

Sin embargo, dichos enfoques lineales presentan problemas cuando las acciones de control se alejan de su punto de equilibrio. Distintas soluciones se han propuesto en enfoques con control robusto y no lineal, como los basados en la estabilidad de Lyapunov [54, 55], lógica difusa [56, 57], control adaptativo (*backstepping control*) [58], control híbrido mediante modelos predictivos no lineales y compensadores difusos [59], control en modo deslizante (SMC) [32], controladores híbridos con redes neuronales [60], control geométrico [33], entre otros. Estos suponen desarrollos con cierta robustez frente a las diferentes condiciones ambientales. A pesar de ello, el diseño de los mismos requieren un modelo matemático preciso y sistemas de control de orden superior, cuyos cálculos de coeficientes son complejos. Así mismo, se cuentan con singularidades cuando se tienen que ejecutar complejas maniobras rotacionales.

Enfoques más recientes combinan métodos clásicos con métodos de aprendizaje de máquina, como [61-63] para diseñar controladores robustos y eficientes que son capaces de auto-sintonizarse, o incluso adaptar los esfuerzos de control a cambios de parámetros en el sistema. Concretamente, [64] implementa un observador de estados adaptable basado en

redes neuronales con controlador de supervisión para implementar un control dinámico, el cual logra incluso reconstruir exitosamente los estados de error a partir del observador neuronal adaptativo, incluso cuando las medidas en la salida son afectadas por el ruido. Además, en [65] se realiza un compensador robusto utilizando política de gradiente determinista (DPG), en donde se logra un controlador preciso y que no necesita de un modelo matemático del sistema. Pese a todo, la implementación requiere de un dispendioso proceso de entrenamiento y de mayores esfuerzos computacionales.

3.3. Técnicas de Aprendizaje Profundo

No obstante, [66-68] abordan distintas técnicas de aprendizaje por refuerzo RL e inteligencia artificial en las que, de manera general, se propone encontrar una solución eficiente mediante la producción de agentes completamente autónomos, que interactúen con entornos y aprendan a replicar comportamientos óptimos de acuerdo al problema dado, reduciendo enormemente los tiempos de aprendizaje. Dichas técnicas incluyen procesos de decisión de Markov (MDP) [69], DPG [65], métodos de actor-crítico [67], redes neuronales profundas (DQN) con memoria de experiencia priorizada [70] y generalización de experiencias [71]. Recientemente proyectos relacionados con aprendizaje profundo, muestran resultados favorables en el rendimiento de agentes inteligentes aplicados a la toma de decisiones. En particular, [72] plantea una red DQN para juegos de atari y, posteriormente, [73] compara el rendimiento del modelo con personas profesionales en el área de videojuegos, en donde se evidencia que en más del 50 % de los juegos se obtiene una efectividad igual o superior al nivel humano. Otras aplicaciones se dan en el campo de la robótica, en donde [74] utiliza el aprendizaje por refuerzo para el desplazamiento de un hexápodo.

Asimismo, se han realizado proyectos e investigaciones netamente enfocados al aprendizaje de máquina para la navegación autónoma de UAVs en entornos 3D como en [75]. Un estudio de la Universidad Santo Tomás [35] presenta además un desarrollo importante en evasión de obstáculos. Estudios más profundos tienen en cuenta distintas variables que pueden afectar el aterrizaje del sistema, como el viento, la lluvia, la carga, dirección del viento, entre otros [27]. Para resolver estos problemas aplican un controlador basado en iteración de política de mínimos cuadrados (LSPI), encontrando en este método robustez y disminución del consumo de energía. En [76] y [77] se comparan algoritmos de aprendizaje por refuerzo (RL) como optimización de políticas próximas (PPO), MDP y procesos de decisión de Markov parcialmente observables (POMDP), en distintos problemas de navegación. Específicamente, en [77] se comparan dichos algoritmos para establecer un sistema de recomendación de

navegación con el cual es posible memorizar ciertas rutas, haciendo que, si el UAV se encuentra con un callejón sin salida pueda retroceder en lugar de recomendar moverse en círculos. En [76] se implementa un POMDP junto con un modelo control predictivo mediante búsqueda de política para procesar información parcial del entorno. Finalmente en [78] se comparan diversos algoritmos de RL con un controlador de vuelo PID, este último artículo está más orientado a lo que se va a trabajar. Sin embargo, todos los estudios anteriores permiten realizar una contextualización acerca de los controladores de vuelo y las distintas técnicas aplicadas sobre estos.

Capítulo 4

Justificación

Los UAVs hacen parte de las tecnologías con mayor impacto a futuro en diferentes áreas. Dada la importancia de ésta tecnología, la asociación para la robótica en Europa SPARC, encargada de evaluar distintas tecnologías y orientarlas a generar impacto económico y social [79], ha destacado varios proyectos que aplican UAVs en su mapa para el desarrollo de la robótica en 2020 [80], alguno de ellos son:

- ADAM (España): movilidad autónoma mediante cooperación de múltiples plataformas.
- DEMUEB (Alemania): investigación sobre tecnologías clave y soluciones operativas para sistemas de aeronaves no tripuladas UAS.
- DECSA (Francia): desarrollo de algoritmos de navegación y percepción para pequeños drones en zonas urbanas medio ambiente.
- SMAVNET (Suiza): desarrollo de enjambres de robots voladores que se pueden implementar en áreas de desastre para crear redes de comunicación para rescatistas.
- AIRobot: robótica de aérea para inspección.
- FIELDCOPTER: agricultura de precisión basada en GPS-EGNOS utilizando UAVs.

Adicionalmente, Markus Christen y colaboradores presentaron, en 2018, un reporte evaluando las oportunidades y los riesgos de la tecnología de drones en la investigación y en la industria en Alemania y Suiza, en el cual se destacan el gran potencial que tiene a futuro y los numerosos efectos positivos en la economía y la sociedad [81]. Como se puede observar, el vuelo autónomo para UAVs es una herramienta con el potencial de expandir los límites de

varios campos que protegen y aportan a la humanidad, donde se encuentra, aparte de los ya mencionados, la agricultura, la seguridad, la investigación, la búsqueda y rescate. Un ejemplo puntual y significativo de la idea anterior se da en la ciudad de Pripyat, Ucrania, que dada la catástrofe ambiental ocurrida en 1986, implica un gran riesgo para la vida humana. En un principio se realizaron misiones de vuelo tripuladas, las cuales afectaron la salud de la tripulación, sin embargo, gracias al uso de vehículos de vuelo no tripulados se evitan afecciones, haciendo más sencillas y seguras las futuras investigaciones en el lugar de la catástrofe [82].

En el caso particular de Colombia, la revista digital Impacto TIC enfatiza la importancia de la tecnología, la innovación y la ciencia para el bienestar y el desarrollo de la comunidad, exponiendo específicamente las aplicaciones que tendrían los drones para la superación del posconflicto y la preservación de la paz [83]; en la lucha contra el narcotráfico [84], el desminado humanitario [85] y el mejoramiento de la productividad del agro colombiano [86]. Además, a partir de la resolución 04201 de 2018, la Unidad Administrativa Especial de Aeronáutica Civil establece en Colombia las normativas para utilizar un UAV dependiendo de su propósito [87]. Lo anterior permite el uso seguro y legal de los drones, lo cual puede incentivar la inversión en este ámbito. Según Goldman Sachs Research, se estima una oportunidad de mercado a nivel global de alrededor de $100 mil millones de dólares para drones entre 2016 y 2020 y un aumento a futuro. Lo anterior, deja en evidencia la necesidad de investigar y desarrollar los posibles prototipos que permitan contar con un vuelo autónomo, flexible dadas las aplicaciones, rentable y seguro en los UAVs en Colombia [88].

En conjunto con la idea anterior, se desea aportar, mediante la exploración de métodos de aprendizaje profundo para el control de UAVs multi-rotor, a un desarrollo reciente en la Universidad Santo Tomás [35], en donde se diseña un sistema de navegación autónoma y evasión de obstáculos basado en aprendizaje reforzado. El agente allí mencionado aprende a tomar decisiones inteligentes y alcanza satisfactoriamente su objetivo en un ambiente simulado. No obstante, si se quiere implementar en un entorno real, los autores concluyeron que es necesario un controlador de vuelo cuya velocidad de respuesta sea alta y que garantice estabilidad frente a diferentes condiciones ambientales, ya que, de lo contrario, se necesitaría tener en cuenta retrasos e incertidumbres en los cálculos de los nuevos estados para la toma de decisiones. Con el desarrollo de este proyecto, se sentaría una base importante en la construcción de drones autónomos y flexibles por parte de la Universidad. Dicha tecnología podría optimizar procesos productivos, mejorar la calidad de vida de la población rural, realizar tareas peligrosas para los humanos y, sobre todo, salvar vidas.

Capítulo 5

Objetivos

5.1. Objetivo General:

Diseñar un sistema de control de vuelo para un UAV multi-rotor basado en técnicas de aprendizaje profundo, para evaluar su desempeño en entornos simulados.

5.2. Objetivos Específicos:

- Seleccionar un sistema de control del estado del arte, que sirva como base para el entrenamiento del sistema de aprendizaje profundo.
- Plantear la arquitectura de una red neuronal profunda, que permita mapear las entradas, salidas y estados del controlador.
- Entrenar la red neuronal profunda, con el objetivo de modelar el comportamiento dinámico del controlador.
- Evaluar el desempeño del sistema de control entrenado mediante parámetros como el tiempo de respuesta, error de estado estacionario y el sobrepaso máximo, sobre el modelo lineal clásico del dron.

Capítulo 6

Marco Teórico

Este capítulo contiene algunos referentes conceptuales que son base para la investigación aquí realizada. En primer lugar se presentan nociones teóricas para comprender el funcionamiento de un *UAV cuadrotor*. Seguidamente, se muestra la estructura básica de un sistema de control y su implementación para un controlador de bajo nivel de un UAV. Finalmente se enuncian algunas de las técnicas del aprendizaje profundo que son claves para el desarrollo de este proyecto.

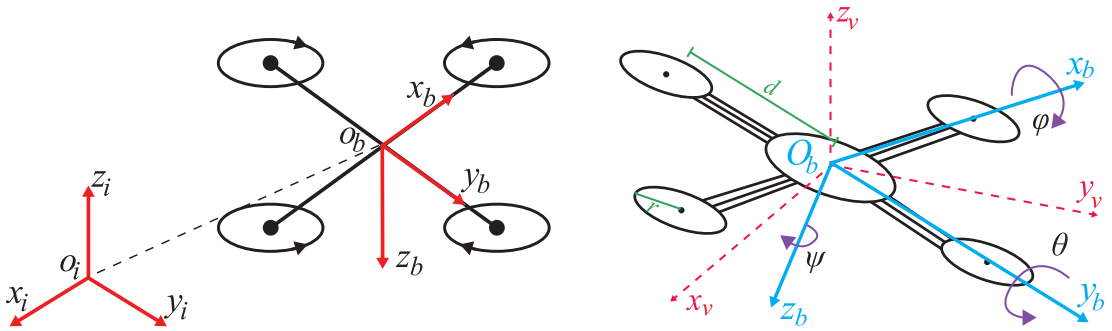
6.1. UAV Cuadrotor

Existen distintos tipos de UAVs, los cuales se pueden clasificar principalmente en aeronaves de ala fija, helicóptero de un solo rotor y multicóptero. Éste último, también llamado multirotor, se subdivide de acuerdo al número de hélices que éste contenga, así pues se encuentran tricópteros (3 motores), cuadricópteros o cuadrotores (4 motores), hexacópteros (6 motores), octacópteros (8 motores), entre otros. De ellos, el más popular es el cuadrotor, el cuál tiene cuatro salidas de control: las cuatro velocidades angulares de los motores [46].

6.1.1. Nociones Preliminares

6.1.1.1. Sistema de Coordenadas

Antes de describir el modelo matemático del dron, es necesario establecer el sistema de coordenadas mediante el cual será posible localizar la aeronave dentro de un espacio tridimensional. Para ello, existen dos marcos de referencias: el primero es el sistema de referencia inercial O_i, en el cual el punto de referencia es arbitrario, global y fijo. El segundo marco de referencia es O_b, también inercial, pero con respecto al baricentro del dron. O_v es el marco del vehículo y representa la proyección O_i sobre O_b, tal como se muestra en la Figura 1 [89].



(A) O_i (B) O_b

FIGURA 1: Sistemas de referencia inerciales.

6.1.1.2. Movimiento Espacial

El sistema de movimiento de la aeronave está directamente relacionado con los sistemas de referencia anteriormente mencionado: el primero es el movimiento del baricentro con respecto al sistema de referencia del mundo, y el segundo es el movimiento al rededor del baricentro. En cada marco de referencia existen movimientos a lo largo de los tres ejes: movimientos traslacionales en los ejes x, y, z, y los movimientos rotacionales $Roll$, $Pitch$, Yaw o ϕ, θ, ψ. Por lo tanto, para describir el movimiento del dron dentro del espacio es necesario de seis grados de libertad SO(3), los cuales incluyen movimientos hacia adelante y atrás, movimientos laterales, movimientos verticales, y rotaciones con respecto a cada uno de los ejes traslacionales. Teniéndose así un vector de estados con la posición lineal y angular del cuadrotor en el mundo $\begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \in \mathbb{R}^6$.

6.1.2. Representación de Orientación

Para describir la orientación o actitud en la que se encuentra el dron dentro del espacio, se utilizan principalmente los ángulos Euler y los cuaterniones.

6.1.2.1. Ángulos Euler

Los ángulos Euler son un conjunto de tres ángulos que describe la orientación de un cuerpo rígido dentro de un espacio euclidiano tridimensional o para describir la orientación relativa frente a otro marco de referencia [90]. Se define cada uno de los ángulos Euler como:

- Ángulo Pitch θ: es el ángulo entre el eje $O_b x_b$ y el plano $O_i x_i y_i$ como se muestra en la figura 2.
- Ángulo Roll ϕ: es el ángulo entre el eje $O_b z_b$ y el plano $O_b O_{xb} O_b^{'}$ como se muestra en la figura 2.
- Ángulo Yaw ψ: es el ángulo entre la proyección del eje $O_b x_b$ en el plano $O_i x_i y_i$ y la extensión de la linea de $O_i x_i$ como se muestra en la figura 2.

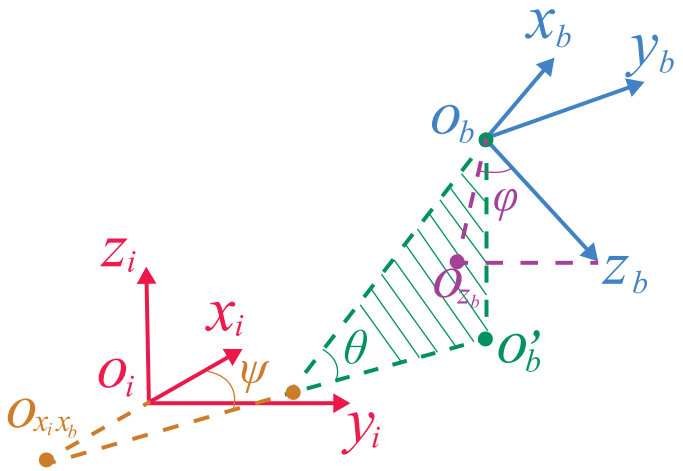


FIGURA 2: Definición de los ángulos Euler
Fuente: [46]

Los ángulos Euler se denotan como $\phi \in]-\pi, \pi]$; $\theta \in]-\frac{\pi}{2}, \frac{\pi}{2}]$; $\psi \in]-\pi, \pi]$ y descrito mediante la ecuaciones 6.1, 6.2 y 6.3.

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{6.1}$$

$$R_y(\theta) = \begin{bmatrix} cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{6.2}$$

$$R_z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.3}$$

Por lo tanto, las coordenadas de posición inercial y las coordenadas de referencia del cuerpo están relacionadas por la matriz de rotación $R_{zyx}(\phi, \theta, \psi) \in SO(3)$:

$$\begin{aligned} R_{zyx}(\phi, \theta, \psi) &= R_z(\psi)R_y(\theta)R_x(\phi) \\ &= \begin{bmatrix} c(\theta)c(\psi) & s(\phi)s(\theta)c(\psi) - c(\phi)s(\psi) & c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi) \\ c(\theta)s(\psi) & s(\phi)s(\theta)s(\psi) + c(\phi)c(\psi) & c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi) \\ -s(\theta) & s(\phi)c(\theta) & c(\phi)c(\theta) \end{bmatrix} \end{aligned} \tag{6.4}$$

en donde $c(\phi) = cos(\phi)$, $s(\phi) = sin(\phi)$, $c(\theta) = cos(\theta)$, $s(\theta) = sin(\theta)$, $c(\psi) = cos(\psi)$, $s(\psi) = sin(\psi)$. Esta matriz describe la rotación del sistema de referencia del cuerpo con respecto al sistema de referencia inercial [89].

Dada una matriz de rotación $R \in \mathbb{R}^3$, se puede determinar los ángulos Euler con respecto a cada eje mediante:

$$\begin{cases} \psi &= arctan(\frac{R_{21}}{R_{11}}) \\ \theta &= arcsin(-R_{31}) \\ \phi &= arctan(\frac{R_{32}}{R_{33}}) \end{cases} \tag{6.5}$$

6.1.2.2. Cuaterniones

Una representación alternativa a las matrices de rotación, son los cuaterniones. Éstos representan las orientaciones y rotaciones en un espacio tridimensional mediante la notación matemática de los cuaterniones de JPL [90].

Se puede expresar un vector de rotación en espacio tridimensional, mediante: $q = q_o + q_x i + q_y j + q_z k$, definiendo un cuaternión como:

$$\begin{aligned} q &= \begin{bmatrix} q_w & q_x & q_y & q_z \end{bmatrix}^T \\ |q|^2 &= q_w^2 + q_x^2 + q_y^2 + q_z^2 = 1 \end{aligned} . \tag{6.6}$$

La asociación del cuaternión con la rotación al rededor de un eje, se realiza mediante:

$$\begin{cases} q_w = & cos(\alpha/2) \\ q_x = & sin(\alpha/2)cos(\beta_x) \\ q_y = & sin(\alpha/2)cos(\beta_y) \\ q_z = & sin(\alpha/2)cos(\beta_z) \end{cases} \tag{6.7}$$

en donde, α es un ángulo de rotación (en radianes), β_x es el ángulo entre el eje de rotación y el eje X, β_y es el ángulo entre el eje de rotación y el eje Y y β_z es el ángulo entre el eje de rotación y el eje Z.

Se puede establecer una matriz de rotación que corresponde a la rotación de dicho elemento en el sentido de las agujas del reloj con respecto al eje de rotación como lo menciona [90]:

$$R = 2 \begin{bmatrix} q_0^2 + q_1^2 - 0.5 & q_1q_2 - q_0q_3 & q_0q_2 + q_1q_3 \\ q_0q_3 + q_1q_2 & q_0^2 + q_2^2 - 0.5 & q_2q_3 - q_0q_1 \\ q_1q_3 - q_0q_2 & q_0q_1 + q_2q_3 & q_0^2 + q_3^2 - 0.5 \end{bmatrix} \tag{6.8}$$

Dada una matriz de rotación $R \in \mathbb{R}^3$, se puede computar un cuaternión que represente la misma rotación que la matriz mostrada por [90]:

$$\begin{cases} |q_0| & = & \sqrt{\frac{Trace(R)+1}{4}} \\ |q_1| & = & \sqrt{\frac{R_{11}}{2} + \frac{1-Trace(R)}{4}} \\ |q_2| & = & \sqrt{\frac{R_{22}}{2} + \frac{1-Trace(R)}{4}} \\ |q_3| & = & \sqrt{\frac{R_{33}}{2} + \frac{1-Trace(R)}{4}} \end{cases} \tag{6.9}$$

en donde, $Trace(R)$ es la traza de la matriz R, definida como la suma de los elementos de la diagonal principal, por lo que $Trace(R) = R_{11} + R_{22} + R_{33} = 4q_0^2 - 1$.

6.1.3. Dinámica de la Aeronave

6.1.3.1. Fuerza y Momento

El cuadrotor experimenta las siguientes fuerzas y momentos [46]: el impulso en el eje z causado por la fuerza de sustentación debido a la rotación de las hélices, el momento en ϕ y θ causado por la diferencia de velocidades de los motores, la fuerza de gravedad, el efecto giroscópico y el momento en ψ. Sin embargo, el efecto giroscópico solo es observable en aeronaves ligeras y el momento en ψ es cancelado si los motores opuestos giran en dirección contraria.

Por lo tanto, la fuerza de la aeronave f_B está dada por la ecuación 6.10.

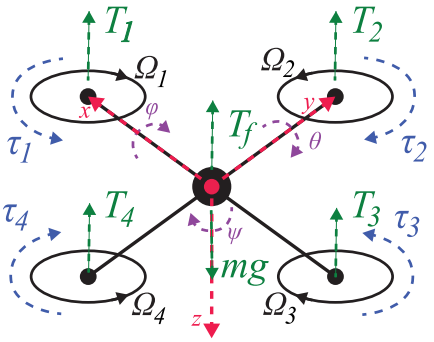


FIGURA 3: Fuerzas del cuadrotor.
Fuente: [91]

$$f_B = mgR^T \cdot \hat{e}_z - f_t\hat{e}_3 + f_w \tag{6.10}$$

En donde $\hat{e}_z$ es el vector unitario en el eje z con respecto al marco de referencia inercial, $\hat{e}_3$ es el vector unitario en el eje z con respecto al cuerpo, g es la aceleración de la gravedad, m es la masa del cuerpo, f_t es el empuje total generado por los rotores y $f_w = \begin{bmatrix} f_{wx} & f_{wy} & f_{wz} \end{bmatrix}^T \in \mathbb{R}^3$ son las fuerzas producidas por el aire en el cuadrotor. El momento externo del cuerpo, m_B está dado por la ecuación 6.11.

$$m_B = \tau_B - g_a + \tau_\omega \tag{6.11}$$

en donde, g_a representa el momento giroscópico causado por la combinación de la rotación de los cuatro motores y la rotación total de la aeronave, $\tau_B = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^T \in \mathbb{R}^3$ son los torques generados por las diferencias de las velocidades de los rotores y $\tau_\omega = \begin{bmatrix} \tau_{\omega x} & \tau_{\omega y} & \tau_{\omega z} \end{bmatrix}^T \in \mathbb{R}^3$ son los torques producidos por el viento. g_a está definido por la ecuación 6.12.

$$g_a = \sum_{i=1}^{4} J_p(\omega_B \times \hat{e}_3)(-1)^{i+1}\Omega_i \tag{6.12}$$

donde J_p es la inercia de cada motor, Ω_i es la velocidad angular del motor i.

6.1.3.2. Actuadores

Las fuerzas y los torques que experimenta el vehículo en función de la velocidad angular de los rotores, basado en [89], se define como:

$$U = \begin{cases} f_t = & b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ \tau_x = & bl(\Omega_3^2 - \Omega_1^2) \\ \tau_y = & bl(\Omega_4^2 - \Omega_2^2) \\ \tau_z = & d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 + \Omega_3^2) \end{cases} \tag{6.13}$$

en donde, l es la distancia entre el rotor y el centro del dron, b es el factor de impulso y d es el factor de arrastre.

Debido a que el sistema de control cuenta con cuatro entradas (torques τ_x, τ_y, τ_z y el empuje f_t) y seis estados (x, y, z, ϕ, θ, ψ), el sistema es considerado sub-actuado, con la imposibilidad de actuar directamente en los ejes x y y.

6.1.3.3. Sensores

Para la estimación de los estados del dron, definidos en la ecuación 6.17, se utilizan distintos sensores, dentro de los cuales, basados en [92] y , algunos son:

- GPS: permite medir la posición y velocidad de un cuerpo en la Tierra mediante el uso del Sistema de Posicionamiento Global (GPS).
- Unidad de medición inercial (IMU): estima la aceleración, orientación y fuerza gravitacional que experimenta un cuerpo a partir de acelerómetros y giróscopos.
- Sonar de altitud: permite medir distancias a través del tiempo que demora una señal ultrasónica en ser emitida y en recibir el eco.
- Barómetro: permite medir la altitud de un cuerpo mediante las variaciones de presión atmosférica
- Magnetómetro: permite estimar la orientación de un cuerpo con respecto al polo norte magnético mediante la fuerza total de dicho campo.
- LiDAR (*light detection and ranging*): permite determinar la distancia desde un cuerpo hasta un objeto o superficie mediante los tiempos de retraso entre un emisor y un detector láser.
- Estimación mediante flujo óptico y/u odometría visual: permite realizar la estimación de la odometría de un cuerpo mediante técnicas basadas en visión computacional como el flujo óptico de imágenes capturadas mediante sensores infrarrojos o cámaras monoculares [93].

6.1.3.4. Efectos Aerodinámicos

El modelo presentado en 6.18 ilustra la dinámica simple de un UAV, sin embargo, cuando se tienen condiciones de vuelo cerca al suelo, o con otros vehículos se pueden presentar los efectos aerodinámicos subsecuentemente relacionados:

- **Efecto suelo (*Ground Effect*):** Cuando un UAV multirrotor despega, aterriza o vuela con baja altitud con respecto a una superficie, la aeronave está sujeta a un mayor empuje causado por la interacción del flujo del aire de las hélices con la superficie. Basados en [94] y en experimentos realizados, en [95] proponen un modelo, ilustrado en la ecuación 6.14 de las contribuciones del impulso G_i en cada motor i:

$$G_i = k_G k_F \left(\frac{r_P}{4h_i}\right)^2 P_i^2 \tag{6.14}$$

  En donde, r_P es el radio de las hélices, P_i es las velocidades angulares, h_i equivale a las altitudes y k_G, k_F son constantes adquiridas en base a datos empíricos.

- **Arrastre (*Drag*):** El arrastre D, o fricción, es una fuerza que se presenta en dirección contraria al movimiento. En [95], se propone un modelo basado en [96] que representa la dinámica:

$$D = -k_D \left(\sum_{i=0}^{3} \frac{2\pi P_i}{60}\right) \dot{x} \tag{6.15}$$

  En donde, $\dot{x}$ es la velocidad lineal de la aeronave y k_D es una constante adquirida experimentalmente.

- **Flujo inducido (*Downwash*):** Es el cambio del flujo de aire desviado por la acción aerodinámica de una aeronave. Cuando dos cuadrotores cruzan caminos a diferentes altitudes, el dron de abajo sufrirá una reducción en la fuerza de sustentación. En [95] modelan este efecto aplicando una fuerza a una partícula puntual, cuya magnitud W depende de las distancias en los ejes x, y y z entre los dos vehículos (δ_x, δ_y, δ_z) y las constantes k_{D_1}, k_{D_2}, k_{D_3}, las cuales fueron encontradas experimentalmente:

$$W = k_{D_1} \left(\frac{r_P}{4\delta_z}\right)^2 exp\left(-\frac{1}{2}\left(\frac{\sqrt{\delta_x^2 + \delta_y^2}}{k_{D_2}\delta_z + k_{D_3}}\right)^2\right) \tag{6.16}$$

6.1.3.5. Modelo No Lineal

Basados en [92], y [97] [89], se define el vector de estados como:

$$X = \begin{bmatrix} \phi & \theta & \psi & p & q & r & u & v & w & x & y & z \end{bmatrix}^T \in \mathbb{R}^{12} \tag{6.17}$$

En donde, u, v y w corresponde a las derivadas de las variables de estado (velocidad) de x, y y z, respectivamente; y p, q y r a las derivadas (velocidad angular) de ϕ, θ, ψ.

La dinámica del cuadrotor en el espacio, al asumir que $\begin{bmatrix}\dot{\phi} & \dot{\theta} & \dot{\psi}\end{bmatrix}^T = \begin{bmatrix}p & q & r\end{bmatrix}^T$, lo cual es verdadero para movimientos de ángulos pequeños, y $\begin{bmatrix}\dot{x} & \dot{y} & \dot{z}\end{bmatrix}^T = \begin{bmatrix}u & v & w\end{bmatrix}^T$, estaría definida por 6.18.

$$X = \begin{cases} \ddot{x} = & -\frac{f_t}{m}\left[sin(\phi)sin(\psi) + cos(\phi)cos(\psi)sin(\theta)\right] \\ \ddot{y} = & -\frac{f_t}{m}\left[cos(\phi)sin(\psi)sin(\theta) - sin(\phi)cos(\psi)\right] \\ \ddot{z} = & g - \frac{f_t}{m}\left[cos(\phi)cos(\theta)\right] \\ \ddot{\phi} = & \frac{I_{yy}-I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{\tau_x}{I_{xx}} \\ \ddot{\theta} = & \frac{I_{zz}-I_{xx}}{I_y}\dot{\phi}\dot{\psi} + \frac{\tau_y}{I_{yy}} \\ \ddot{\psi} = & \frac{I_{xx}-I_{yy}}{I_{zz}}\dot{\phi}\dot{\theta} + \frac{\tau_z}{I_{zz}} \end{cases} \tag{6.18}$$

Adicionalmente, en 6.19 se define un vector de perturbaciones causados, principalmente, por cambios en la dirección y magnitud del viento, las cuales alteran la fuerza y el torque en cada una de las dimensiones de la aeronave.

$$Z = \begin{bmatrix}f_{wx} & f_{wy} & f_{wz} & \tau_{wx} & \tau_{wy} & \tau_{wz}\end{bmatrix}^T \in \mathbb{R}^6 \tag{6.19}$$

6.1.3.6. Modelo de Pequeña Señal

Para pequeñas oscilaciones se puede aproximar a un modelo utilizando la aproximación de pequeña señal o de ángulos pequeños en [98]. De esta manera, la dinámica del sistema se reduce a:

$$f(x,u) = \begin{cases} \dot{\phi} \approx & p + r\theta + q\phi\theta \\ \dot{\theta} \approx & q - r\phi \\ \dot{\psi} \approx & r + q\phi \\ \dot{p} \approx & \frac{I_y - I_z}{I_x}rq + \frac{\tau_x + \tau_{wx}}{I_x} \\ \dot{q} \approx & \frac{I_z - I_x}{I_y}pr + \frac{\tau_y + \tau_{wy}}{I_y} \\ \dot{r} \approx & \frac{I_x - I_y}{I_z}pq + \frac{\tau_z + \tau_{wz}}{I_z} \\ \dot{u} \approx & rv - qw - g\theta + \frac{f_{wx}}{m} \\ \dot{v} \approx & pw - ru + g\phi + \frac{f_{wy}}{m} \\ \dot{w} \approx & qu - pv + g + \frac{f_{wz} - f_t}{m} \\ \dot{x} \approx & w(\phi\psi + \theta) - v(\psi + \phi\theta) + u \\ \dot{y} \approx & v(1 + \phi\theta\psi) - w(\phi - \psi\theta) + u\psi \\ \dot{z} \approx & w - u\theta + u\phi \end{cases} \tag{6.20}$$

6.1.4. Modelo Lineal en Espacio de Estados

La linealización del sistema se realiza a partir de un punto de equilibrio en el vector de entradas $\bar{x}$, obtenido al despejar la función $f(\bar{x}, \bar{u}) = 0$, definiéndose como punto de equilibrio, tanto para el vector de estados, como para el vector de entradas [89]:

$$\bar{X} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{x} & \bar{y} & \bar{z} \end{bmatrix}^T \in \mathbb{R}^{12} \tag{6.21}$$

$$\bar{u} = \begin{bmatrix} mg & 0 & 0 & 0 \end{bmatrix}^T \in \mathbb{R}^4 \tag{6.22}$$

Posteriormente en 6.23, 6.24 y 6.28 se definen las matrices de estados del sistema.

$$A = \left.\frac{\partial f(x,u)}{\partial x}\right|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ g & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{6.23}$$

$$B_u = \left.\frac{\partial f(x,u)}{\partial u}\right|_{\substack{x=\bar{x} \\ u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.24}$$

$$W = \left.\frac{\partial f(x,u,z)}{\partial z}\right|_{\substack{x=\bar{x}\\u=\bar{u}}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{I_z} \\ \frac{1}{m} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.25}$$

Por lo tanto, el modelo lineal en espacio de estados con el vector de perturbaciones, es:

$$\dot{x} = A \cdot x + B_u \cdot u + W \cdot w \tag{6.26}$$

$$\begin{cases} \dot{\phi} = & p \\ \dot{\theta} = & q \\ \dot{\psi} = & r \\ \dot{p} = & \frac{\tau_x + \tau_{wx}}{I_x} \\ \dot{q} = & \frac{\tau_y + \tau_{wy}}{I_y} \\ \dot{r} = & \frac{\tau_z + \tau_{wz}}{I_z} \\ \dot{u} = & -g\theta + \frac{f_{wx}}{m} \\ \dot{v} = & g\phi + \frac{f_{wy}}{m} \\ \dot{w} = & \frac{f_{wz} - f_t}{m} \\ \dot{p} = & u \\ \dot{q} = & v \\ \dot{r} = & w \end{cases} \tag{6.27}$$

Finalmente el vector de salidas se define como:

$$C_y = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \mathbb{R}^{12} \tag{6.28}$$

6.1.5. Simulación en Pybullet

Pybullet es una interfaz de *Python*, de código abierto, que permite acceder a la librería *Bullet Physics SDK*, especializada en simulación de físicas, robótica y aprendizaje por refuerzo profundo [99]. La simulación

de drones en esta plataforma, se realiza a través de *OpenAI Gym* y tomando como base el proyecto *gym-pybullet-drones*[100].

6.1.5.1. Herramientas de Software

- **OpenAI Gym**: es un kit de herramientas desarrollado para realizar principalmente tareas de aprendizaje profundo. Permite diseñar un entorno de simulación y es compatible con distintas librerías de computo numéricas como *Tensorflow*.
- **gym-pybullet-drones**: este proyecto de código abierto es un entorno de *Open AI Gym* basado en *Pybullet*, el cual cuenta con herramientas de simulación para sistemas de uno y múltiples agentes de drones. Así mismo permite el control a bajo nivel de vehículos aéreos y simulación de efectos aerodinámicos como la fricción, cambios de dirección en el viento y el efecto suelo.

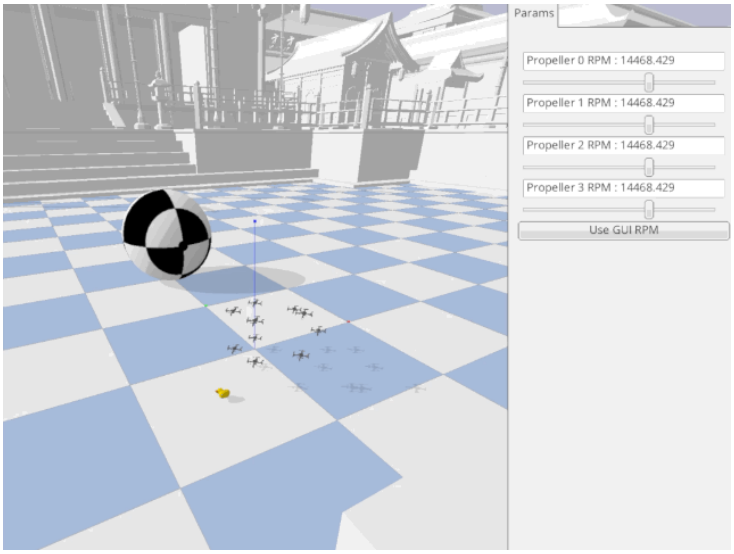


FIGURA 4: Interfaz gráfica de *gym-pybullet-drones*
Fuente: *gym-pybullet-drones*

6.1.5.2. Modelos

- **Hb**: El modelo en cuestión es un cuadrotor genérico con propiedades inerciales del *AscTec Hummingbird* de *Ascending Technologies GmbH* relacionadas en la tabla 2 y mostrado en la figura 5. El dron ofrece una navegación en espacios restringidos. Es pequeño y resistente a choques. El software integrado de la plataforma ofrece interfaces de comando de alto y bajo nivel [97] [101].
- **Cf2x/Cf2p**: El modelo en cuestión es el dron *Bitcraze Craziflie 2.0* [102] en configuración X o $+$, el cual cuenta con las características enunciadas en la Tabla 3 y mostrado en la figura 6. También

[1]Coeficientes de efectos aerodinámicos no implementados en *Pybullet* para el model *Hb*

TABLA 2: Parámetros físicos del dron *AscTec Hummingbird 1*

Parámetro	Descripción	Valor
m_{quad}	Masa	560.0 $[g]$
Tamaño	WxHxD	$540x540x85.5$ $[mm]$
d	Longitud brazo	175.0 $[mm]$
r	Radio del rotor	10.00 $[mm]$
I_{xx}	Momento de inercia sobre el eje x	$3.650e^{-3}$ $[kg * m^2]$
I_{yy}	Momento de inercia sobre el eje y	$3.680e^{-3}$ $[kg * m^2]$
I_{zz}	Momento de inercia sobre el eje z	$7.030e^{-3}$ $[kg * m^2]$
k_F	Coeficiente de impulso	$6.11e-8$ $[N/rpm^2]$
k_M	Coeficiente de torque	$1.5e-9$ $[Nm/rpm^2]$
$C_{D_{xy}}$ [1]	Coeficiente de fricción en eje xy	0
C_{D_z}	Coeficiente de fricción en eje z	0
k_G	Coeficiente de efecto suelo	0
k_{D_1}	Coeficiente de flujo inducido 1	0
k_{D_2}	Coeficiente de flujo inducido 2	0
k_{D_3}	Coeficiente de flujo inducido 3	1

FIGURA 5: Hummingbird 1

denominados como *nanoquads*, tiene un tamaño reducido y un bajo peso, lo que permite que se movilice fácilmente en interiores y realice maniobras agresivas. Así mismo es una plataforma de desarrollo de bajo costo [103].

FIGURA 6: Crazyflie 2.0

TABLA 3: Parámetros físicos del dron *Bitcraze Craziflie 2.0*

Parámetro	Descripción	Valor
m_{quad}	Masa	270 $[g]$
Tamaño	WxHxD	$92x92x29$ $[mm]$
d	Longitud brazo	39.73 $[mm]$
r	Radio del rotor	23.13 $[mm]$
I_{xx}	Momento de inercia sobre el eje x	$1.395e^{-5}$ $[kg * m^2]$
I_{yy}	Momento de inercia sobre el eje y	$1.436e^{-5}$ $[kg * m^2]$
I_{zz}	Momento de inercia sobre el eje z	$2.173e^{-5}$ $[kg * m^2]$
k_F	Coeficiente de impulso	$3.160e^{-10}$ $[N/rpm^2]$
k_M	Coeficiente de torque	$7.940e^{-12}$ $[Nm/rpm^2]$
$C_{D_{xy}}$	Coeficiente de fricción en eje xy	$9.170e^{-7}$
C_{D_z}	Coeficiente de fricción en eje z	$10.31e^{-7}$
k_G	Coeficiente de efecto suelo	11.36
k_{D_1}	Coeficiente de flujo inducido 1	2267
k_{D_2}	Coeficiente de flujo inducido 2	0.160
k_{D_3}	Coeficiente de flujo inducido 3	-0.110

6.1.5.3. Controladores

- **Hb SimplePID:**
  El controlador en cuestión fue diseñado para el modelo *Hb*. Los motores fueron modelados como actuadores de fuerza y no se tuvo en cuenta ningún efecto aerodinámico. Los parámetros del controlador deben ser sintonizados y completados, ya que no se tiene control sobre el eje ψ [104].

TABLA 4: Parámetros del controlador SimplePID

Eje	k_P	k_I	k_D
x	0.1	1^{-4}	0.3
y	0.1	1^{-4}	0.3
z	0.2	1^{-4}	0.4
ϕ	0.3	1^{-4}	0.3
θ	0.3	1^{-4}	0.3
ψ	0.05	1^{-4}	0.5

- **C2fx DSLPID:**

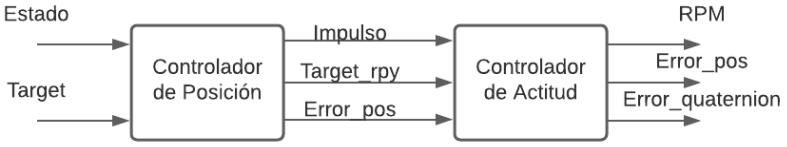
  El control *DSLPID* está basado en un trabajo realizado en Instituto de Estudios Aeroespaciales de la Universidad de Toronto en el Laboratorio de Sistemas Dinámicos (DSL) por SiQi Zhou and James Xu [2] diseñado para el modelo *Cf2x/Cf2p* [95].

  En la Figura 7a se muestra el esquema general del controlador. Debido a que se trabaja con un sistema subactuado, con imposibilidad de control en los ejes x, y; es necesario de un controlador

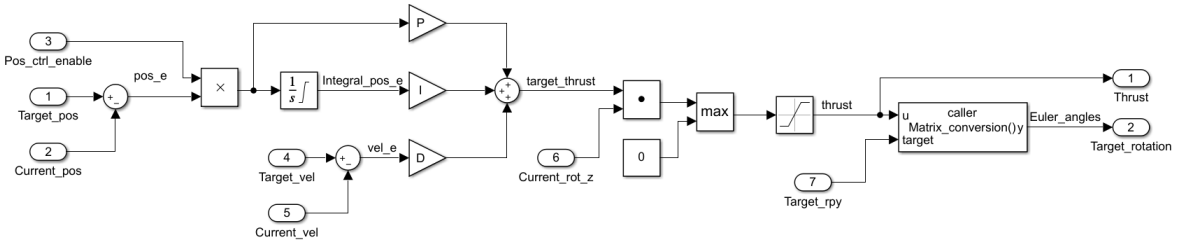
[2]Código disponible en el repositorio de *Github* de *gym-pybullet-drones*

TABLA 5: Parámetros del controlador DSLPID

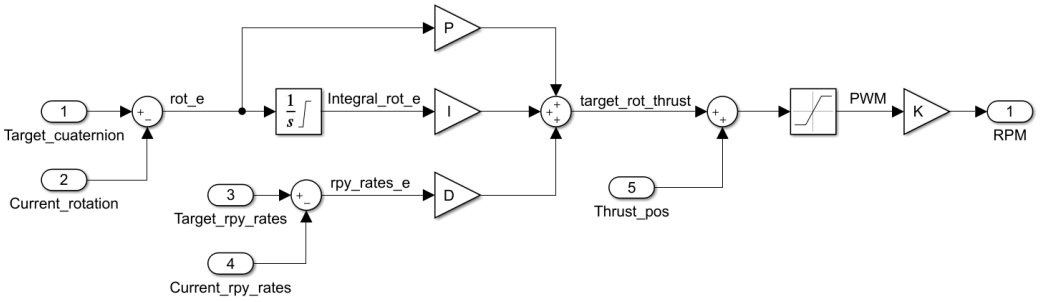
Eje	k_P	k_I	k_D
x	0.4	0.05	0.2
y	0.4	0.05	0.2
z	1.25	0.05	0.5
ϕ	$70e^3$	0	$20e3$
θ	$70e^3$	0	$20e3$
ψ	60^3	500	12^3



(A) Esquema general de Cf2x DSLPIDControl



(B) Controlador de movimientos traslacionales



(C) Controlador de actitud (movimientos rotacionales)

FIGURA 7: Diagrama de bloques de Cf2x DSLPIDControl de Pybullet
Fuente: *gym-pybullet-drones*

de doble lazo, puesto que dichos ejes son manipulables indirectamente mediante ϕ y θ, por lo que el controlador de posición calculará un impulso (que actúa sobre el eje z) y una rotación deseada, necesaria para desplazar lateralmente al dron. Internamente estos controladores son tipo PID con configuración como se muestra en las figuras 7b y 7c para control de posición y de actitud, respectivamente.

6.1.5.4. Motores de Físicas

- PYB: Motor de físicas de Pybullet básico.
- DYN: Actualización con un modelo explícito dinámico.
- PYB_GND: Motor de físicas de Pybullet básico con efecto suelo.
- PYB_DRAG: Motor de físicas de Pybullet básico con arrastre.
- PYB_DW: Motor de físicas de Pybullet básico con flujo inducido.
- $PYB\_GND\_DRAG\_DW$: Motor de físicas de Pybullet básico con efecto suelo, arrastre y flujo inducido.

6.1.5.5. Ambientes de Control

- $BaseAviary$: Clase base de vuelo del dron.
- $CtrlAviary$: Entorno de múltiples drones para aplicaciones de control de posición y actitud a bajo nivel (velocidades angulares de los motores).
- $DynAviary$: Entorno de múltiples drones para aplicaciones de control con impulso y torques deseados.
- $VelocityAviary$: Entorno de múltiples drones para aplicaciones de control de alto nivel de velocidad (planificación y ejecución de trayectorias).
- $VisionAviary$: Entorno de múltiples drones para aplicaciones de control utilizando visión computacional.
- $BaseSingleAgentAviary$: Entorno de un único dron para aplicaciones de aprendizaje por refuerzo.
- $BaseMultiagentAgentAviary$: Entorno de múltiples drones para aplicaciones de aprendizaje por refuerzo

6.2. Control de Sistemas Dinámicos

6.2.1. Estructura de un Sistema de Control

Un sistema de control tiene como objetivo principal reducir el error (e) una señal de salida o variable de estado (y) y la variable de referencia o *set-point* (r) [105]. El $error$ es ingresado a un controlador, el cual calcula el esfuerzo que deben ejercer los actuadores sobre la planta (u). Finalmente los sensores miden el estado de las variables (x) y se realimenta para corregir ese error. Los factores externos no medidos o no tenidos en cuenta en el proceso se modela como perturbaciones (w). La estructura de un sistema de control en lazo cerrado se presenta en la figura 8.

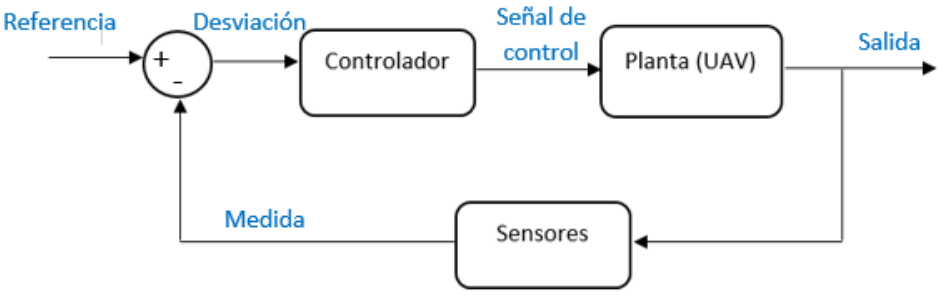


FIGURA 8: Estructura general de control a lazo cerrado

6.2.2. Análisis de Respuesta en Tiempo

El rendimiento de un sistema de control puede ser evaluado basado en mediante su respuesta en el tiempo, por medio del tiempo de establecimiento (t_s), sobrepico (o_s), error en estado estacionario (e_{ss}) y otros parámetros.

- **Tiempo de establecimiento t_s:** se define como el tiempo que le toma a la variable de estado (x) mantenerse dentro del $2\,\%$ de $|x_i - x_f|$, siendo x_i el valor inicial y x_f el valor final.
- **Tiempo de subida t_r:** está definido como el tiempo de transición entre $10\,\%$ y el $90\,\%$ del x_i para la variable de control (x).
- **Sobrepico o_s:** es el punto máximo para que la señal exceda su nivel objetivo. Está definido por la ecuación 6.29

$$Os = \frac{max(y) - y_{ref}}{y_{ref}} \tag{6.29}$$

- **Error en estado estacionario e_{ss}:** es la diferencia entre los valores deseados (w) y reales de la salida (x) de un sistema en el límite del tiempo cuando tiende a infinito. Dado que no es posible evaluar el sistema en $t = \infty$, el error de estado estacionario se representa como un error de tiempo cuadrático integral ($ITSE$) mediante la ecuación 6.30.

$$ITSE = \frac{1}{len(t)} \sum_{i=1}^{len(y)} t * (y_i - u)^2 \tag{6.30}$$

- **Estabilidad:** Un sistema es estable si, al momento de ingresar una señal de amplitud limitada este sistema tiene a su salida una señal igualmente limitada en amplitud. Así mismo, su respuesta natural tiende a cero cuando el tiempo tiende a infinito.

6.2.3. Controlabilidad y Observabilidad

Una forma de establecer si el modelo matemático-físico abstraído en espacio de estados para una planta es útil en el diseño de un sistema de control, es determinando la controlabilidad y observabilidad del

mismo. Si el sistema no es controlable, significa que el modelo no permite ejercer la acción de control sobre el mismo, por otro lado si el sistema no es observable no se podrá determinar el estado dadas las variables que se quieren observar en el modelo.

- **Controlabilidad**
  Un modelo en espacio de estados es controlable si, dado un tiempo inicial t_0 se puede aplicar un vector de control que cambie el estado inicial del sistema a cualquier otro estado, en un tiempo determinado.
- **Observabilidad**
  Un sistema es observable si, es posible determinar el estado del sistema a un determinado tiempo, simplemente observando la salida.

6.2.4. Control PID

La estructura del controlador PID (proporcional, integral, derivativo) ha sido encontrada empíricamente muy útil para resolver problemas de control LTI [106]. Consiste en la combinación de una acción, proporcional, integral y derivativa sobre el error para ejercer la acción de control. La fuunción matemática que describe la salida de control (y) de un PID se relaciona en la ecuación 6.31.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{\delta e(t)}{\delta t} \tag{6.31}$$

Equivalentemente, se establece la función de transferencia en el dominio de Laplace en la ecuación 6.32.

$$u(s) = (Kp + \frac{K_i}{s} + K_d s)e(s) \tag{6.32}$$

6.2.5. Control de Posición Para Drones

Dado que el cuadrotor es un modelo sub-actuado, solo se puede actuar directamente sobre ϕ, θ, ψ y z. Para controlar x y y se necesita un controlador en cascada. En esta arquitectura de control, el controlador x-y recibe las señales de estado actuales del dron y calcula la acción de control necesaria para ϕ y θ según la referencia en x y y. Luego, el controlador de actitud calcula la acción de control correspondiente. Las señales se mezclan junto con las acciones de los controladores de z y ψ y envían como comandos a los motores. En la figura 9 se ilustra el esquema de control de posición.

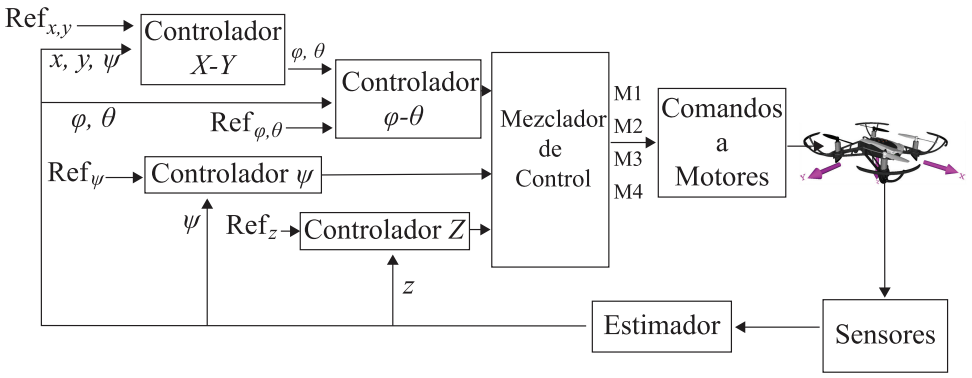


FIGURA 9: Esquema general de control para posición de UAV

6.3. Aprendizaje Profundo

6.3.1. Introducción

El aprendizaje profundo (o *deep learning*) es un subconjunto dentro del área de la inteligencia artificial y el aprendizaje de maquina, inspirado en teorías acerca del funcionamiento del cerebro humano, el cual involucra diferentes arquitecturas y técnicas de entrenamiento de modelos para generar sistemas computacionales que "aprenden" o emulan comportamientos específicos o "inteligentes" a partir de la experiencia, como lo indican en [107]. El objetivo principal de los algoritmos de aprendizaje profundo es lograr una implementación automatizada de un análisis estadístico para encontrar ciertos patrones significativos en grupos de datos, mediante la utilización de distintos modelos redes neuronales profundas, como se observa en la figura 10.

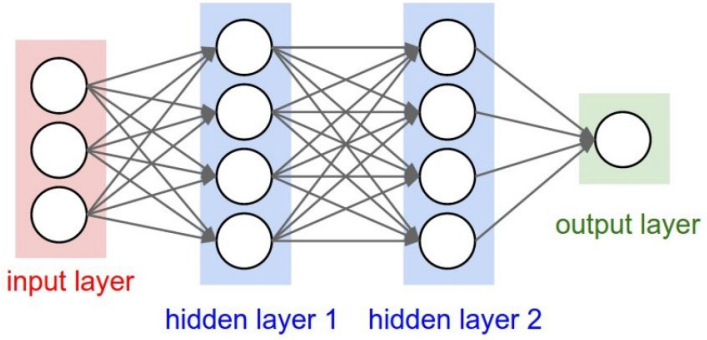


FIGURA 10: Red neuronal con 2 capas ocultas
Fuente: datasciencearth

6.3.2. Redes Neuronales Artificiales

Una red neuronal artificial es la conexión de varias neuronas artificiales dada una arquitectura determinada, empezando por una capa de entrada, las capas ocultas y la capa de salida como se puede observar en la Figura 10. De la misma forma en que se da para una neurona una serie de parámetros, para cada capa de una red neuronal se tienen los pesos en las entradas y la función de activación a la salida de cada neurona. Para entrenar las redes neuronales es necesario tener una función de perdida que indicara el rendimiento de la red en cada época de entrenamiento, un algoritmo de optimización que ayudara a minimizar el error obtenido o maximizar una recompensa, mediante la función de perdida aplicando el algoritmo de *back-propagation* [108]. Finalmente para mejorar el rendimiento de una red dados diferentes entrenamientos, se sintonizan los hiperparámetros, los cuales varían dependiendo del modelo y del algoritmo de optimización que se quiera implementa. La forma en que el ajuste de los hiperparámetros afectan a a la red se describirá en detalle en la Sección 6.3.2.4.

6.3.2.1. Optimizadores

En el proceso para la actualización de los parámetros de una red neuronal, mediante el uso del *back-propagation* es necesario utilizar un optimizador. A nivel general, los optimizadores utilizados para el entrenamiento de redes neuronales funcionan manejando una función objetivo, de la cual se quiere obtener un valor máximo o mínimo, mediante la aplicación de diferentes algoritmos que modifican los parámetros de la red con principios matemáticos, el funcionamiento de estos algoritmos se relaciona a continuación:

- **Descenso del gradiente (SGD):** El descenso del gradiente separa las actualizaciones de los pesos por pequeños grupos de datos llamados batches, determinando la derivada parcial con respecto a cada uno de los parámetros que entran a la red neuronal y actualizando los pesos en sentido negativo a la derivada parcial, como lo indica la ecuación 6.33.

$$w(t+1) = w(t) - \frac{\delta Loss}{\delta w} * \alpha \tag{6.33}$$

  Siendo w(t) los pesos actuales, α la tasa de aprendizaje y Loss la función de perdida.

- **Propagación cuadrática media (RMSprop):** Es un algoritmo donde la actualización de cada peso se realiza al igual que en el descenso del gradiente, con derivadas parciales pero en este caso se dividen entre el cuadrado medio del gradiente, como se puede observar en 6.35.

$$S\delta w(t) = \beta S\delta w(t-1) + (1-\beta)(\frac{\delta Loss}{\delta w})^2 \tag{6.34}$$

$$w(t) := w(t) - \alpha \frac{\delta Loss}{\delta w} \frac{1}{\sqrt{S\delta w(t) + \epsilon}} \tag{6.35}$$

Siendo $Sdw(t)$ el cuadrado medio del gradiente, β y ϵ nuevos hiperparámetros, donde generalmente β tiene un valor de 0.9 y para ϵ igual a 1e-8, con la intención de mejorar el rendimiento solo ajustando α.

- **Estimación de momento adaptativo (Adam):** Es un optimizador que calcula el algoritmo RMSprop y el momento del gradiente, calculando el momento ($V\delta w$) con la ecuación 6.36

$$V\delta w(t) = \beta_2 V\delta w(t-1) + (1-\beta_2)\frac{\delta Loss}{\delta w} \tag{6.36}$$

Actualizando los pesos mediante:

$$w(t) := w(t) - \alpha\frac{V\delta w(t)}{\sqrt{S\delta w(t)+\epsilon}} \tag{6.37}$$

Destacando que el β_2 generalmente tiene un valor de 0.999 y es diferente al β_1 del calculo del RMS.

6.3.2.2. Pérdida

Las redes neuronales son entrenadas mediante un proceso de optimización que requiere una función matemática que calcule el error del modelo. Esta función se conoce como *loss* o pérdida. Para un sistema de regresión, dentro de las funciones más comunes se encuentra:

- **Error absoluto medio (MAE):** También conocida como pérdida L1. Esta función mide el promedio de la magnitud de los errores entre los valores reales y y los valores predichos $\hat{y}$ en un conjunto de datos. En general, el MAE es más robusto frente a valores atípicos, por lo que es sumamente útil cuando el conjunto de datos contiene datos atípicos.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}| \tag{6.38}$$

- **Error cuadrático medio (MSE):** También conocida como pérdida L2. Esta función de pérdida es comúnmente utilizada para problemas de regresión. Mide el promedio del cuadrado de los errores entre los valores reales y y los valores predichos $\hat{y}$ en un conjunto de datos. Esta función aumenta con el cuadrado del error, por lo que castiga fuertemente grandes errores. En general, el MSE es más sencillo de que converja debido a que el gradiente de pérdida es alto para errores grandes y disminuye conforme el error se acerca a cero, por lo que es más preciso al final del entrenamiento, sin embargo, es sensible ante anomalías o valores atípicos.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{6.39}$$

- **Función de Huber:** También conocida como error absoluto medio suave. Básicamente mide la pérdida como MSE o MAE, de acuerdo a si la magnitud del error es superior a un hiperparámetro

δ. La pérdida de Huber se comporta como MSE cuando $\delta \sim 0$ y MAE cuando $\delta \sim \infty$. Esta función es útil cuando se tienen demasiados valores atípicos, pero se desea garantizar una convergencia rápida.

$$L_\delta(y, \hat{y}) = \begin{cases} MSE & para \quad |y_i - \hat{y}_i| \leq \delta, \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & para \quad |y_i - \hat{y}_i| > \delta \end{cases} \tag{6.40}$$

6.3.2.3. Funciones de Activación

Las funciones de activación alteran la salida de una neurona, permitiendo agregar no linealidades al modelo, obteniendo así un mejor ajuste de la red neuronal con respecto a los datos. Dentro de las funciones de activación más comunes se encuentran:

- **Sigmoide:**

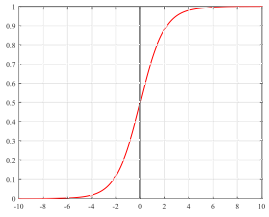


FIGURA 11: Sigmoide

$$f(x) = \frac{1}{1 - e^{-x}} \tag{6.41}$$

- **Tanh:**

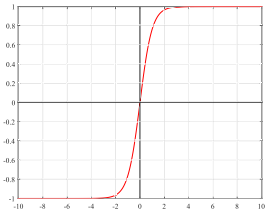


FIGURA 12: Tanh

$$f(x) = tanh(x) \tag{6.42}$$

- **ReLU:**

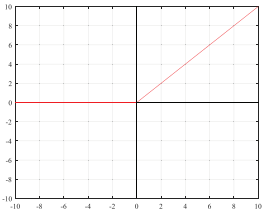


FIGURA 13: ReLU

$$f(x) = max(0, x) \tag{6.43}$$

6.3.2.4. Hiperparámetros

Los hiperparámetros son parámetros que no se ajustan con ayuda del optimizador de la red neuronal y se ajustan con el paso de los entrenamientos, para obtener el mejor rendimiento de la red dadas las diferentes aplicaciones, los hiperparámetros mas comunes en ajustar son:

- Tasa de aprendizaje: Es un numero que determina con que magnitud se actualizaran los pesos de la red neuronal, si se maneja un valor muy grande la red oscilara entre el valor óptimo, si es un valor muy pequeño la red demorara mucho en optimizar la función de perdida.
- Numero de iteraciones: Es el numero de épocas que se ejecutara el optimizador.
- Numero de capas ocultas: Es el numero de capas que tendrá la red sin contar las capas de entrada y salida.

- Número de unidades ocultas: El numero de neuronas utilizadas en cada capa oculta.
- Función de activación: Es una función que se aplica a la salida de cada red neuronal y se explica mas detalladamente en 6.3.2.3
- Optimizador: Algoritmo encargado de disminuir el error o aumentar la utilidad y se explica mas a detalle en 6.3.2.1
- Función de perdida: Es la función encargada de evaluar durante el entrenamiento, la predicción de la red en comparación con los datos tomados de las salidas y si se desea obtener una explicación mas especifica se sugiere dirigirse a 6.3.2.2

6.3.2.5. Auto-Sintonizador Hyperband

Es un algoritmo que ayuda a encontrar la mejor combinación de hiper-parametros para una red neuronal, el cual fue presentado en Abril del 2018 en el articulo [109] y basa su metodología en algoritmo *SuccessiveHalving* para seleccionar los mejores modelos.

- **SuccessiveHalving** : Este algoritmo evalúa el mejor modelo de un grupo de modelos con unos recursos fijos asignados, requiere 2 variables de entrada, n como la cantidad de posibles combinaciones de hiperparametros a probar (configuraciones) y B como la máxima cantidad de dataset, características y tiempo de entrenamiento (recursos), disponibles para asignar a un grupo de configuraciones a probar. El funcionamiento de este algoritmo se basa en asignar un presupuesto de valor $B = \frac{B}{n}$ a un grupo de configuraciones inicial y evaluar las configuraciones con los recursos asignados para tomar la mejor mitad del grupo de configuraciones y eliminar la peor, posteriormente se vuelve a probar las configuraciones seleccionadas actualizando el valor de n y asignando un nuevo presupuesto $B = \frac{B}{n}$, se toma la mejor mitad y se elimina la peor mitad, siguiendo este ciclo hasta llegar a la mejor configuración o modelo.

Sin embargo esta propuesta tiene problemas dado que puede tomar mas del tiempo necesario para encontrar el mejor modelo si se asigna un B demasiado grande o puede calificar mal una configuración si se asigna un B muy pequeño y al momento de evaluar no se consigue que el modelo converja. Es por esto que se propone el algoritmo Hyperband donde se prueban en un principio pocas configuraciones con una enorme cantidad de recursos y gran cantidad de configuraciones con poca cantidad de recursos, garantizando que no se evaluara incorrectamente alguna configuración. Para entender mejor el funcionamiento de esta propuesta se procede a explicar la ejecución del algoritmo presentado en la figura 1

Para este algoritmo se recibe como entrada R que representa la cantidad máxima de recursos a usar y η que ayudara a determinar que porción de configuraciones se van a descartar por cada ejecución del ciclo interno del algoritmo, posteriormente se define el presupuesto máximo a utilizar con la ecuación $B = (S_{max} + 1)R$ y se define S_{max} que representa la cantidad de veces a probar n configuraciones

Algorithm 1 Pseudocódigo Hyperband

Input: R, η, (por defecto $\eta = 3$)
Output: Configuración con el menor error encontrado.
for s $\in \{s_{max}, s_{max} - 1, ..., 0\}$ **do**
$n = \frac{R\eta^s}{R(s+1)}$ $r = Rn^{-s}$
Empieza ejecución de rutina del SuccessiveHalving con r recursos y n configuraciones.
T=$get\_hyperparameter\_configuration(n)$
for i $\in \{0, ..., s\}$ **do**
$n_i = [n\eta^{-i}]$
$r_i = [r\eta^i]$
for t $\in$ T **do**
$L.append(run\_then\_return\_val\_loss(t_i, r_i))$
end for
T=top_k(T,L,[$\frac{n_i}{\eta}$])
end for
end for

con r recursos máximos, luego se calcula la cantidad de configuraciones a probar (n), basándose en el presupuesto y la cantidad máxima de recursos, con la ecuación 6.44

$$n = \frac{B\eta^s}{R(s+1)} \tag{6.44}$$

Luego se calcula r, para distribuir los recursos inversamente a la cantidad de configuraciones calculadas. Empezando con muchas configuraciones y pocos recursos para explorar muchas configuraciones y gastar poco tiempo en descartar los peores modelos.

$$r = R\eta^{-s} \tag{6.45}$$

Posteriormente se toman T configuraciones de tamaño n del espacio de hiper-parámetros definido y se ingresa a un ciclo donde se itera sobre diferentes valores de r_i y n_i para evaluar con la función $run\_then\_return\_val\_loss(t, r_i)$ las configuraciones e ir reduciendo la cantidad de posibles modelos que tiene T con la función $top\_k(T, L, [n_i/\eta])$, que toma las mejores n_i/η configuraciones basándose en las perdidas de cada configuración L. Y así se va iterando sobre diferentes valores de n_i y r_i y reduciendo la cantidad de posibles configuraciones (T) hasta retornar el modelo con el menor error.

6.3.2.6. Sobreajuste

El sobreajuste u *overfitting* es un problema común en el campo del aprendizaje supervisado. Sucede cuando el rendimiento del modelo en la fase de aprendizaje es muy superior a la fase de prueba, es decir, el modelo no generaliza para datos jamás vistos. A continuación se enuncian algunas de las causas más comunes del sobreajuste, según [110]:

- Cuando el dataset de entrenamiento es demasiado pequeño o los datos existentes tienen demasiado ruido o datos atípicos en comparación con los datos representativos.
- La complejidad de la hipótesis o del modelo es demasiado alta. Pueda que se tengan demasiadas características, características redundantes/correlacionadas o el modelo es excesivamente complejo para los datos o es incorrecto.

Así mismo, basados en [110], se enuncian algunos métodos para ayudar a controlar este problema:

- **Early-stopping:** Es probable que, después de un tiempo de entrenamiento, el rendimiento del modelo en la fase de validación no mejore. Este método consiste en detener el entrenamiento cuando la precisión del modelo empiece a disminuir o no mejore lo suficiente.
- **Reducción de la red:** Se propone reducir la complejidad del modelo mediante algoritmos *pre-pruning* y *post-pruning*, que consisten en identificar aquellos parámetros de la red que no alteran la predicción de la red. Para ello se realiza un análisis de los datos, basado distribuciones de ejemplos, identificando las características redundantes, correlacionadas o irrelevantes para el modelo.
- **Expansión de los datos de entrenamiento** Se puede solucionar el problema de *overfitting* añadiendo más datos de entrenamiento, generando ruido aleatorio a los datos, pre-procesar los datos mediante filtros y transformaciones lineales o generar nuevos ejemplos de entrenamiento mediante modelos de *data augmentation*.
- **Regularización:** Este método consiste en minimizar los parámetros de la red que tengan poca influencia en la inferencia del modelo. Para ello se añade un "término de penalización" en la función de costos. En la siguiente Sección se comentan 3 métodos generales de regularización:
  - *L1:* Este método se aplica principalmente para la función de costos MAE y utiliza la teoría de regresión de lazo (*Lasso Regression*)penaliza la función de costos mediante la sumatoria del valor absoluto de todos los pesos de la red:

$$\omega(\omega) = \|w\|_1 = \sum_i |w_i| \tag{6.46}$$

  - *L2 (Weight Decay):* En este método se aplica la distancia euclideana como término de penalización y utiliza la teoría de regresión de crestas: (*Ride Regression*)

$$\Omega(\omega) = \|w\|_2 = \sqrt{\sum_i w_i^2} \tag{6.47}$$

  - *Abandono:* También conocido como *Dropout*. Este método consiste en eliminar conexiones internas dentro de cada neurona de manera aleatoria durante el entrenamiento, esto evita que cada neurona se adapte demasiado. Es una forma eficiente de realizar promedios de

modelos con redes neuronales.

$$\hat{w_{ij}} = \begin{cases} w_{ij} & para \quad c >= P \\ 0 & para \quad c < P \end{cases} \tag{6.48}$$

En donde $\hat{w_{ij}}$ es el valor del peso de la neurona i en la capa j; P es la probabilidad de poda, que es la probabilidad de mantener la neurona activa y c es un número aleatorio que decide si la neurona se activa o no.

6.3.2.7. Capas Convolucionales de 1 Dimensión

La convolución es una operación matemática definida entre 2 señales. En tiempo discreto, está dada por la ecuación 6.49.

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k] \tag{6.49}$$

En donde, h corresponde a un filtro o kernel, x es la señal a operar y y es la señal resultante.

Las capas convolucionales en 1 dimensión tienen como propósito principal el análisis de series de tiempo, donde los kernels son de 1 dimensión y se desplazan a través de la entrada, aplicando el producto punto entre cada elemento del vector de entrada y el kernel para calcular la salida como lo indica la figura 14.

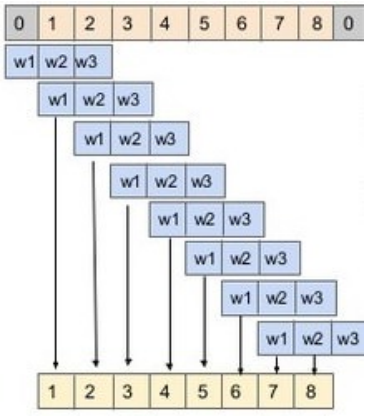


FIGURA 14: Estructura capas convolucionales 1D
Fuente: CNN1D

Donde $w1 = 0, w2 = 1, w3 = 0$ son los pesos del kernel, cuyo valor se va ajustando a medida que se ejecuta el entrenamiento para la extracción de características.

6.3.2.8. Redes de Memoria Largo-Corto Plazo

Las redes de Memoria a Largo-Corto Plazo, hacen parte de las redes recurrentes, las cuales son redes encargadas de solucionar problemas donde es necesario tener en cuenta la secuencia de estados a través del tiempo y no solo un estado de la secuencia para resolver dicho problema, esto lo realizan mediante la realimentación de una neurona de la forma que lo indica la figura 15 .

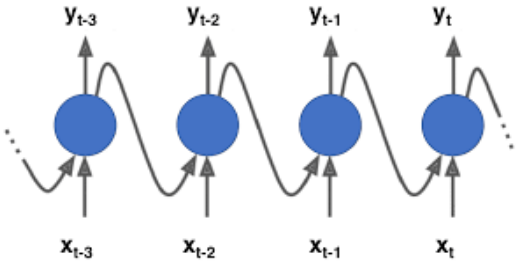


FIGURA 15: Estructura redes recurrentes

Sin embargo dicha realimentación involucra diferentes operaciones matemáticas dependiendo del tipo de unidad recurrente, el tipo de unidad para las redes de Memoria a Largo-Corto Plazo o unidad LSTM, busca relacionar estados que se encuentran ampliamente separados en tiempo para las secuencias de entrenamiento, esto lo realiza mediante la ejecución de operaciones que indica la figura 16.

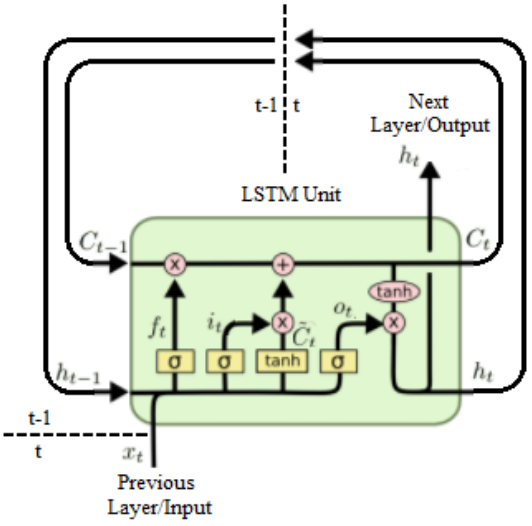


FIGURA 16: Estructura unidades LSTM
Fuente: Toward Science

Donde el ciclo interno involucra la realimentación de la información a retener y el ciclo externo involucra la realimentación de salidas de la red, teniendo esto en cuenta las ecuaciones que describen las operaciones realizadas paso por paso en estas unidades se detallan a continuación.

Donde lo primero que se hace es definir el valor de f_t que indicara que tanto se desea olvidar la información retenida, ya que dicho valor se obtiene de la implementación de una capa neuronal con función sigmoide a la salida que recibe a la entrada la predicción anterior y el estado actual, la cual determinara un valor entre 0 y 1, que establecerá que tanta información a retener, modificando C_t como se indica en la ecuación 6.52.

$$f_t = \sigma(W_f[h_{h-t}, x_t] + b_f) \tag{6.50}$$

Lo siguiente a realizar es obtener el valor de i_t y $\tilde{C}_t$, los cuales seleccionaran la información a añadir, siendo el valor de $\tilde{C}_t$ un vector con información candidata a retener obtenido mediante una capa neuronal con función de activación tangente hiperbólica y el valor de i_t un selector de la información que al multiplicarse indicara que información se agregara, dicho selector se obtiene mediante la implementación de otra capa neuronal con función de activación sigmoide.

$$i_t = \sigma(W_i[h_{h-t}, x_t] + b_i) \tag{6.51}$$

$$\tilde{C}_t = tanh(W_C[h_{h-t}, x_t] + b_C) \tag{6.52}$$

Después se actualiza la información a retener mediante la multiplicación de f_t y la suma de la nueva información a retener.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6.53}$$

Finalmente se genera la salida de la unidad LSTM (O_t) con una capa neuronal que posee función de activación sigmoide, a la cual se le añade la información retenida pasando la misma por una función tangente hiperbolica y multiplicando la salida de dicha función junto con la salida de la unidad (O_t) generando la perdición (h_t).

$$O_t = \sigma(W_o[h_{h-t}, x_t] + b_o) \tag{6.54}$$

$$h_t = O_t * tanh(C_t) \tag{6.55}$$

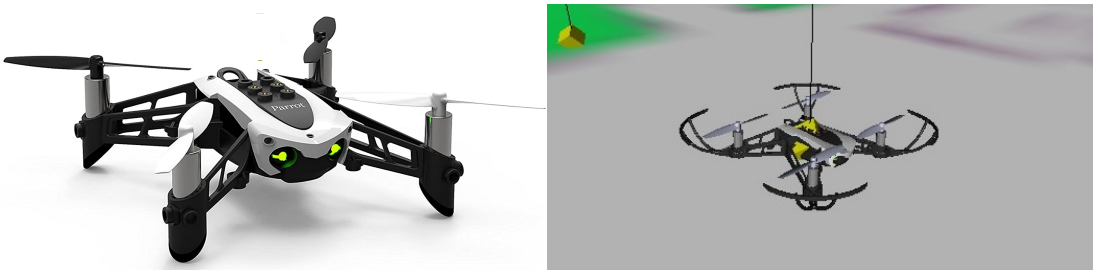
Capítulo 7

Trabajo Preliminar

Para este apartado se presenta un trabajo desarrollado en paralelo a este proyecto, el cual se encuentra bajo revisión para ser publicado en la conferencia IAV (Vehículos Inteligentes Autónomos) de IFAC (International Federation Of Automatic Control), bajo el nombre Çontrol PID Óptimo para el eje ϕ en UAV Cuadrotor basado en PSO (Optimización por Enjambre de Partículas) Multi-Objetivo". El trabajo propone un algoritmo PSO modificado que sintoniza un controlador PID por defecto ofrecido por MATLAB/Simulink, esta sintonización se da con la intención de tomar un controlador base, reduciendo el tiempo de estabilización y el sobre-paso. Para ello, se realiza dicha sintonización con la implementación de diferentes funciones de perdida propuestas dentro de dos funciones multi-objetivo. Dicho trabajo soporta algunas secciones del desarrollo de este proyecto.

7.1. Planteamiento del Problema

Este articulo se centra en sintonizar las constantes de un controlador PID de forma automática mediante el enfoque del PSO Multi-objetivo. Las pruebas e implementacion del PID se realizan en un entorno virtual para un modelo de mini-dron comercial. Siendo seleccionado el mini-dron Parrot Mambo debido a la exactitud del modelo virtual y la documentación abierta para la investigación y el desarrollo. La imagen fisica del dron junto con el modelo virtual son mostrados en la figura 17. Los parámetros del dron son presentados en la tabla 6. Aunque el enfoque experimental se basa en mini-dron Parrot Mambo, el sistema propuesto puede ser aplicado para cualquier modelo de dron. Adicionalmente, el sistema propuesto es independiente de la plataforma robótica utilizada en la evaluaci'pn experimental de este trabajo.



(A) Plataforma Física (B) Modelo Virtual

FIGURA 17: Parrot Mambo
Fuente: Pegasus Aereogrup

TABLA 6: Parámetros del Parrot Mambo

Parámetro	Descripción	Valor
m	Masa	0.063 $[kg]$
Tamaño	WxHxD	$160x78x9.80$ $[mm]$
d	Longitud de Brazo	62.40 $[mm]$
r	Radio del Rotor	33.00 $[mm]$
I_{xx}	Momento de Inercia en Roll$x$	$58.28x10^{-6}$ $[kg * m^2]$
I_{yy}	Momento de Inercia en Pitch y	$71.69x10^{-6}$ $[kg * m^2]$
I_{zz}	Momento de Inercia en Yaw z	$10.00x10^{-6}$ $[kg * m^2]$

7.2. Sistema Propuesto

El objetivo es optimizar el controlador del eje ϕ, implementado como un controlador PID con un sistema anti-windup. Para este objetivo, se uso la arquitectura de control presentada en la figura 18. Este proceso de optimización se llevo acabo usando un enfoque PSO basado en reducir el error de la respuesta transitoria con parámetros como el tiempo de estabilización, sobre-pico y error de estado estacionario.

7.2.1. Sistema de Control General

Debido a que el cuadrotor es un sistema sub-actuado, se necesita un controlador PD-PID para controlar los ejes X e Y. En esta arquitectura de control, el controlador X-Y recibe a la entrada el estado actual de las señales y calcula la acción de control necesario para ϕ y θ basándose en la referencia en X e Y. Posteriormente, el controlador de ϕ y θ calcula la acción de control correspondiente y las señales de salida son combinadas para ser enviadas como comandos de motor al dron.

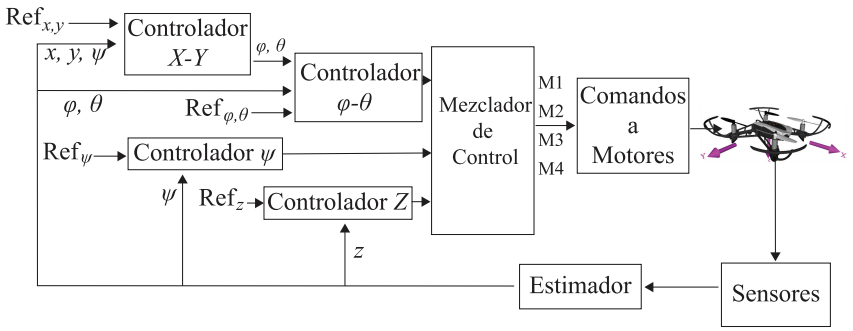


FIGURA 18: Esquema general de control

Las constantes de control inicial son $k_p = 0.01$, $k_i = 0.005$, $k_d = 0.0028$ y $AntiWU = 0.001$. La figura 19 muestra el esquema de control en ϕ aplicado en este caso.

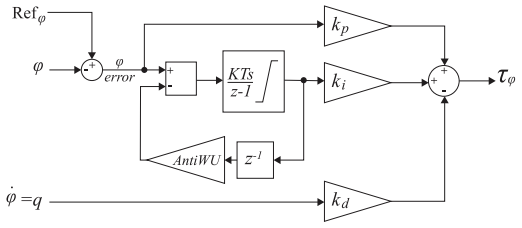


FIGURA 19: Controlador PID ϕ

7.2.2. Optimización por Enjambre de Partículas (PSO)

El algoritmo PSO es un algoritmo bio-inspirado que ha sido efectivo para resolver problemas de optimización no lineales [111]. El algoritmo esta compuesto por un enjambre de partículas que son ubicados en posiciones aleatorias. Posteriormente, la posición de las partículas es actualizada con base en la distancia de la mejor posición del enjambre y la mejor posición histórica personal. Este algoritmo es flexible, siendo modificado para mejorar el desempeño [112].

7.2.2.1. Algoritmo PSO Implementado:

Con un espacio de búsqueda finito, limitado por los vectores $x_{lim} = [x_{min}, x_{max}]$, $x_i = [k_p, k_i, k_d] \in \mathbb{R}^N$ y v_i son definidos como la posicion y velocidad para la particula i respectivamente.

La posición de la partícula es actualizada como lo indica la ecuación 7.1:

$$x_i(t+1) = x_i(t) + \delta t v_i(t) \tag{7.1}$$

Donde, δt es una constante que determina el tiempo de paso para la simulación de partículas.

La velocidad v_i de cada partícula es actualizada basada en dos factores: la mejor posición de cada partícula Pb_i y la mejor posición del enjambre Gb. En cada iteración, la velocidad de cada partícula es actualizada como lo indica la ecuación 7.2:.

$$\begin{aligned} v_i(t+1) =& wv_i(t) + c_1 r_1(Pb_i(t) - x_i(t)) \\ &+ c_2 r_2(Gb(t) - x_i(t)) \end{aligned} \tag{7.2}$$

El peso de inercia (w) indica que tan resistente es un agente modificar su comportamiento. En contraste, los factores cognitivos y sociales (c_1 y c_2) indican respectivamente que tan probable es que un agente siga una tendencia social o una tendencia individual. Finalmente, los valores aleatorios (r_1 y r_2) determinan que tan seguido el agente alterna entre seguir una tendencia individual o social.

En [112], el autor sugiere estrategias para prevenir que el enjambre quede atrapado en un mínimo local, añadiendo un factor de repulsión a la ecuación que actualiza la velocidad 7.2.

$$V_i(t+1) = -K * |\|Cb_i(t) - X_i(t)\|_2 - d| * X_i(t) \tag{7.3}$$

Donde, K es una constante que corresponde a la fuerza de repulsión; d corresponde a la distancia cómoda, la mínima distancia entre las partículas o el radio de colisión.

Adicionalmente, basándose en [113], una versión simplificada de una estrategia de exploración-explotación es implementada, la cual consiste en no siempre ir hacia la mejor posición sino eventualmente tomar decisiones aleatorias. Esta implementación es hecha mediante la ecuación 7.4:

$$V_i(t+1) = \begin{cases} random & for \quad rdn(t) \geq \epsilon \\ V_i(t+1) & for \quad rdn(t) < \epsilon \end{cases} \tag{7.4}$$

Donde, rdn es un numero aleatorio generado en cada iteración y ϵ es la probabilidad de exploración.

Así mismo, se aplica un efecto de reflejo en la velocidad de las partículas en los ejes en los que se salen del espacio de búsqueda.

$$v_{ij}(t+1) = \begin{cases} v_{ij}(t+1) & for \quad x_{j_{min}} < x_{ij}(t+1) < x_{j_{max}} \\ -v_{ij}(t+1) & for \quad otherwise \end{cases} \tag{7.5}$$

Posteriormente, la velocidad y posición de las partículas son limitadas para garantizar la convergencia como se muestra en la ecuación 7.6.

$$v_i(t+1) = \begin{cases} v_{min} & for \quad v_{min} \geq v_i(t+1) \\ v_i(t+1) & for \quad v_{min} < v_i(t+1) < v_{max} \\ v_{max} & for \quad v_i(t+1) \geq v_{max} \end{cases} \tag{7.6}$$

$$x_i(t+1) = \begin{cases} x_{min} & for \quad x_{min} \geq x_i(t+1) \\ x_i(t+1) & for \quad x_{min} < x_i(t+1) < x_{max} \\ x_{max} & for \quad x_i(t+1) \geq x_{max} \end{cases} \tag{7.7}$$

$$v_{lim} = \pm \delta t * (x_{max} - x_{min}) \tag{7.8}$$

Finalmente, la probabilidad de exploración es actualizada usando la ecuación 7.9. La cual asegura que las partículas se mueven más alrededor del espacio definido en el principio y menos a medida que va finalizando la ejecución.

$$\epsilon = (\epsilon_i - \epsilon_o) \left(1 - \left(\frac{t}{\delta t * T}\right)^{\gamma}\right) + \epsilon_o \tag{7.9}$$

Donde, ϵ_i es el valor inicial, ϵ_o es el valor final, γ es el factor de amortiguamiento, t es la iteración actual y T es el numero máximo de iteraciones del algoritmo.

En el Algoritmo 2 se relaciona el pseudocódigo del algoritmo PSO para minimizar una función de costos L.

7.2.3. Funciones de Pérdida

La evaluación del desempeño del sistema propuesto se lleva acabo mediante el uso de 4 funciones de perdida. La primeras dos funciones son el error promedio absoluto (MAE) y el error promedio cuadrático (MSE) evaluando la diferencia entre la posición actual y la posición de referencia del dron como se muestra respectivamente en las figuras (7.10) y (7.11). Las dos ultimas funciones de perdida son F1 y F2, las cuales tienen un enfoque multi-objetivo basado en parámetros de control y el consumo de energía como se muestra en (7.12, 7.13) y (7.14).

$$L_{MAE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y_i}| \tag{7.10}$$

$$L_{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 \tag{7.11}$$

Algorithm 2 Pseudocódigo PSO

Input: N, X_{lim}, T, δt
Output: Gb
for t=0 hasta $\delta t * T$ **do**
  Inicializar enjambre con N partículas;
  Inicializar parámetros de cada partícula dentro de X_{lim};
  for i=1 hasta N **do**
    Actualizar Velocidad de acuerdo con 7.2 y 7.3;
    Limitar Velocidad de acuerdo con 7.6;
    Explotar o Explorar velocidad de acuerdo con 7.4;
    Actualizar posición de acuerdo con 7.1;
    Reflejar velocidad de acuerdo con 7.5;
    Limitar Posición de acuerdo con 7.7;
    if $L(X_i) < L(Pb_i)$ **then**
      $Pb_i = X_i$;
      if $L(X_i) < L(Gb)$ **then**
        $Gb = Pb_i$;
      end if
    end if
  end for
  Actualizar w y ϵ con 7.9;
end for

7.2.3.1. Función Multi-objetivo Para Los Parámetros de Control (F1):

Esta función mide el desempeño del controlador basada en el tiempo de estabilización, el sobre-pico y el error de estado estacionario. El tiempo de estabilización (ts) se define como el tiempo que toma la salida y en estar por debajo del 2 % de $|y_{initial} - y_{final}|$. El sobre-pico ($Os$) se define como lo indica la ecuación 7.12. El error de estado estacionario (Ss) es la diferencia entre el valor deseado y el valor real del sistema cuando el tiempo final tiende al infinito. Debido a que no es posible evaluar el sistema cuando $t = \infty$, el error de estado estacionario es calculado mediante el error cuadrático integral ($ITSE$) con la ecuación 7.13. Por lo tanto, $F1 = [\, ts, Os, ITSE \,]$.

$$Os = \frac{max(y) - y_{ref}}{y_{ref}} \tag{7.12}$$

$$ITSE = \frac{1}{len(t)} \sum_{i=1}^{len(y)} t * (y_i - u)^2 \tag{7.13}$$

7.2.3.2. Función Multi-objetivo Para Parámetros de Control y Esfuerzo de Control (F2):

El esfuerzo de control es la cantidad de energía o potencia necesario para que el controlador cumpla con su propósito, como se muestra en [114]. Para el estado transitorio, el esfuerzo de control Q es definido por la ecuación 7.14. Esta función añade un parámetro de esfuerzo de control a la función multi-objetivo $F1$ y se define como $F2 = [F1\ Q] = [ts\ Os\ ITSE\ Q]$.

$$Q = \frac{1}{t}\int_0^T |u(t) - u_{ss}|\delta t \tag{7.14}$$

Donde, u es la salida del controlador, u_{ss} es el valor de estado estacionario para u y T es el tiempo de integración. Para este problema $u_{ss} = 236.7370$ rpm.

7.3. Simulación y Resultados

La dinamica del modelo y el controlador son implementadas usando el proyecto *MATLAB & Simulink Quadcopter* project [115]. El modelo del dron tiene integrado acelerómetros, giroscopios, un sensor de ultrasonido, un sensor de presión y una cámara RGB.

Para encontrar las constantes del controlador PID ϕ del Parrot Mambo mini-drone, se proponen 2 tipos de experimentos en MATLAB/Simulink:

- Primer experimento, respuesta al escalón con referencia $y_{ref}(t = 5s) = 1m$.
- Segundo experimento, referencia $y_{ref}(t) = 0m$ y perturbaciones en el eje ϕ.

El tiempo de simulación (T) es 50 segundos para ambos experimentos, el tiempo de muestreo (Ts) es 5ms y el dron vuela en $Z = 1m$.

Los parámetros elegidos para desempeñar el algoritmo PSO propuesto son presentados en la tabla 7.

Donde K_{pmin}, K_{pmax}, K_{imin}, K_{imax}, K_{dmin} y K_{dmax} son los limites del espacio de búsqueda, estos son definidos tomando como referencia PID_{base} (el controlador propuesto por defecto en *MATLAB/Simulink*), con constantes $K_{pb} = 0.01$, $K_{ib} = 0.005$ y $K_{db} = 0.0028$. También obteniendo entonces el valor de $d = 0.0028$.

De acuerdo a la función de perdida y el tipo de experimento, se encontraron valores de k_p, k_i y k_d, tal como muestra la tabla 8 y 9.

Después, se evalúa la respuesta al escalón para cada controlador obtenido, las figuras 20 y 21 muestran la señal de respuesta.

TABLA 7: Parámetros generales de ejecución del PSO

Parámetro	Valor	Parámetro	Valor
w	1.00	K	1.00
c_1	0.20	d	$min(PID_{base})$
c_2	0.50	K_{pmin}	$0.5K_{pb}$
ϵ_i	1.00	K_{pmax}	$2K_{pb}$
ϵ_o	0.20	K_{imin}	$0.5K_{ib}$
γ_ϵ	0.50	K_{imax}	$2K_{pb}$
N	50	K_{dmin}	$0.5K_{db}$
T	100	K_{dmax}	$2K_{db}$
dt	0.15		

TABLA 8: Resultados de los controladores PSO propuestos para el experimento 1

Param.	$PID\ MSE$	$PID\ MAE$	$PID\ F2$	$PID\ F1$
k_p	0.148	0.148	0.011	0.046
k_i	0.089	0.089	0.005	0.045
k_d	0.011	0.011	0.003	0.005
$L(y, \hat{y})$	L_{MSE}	L_{MAE}	L_{F2}	L_{F1}
Perdida	$44.45x10^{-6}$	$2.250x10^{-3}$	0.724	0.848

TABLA 9: Resultados de los controladores PSO propuestos para el experimento 2

Param.	$PID\ MSE$	$PID\ MAE$	$PID\ F2$	$PID\ F1$
k_p	0.057	0.056	0.046	0.044
k_i	0.021	0.048	0.045	0.113
k_d	0.010	0.009	0.005	0.010
$L(y, \hat{y})$	L_{MSE}	L_{MAE}	L_{F2}	L_{F1}
Perdida	$6.006x10^{-8}$	$197.7x10^{-6}$	0.738	0.866

TABLA 10: Evaluación controladores del Experimento 1

Param.	$PID\ MSE$	$PID\ MAE$	$PID\ F2$	$PID\ F1$
$y_{t_r}[s]$	0.450	0.450	0.450	0.510
$y_{t_s}[s]$	9.380	9.380	37.85	5.940
$y_{o_s}[\%]$	66	66	97	41
$y_{t_p}[s]$	1.840	1.840	1.990	1.780
$y_{e_s}[\%]$	0.190	0.190	0.270	0.120
$MSEx10^5$	1.668	1.668	1.132	1.131
$Total$	0.550	0.550	0.710	0.260

Con base a estas señales de respuesta se califica cada controlador, estos valores son relacionados en las tablas 10 y 11.

Donde y_{t_r} es el tiempo de subida en el eje y, medido como la diferencia del tiempos en el que la señal alcanza el $10\,\%$ y el $90\,\%$ del valor final, y_{t_s} es el tiempo de estabilización en el eje y, medido bajo el

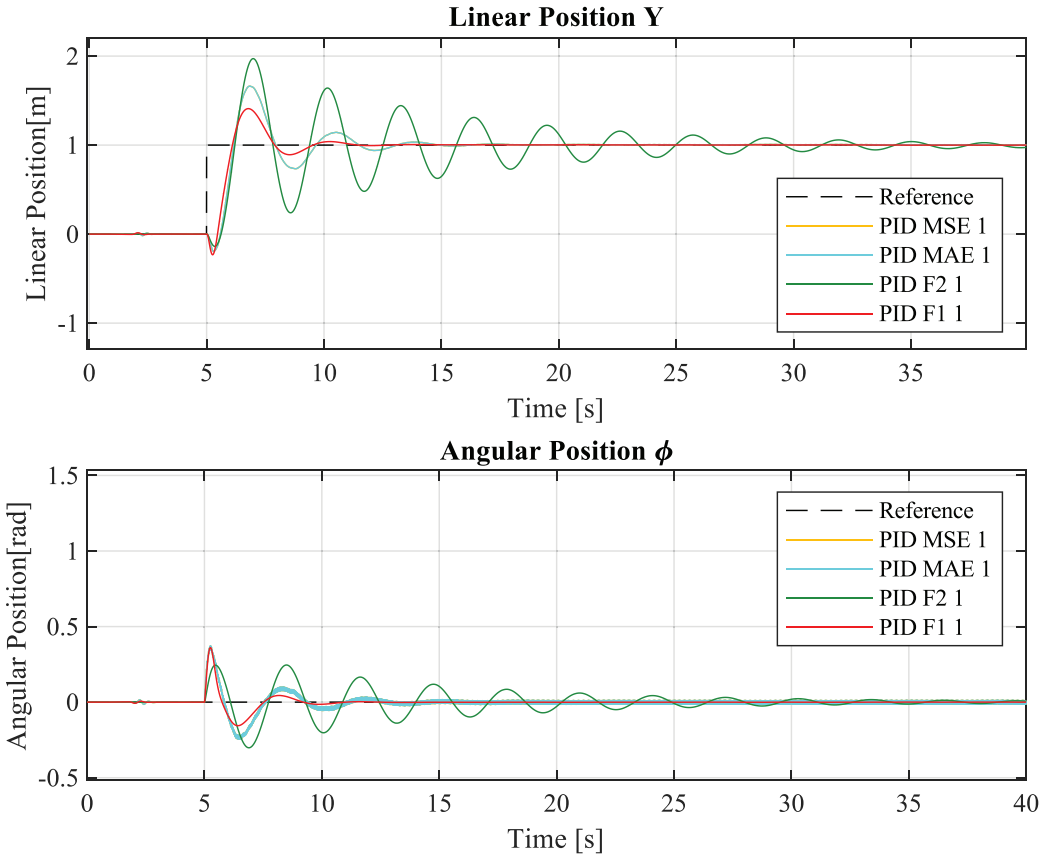


FIGURA 20: Respuesta al escalón/perturbación controladores del primer experimento.

TABLA 11: Evaluación controladores del Experimento 2

Param.	$PID\ MSE$	$PID\ MAE$	$PID\ F2$	$PID\ F1$
$y_{t_r}[s]$	0.500	0.470	0.510	0.370
$y_{t_s}[s]$	9.040	6.820	5.940	7.658
$y_{o_s}[\%]$	59	53	41	63
$y_{t_p}[s]$	1.870	1.770	1.780	1.560
$y_{e_s}[\%]$	0.140	0.130	0.120	0.050
$MSEx10^5$	1.113	1.113	1.113	1.113
$Total$	0.350	0.260	0.260	0.090

criterio del $2\,\%$, Yo_s es el porcentaje de sobre-pico en el eje y, y_{t_p} el tiempo donde el controlador alcanza el valor de y_{o_s}, y_{e_s} el error de estado estacionario en el eje y. MSE es la suma de perdidas MSE entre u_{ss} y cada una de las 4 señales de control, y $Total$ es la suma de puntajes normalizados en cada parámetro de evaluación, la cual es dividida en el numero de parámetros. Siendo necesario resaltar que para normalizar los puntajes, los valores máximos y mínimos son tomados del puntaje obtenido por todos los controladores (incluyendo los puntajes del controlador por defecto, los cuales se muestran en

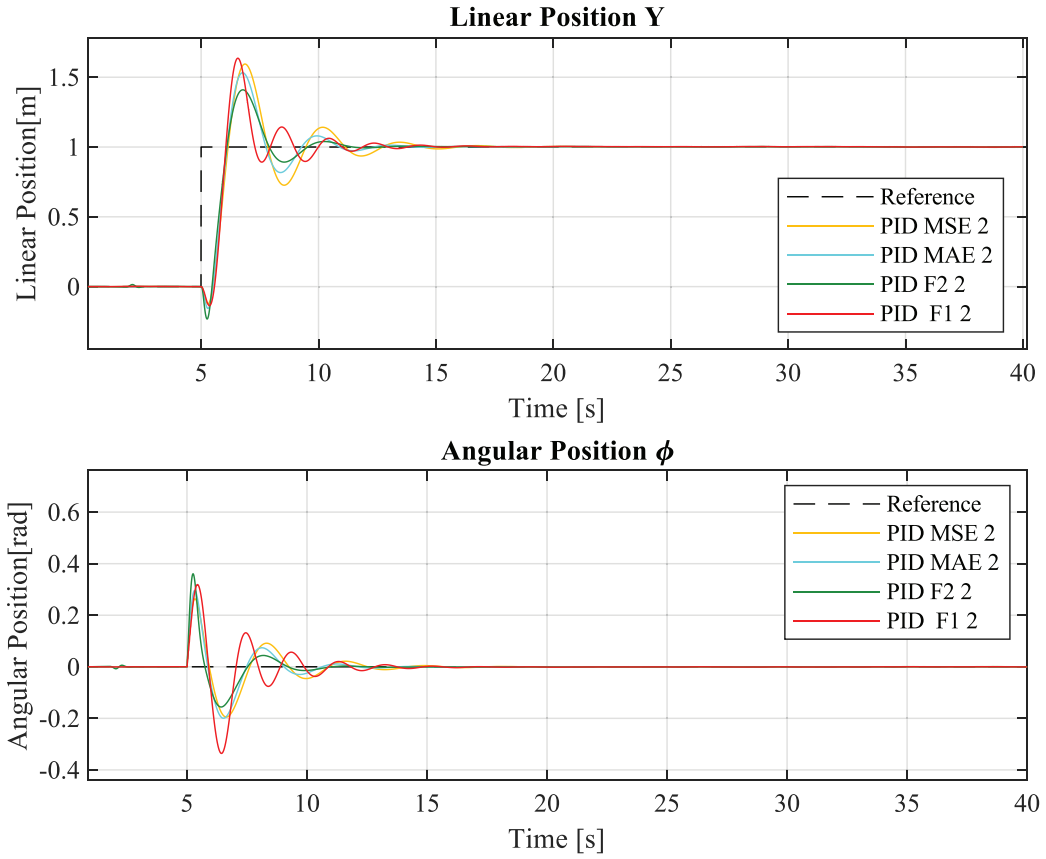


FIGURA 21: Respuesta al escalón/perturbación controladores del segundo experimento.

la tabla 12).

De la fila $Total$, se puede notar que el controlador mejor calificado del primer episodio planteado es el controlador PID F1; por otro lado, el controlador mejor calificado del episodio 2 es el controlador PID F1. Por lo tanto, estos controladores serán comparados con el controlador por defecto propuesto por MATLAB/Simulink. Esta comparación sera ejecutada bajo las mismas condiciones propuestas para evaluar los controladores sintonizados. Los resultados de esta comparación son presentados en la figura 22 y la tabla 12.

Al analizar los resultados, es apreciable que ambos controladores sintonizados obtuvieron un mejor desempeño que el controlador por defecto, donde el mejor controlador es el controlador del segundo episodio, sintonizado con la función de perdida F_1, mejorando (cuando es tomado el controlador por defecto como referencia), y_{t_r} un $9.75\,\%$, y_{t_s} un $80.19\,\%$, y_{o_s} un $36.11\,\%$ y y_{e_s} un $63.04\,\%$. Adicionalmente, cuando la respuesta de control es observada para estos controladores, se puede notar que los mejores controladores sintonizados logran una mejor respuestas sin saturar la salida de control, lo cual asegura

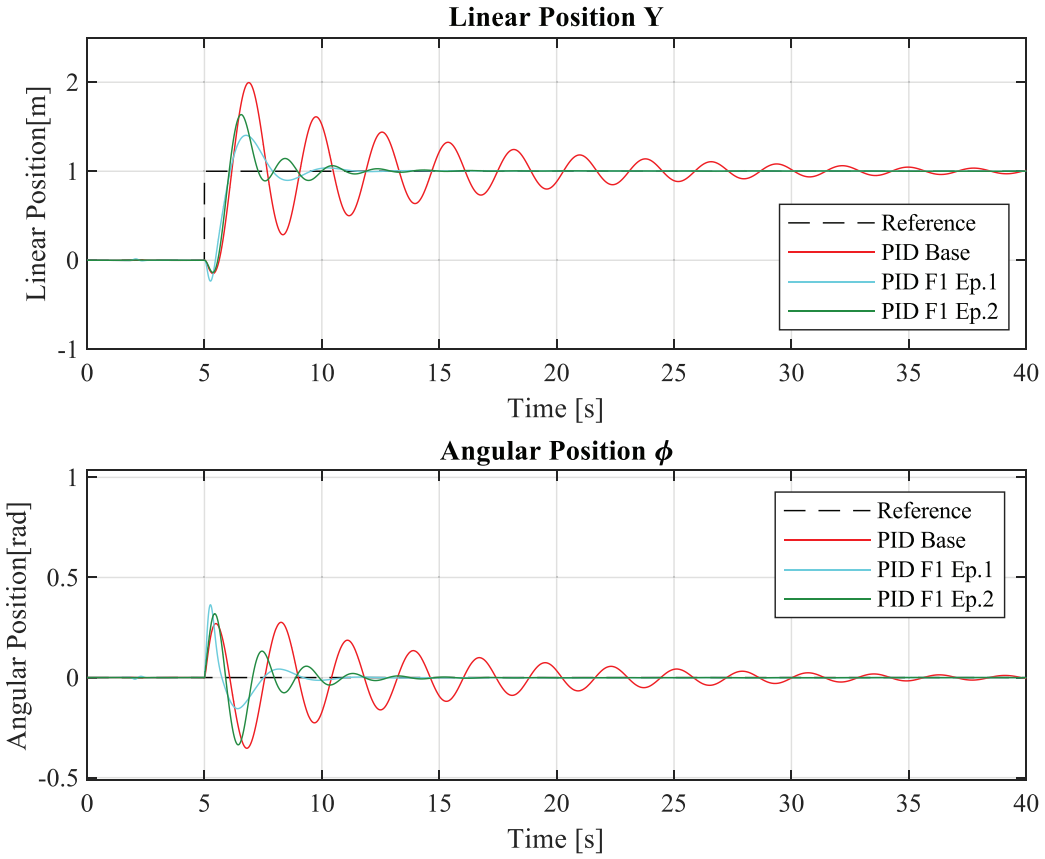


FIGURA 22: Respuesta al escalón de controladores PSO vs controlador base

TABLA 12: Comparación de controladores PSO vs controlador base

Param.	PID_{BASE}	$PID\ F1\ Ep.1$	$PID\ F1\ Ep.2$
k_p	0.010	0.046	0.044
k_i	0.005	0.048	0.113
k_d	0.0028	0.0054	0.0108
$y_{t_r}[s]$	0.410	0.510	0.370
$y_{t_s}[s]$	38.63	5.940	7.650
$y_{o_s}[\%]$	100	41	63
$y_{t_p}[s]$	1.890	1.780	1.500
$y_{e_s}[\%]$	0.150	0.120	0.050
$MSEx10^5$	1.133	1.130	1.131
$Total$	0.550	0.260	0.090

que el sistema no es llevado al limite para obtener una respuesta de mejor calidad en ambos controladores.

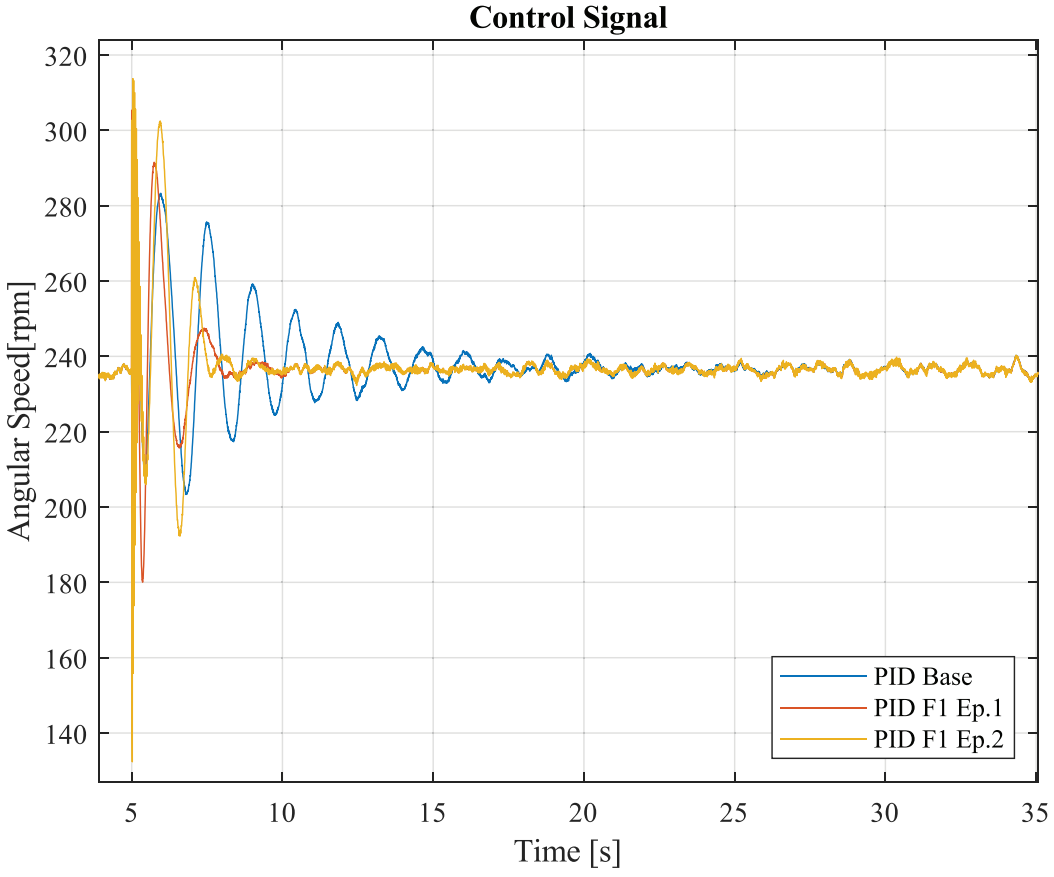


FIGURA 23: Comparación señal de control controlador PSO vs controlador base

Capítulo 8

Diseño y Ejecución del Proyecto

En primer lugar, es pertinente indicar que el presente proyecto es una investigación orientada a explorar la funcionalidad y el rendimiento de técnicas de aprendizaje profundo para el diseño de sistemas de control para vehículos aéreos multi-rotor en modelos virtuales. Para ello, se emplea un estudio basado principalmente en métodos empíricos. Los experimentos a realizar corresponden a observaciones del entorno de simulación que se menciona en la sección 8.2. La estructura metodológica llevada a cabo para la ejecución de este proyecto se presenta en la figura 24:

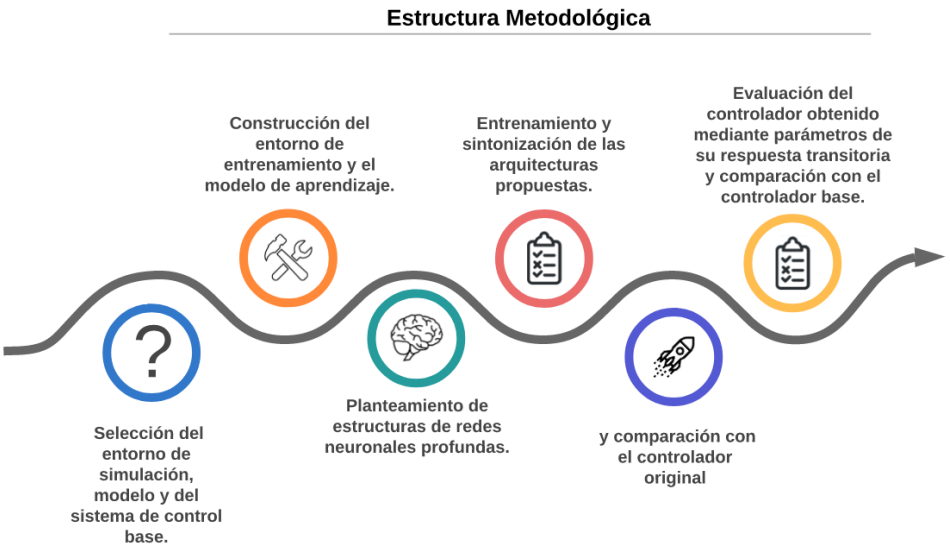


FIGURA 24: Estructura metodológica

8.1. Especificaciones Técnicas

Las características del *hardware* utilizado para el desarrollo de este proyecto se presentan en la tabla 13:

TABLA 13: Especificaciones de *hardware*

Recurso	Descripción
Tarjeta gráfica (GPU)	NVIDIA GeForce GTX 1050 4GB
Procesador (CPU)	Intel(R) Core(TM) i5-8300H
Memoria (RAM)	16384MB

Las características del *software* utilizado para el desarrollo de este proyecto se presentan en la tabla 14:

TABLA 14: Especificaciones de *sotfware*

Recurso	Descripción
Sistema operativo (OS)	Windows 10 pro 64 bits
Lenguaje de programación	Python 3.8.10
Biblioteca de cómputo numérico	Numpy 1.19.5
Biblioteca de visualización	Matplotlib 3.3.4
Biblioteca para el análisis estadístico	Statsmodels 0.12.2
Plataforma de aprendizaje automático	Tensoflow-gpu 2.5.0
Biblioteca de redes neuronales	Keras 2.5.0rc0
Biblioteca para la sintonización de hiperparámetros	Keras_tuner 1.0.3
Simulador	Pybullet 3.1.6
Proyecto base	gym_pybullet_drones 0.6.0
Interfaz	Gym 0.20.0

8.2. Selección del Entorno de Simulación, del Modelo y del Sistema de Control Base

Debido a que los sistemas de control tradicionales son altamente dependientes al modelo, y no todos los simuladores permiten modificar los parámetros del entorno para simular cualquier modelo, la selección del sistema de control base para el entrenamiento del sistema de aprendizaje está condicionado al modelo que permita el simulador.

8.2.1. Entornos de Simulación

Se propone encontrar un entorno de simulación que permita realizar el control de bajo nivel de cada uno de los motores. Para ello se compararán distintos simuladores disponibles gratuitamente, seleccionando aquellos de acuerdo a parámetros como los motores de físicas, el acceso a las variables de estado relacionadas en la ecuación 6.17, control de bajo nivel, sensores compatibles o la documentación disponible.

En la tabla 15 se muestra la comparación de algunos de los simuladores considerados.

TABLA 15: Comparación de simuladores de UAV

Simulador		MATLAB-Simulink	Pybullet	Airsim	Gazebo	Webots
Información General	Desarrollador	Mathworks	Código abierto	Microsoft	OSRF	Cyberbotics Ltd.
	Precio	USD $275 [1]	Gratis	Gratis	Gratis	Gratis
	Plataforma	Linux, MacOS, Windows	Linux, Windows	Linux, Windows	Linux, MacOS, Windows	Linux, MacOS, Windows
Información Técnica	Lenguaje de programación	Matlab	Python	C++, Python, C#, Java	C++	C++
	Extensibilidad	Matlab AddOns and Community AddOns	OpenAI-Gym	Unity Plugin	Plugins (C++)	API, PROTOs, Plugins (C/C++)
	APIs Externas	C, C++. MEX, Java, Python, .NET, COM	Bullet C-API, C++	Python, C++, Boo	C++	C, C++, Python, Java, Matlab, ROS
	Documentación	Si	Si	Si	Si	Si
	Estado de Desarrollo	Activo	Activo	Activo	Activo	Activo
Características	API *Open AI Gym*	Si	Si	No	Si	Si
	Múltiples vehículos *Open AI Gym*	Si	Si	Si	Si	Si
	Control de transmisión en tiempo real	Si	Si	Si	Si	Si
Entorno de Simulación	Motor de Renderizado 3D	OpenGL 3D	TinyRenderer, OpenGL	Unreal Engine 4	OGRE	WREN
	Motor de Físicas	CapSim	Bullet, Nvidia PhysX (experimental)	PhysX	ODE, Bullet, Simbody, DART	Fork of ODE
	Middleware Robótico	ROS	ROS	Sockets, ROS	ROS, Player, Sockets	Sockets, ROS, NaoQI

[1] Valor estimado de licencia 1 año para campos de educación e investigación para un estudiante, sin embargo ya se cuenta con suscripción

Sensores	Odometría	Si	Si	Si	Si	Si
	IMU	Si	No [2]	Si	Si	Si
	Colisión	Si	Si	Si	Si	Si
	GPS	Si	No	Si	Si	Si
	Barómetro	Si	No	Si	Si	No
	Magnetómetro	No	No	Si	Si	Si
	Sonar de altitud	Si	No	No	Si	Si
	Cámaras monoculares	Si	Si	Si	Si	Si
	Cámaras de profundidad	Si	Si	Si	Si	Si
	Escáner Láser 2D	Si	Si	Si	Si	Si
	Escáner Láser 3D	Si	Si[3]	Si	Si	Si
Actuadores	Cadenas cinemáticas genéricas	Si	Si	Si	Si	Si
	Control PWM actuadores	Si	Si	Si	Si	No
	Movimiento controlado por fuerza	Si	Si	Si	Si	Si

De los anteriores simuladores, se decidie que lo más conveniente para el proyecto era *Pybullet*, debido a su motor de renderizado 3D, ya que se podrían realizar simulaciones con bajos recursos. Además de la extensibilidad con el *framework* de *OpenAI Gym*, el cual facilita el vínculo de agentes inteligentes a la simulación. *Pybullet* permite, además, la posibilidad de realizar simulaciones tanto en un entorno ideal, como en uno dinámico con los efectos aerodinámicos mencionados en la sección 6.1.3.4. Finalmente la disponibilidad de la documentación y el acceso libre al código fuente en Python fueron también factores determinantes.

8.3. Construcción del Entorno de Entrenamiento y del Modelo de Aprendizaje

La metodología para la obtención del modelo entrenado a partir del controlador base, se realiza planteando el sistema como un problema de regresión. Para ello se utilizan técnicas de aprendizaje profundo supervisado, por lo cual es necesario primero construir un conjunto de datos de entrenamiento que reflejen el comportamiento del controlador y posteriormente ejecutar el modelo de aprendizaje supervisado propuesto, como se especificara mas adelante en esta sección.

[2]Se puede implementar la simulación de una unidad de movimiento inercial (IMU) mediante la posición global del UAV y un modelo de ruido

[3]Utilizando un *framework* externo

8.3.1. Entorno de Simulación en Pybullet

La implementación del controlador base, así como el sistema de simulación utilizado para adquirir el conjunto de datos y la evaluación del modelo entrenado se realizan en *Pybullet* (ver sección 6.1.5).

Para el proceso de simulación se implementan tres clases *BaseControl*, *Drone* y *PybulletSimDrone*:

- En *BaseControl* se definen las propiedades y los métodos del controlador según su tipo (para este proyecto *DSLPIDControl*, *SimplePIDControl* y *NNControl*). Si el controlador a usar es un PID, se definen los coeficientes de dicho controlador. Si es un controlador neuronal se carga el modelo desde un archivo de *Tensorflow* (.h5). El controlador tiene métodos para calcular la acción de control y procesar los estados actuales.
- En *Drone* se definen las propiedades del dron: modelo, controlador, posición y orientación inicial, nombre y acción. El dron tiene métodos para observar los estados actuales y ejecutar la acción de control.
- En *PybulletSimDrone* se definen las propiedades del entorno de simulación (motor de físicas, frecuencia de simulación, frecuencia de control, duración, ambiente, obstáculos), depuración (interfaz gráfica, grabación de video, salida por consola, gráfica de datos, almacenamiento de datos) y comandos (trayectorias, perturbaciones). Los métodos del simulador son inicializar entorno, establecer drones, inicializar trayectorias, correr simulación, correr un paso de tiempo, registrar datos.

La configuración del entorno de simulación para las pruebas realizadas se muestra en la tabla 16.

TABLA 16: Descripción del entorno de simulación

Motor de Físicas	Pyb
Modelo de dron	$C2fx$
Controlador	$DSLPID$
Frecuencia de simulación	240 Hz
Frecuencia de control	48 Hz
Obstáculos	No
Ambiente de control	*CtrlAviary*

8.4. Selección del Controlador Base Utilizado

De los controladores ya implementados en $Pybullet$ se realizó la comparación de la respuesta de control ante un paso en cada uno de los ejes de control, como se evidencia en 25. Los parámetros de simulación se muestran en la tabla 16.

- SimplePID: Se evalúa la respuesta del controlador con el modelo Hb con una respuesta a un paso unitario con referencias en $x = 0.1m$, $y = 0.1m$, $z = 0.1m$. No se tiene respuesta para ψ debido a que no está implementado en el controlador. La figura 25 muestra los resultados. Los parámetros de la respuesta se relacionan en la tabla 17.

TABLA 17: Parámetros de la respuesta del controlador SimplePID ante un escalón

Medida	x	y	z	ψ
Tiempo de subida (t_r) [s]	1.787	1.787	0.763	-
Tiempo de establecimiento (t_s) [s]	12.687	12.687	11.70	-
Sobrepico (O_s) [%]	39.12	39.12	130.7	-
Error estado estacionario (e_{ss}) [%]	0.301	0.301	1.254	-
$ITSE$	4400	4400	4054	-

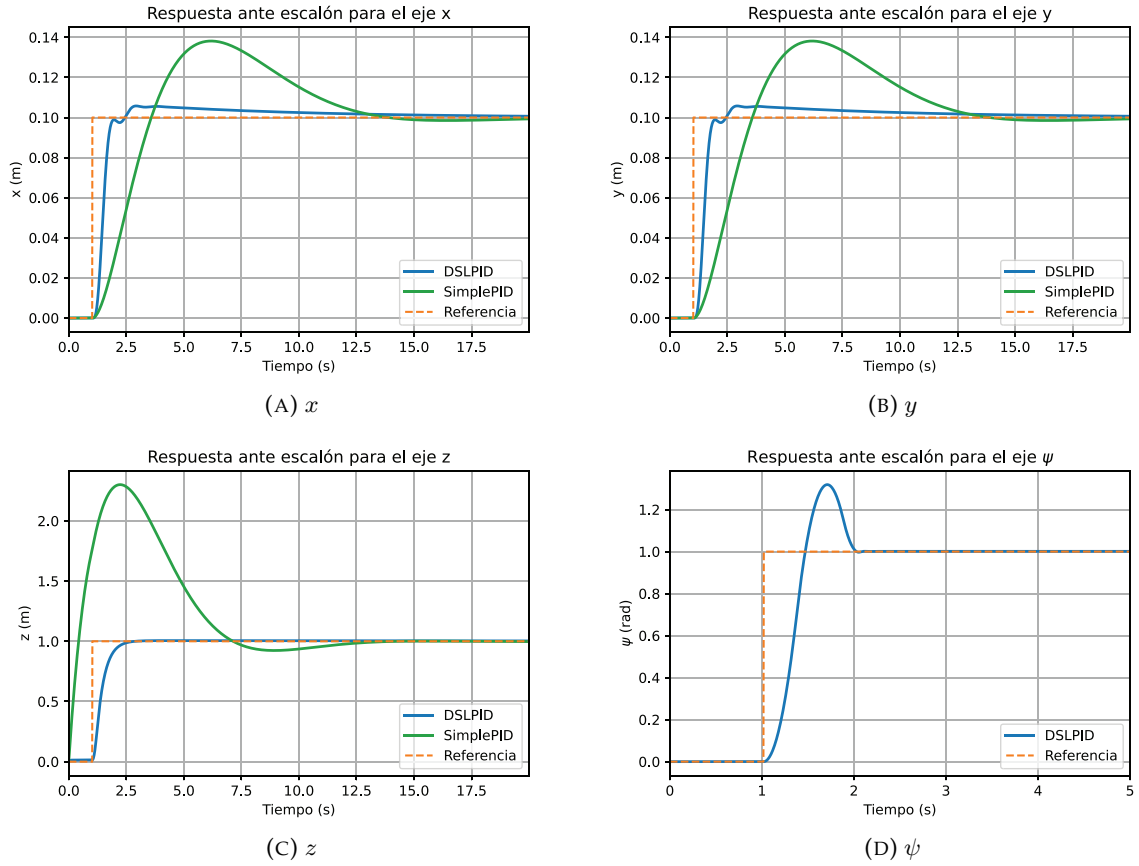
- DSLPID: Se evalúa la respuesta del controlador con el modelo $C2fx$ con una respuesta a un paso unitario con referencias en $x = 0.1m$, $y = 0.1m$, $z = 0.1m$ y $\psi = 1$. La figura 25 muestra los resultados. Los parámetros de la respuesta se relacionan en la tabla 17.

TABLA 18: Parámetros de la respuesta del controlador DSLPID ante un escalón

Medida	x	y	z	ψ
Tiempo de subida (t_r) [s]	0.470	0.470	0.763	0.283
Tiempo de establecimiento (t_s) [s]	11.98	11.98	1.616	1.000
Sobrepico (O_s) [%]	5.098	5.098	0.264	31.73
Error estado estacionario (e_{ss}) [%]	1.630	1.630	0.707	1.199
$ITSE$	1024	1024	332.7	182.4

‘

De los controladores anteriores, se selecciona el controlador $DSLPID$ (ver sección 6.1.5.3), debido a que su implementación se realizó para todos los ejes de control. Así mismo, presenta un mejor rendimiento en términos del tiempo de establecimiento (t_s), sobrepico (o_s) y error en estado estacionario (e_{ss}). Del mismo modo, el modelo de referencia es el $C2fx$ (ver sección 6.1.5.2), debido a que el controlador $DSLPID$ fue diseñado específicamente para ese modelo.



(A) x (B) y

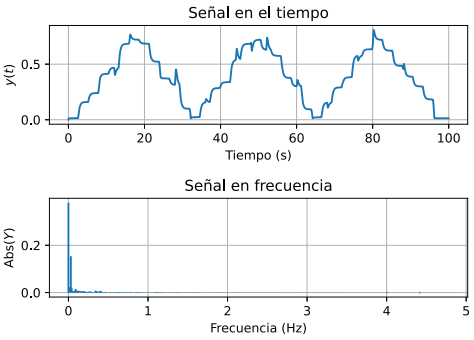
(C) z (D) ψ

FIGURA 25: Respuesta de los controladores ante escalón

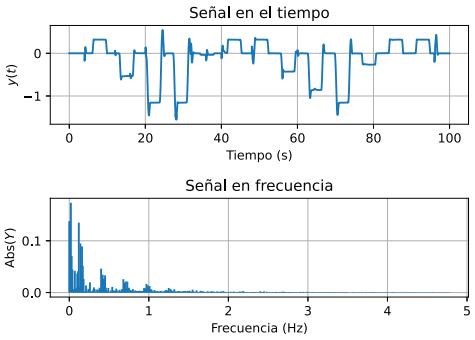
8.4.1. Conjunto de Datos

Para la elaboración del conjunto de datos de entrenamiento se evaluaron distintas trayectorias, buscando maximizar el número de componentes armónicas presentes, la distribución uniforme de los datos y la información en estado transitorio y estable. Para ello, se proponen señales escalonadas, sinusoidales, de ruido, barridos en frecuencia, rampas y nulas con perturbaciones (ver figura 26).

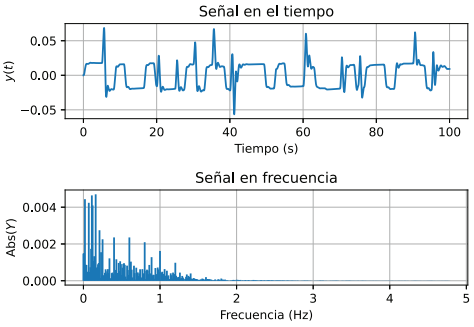
Dentro de las señales que mayores componentes frecuenciales aporta, se encuentra el barrido sinusoidal en frecuencia relacionado en la Figura 26h, sin embargo, dado a que la señal de referencia está acotada dentro de un rango y es una señal periódica que repite valores, esta señal tiende a generar estados repetidos. En el caso de la señal nula con perturbaciones, relacionada en la Figura 26g, contiene un alto rango de frecuencias y permite que el modelo aprenda a controlar a cero, lo cual podría ayudar a la estabilidad del sistema. La señal de ruido dentro de los rangos de operación del controlador, relacionada en la Figura 26f permite explorar un mayor número de estados y ayuda a controlar sobre-ajuste (si es que existe). Finalmente, las señales escalonadas de las Figuras 26a, 26b, 26d, 26c y 26e, suponen una gran importancia dentro del conjunto de datos, debido a que en este tipo de señales se evidencia el estado transitorio y el estado estable del sistema desde distintas referencias.



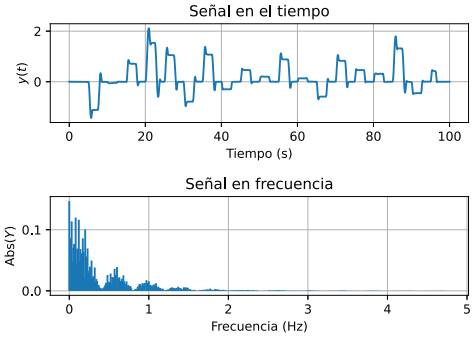
(A) Señal sinusoidal escalonada



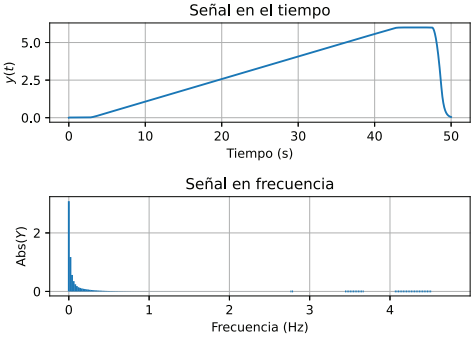
(B) Señal sinusoidal escalonada con retorno a cero



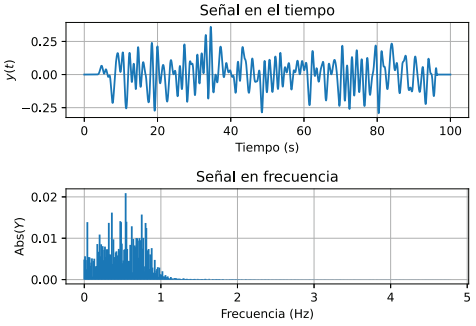
(C) Señal de pasos aleatorios



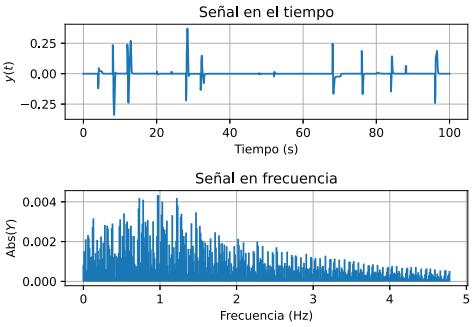
(D) Señal de pasos aleatorios con retorno a cero



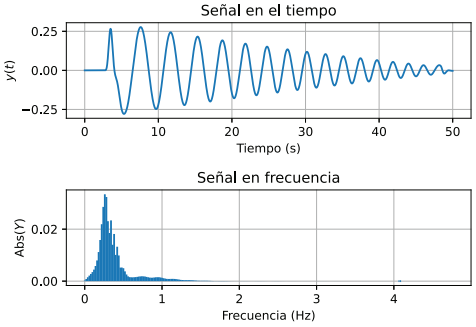
(E) Señal rampa



(F) Señal de ruido de baja frecuencia



(G) Señal nula con perturbaciones



(H) Barrido de frecuencia sinusoidal

FIGURA 26: Señales de referencia para la generación del conjunto de datos de entrenamiento

Adicionalmente se añadieron perturbaciones en los sensores del dron dentro de los límites de estabilidad a todas las señales. Es necesario mencionar que para la correcta operación del modelo entrenado, todas las señales de referencia deben situarse dentro del rango estable del controlador base.

La construcción del conjunto de datos se realiza de forma automática. El tipo de señal y los parámetros propios de la misma son seleccionadas de forma aleatoria. Las señales que el controlador no pueda ejecutar bien o que generen inestabilidad son rechazadas. Así mismo, se rechazan trayectorias con referencias repetidas. Debido a que la posición y orientación es estimada en base a ellos a los estados de p, q, r y vz, las perturbaciones se realizan en estos estados.

8.5. Planteamiento de la Arquitectura de la Red Neuronal Profunda

8.5.1. Ventana de Inferencia

Dado que las señales del sistema son series de tiempo, es necesario definir la cantidad de muestras o estados anteriores necesarios para la inferencia del modelo. Para ello, se plantea un balance entre el tiempo de inferencia, cantidad de memoria y la precisión del modelo. Para definir este valor se propone 2 métodos, un método empírico y otro basado en la correlación parcial.

8.5.1.1. Autocorrelación Parcial (PACF)

La función de autocorrelación (ACF) muestra la correlación entre las observaciones actuales y las observaciones en tiempos anteriores [116]. El coeficiente de autocorrelación ρ_k función indica la correlación entre las observaciones para k muestras anteriores. La función de autocorrelación está definida por la ecuación 8.1.

$$\rho_k = \frac{\sum_{t=1}^{N-K}(x_t - \mu)(x_{t+k} - \mu)}{\sum_{t=1}^{N}(x_t - \mu)^2} \tag{8.1}$$

En donde, x es la señal a operar y μ es el promedio de los datos, definido como $\mu = \sum_{t=1}^{n} \frac{x_t}{n}$.

La función de autocorrelación parcial (PACF) muestra la autocorrelación entre las observaciones actuales y las observaciones en tiempos anteriores, pero teniendo en cuenta los estados intermedios [116]. El coeficiente de autocorrelación parcial se denota como Φ_{kk} y es definido por la ecuación 8.2.

$$\Phi_{kk} = Corr(x_t, x_{t-k} | x_{t-1}, x_{t-2}, ..., x_{t-k+1}) \tag{8.2}$$

Una correlación positiva entre los valores de corriente y de retardo significa que los valores actuales grandes suelen estar asociados con valores de retardo grandes, y una correlación negativa significa que

los valores actuales grandes suelen estar asociados con valores de retardo pequeños. El valor absoluto de una correlación mide la fuerza de la asociación entre los valores actuales y retardo.

Para el cálculo de los coeficientes de autocorrelación parcial, se utiliza la función *pafc* de la librería *statsmodels*. Para determinar el tamaño de la ventana de inferencia del modelo, primero se calculan todos los coeficientes de autocorrelación parcial entre 1 y 100 muestras anteriores para cada una de las señales del dron x, y, z y ψ utilizando todo el conjunto de datos. Posteriormente se calcula el promedio en cada uno de los ejes. Finalmente se calcula el promedio de todos los ejes. El coeficiente que posea mayor correlación es el número de muestras anteriores con el que se espera tener una mejor predicción. En la figura 27 se muestra la gráfica de los valores absolutos de los coeficientes de autocorrelación parcial para cada eje.

TABLA 19: Número de muestras con mayor autocorrelación parcial por eje

Posición	x	y	z	ψ	Media
1	75	2	76	57	2
2	60	63	59	66	75
3	93	5	2	4	58
4	29	96	40	78	64
5	2	21	47	40	67

De los valores obtenidos, los máximos valores se encuentran con entre 1 y 5 y entre 57 y 75 muestras retrasadas. Debido a que la frecuencia de muestreo es alta (240Hz), con entre 1 y 5 estados retrasados, el modelo solo tendría información de entre $4.1ms$ y $20ms$, y, debido a que los tiempos de subida del controlador $DSLPID$ se sitúan entre $283ms$ y $763ms$, el controlador entrenado tendría poca información temporal, por tanto se decide seleccionar un valor de muestras en el intervalo entre 57 y 75 muestras. En cuestiones de implementación, es más sencillo almacenar y procesar datos cuando son potencias de 2, por lo que se selecciona una ventana de 64 muestras (la muestra actual y 63 anteriores), procesándose información de $266ms$, que equivale a la tercera parte del máximo tiempo de subida del controlador $DSLPID$.

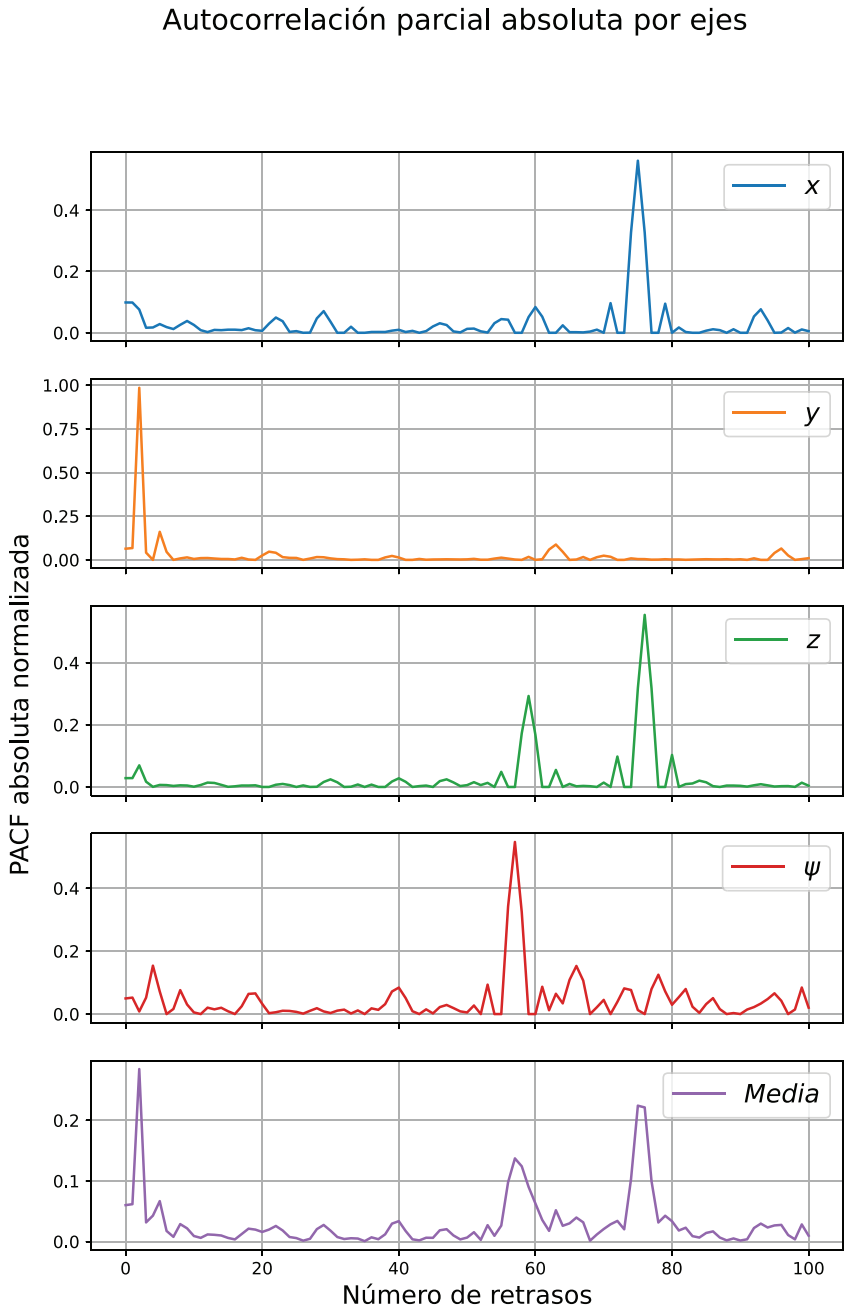


FIGURA 27: Resultados autocorrelación parcial absoluta

8.5.1.2. Método Empírico

Dicho método hace la inferencia del modelo inteligente con un barrido para los valores de la ventana entre 0 y 1000, tomando 100 valores separados logarítmicamente dentro de este rango. Se calcula el tiempo de inferencia promedio T_i y el error cuadrático medio (MSE) para el mejor modelo (ver sección 9.2) con 4096 muestras del conjunto de datos de prueba para cada tamaño de ventana. Se define la función de rendimiento P como se muestra en la ecuación 8.3.

$$P = \frac{1}{MSE * T_i} \tag{8.3}$$

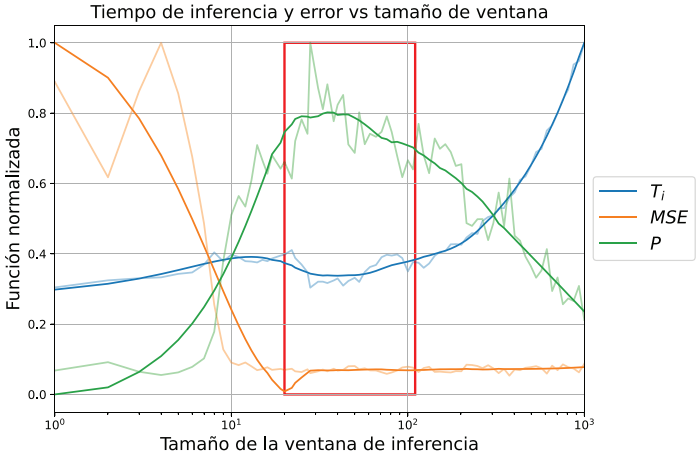


FIGURA 28: Resultados método empírico

Del procedimiento, se obtiene que el número de muestras retrasadas que minimiza el tiempo de inferencia es de 23, el que minimiza el error cuadrático medio es 18 y el que maximiza la función de rendimiento es de 23. Sin embargo, este método fue evaluado solamente teniendo en cuenta el conjunto de datos de prueba, el cual es reducido. Además, debido a la naturaleza estocástica del generador de lotes, se puede presentar ruido en las funciones. Se observa que tanto el tiempo de inferencia, como el error cuadrático medio, en la región entre 20 y 90 muestras, no varían demasiado, por lo que seleccionar un valor entre este rango apenas afecta el rendimiento. De este método se decide optar por mantener el número de 64 muestras obtenido con el método de autocorrelación parcial.

8.5.1.3. Entradas Propuestas

Los estados que se propone ingresar al modelo inteligente está dado por la ecuación 6.17. Se basan en el vector de velocidades entregado por el estimador de estados de *Pybullet*, el cual esta compuesto por las velocidades angulares y lineales con respecto a cada eje. A este vector se le añade la parte integral

(posición) y la parte derivativa (aceleración). De acuerdo con los métodos propuestos para la definición de la ventana de inferencia, se establecen 64 estados a utilizar (la muestra actual y 63 muestras más).

8.5.2. Arquitecturas Planteadas

8.5.2.1. Red Neuronal Artificial (ANN)

En principio se plantea empezar implementando un perceptrón multi-capa, usando varias capas *fully conected* en cascada, dado que esta arquitectura es la mas sencilla y la que menos recursos computacionales consume.

8.5.2.2. Red Neuronal Artificial Realimentada (ANN feedback)

La segunda arquitectura planteada se encuentra basada en la ANN, sin embargo, en este caso se ingresa como entrada adicional los estados anteriores de las predicciones que realiza la red.

8.5.2.3. Memoria a Largo-corto Plazo (LSTM)

El tercer tipo de arquitectura planteada consiste en una estructura compuesta por capas LSTM para el análisis de la secuencia de estados anteriores que se ingresa a la red y posteriormente se conecta la salida de las capas LSTM con capas *fully conected*, las cuales abstraerán información de la respuesta proporcionada por las LSTM para generar la salida de la red.

8.5.2.4. Combinación Memoria a Largo-corto Plazo Con Capas Convolucionales 1d

El cuarto y quinto tipo de arquitectura se plantean dados los resultados encontrados en el área del análisis de secuencias al momento de combinar las redes convolucionales con las redes LSTM, en este caso se hace uso de capas convolucionales de una dimensión (CNN1D), estas capas permiten extraer las características de la señal en frecuencia similar a los filtros lineales. Las diferencias en estos dos tipos de estructuras se especifican a continuación:

- Memoria a largo-corto plazo con capas convolucionales 1d intercaladas (LSTM-CNN1D): Esta arquitectura consiste en implementar al principio bloques en cascada, los cuales están compuestos por una capa LSTM y posteriormente una capa convolucional 1D. la salida de estos bloques posteriormente se conectan a varias capas *fully conected* en cascada que procesaran la salida de la combinación LSTM-CNN1D y generaran la salida con base en las características extraídas.

- Red neuronal convolucional con memoria a largo-corto plazo (CLSTM): Para esta arquitectura se planea implementar al principio bloques compuestos de una capa convolucional en 1D y una capa de *MaxPooling* posteriormente. Esta capa *MaxPooling* permite submuestrear la señal, extrayendo las características mas importantes o con máximo valor. La salida de estos bloques se conecta con capas LSTM para el análisis de las secuencias cuya salida finalmente se conecta en cascada a varias capas *fully conected*, que generaran la salida de la red.

8.6. Entrenamiento y Sintonización de las Arquitecturas Propuestas

8.6.1. Preprocesamiento del Conjunto de Datos

Primero se aplica una normalización media a los datos, tal como se muestra en la ecuación 8.4.

$$x_{norm_i} = \frac{x_i - \mu}{x_{max} - x_{min}} \tag{8.4}$$

En donde, μ es el promedio de la señal; x_{max} y x_{min} son los valores máximos y mínimos de la señal, respectivamente; y x_i es la muestra a normalizar.

Una vez normalizados los datos, se divide el conjunto de datos para las operaciones de entrenamiento, validación y prueba. Los datos de entrenamiento son con los que se ajustan los parámetros del modelo. Los datos de validación permiten identificar si el modelo posee problemas de sobre-ajuste, es decir, si el modelo realmente está generalizando. Los datos de prueba permiten establecer una medida del rendimiento del modelo. La división de los datos se realiza de forma aleatoria, definiéndose la proporción de cada conjunto como se relaciona en la tabla 20.

TABLA 20: División de conjuntos de datos

Conjunto	Número de trayectorias	Porcentaje (%)
Entrenamiento	142	84
Validación	16	10
Prueba	10	6

Se decide realizar un entrenamiento con lotes o *mini-batches*, es decir, se actualizan los parámetros del modelo por pasos con un tamaño reducido del conjunto de datos. Debido a que se maneja una cantidad enorme de estados y secuencias grandes para el entrenamiento de las LSTM, se define una clase tipo *DataGenerator*, la cual permite cargar las muestras del conjunto de datos conforme se vaya necesitando, por lo tanto, se evita que se llene la memoria disponible. El funcionamiento de este generador se basa en tomar los datos normalizados previamente y generar muestras. En cada iteración el generador selecciona una trayectoria aleatoria del tamaño de la ventana de inferencia con índice aleatorio. Finalmente se

ajustan las dimensiones para ser procesadas por el modelo entrenado. El proceso se repite hasta que el lote esté lleno y posteriormente el optimizador ajusta los parámetros con el algoritmo correspondiente.

8.6.2. Llamados de Funciones

Una vez acabe cada episodio de entrenamiento, se llaman funciones de monitoreo y almacenamiento de datos, los cuales se relacionan a continuación:

- Graficado (*Plotting*): Esta función grafica la pérdida de entrenamiento y de validación obtenida obtenida en cada época hasta el momento.
- Punto de control (*ModelCheckpoint*): Esta función ofrecida por *Keras*, permite almacenar el mejor modelo obtenido con base a la función de pérdida definida. Esto con el objetivo de poder recuperar el proceso en caso de que se vea interrumpido por una falla externa.
- Detención anticipada (*EarlyStopping*): Esta función ofrecida por *Keras*, permite hacer seguimiento del error de validación obtenido en cada época de entrenamiento y si después de N épocas el error no mejora, se detiene el entrenamiento con el fin de evitar el sobre-ajuste.

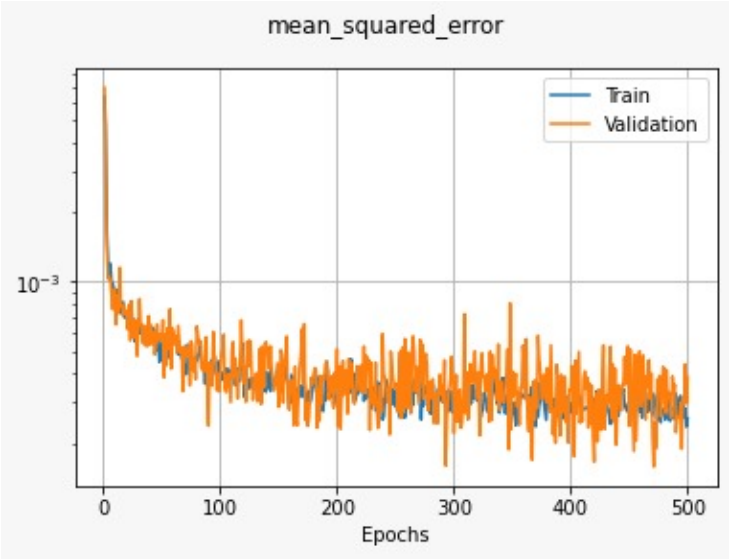


FIGURA 29: Curvas de entrenamiento y validación

8.6.3. Compilado de la Arquitectura

Finalmente se define el modelo de red neuronal a utilizar, la cantidad de capas y los hiperparámetros que se desea usar. Posteriormente se define el optimizador y un valor de recorte de gradiente (*gradient clipping*) para evitar la explosión de pesos en los modelos (esto último principalmente para las

arquitecturas más complejas). Finalmente se define la tasa de aprendizaje (α) a usar con el optimizador y se compila el modelo. Los hiperparámetros con los que se decide ejecutar el algoritmo de entrenamiento se relacionan en la tabla 21.

TABLA 21: Hiperparámetros del entrenamiento

Parámetro	Valor
Pérdida	MSE
Optimizador	Adam
Recorte de gradiente	0.5
Tasa de aprendizaje	$1x10^{-4}$
Tamaño de lote	1024
Tamaño de la ventana de inferencia	64
Número de épocas	500
Pasos de entrenamiento	142
Pasos de validación	16
Paciencia *EarlyStopping*	15

8.6.4. Sintonización de Hiperparámetros

Para las arquitecturas *ANN*, *ANN feedback* y *LSTM*, los hiperparámetros fueron obtenidos a partir de la evaluación del error final en numerosos entrenamientos con hiperparámetros combinados de forma aleatoria, implementando finalmente la arquitectura con menor error de este tipo. Para las arquitecturas *LSTM-CNN* y *CLSTM*, se lleva a cabo una sintonización automática basada en el algoritmo $hyperband$ (ver sección 6.3.2.5). Los hiperparámetros a sintonizar se presentan en la tabla 22.

TABLA 22: Rangos de Exploración Sintonización Automática

Hiperparámetro	**Por defecto**	**Min**	**Max**	**Paso**	**Valores Posibles**
Hidden_LSTMlayers	5	1	5	1	
LSTM_units	256	32	512	32	
Hidden_CNNlayers	5	1	5	1	
Conv1_filters	128	32	512	32	
Conv1_activation	relu				[relu, tanh, sigmoid]
Hidden_FClayers	5	1	5	1	
Hidden_units	128	32	512	32	
Dense_activation	relu				[relu, tanh, sigmoid]
Learning_rate	1e-3	1e-4	0.01	log	

Donde, la columna Paso significará que tanto se puede modificar el valor de un parámetro al ajustarlo, el parámetro $Hidden\_LSTMlayers$ es la cantidad de capas LSTM a utilizar, $LSTM\_units$ es la cantidad de neuronas de cada capa LSTM, $Hidden\_CNNlayers$ es la cantidad de capas convolucionales, $Conv1\_filters$ y $Conv1\_activation$ corresponden a la cantidad de filtros y la función

de activación a utilizar en cada capa convolucional, respectivamente, $Hidden\_FClayers$ la cantidad de capas *fully conected* a utilizar, $Hidden\_units$ y $Dense\_activation$ son la cantidad de neuronas y la función de activación en cada capa *fully conected*, y $Learning\_rate$ es la tasa de aprendizaje utilizada en el modelo. La tasa de aprendizaje posee un paso tipo *log*, lo cual significa que cada ajuste a este parámetro se hará con una separación logarítmica.

Después de definir el modelo se definen los parámetros de ejecución del *hyperband*, los cuales se relacionan en la tabla 23.

TABLA 23: Parámetros del *hyperband*

Parámetro	Valor
Pérdida	MSE
Optimizador	Adam
Ejecuciones por intento	2
Rondas de sintonización	3
Máximo de épocas	150
Tamaño de lote	512
Ventana de inferencia	64
Pasos de entrenamiento	142
Pasos de validación	10
Kernel CNN	3

Una vez finalizado el algoritmo *hyperband*, se procede a entrenar los 5 mejores modelos (evaluados con el error cuadrático medio de validación) con los hiperparámetros de la tabla 21.

8.7. Selección de la Mejor Red

La selección del mejor modelo se realizara con base en el error cuadrático medio con respecto al conjunto de datos de prueba, así como el desempeño en la implementación para realizar el seguimiento de una referencia en comparación con el controlador original. Posteriormente, la mejor red obtenida es puesta a prueba en diferentes trayectorias propuestas junto con el controlador original, evaluando el desempeño de la misma en términos de tiempo de estabilización, sobre-paso y error de estado estacionario de acuerdo a la función F_1 descrita en el trabajo relacionado (ecuación 7.2.3.1). Adicionalmente se evalúa el esfuerzo de control Q definido por la ecuación 7.14.

Capítulo 9

Resultados

9.1. Conjunto de Datos de Entrenamiento

Las trayectorias del conjunto de datos final, aplicado en los entrenamientos de los mejores modelos y obtenido con base a la metodología definida se presenta en la figura 26.

9.1.1. Descripción

Se construye entonces, un conjunto de datos de entrenamiento con un total de 168 trayectorias de duración entre 15 y 100 segundos, distribuidas como se muestra en las tablas 24, 25 y 26.

TABLA 24: Descripción del conjunto de datos

Longitud	168 señales
Tamaño	0.778 GB
Entradas	14 variables [1] [2]
Control	4 variables [3]
Salidas	4 variables [4]
Duración promedio de simulación	76.55 s
Total de muestras	3.068 M

El formato de visualización de una de las trayectorias generadas se muestra en la figura 30.

[1] Variables de entrada = ['x', 'y', 'z', 'p', 'q', 'r', 'vx', 'vy', 'vz', 'wp', 'wq', 'wr', 'ax', 'ay', 'az', 'ap', 'aq', 'ar']
[2] Para *Pybullet*, el marco de referencia en el eje z es positivo hacia arriba, diferente al modelo planteado en 6.1.3.5
[3] Variables de control = ['ux', 'uy', 'uz', 'ur']
[4] Variables de salida = ['RPM0', 'RPM1', 'RPM2', 'RPM3']
[5] Conformado por señales especiales de prueba como señales sinusoidales, rampas, lemniscatas o exponenciales
[6] Valor de altitud con el motor de físicas Pyb

TABLA 25: Longitud de trayectorias del conjunto de datos

Duración	Cantidad
100 s	104 señales
50 s	38 señales
20 s	22 señales
15 s	4 señales

TABLA 26: Distribución de señales en el conjunto de datos

Señal	x		y		z		ψ		Total	
	N	%	N	%	N	%	N	%	N	%
Señal nula con perturbaciones	46	27.4	56	33.3	44	26.2	99	58.9	245	36.5
Pasos aleatorios con retorno a cero	61	36.3	63	37.5	66	39.3	53	31.5	243	36.2
Pasos aleatorios sin retorno a cero	23	13.7	17	10.1	20	11.9	3	1.8	63	9.4
Señal de ruido	7	4.2	8	4.8	19	11.3	4	2.4	38	5.7
Sinusoidal escalonada sin retorno a cero	15	8.9	6	3.6	11	6.5	4	2.4	36	5.4
Sinusoidal escalonada con retorno a cero	9	5.4	10	6.0	3	1.8	1	0.6	23	3.4
Barrido en frecuencia	2	1.2	2	1.2	0	0.0	0	0.0	4	0.6
Otras [5]	5	3.0	6	3.6	5	3.0	4	2.4	20	3.0

TABLA 27: Descripción de señales de control del conjunto de datos

Medida	r_x [m]	r_y [m]	r_z [m]	r_ψ [rad]
Mínimo	-0.800	-0.800	0.000	$-\pi$
Promedio	$6.439x10^3$	$3.417x10^3$	0.878	0.012
Máximo	-0.800	-0.800	7.98 [6]	π
Desviación estándar	0.105	0.101	0.764	0.441

9.1.2. Correlación Entre Variables de Control y Referencias

Con la intención de analizar si dentro del conjunto de datos el controlador realiza un adecuado seguimiento de las señales de referencia propuestas, se evalúa la correlación entre los estados de posición ingresados al controlador y las señales de referencia en posición. La correlación de las variables indica si la relación de cambio de las dos variables es de manera constante, sin embargo no se pueden hacer afirmaciones de causa o efecto. Un valor cercano a 1 indicará que los estados del controlador se relacionan linealmente a la referencia. Por otro lado, un valor cercano a 0 indica que la

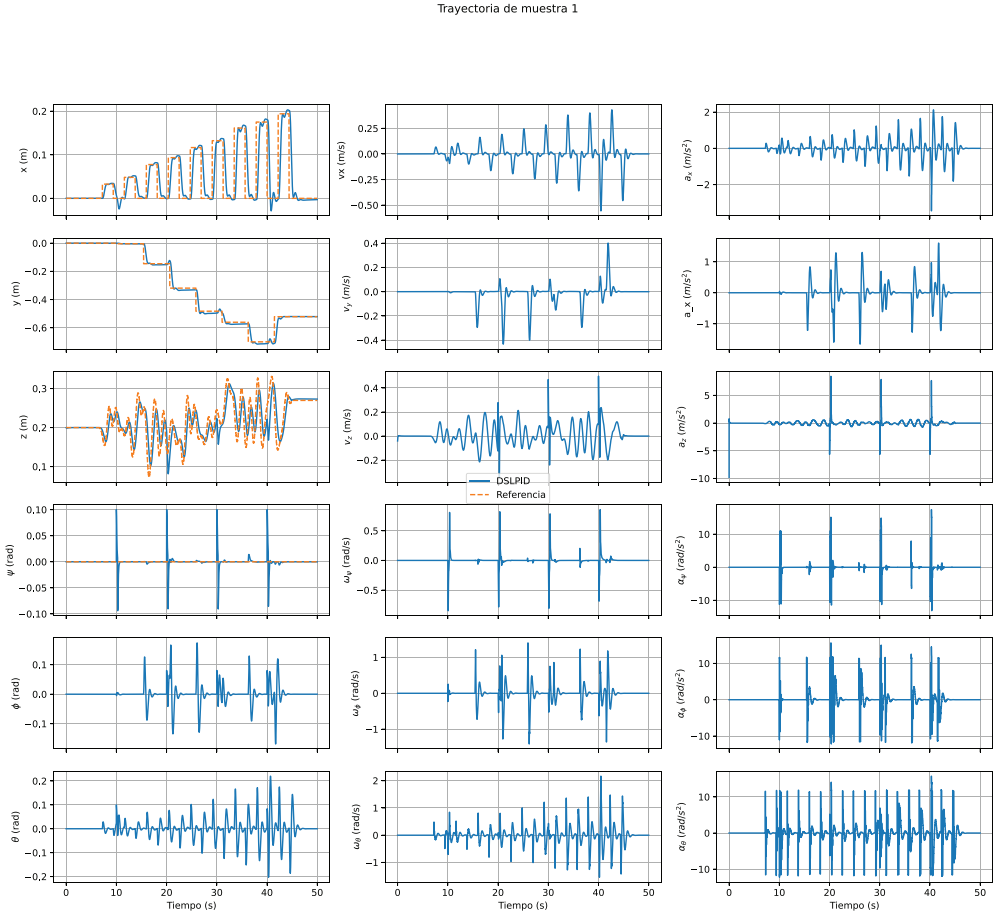


FIGURA 30: Visualización de trayectorias

señal de control y la señal de referencia no están relacionadas, lo que permitiría saber si el conjunto de datos posee trayectorias donde el controlador se desestabiliza. Esta medición permite identificar qué tan bien son seguidas las señales de referencia en el sistema. A partir de esta evaluación se genera la matriz presentada en la figura 31.

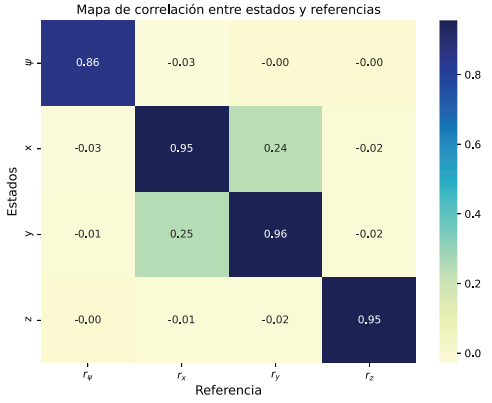


FIGURA 31: Matriz de correlación posición/referencia.

Donde ψ, x, y, z representan los estados en posición y r_ψ, r_x, r_y y r_z las referencias de ψ, x, y y z, respectivamente. Para esta matriz se puede observar que, dentro del conjunto de datos construido, los estados del controlador se relacionan en dirección y magnitud con las referencias propuestas por encima de 0.86. Adicionalmente en la matriz se identifica un valor mínimo de correlación entre la posición en y con la referencia en x, que tiene un valor similar para la posición en x y la referencia en y. Esto se puede interpretar como un movimiento generado en x o y dada la inclinación del dron al seguir un cambio en la referencia del eje contrario debido a la naturaleza acoplada de estos dos ejes.

9.1.3. Diagrama de Caja

Posteriormente se evalúa la distribución del error en el conjunto de datos para cada uno de los ejes, calculando el error como la resta entre la posición y la referencia ($e_y = y - r_y$). Se realizo el diagrama de caja presentado en las figuras 32 y 33.

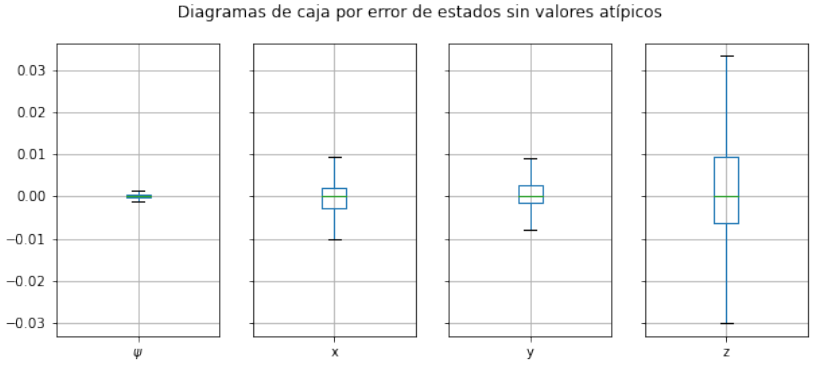


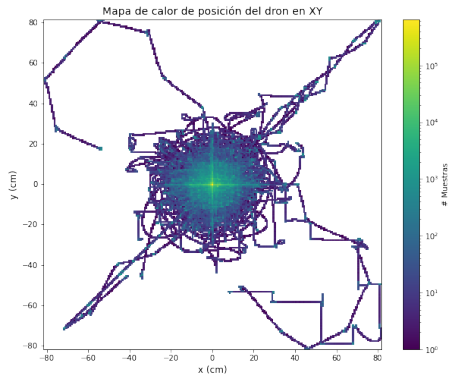
FIGURA 32: Diagrama sin valores atípicos

FIGURA 33: Diagrama con valores atípicos

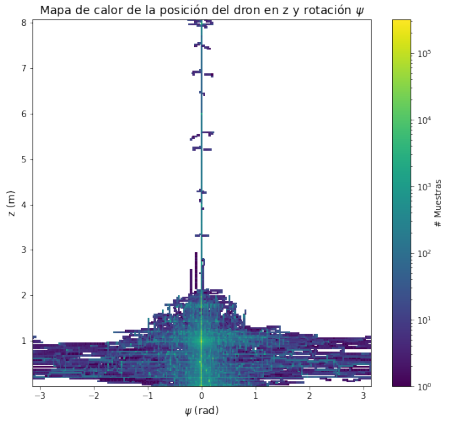
Se puede observar que los valores atípicos en ψ son muy grandes. Para este caso,los valores atípicos se encontrarían en el error al momento que se cambia la magnitud de la referencia y el controlador empieza a responder. Conforme pasa el tiempo, el error debería tender a cero para el controlador. Dicho proceso se refleja en el diagrama de caja con la media $\mu = 0$ para x, y, z y ψ. También se puede notar en la gráfica sin valores atípicos que la distribución de datos del error no es simétrica con respecto a los otras variables. Los valores típicos de error que presenta el controlador se dan en un rango de -0.10 y 0.10 metros aproximadamente para los ejes x e y, el error en Z se encuentra en un rango de -0.30 y 0.32 metros, el rango de error en yaw es de $-1x10^{-3}$ y $1x10^{-3}$ radianes. Debido a esto se encuentra que la respuesta en x e y del controlador es mejor que la respuesta en z para las señales de referencia ingresadas. Lo anterior también se debe a que los rangos de referencia son mayores en z que en algún otro eje (ver Tabla 27). Adicionalmente se confirma que en todas las trayectorias el controlador realiza un seguimiento sin error de estado estacionario en ningún eje, al encontrar la distribución de los datos centrada en 0 ($\mu = 0$ para x, y, z y ψ).

9.1.4. Visualización de Muestras

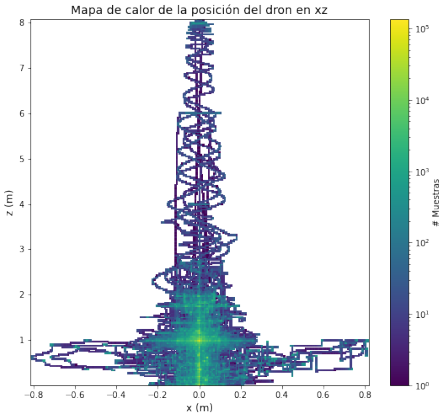
Con el fin de verificar que la distribución de señales de control se encuentren distribuidos de manera uniforme, se crearon los mapas de calor presentados en la figura 34. Esto permite analizar los rangos en los que el controlador tiene mayor información. Los rangos en los que se presentan menor cantidad de muestras son fundamentales para determinar si el controlador entrenado ha generalizado el comportamiento del controlador base.



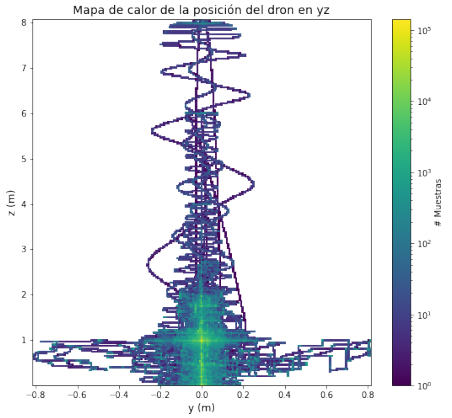
(A) Plano XY



(B) Plano ΨZ



(C) Plano XZ



(D) Plano YZ

FIGURA 34: Mapa de calor de las muestras en el conjunto de datos

En la figura 34a se observa que las trayectorias exploradas en x e y se encuentran principalmente dentro del rango de -0.20 y 0.20 metros, manejando una distribución aproximadamente gaussiana. Los estados con más muestras se encuentran en el vector X_0= $\begin{bmatrix} \bar{x}=0 & \bar{y}=0 & \bar{z}=1 & \bar{\psi}=0 \end{bmatrix}^T$, que corresponden con el punto de operación definido en la ecuación 6.21. De las figuras 34c, 34d y 34b se observa que la mayoría de estados se exploran a menor altura, manejando una distribución no uniforme. Así mismo se evidencian los límites definidos por la tabla 27.

9.1.5. Análisis en Frecuencia

Finalmente, se plantea realizar un análisis en frecuencia mediante la transformada de Fourier con la intención de evaluar las componentes frecuenciales presentes en el conjunto de datos. Para esto se

calcula la transformada de Fourier para cada una de las trayectorias para la posición del controlador y la referencia en los ejes x, y, z y ψ. Posteriormente calcula el promedio por eje y se grafica la respuesta en la figura 35.

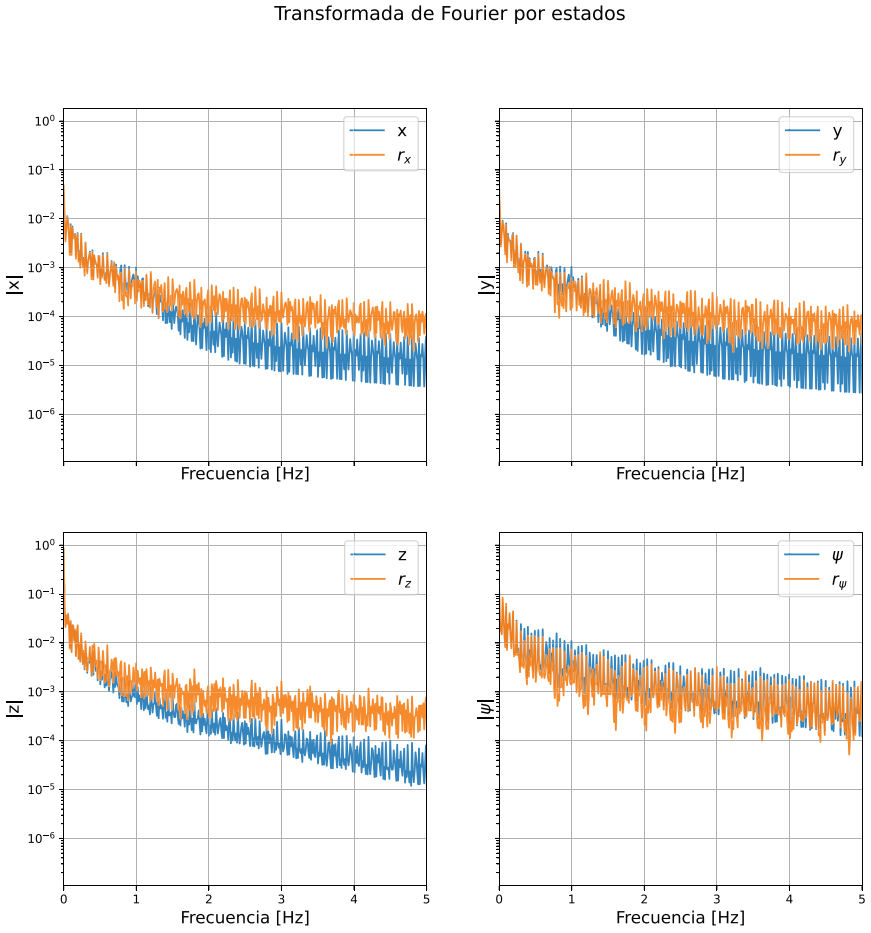


FIGURA 35: Respuesta en Fourier de las trayectorias

De la figura 35 se determina que la respuesta en frecuencia del sistema controlador-UAV responde correctamente para bajas frecuencias (menores a 0.9Hz). De las trayectorias implementadas se obtiene una respuesta más semejante en el eje ψ.

9.2. Arquitectura del Modelo Inteligente

Las arquitecturas de los mejores modelos entrenados con los procesos de sintonización manual o automática, junto con sus evaluaciones se presentan en esta sección. La primera evaluación se realiza

en términos pérdida (MSE) evaluada con el conjunto de datos de prueba, al tiempo de inferencia (T_i) y a la función de evaluación (F_1) (ver sección 7.2.3.1). La función F_1 fue calculada para una señal escalón en los estados de control ($r_x = 0.1m$, $r_y = 0.1m$, $r_z = 0.1m$, $r_\psi = 0.1rad$). La función F_1 mide el el tiempo de estabilización (T_s), el sobrepico (O_s), el error en estado estacionario como una función ITSE. La función se calcula para cada uno de los ejes y se normaliza con respecto al controlador base. Finalmente se promedian cada uno de los ejes y se obtiene una medición. Si $F_1 > 1$ significa que el controlador evaluado responde en promedio en todos sus ejes peor en términos de T_s, O_s e ITSE que el controlador base; si $F_1 < 1$, el controlador evaluado es mejor; y si $F_1 = 1$, el rendimiento es similar. Los modelos fueron implementados en *Pybullet* con los parámetros relacionados en la tabla 16.

9.2.1. Red Neuronal Artificial (*ANN*)

De las arquitecturas exploradas para la red neuronal artificial (*ANN*), se relacionan las 5 mejores en las tablas 42, 43, 44, 45, y 46. La sintonización de los parámetros se realizó manualmente mediante la selección de parámetros aleatorios predefinidos. En la figura 36 se comparan las señales para una respuesta al escalón con los controladores implementados en *Pybullet* para la arquitectura *ANN*.

Así mismo, las funciones de evaluación para cada una de las arquitecturas se muestran en la tabla 28.

TABLA 28: Evaluación ANN

ANN	**MSE**	T_i **[s]**	F_1
1	$2.22x10^{-3}$	$93x10^{-3}$	5.145
2	$3.04x10^{-3}$	$89x10^{-3}$	3.031
3	$3.31x10^{-3}$	$87x10^{-3}$	1.439
4	$3.11x10^{-3}$	$113x10^{-3}$	5.830
5	$3.12x10^{-3}$	$93x10^{-3}$	1.480

Por tanto, debido a su bajo tiempo de inferencia y buen rendimiento en la función F_1, se decide que el mejor modelo entrenado es la ANN 3 (ver tabla 44).

9.2.2. Red Neuronal Artificial Realimentada (*ANN Feedback*)

La estructura de la red neuronal artificial realimentada (*ANN Feedback*) es similar a la *ANN*. La diferencia radica en las entradas, dado que a esta red se ingresan no solo los estados establecidos en la metodología sino los estados anteriores de la salida también. La sintonización de los parámetros se realizó manualmente mediante la selección de parámetros aleatorios predefinidos. Las 5 mejores arquitecturas exploradas relacionan en las tablas 47, 48, 49, 50, y 51. En la figura 37 se comparan las señales para una respuesta al escalón con los controladores implementados en *Pybullet*.

Así mismo, las funciones de evaluación para cada una de las arquitecturas se muestran en la tabla 29.

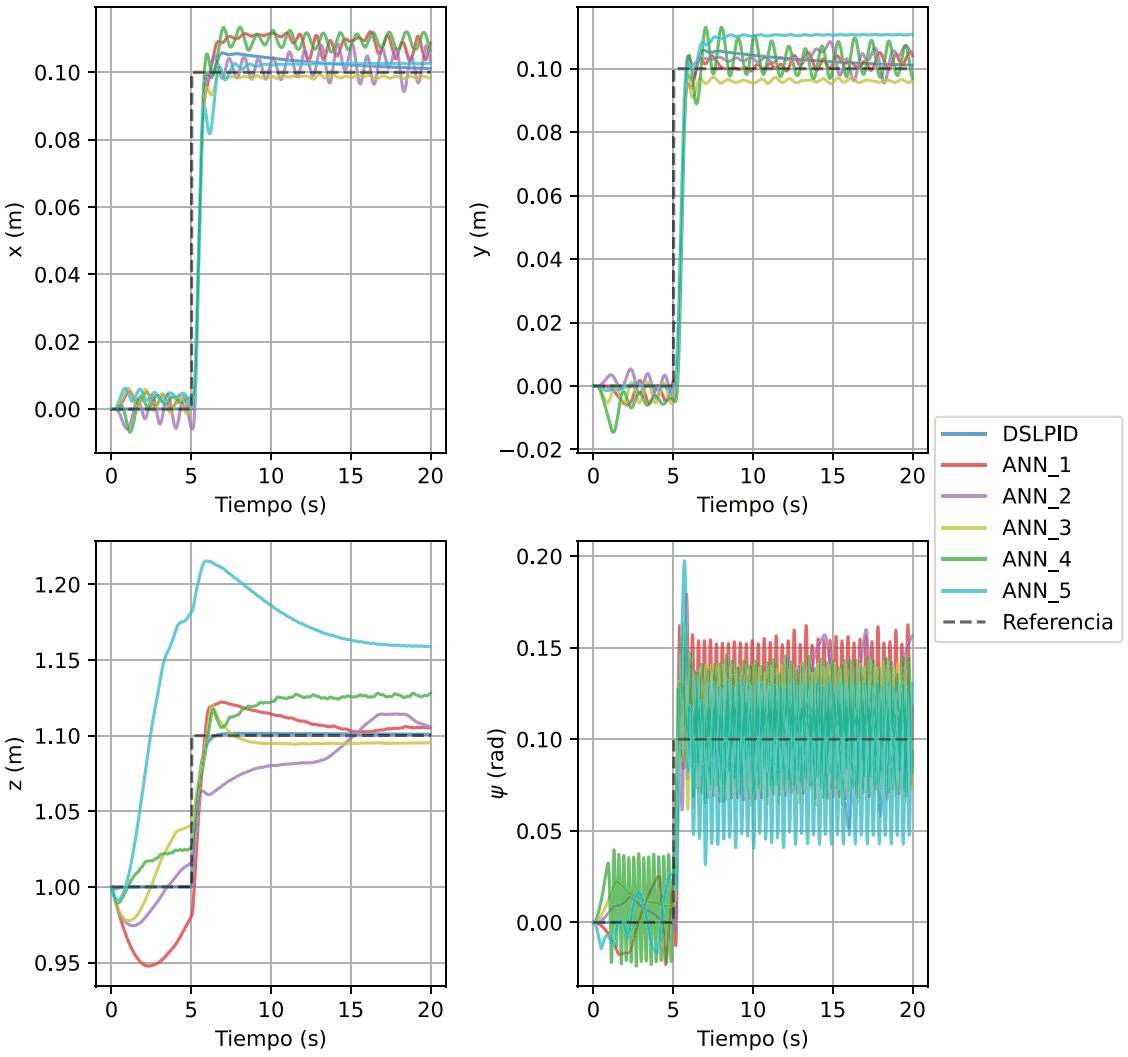


FIGURA 36: Respuesta de controladores ANN ante un escalón

Debido a que ninguna arquitectura demostró estabilidad en la implementación en el simulador, se descarta esta arquitectura para futuras evaluaciones. Sin embargo, se señala que esta arquitectura obtuvo menor pérdida con los datos de prueba, por lo que muestra una mejor capacidad de adaptación. Lo anterior conlleva a que se exploren modelos secuenciales como las redes LSTM.

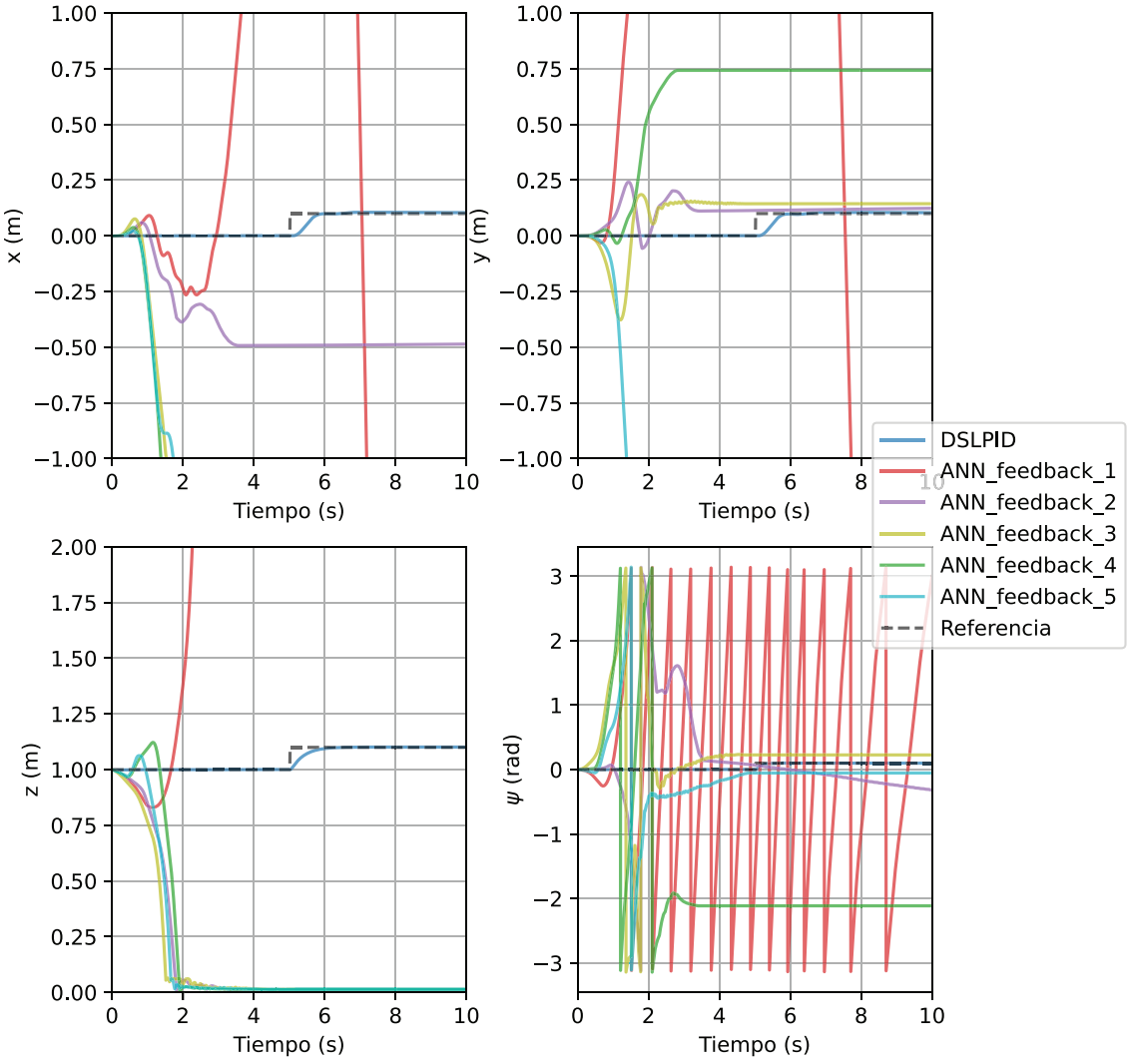


FIGURA 37: Respuesta de controladores ANN Feedback ante escalón

9.2.3. Memoria-Largo Corto Plazo (LSTM)

La arquitectura LSTM se sintonizó manualmente mediante la selección aleatoria del número de neuronas en cada capa. Las 5 mejores arquitecturas exploradas relacionan en las tablas 52, 53, 54, 55, y 56. En la figura 38 se comparan las señales para una respuesta al escalón con los controladores implementados en *Pybullet*.

TABLA 29: Evaluación ANN Feedback

ANN	**MSE**	T_i **[s]**	F_1
1	$6.97x10^{-4}$	$110x10^{-3}$	NAN
2	$7.79x10^{-4}$	$109x10^{-3}$	NAN
3	$7.06x10^{-4}$	$106x10^{-3}$	NAN
4	$7.63x10^{-4}$	$109x10^{-3}$	NAN
5	$7.35x10^{-4}$	$113x10^{-3}$	NAN

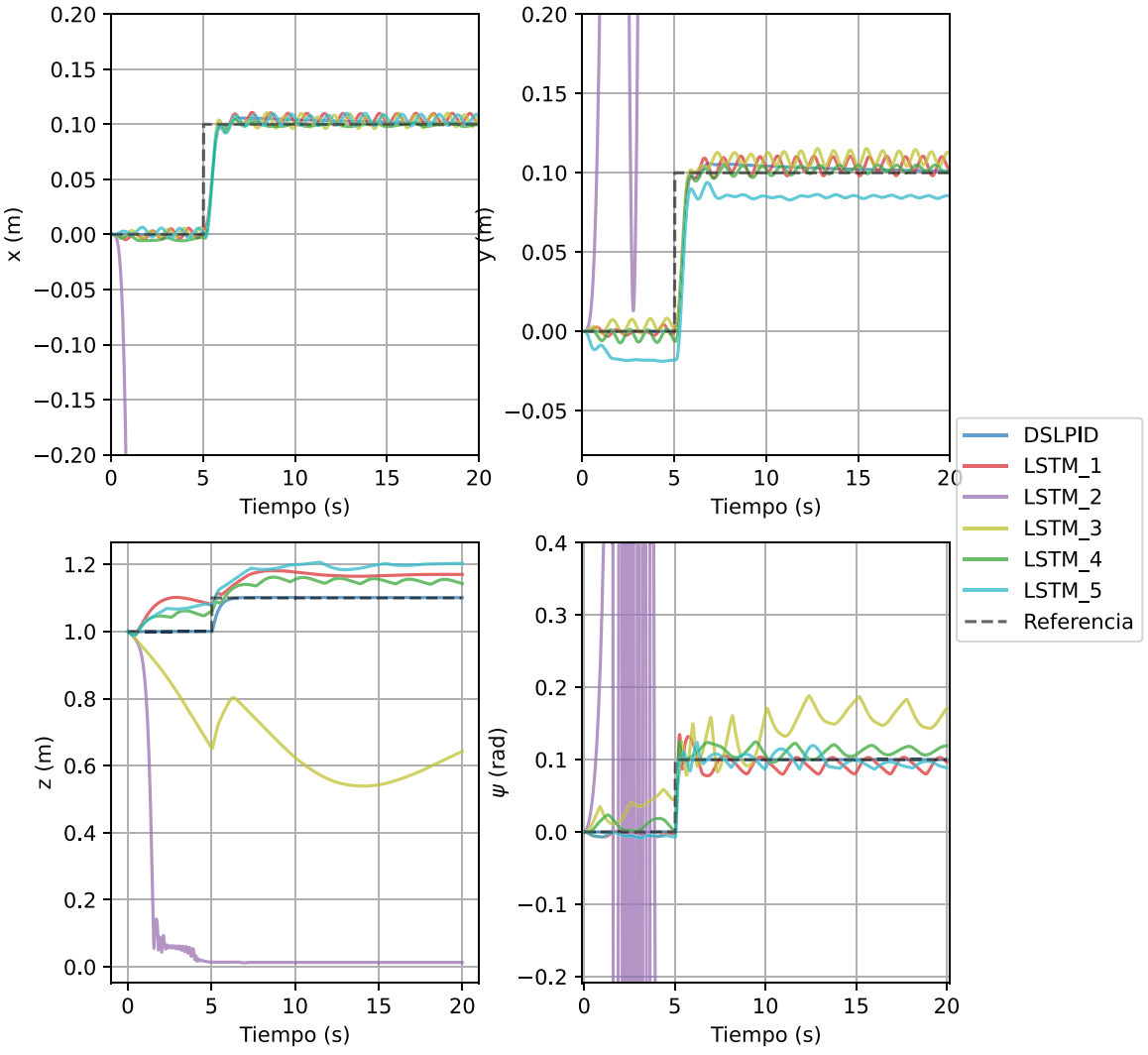


FIGURA 38: Respuesta de controladores LSTM ante escalón

Así mismo, las funciones de evaluación para cada una de las arquitecturas se muestran en la tabla 30.

TABLA 30: Evaluación LSTM

LSTM	**MSE**	T_i **[s]**	F_1
1	$1.52x10^{-3}$	$151x10^{-3}$	1.672
2	$7.717x10^{-4}$	$112x10^{-3}$	NA
3	$1.51x10^{-3}$	$161x10^{-3}$	1.579
4	$2.36x10^{-3}$	$133x10^{-3}$	0.983
5	$1.74x10^{-3}$	$248x10^{-3}$	1.576

Debido a su rendimiento en la función F_1 y a su bajo tiempo de inferencia, se selecciona al controlador LSTM 4 (ver tabla 55 como el mejor en esta arquitectura.

9.2.4. Memoria a Largo-corto Plazo Con Capas Convolucionales 1D Intercaladas (LSTMCNN):

Para las redes LSTM combinadas con las capas convolucionales (LSTMCNN), se encontraron los parámetros haciéndose uso del algoritmo *Hyperband* (ver sección 6.3.2.5). La figura 39 muestra la pérdida por época para los modelos entrenados. Para el algoritmo se entrenaron un total de 64 modelos, en las tablas 57, 58, 59, 60 y 61, se muestran los 5 mejores.

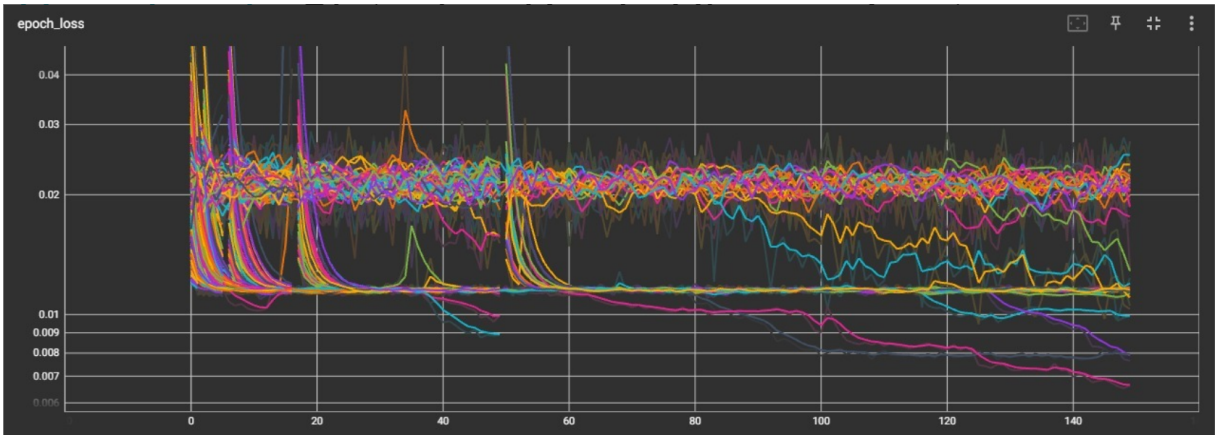


FIGURA 39: Resultados algoritmo hyperband para capas LSTMCNN

Finalmente, en la figura 40 se comparan las señales para una respuesta al escalón con los controladores implementados en *Pybullet* con los parámetros relacionados en la tabla 16 para la arquitectura *LSTMCNN*.

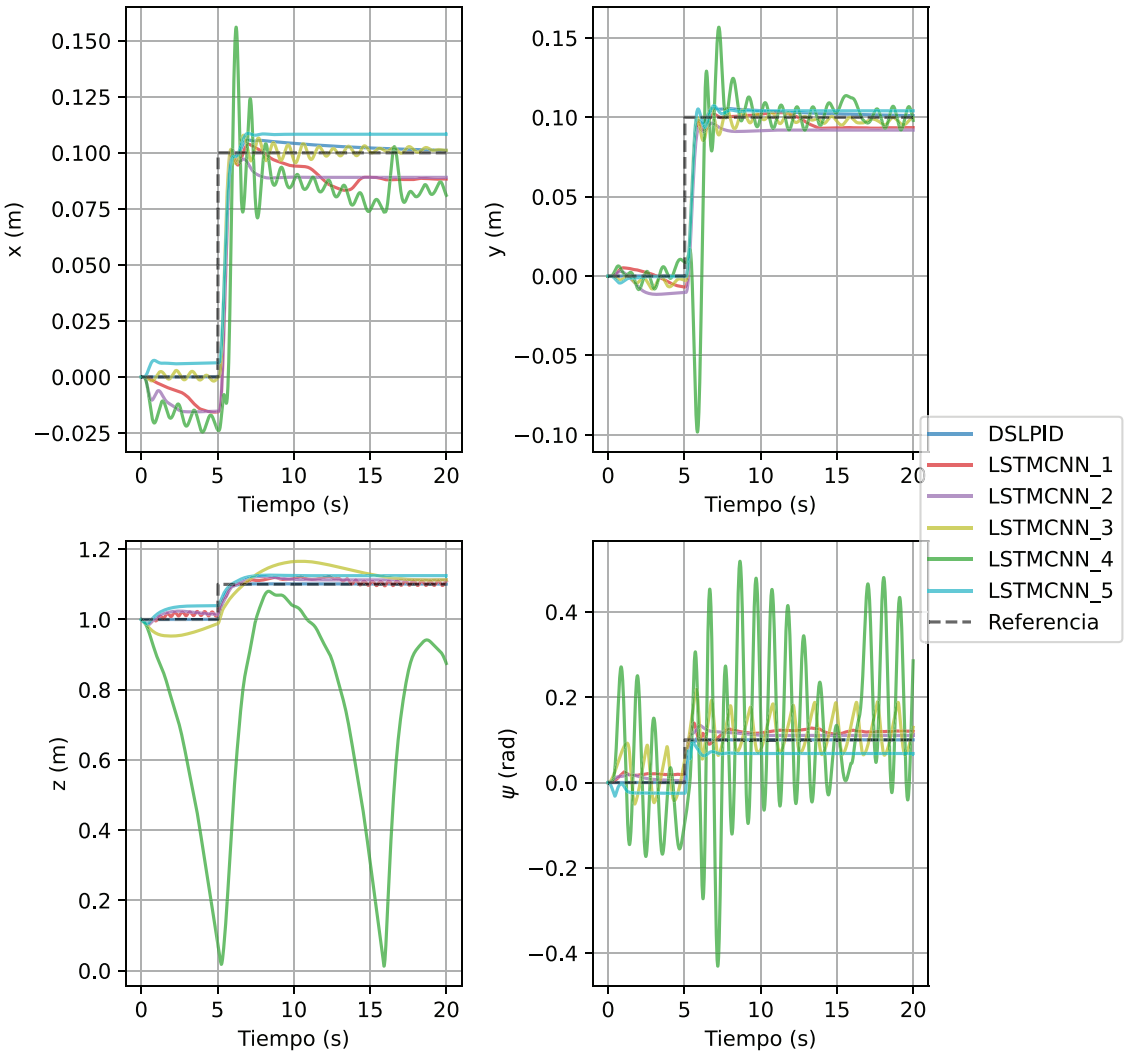


FIGURA 40: Respuesta controladores LSTMCNN ante escalón

Así mismo, las funciones de evaluación para cada una de las arquitecturas para la LSTMCNN se muestran en la tabla 31.

Debido a su rendimiento en la función F_1, la red LSTMCNN 3 debería ser considerada la mejor. Sin embargo, la función no que evalúa el error en estado estacionario no tiene en cuenta si el controlador oscila al rededor del punto, por lo que obtiene mejor evaluación que una señal con error en estado estacionario. Por estabilidad del dron, se decide seleccionar el controlador LSTMCNN 5, puesto que tiene un buen rendimiento en la función F_1 y no cuenta con oscilaciones.

TABLA 31: Evaluación LSTMCNN

LSTM	**MSE**	T_i **[s]**	F_1
1	$4.32x10^{-3}$	$201x10^{-3}$	1.622
2	$2.49x10^{-3}$	$178x10^{-3}$	1.484
3	$2.03x10^{-3}$	$161x10^{-3}$	1.268
4	$1.97x10^{-3}$	$133x10^{-3}$	5.471
5	$3.27x10^{-3}$	$175x10^{-3}$	1.393

9.2.5. Red Neuronal Convolucional Con Memoria a Largo-corto Plazo (CLSTM):

Para las redes convolucionales combinadas con las capas LSTM (CLSTM), se encontraron los parámetros haciéndose uso del algoritmo *Hyperband* (ver sección 6.3.2.5) con los parámetros descritos en la sección 8.6.4. La figura 41 muestra la pérdida por época para los modelos entrenados. Para el algoritmo se entrenaron un total de 62 modelos, en las tablas 62, 63, 64, 65 y 66, se muestran los 5 mejores.

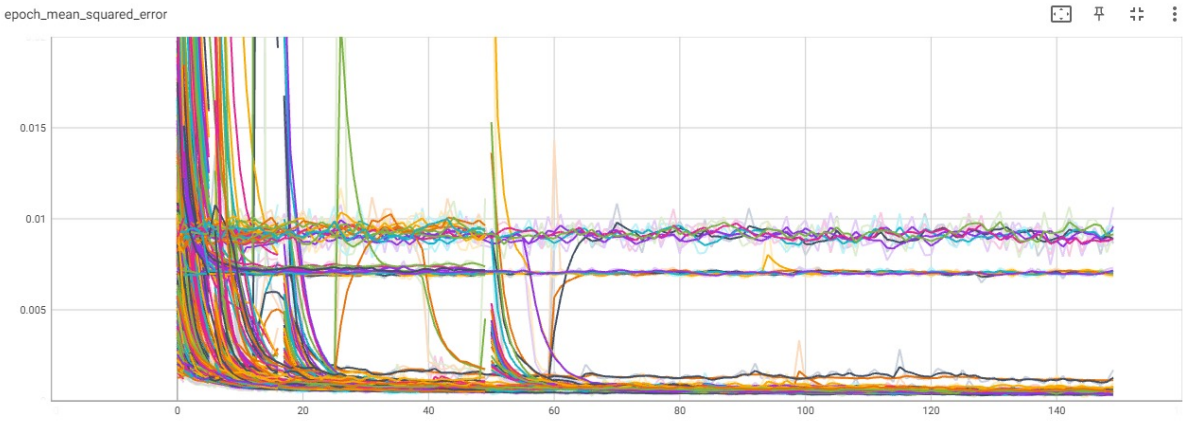


FIGURA 41: Resultados algoritmo hyperband para capas CLSTM

Finalmente, en la figura 42 se comparan las señales para una respuesta al escalón con los controladores implementados en *Pybullet* con los parámetros relacionados en la tabla 16 para la arquitectura *CLSTM*.

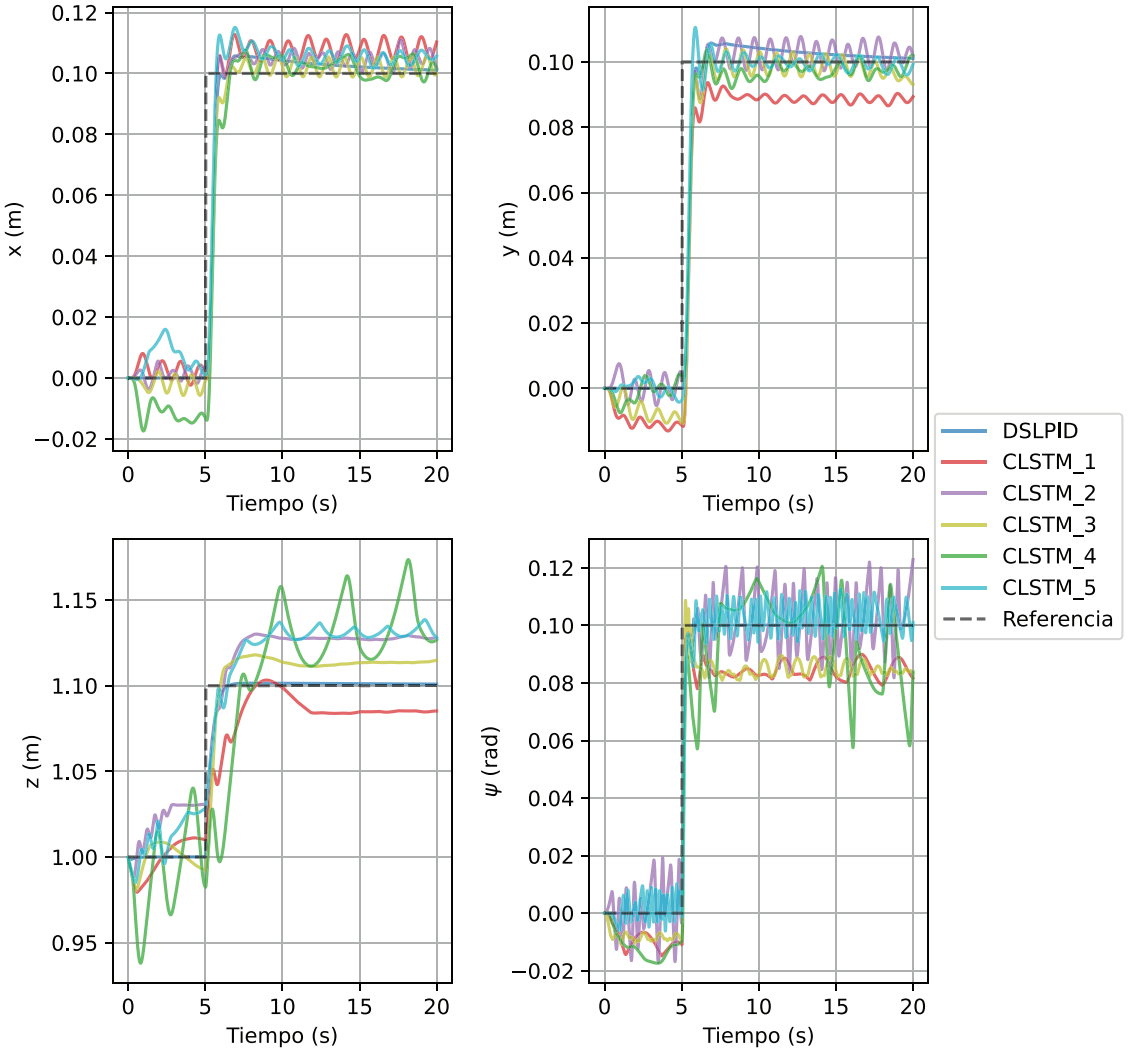


FIGURA 42: Respuesta de controladores CLSTM ante escalón

Así mismo, las funciones de evaluación para cada una de las arquitecturas para la LSTMCNN se muestran en la tabla 31.

Debido a su rendimiento en la función F_1, se selecciona la red CLSM 3.

TABLA 32: Evaluación CLSTM

LSTM	**MSE**	T_i **[s]**	F_1
1	$2.41x10^{-3}$	$187x10^{-3}$	2.015
2	$2.13x10^{-3}$	$167x10^{-3}$	1.656
3	$2.39x10^{-3}$	$159x10^{-3}$	1.175
4	$2.32x10^{-3}$	$149x10^{-3}$	1.440
5	$2.35x10^{-3}$	$173x10^{-3}$	2.001

9.2.6. Evaluación Mejores Modelos

9.2.6.1. Respuesta al Escalón

Posteriormente se compara cada uno de los mejores modelos en *Pybullet*. La figura 43 muestran la respuesta de los controladores ante el experimento propuesto (ver sección 9.2).

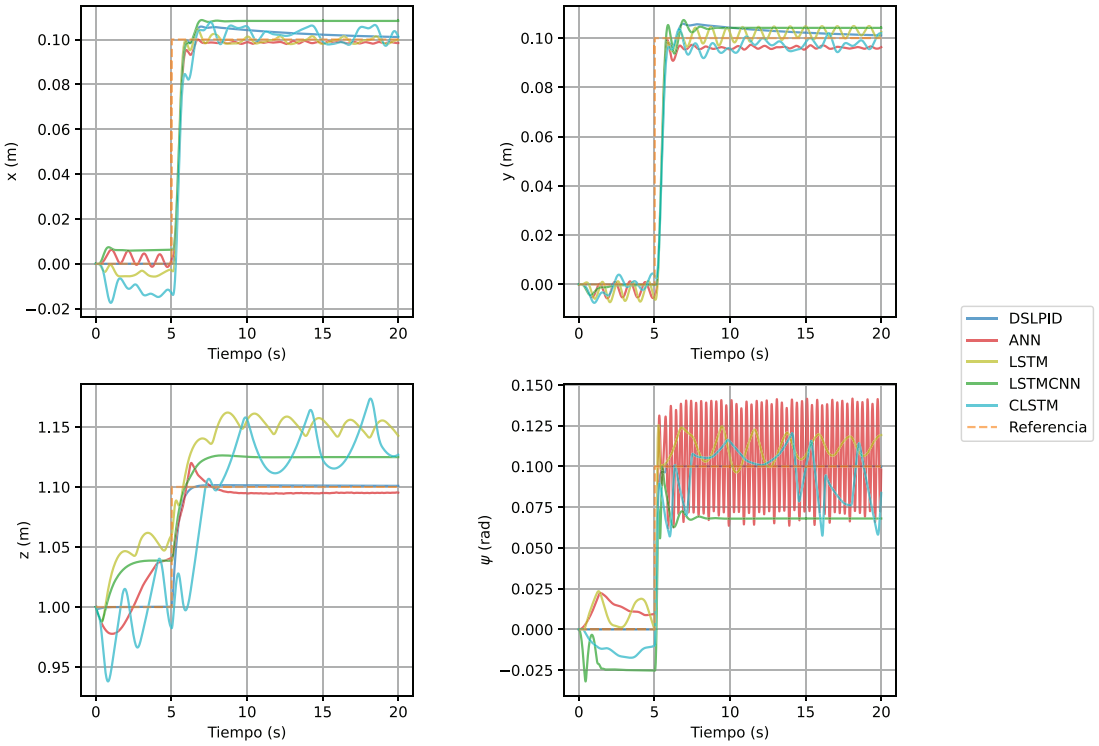


FIGURA 43: Respuesta de los mejores controladores ante escalón

Las mediciones del tiempo de estabilización, el sobre-pico y el error cuadrático de tiempo integral para cada modelo en este experimento se muestra en las tablas 36, 35, 33 y 34.

TABLA 33: Evaluación mejores modelos en x para $r_x = 0.1m$

Medida	DSLPID	ANN	LSTM	LSTMCNN	CLSTM
Ts	10.12	1.88	14.558	1.75	14.904
Os $[\%]$	5.786	0.165	4.616	8.687	7.723
ITSE	1725	1316	1232	4326	2613
F_1	1.000	0.316	0.983	1.393	1.440

TABLA 34: Evaluación mejores modelos en y para $r_y = 0.1m$

Medida	DSLPID	ANN	LSTM	LSTMCNN	CLSTM
Ts	10.15	6.425	14.89	2.679	24.72
Os $[\%]$	5.813	0.739	4.89	7.322	3.714
ITSE	1737	1316	2373	2412	1960
F_1	1.000	0.708	1.104	0.970	1.072

TABLA 35: Evaluación mejores modelos en z para $r_z = 0.1m$

Medida	DSLPID	ANN	LSTM	LSTMCNN	CLSTM
Ts	0.816	2.033	14.16	1.470	13.77
Os $[\%]$	0.134	1.816	5.634	2.388	6.688
ITSE	88.38	308.6	2159	1097	1484
F_1	1.000	6.487	27.86	10.64	27.77

TABLA 36: Evaluación mejores modelos en ψ para $r_\psi = 0.1rad$

Medida	DSLPID	ANN	LSTM	LSTMCNN	CLSTM
Ts	0.808	14.99	14.76	3.17	14.99
Os $[\%]$	2.247	41.64	25.23	0.269	20.51
ITSE	225.2	10782	5683	15109	6630
F_1	1.000	28.31	18.23	23.70	19.03

Donde la fila F_1, representa el promedio de las medidas normalizadas de cada columna para los ejes evaluados. Estas medidas se normalizan tomando como máximos los valores del controlador original.

También se evalúa el error a través del tiempo junto con el error acumulado de cada modelo en comparación con el controlador base en las figuras 44 y 45.

Posteriormente se calcula el promedio de todos los ejes para cada controlador para la función F_1. Así mismo, se calcula el esfuerzo de control Q (ver ecuación 7.14) promedio para los 4 motores, con valor en estado estable $u_{ss} = 14468$. Lo anterior con el objetivo de minimizar el consumo de energía y evitar que el controlador se sature. Además se relaciona el error acumulado promedio en todos los ejes e_{acum}. Finalmente se calcula el promedio normalizado con respecto al controlador entre F_1, Q y e_{acum}, relacionado en la fila Total. Los resultados de relacionan en 37.

De la tabla 37, se define que la mejor red es la LSTMCNN y por lo tanto esta red es la que se selecciona para comparar con el controlador base.

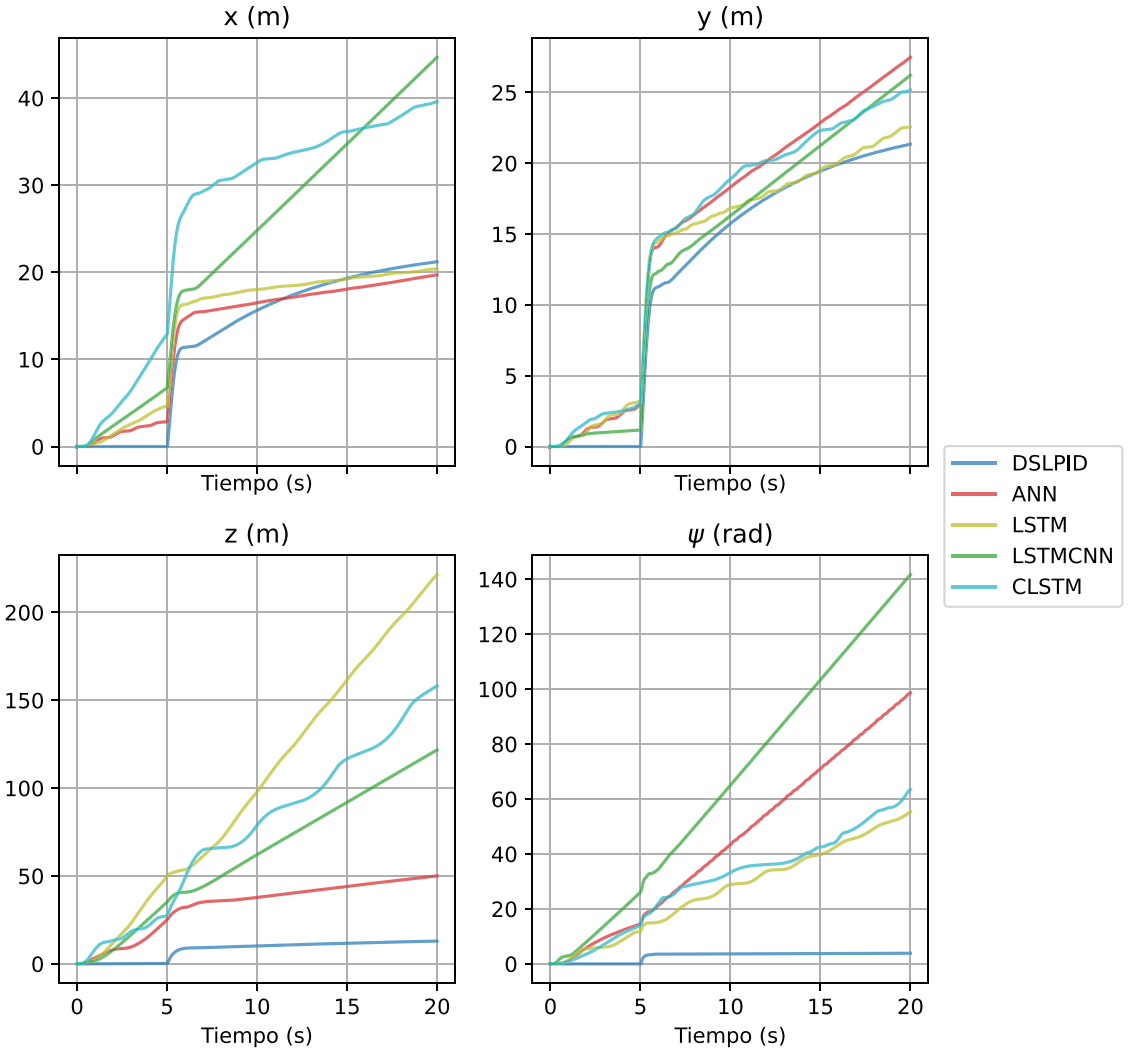


FIGURA 44: Error acumulado mejores modelos respuesta escalón

9.3. Validación Ventana de Inferencia

Para validar que la cantidad de estados anteriores que se ingresan a la red es la cantidad adecuada con base a los obtenidos en 8.5.1, se realiza una prueba donde para el mismo controlador LSTMCNN se ingresa diferentes estados retrasados. Esto debido a que la capa LSTM de entrada puede funcionar con distintas cantidades de estados retrasados. Dentro de esta prueba se mueve la referencia en $r_y = 0.1m$,$r_x = 0.1m$ y $r_z = 0.1m$ y $r_\psi = 0.1rad$. Posteriormente se vuelve a la posición a la posición inicial posteriormente para evaluar la respuesta transitoria de cada controlador como lo muestra la figura 46.

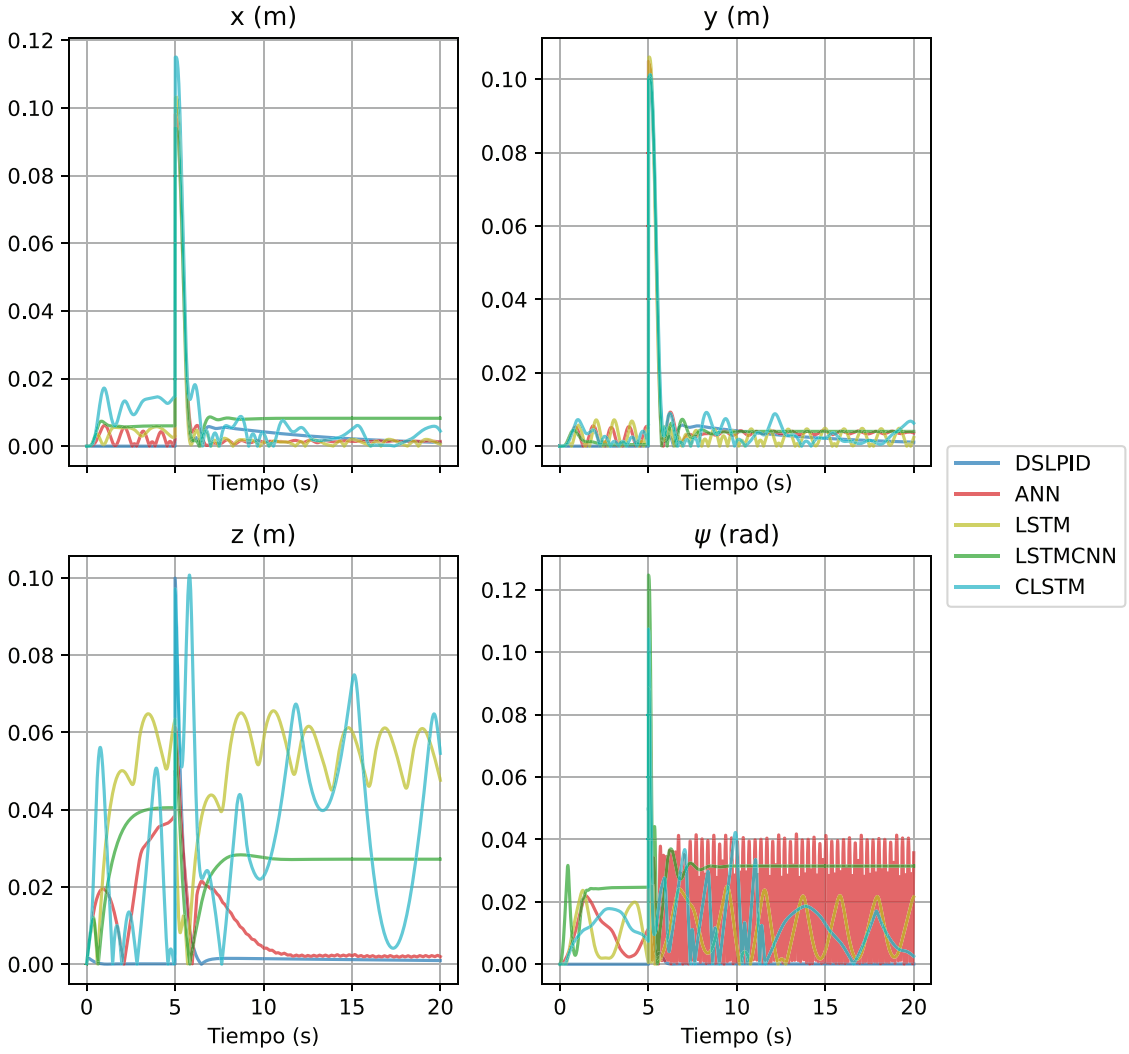


FIGURA 45: Error mejores modelos respuesta escalón

Las medidas ante este cambio en la referencia en el segundo 6 para cada eje, se presentan en las tablas 38, 39, 40 y 40.

Adicionalmente se midió el error y el error acumulado de cada controlador como se presenta en las figuras 47 y 48.

Donde al sumar la puntuación de cada controlador en cada eje se tiene una puntuación de 4 para DSLPID, 90.75 para $LSTMCNN\_8$, 27.44 para $LSTMCNN\_16$, 28.53 para $LSTMCNN\_32$, 28.53 para $LSTMCNN\_64$, 28.68 para $LSTMCNN\_128$. Adicionalmente se midió el tiempo de inferencia de cada red, siendo el tiempo de inferencia de $LSTMCNN\_8 = 95ms$, $LSTMCNN\_16 = 106ms$,

TABLA 37: Evaluación mejores modelos para respuesta escalón

Medida	DSLPID	ANN	LSTM	LSTMCNN	CLSTM
Q	$70.01x10^3$	$415.6x10^3$	$69.74x10^3$	$59.35x10^3$	$78.37x10^3$
$\bar{F}_1$	1.000	8.955	12.04	9.175	12.328
e_{acum}	14.85	48.98	79.96	83.59	71.62
Total	1.000	6.061	6.140	5.217	6.09

TABLA 38: Respuesta transitoria en x comparación de tamaño de ventana

	DSLPID	Ventana 8	Ventana 16	Ventana 32	Ventana 64	Ventana 128
Ts	10.12	10.51	12.76	13.30	12.87	12.87
Os [%]	5.786	1.771	6.027	2.263	1.702,26	1.702
ITSE	1725	4346	9627	45987	5027	5007
Total	1.000	0.772	1.634	0.940	0.880	0.880

TABLA 39: Respuesta transitoria en y comparación de tamaño de ventana

	DSLPID	Ventana 8	Ventana 16	Ventana 32	Ventana 64	Ventana 128
Ts	10.15	11.36	13.33	13.94	13.950	13.950
Os [%]	5.813	4.326	1.542	10.75	10.73	10.73
ITSE	1737	4747	4981	3377	3384	3396
Total	1.000	1.021	1.300	0.680	0.640	0.640

TABLA 40: Respuesta transitoria en z comparación de tamaño de ventana

	DSLPID	Ventana 8	Ventana 16	Ventana 32	Ventana 64	Ventana 128
Ts	0.816	9.200	9.340	11.85	11.69	11.70
Os [%]	0.134	13.05	10.97	12.80	12.83	12.83
ITSE	88.38	2118	671	1459	1491	1486
Total	1.000	4.280	2.040	3.333	3.333	3.328

TABLA 41: Respuesta transitoria en ψ comparación de tamaño de ventana

	DSLPID	Ventana 8	Ventana 16	Ventana 32	Ventana 64	Ventana 128
Ts	0.808	12.64	13.30	12.85	12.69	12.69
Os [%]	2.247	2.451	4.785	4.995	5.059	5.059
ITSE	225.2	5714	15.158	14.143	13.941	13.965
Total	1.000	84.69	23.95	23.78	23.84	23.85

$LSTMCNN\_32 = 133ms$, $LSTMCNN\_64 = 176ms$, y $LSTMCNN\_128 = 265ms$. Después de estas pruebas se valida que la red con menor error en la respuesta transitoria y menor tiempo de inferencia es la red a la que se le ingresan 32 estados. De igual manera, la ventana previamente seleccionada (64 muestras) ofrece un rendimiento similar que la ventana de 32 estados.

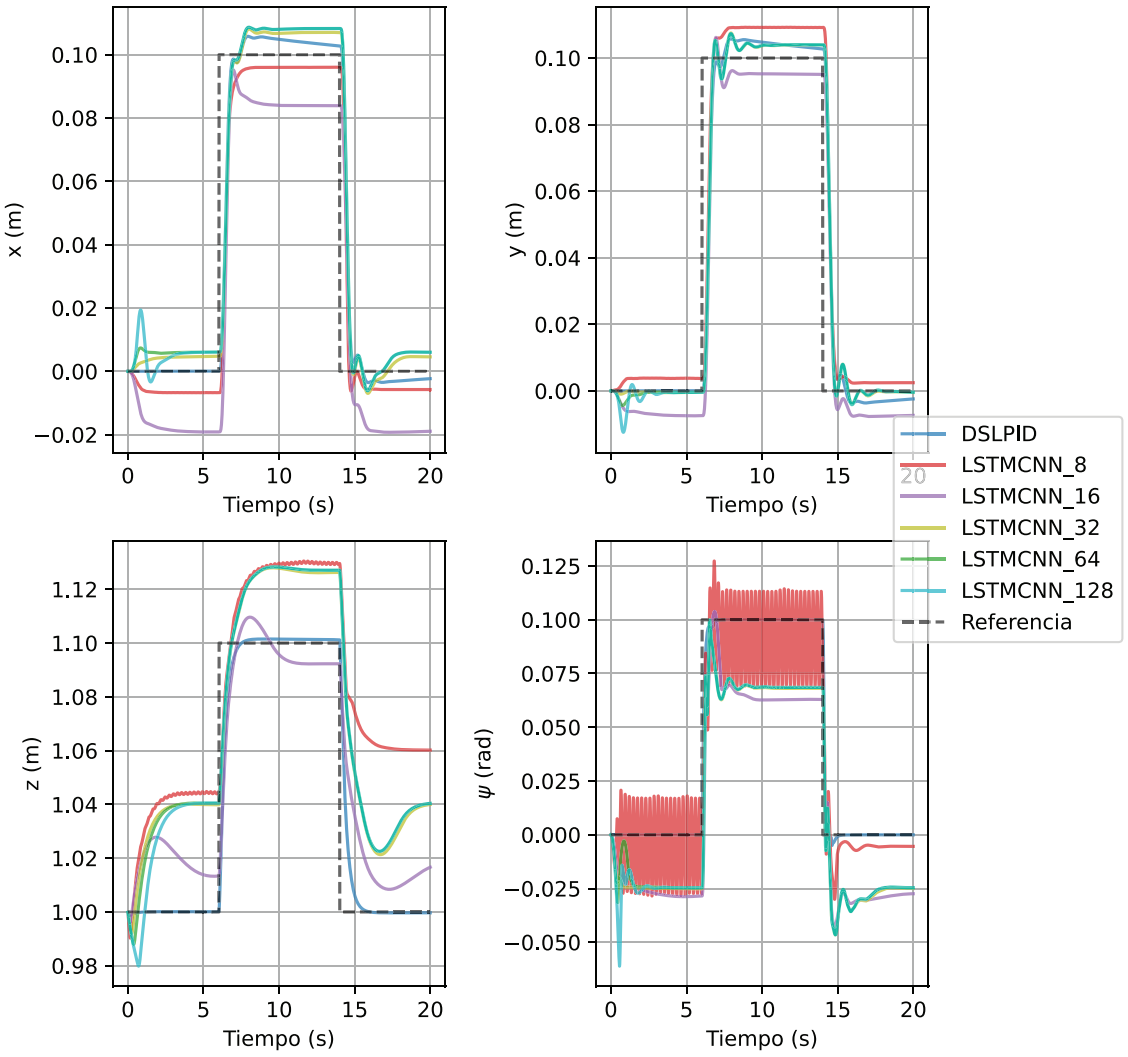


FIGURA 46: Estados de la respuesta al escalón para LSTMCNN con distintas ventanas

9.4. Comparación LSTMCNN vs DLSPID

En esta sección se explora a fondo el rendimiento del controlador LSTMCNN con respecto al controlador base ante distintos experimentos.

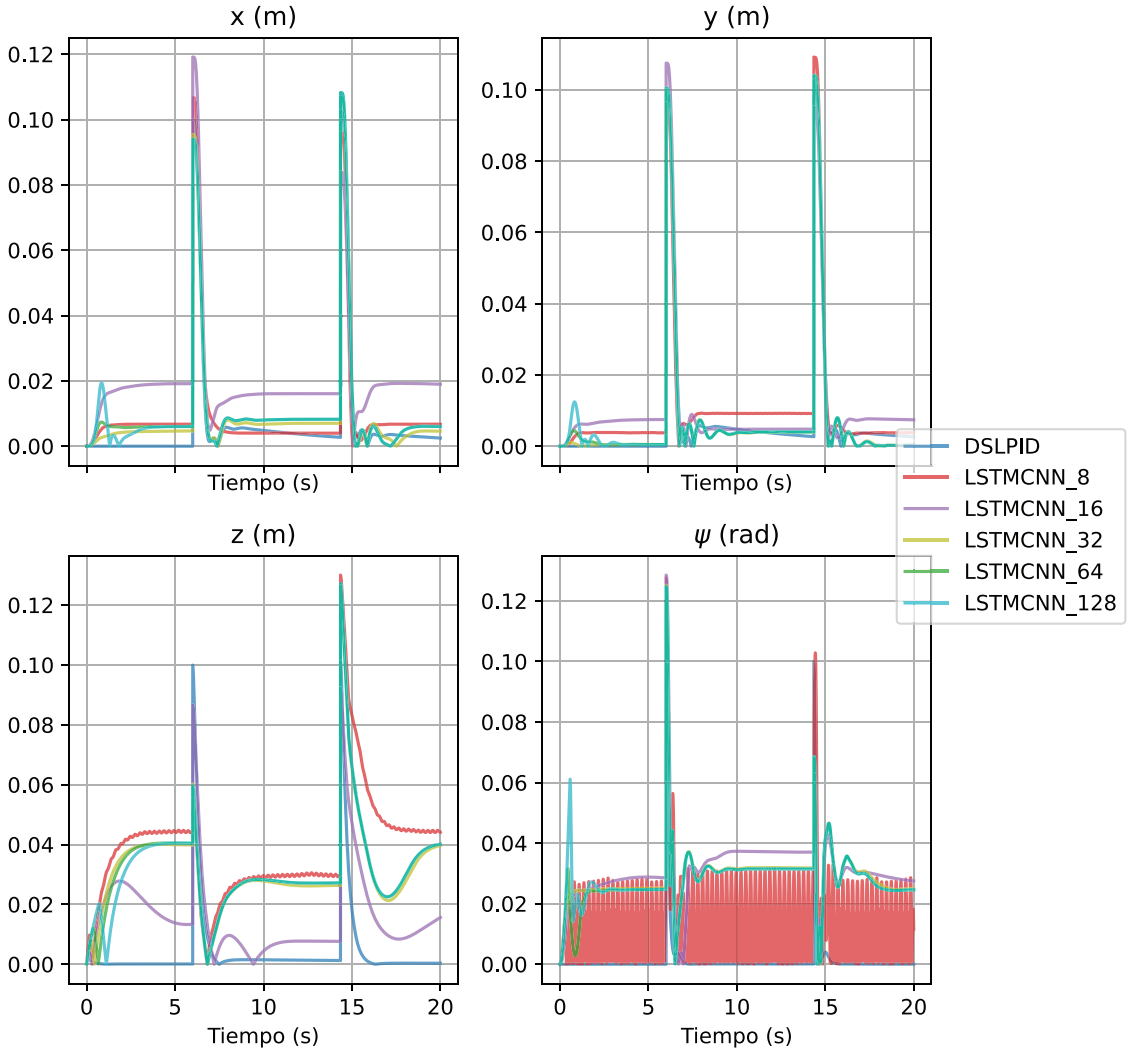


FIGURA 47: Error respuesta al escalón para LSTMCNN con distintas ventanas

9.4.0.1. Respuesta al Escalón

Se plantea la respuesta al escalón con referencias en $r_x = 0.2m$, $r_y = 0.2m$, $r_z = 0.2m$ y $r_\psi = 0.2rad$. El resultado del experimento se evidencia en las figuras 49, 50, 51, 52 y 54.

De la figura 49 se observa que el controlador LSTMCNN en ψ no realiza un buen seguimiento de la señal, manteniendo un error en estado estacionario. Así mismo, en el eje z, antes de ingresar la señal el controlador tiene un error en estado estacionario. Los ejes x y y responden de manera adecuada, sin embargo, la aceleración y los ángulos de actitud (ϕ, θ) en dichos ejes es mayor. La aceleración en el eje z

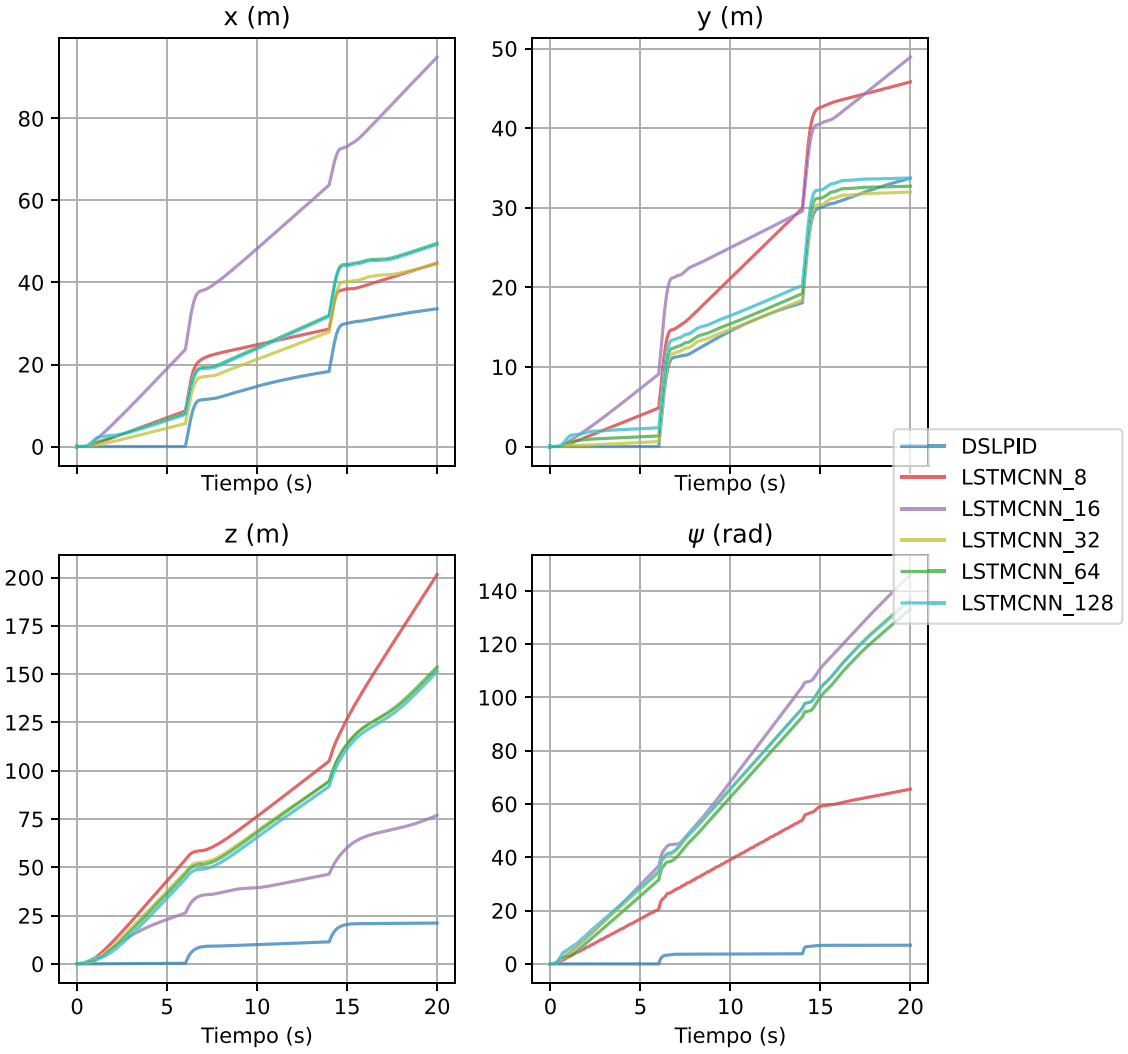


FIGURA 48: Error acumulado respuesta al escalón para LSTMCNN con distintas ventanas

es menor, lo que conlleva a una transición más suave. De las figuras 52 y 53 se observa que la salida del controlador LSTMCNN es más suave, es decir, se minimiza el pico máximo del esfuerzo de control.

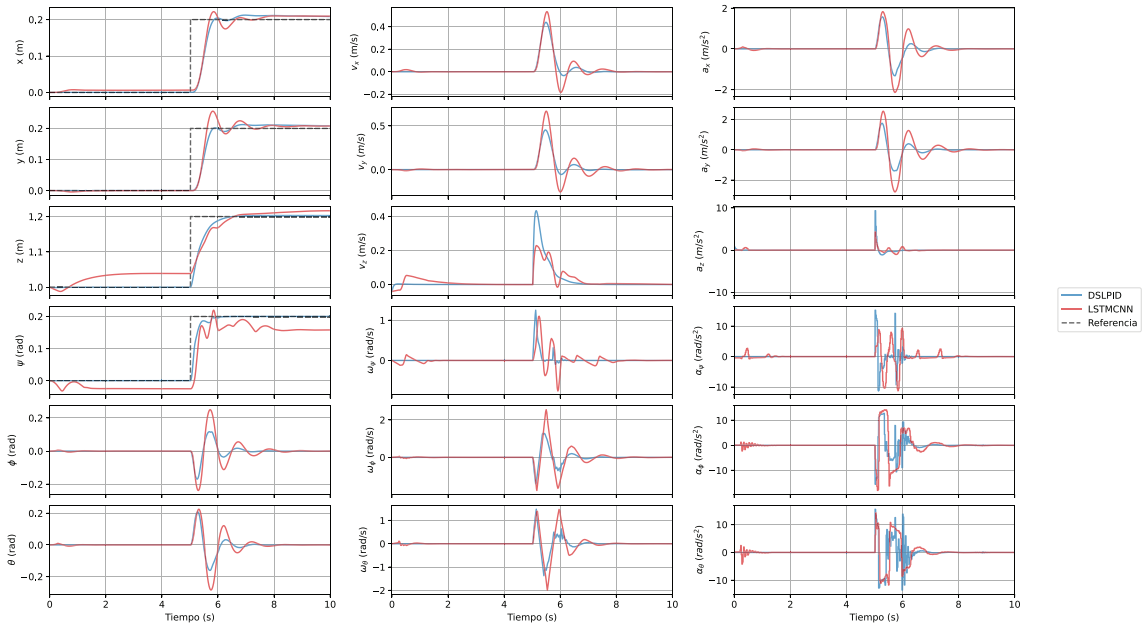


FIGURA 49: Estados de la respuesta al escalón para LSTMCNN vs DSLPID

9.4.0.2. Respuesta al Gran Escalón

En un segundo experimento se evalúa el seguimiento a la referencia para una señal escalón, probando cada eje por separado (es decir cuando $r_x != 0$, $r_y = 0$, $r_z = 0$, y $r_\psi = 0$. La magnitud de referencia para cada uno de los ejes es $r_x = 0.4m$, $r_y = 0.4m$, en $r_z = 4m$ y $r_\psi = 3rad$. Estos valores fueron determinados, ya que se encuentran fuera del punto de operación para el controlador base, por lo que éste es inestable. Con este experimento se quiere comprobar si el modelo entrenado fue capaz de generalizar el comportamiento del controlador, manteniendo la estabilidad fuera del la región de operación. Los resultados de este experimento se pueden observar en la figura 55.

Se observa que el controlador LSTMCNN mantiene la referencia en los ejes x, y, incluso cuando el controlador DSLPID no. En el eje z, el controlador entrenado es estable, aunque mantiene error en estado estacionario y pierde altitud en el transitorio, pero recupera el vuelo. En el eje ψ, el controlador sigue la referencia, pero no la mantiene.

9.4.0.3. Rechazo a Perturbaciones

Para evaluar la estabilidad del controlador, se propone perturbar el estado del dron con amplitudes el doble de las presentes en el conjunto de datos de entrenamiento. Se define como factor de perturbación en $W_{v_z} = 2$, $W_q = -0.4$, $W_p = 0.4$, $W_r = 0.5$. La respuesta del sistema se muestra en la figura 56

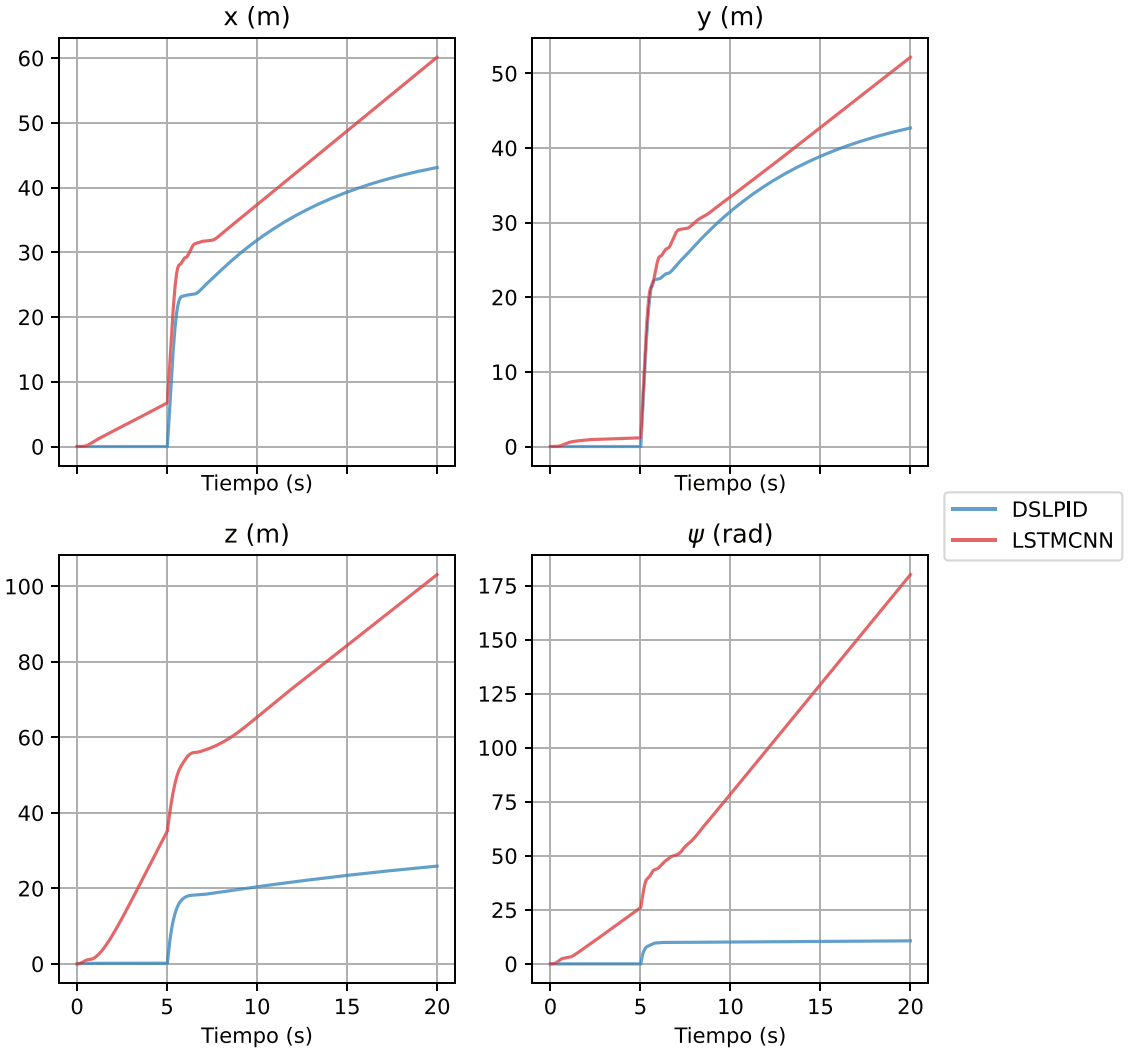


FIGURA 50: Error acumulado de la respuesta al escalón para LSTMCNN vs DSLPID

El controlador LSTMCNN es capaz de rechazar las perturbaciones del mismo modo que el controlador DSLPID. Los problemas de errores de estado estacionario persisten en los ejes x, z y ψ después de la perturbación. En el eje y el controlador LSTMCNN se recupera más rápido y con menor sobrepaso que el DSLPID.

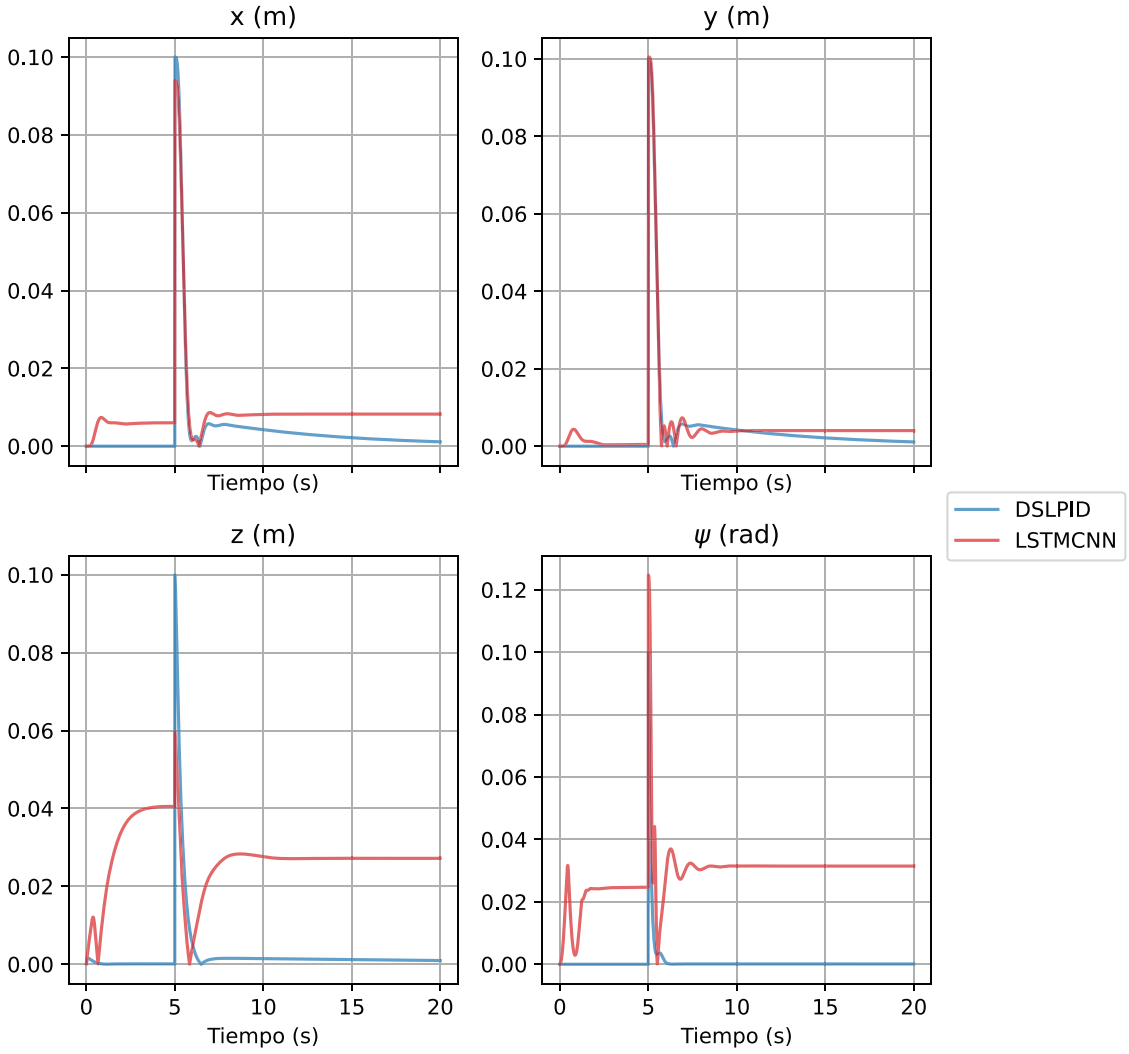


FIGURA 51: Error de la respuesta al escalón para LSTMCNN vs DSLPID

9.4.0.4. Señales Varias

Se propone evaluar el controlador con diferentes señales en todos los ejes con señales de ruido en x de amplitud $A = 0.2m$ y frecuencia de corte $f_0 = 0.1Hz$, ruido en ψ con amplitud $A = 1rad$ y frecuencia de corte $f_0 = 0.1Hz$, sinusoidal en y con amplitud $A = 0.2m$ y frecuencia $\omega = 0.1Hz$ y rampa en z con pendiente $m = 0.1m/s$. La respuesta del sistema se muestra en las figuras 57, **??** y 58.

De la figura 57 se observa que el controlador LSTMCNN mantiene un mayor error con respecto a la trayectoria en todos los ejes. Sin embargo, el controlador entrenado mantiene la estabilidad y realiza el

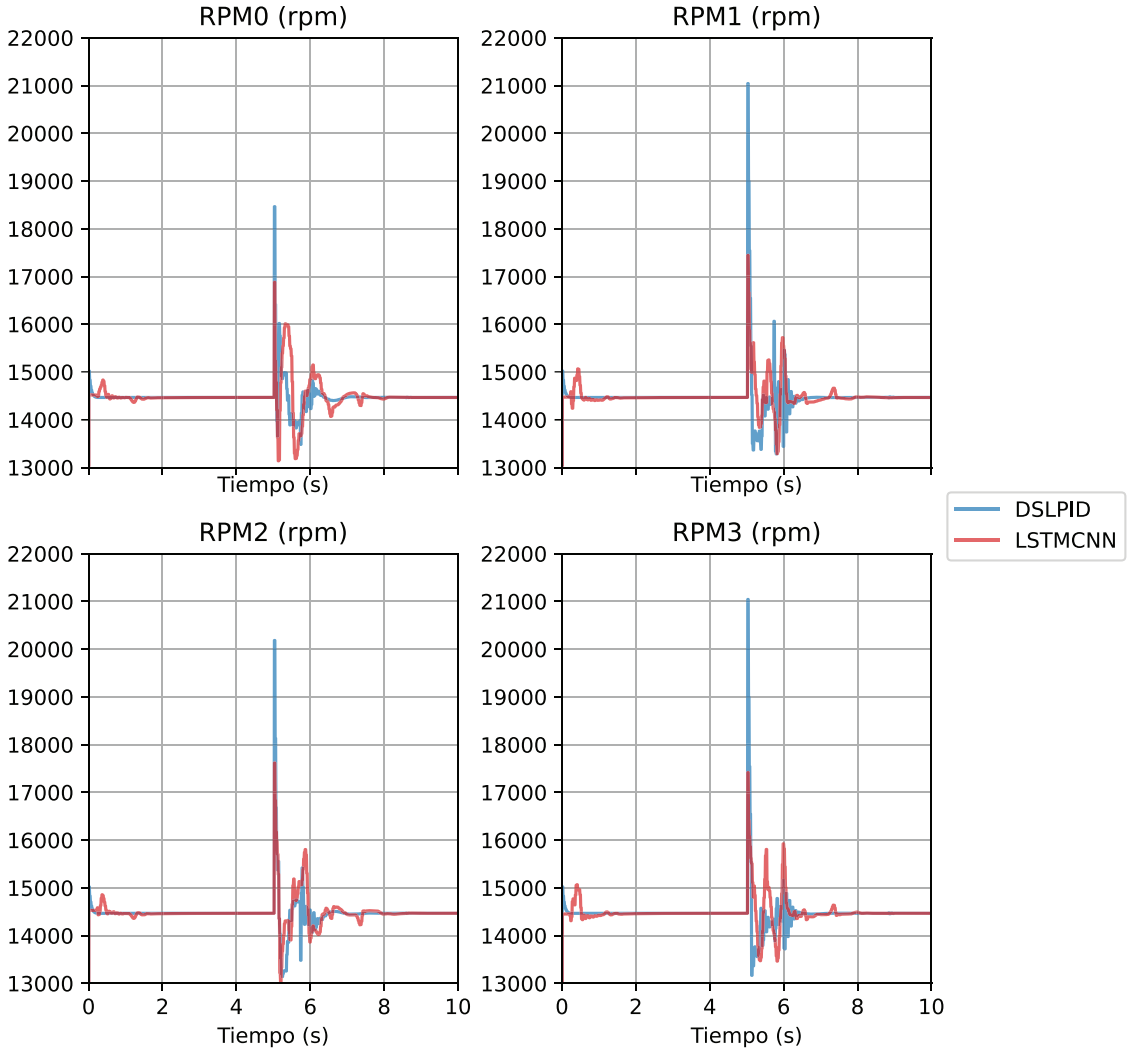


FIGURA 52: Esfuerzo de control para LSTMCNN vs DSLPID

seguimiento de la trayectoria, incluso con una perturbación a los 10s.

9.4.0.5. Espiral de Subida

Uno de los problemas del controlador lineal DSLPID es que no responde bien a altas frecuencias y grandes velocidades. Este problema es crucial para la ejecución de trayectorias agresivas en tiempos de respuesta cortos mencionadas en [35]. Por tal motivo se propone evaluar el controlador realizando una trayectoria circular a aceleraciones de $a_x = 2.5m/s^2$ y $a_y = 3.5m/s^2$, mientras el dron se eleva a una

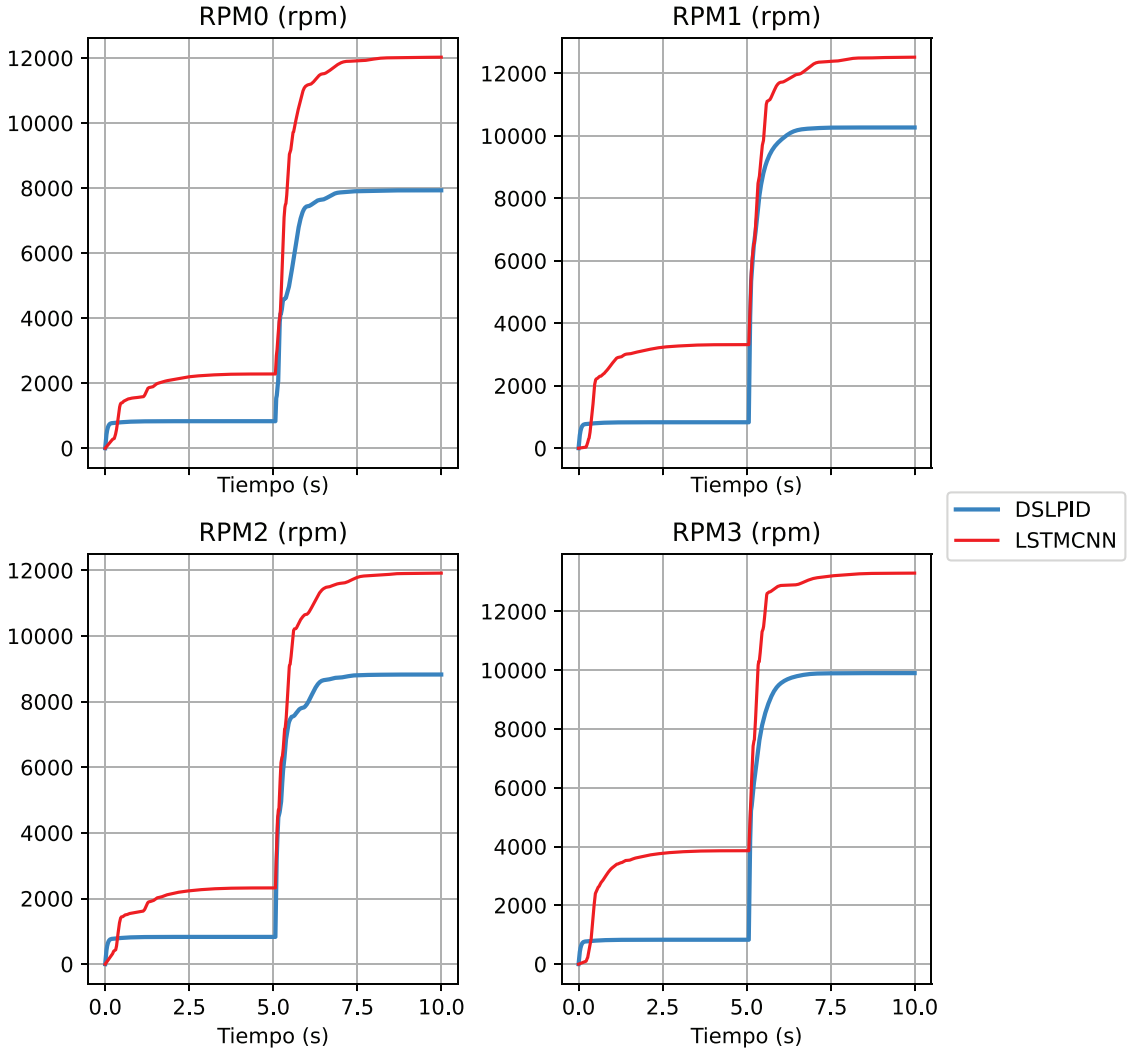
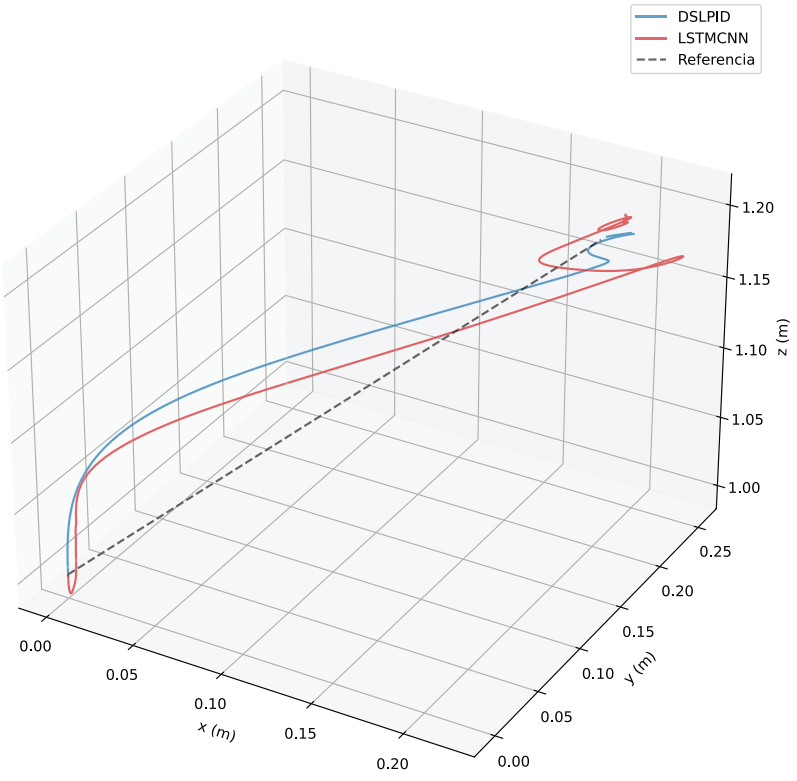


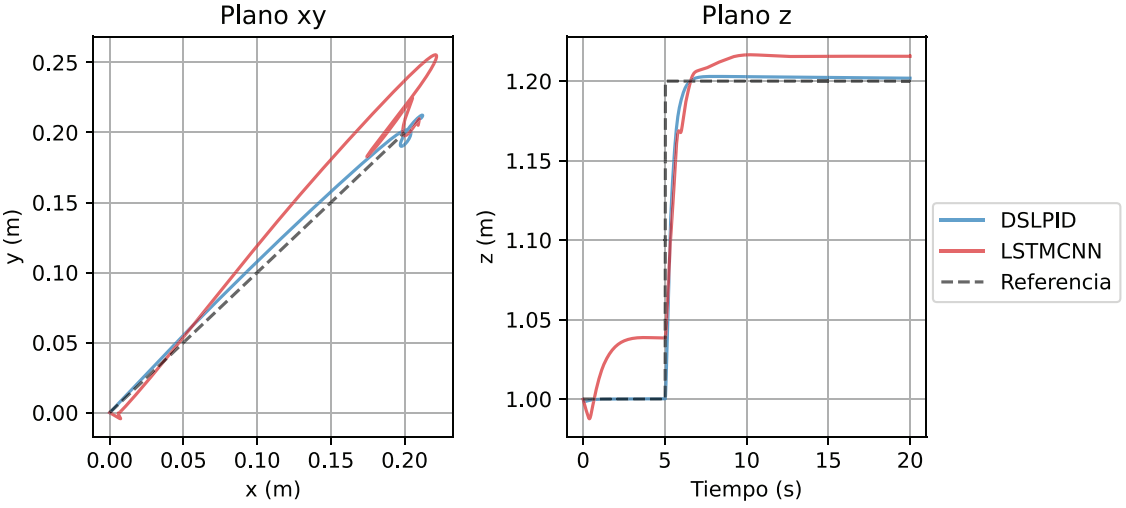
FIGURA 53: Esfuerzo de control acumulado para LSTMCNN vs DSLPID

velocidad de $v_z = 0.25m/s$ y mantiene la orientación en $\psi = 0$ rad. La referencia en $r_x = 0.3sin(\pi t)$ y $r_r = 0.3cos(\pi t)$.

De la figura 60 se observa que el controlador LSTMCNN mantiene la estabilidad y sigue la trayectoria para una trayectoria agresiva. Así mismo se señala que el ángulo de actitud del dron LSTMCNN supera los 0.4 radianes (23 grados), el cual es un ángulo de inclinación considerable, al cual el controlador DSLPID no logra mantener, ya que pierde la estabilidad y cae. Lo anterior debido a que la aproximación realizada para ángulos pequeños en el modelo de pequeña señal (ver sección 6.1.3.6), solo es aplicable para ángulos menores a 10 grados aproximadamente [98].

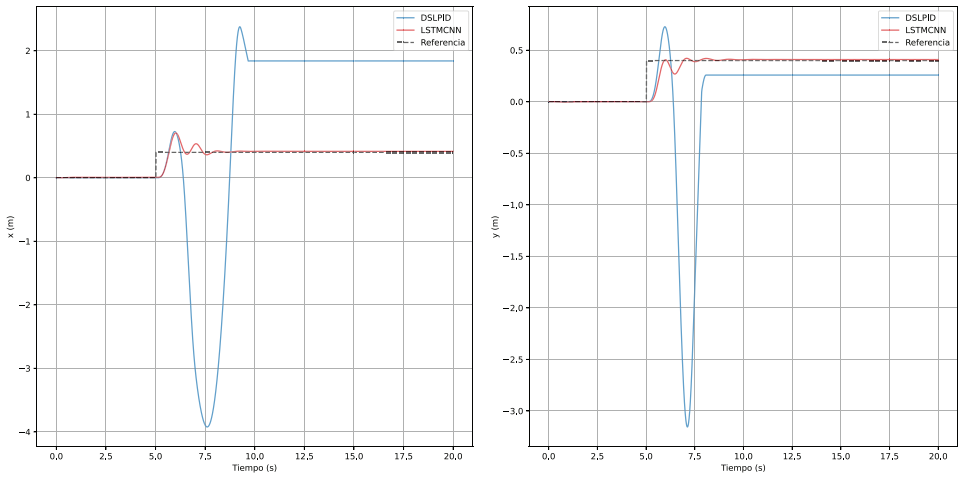


(A) Vista 3D

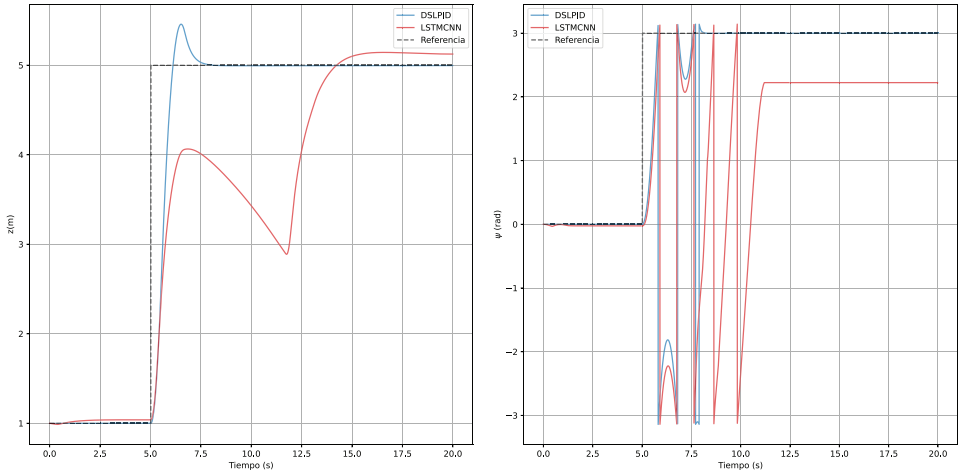


(B) Vista Planos XY, Z

FIGURA 54: Vista complementaria de la respuesta al escalón para LSTMCNN vs DSLPID



(A) x (B) y



(C) z (D) ψ

FIGURA 55: Respuesta al gran escalón para LSTMCNN vs DSLPID

9.4.0.6. Lemniscate

Posteriormente, se plantea el experimento de la función Lemniscate o infinito. Esta trayectoria es agresiva porque involucra cambios bruscos de velocidad. Esta referencia no fue posible de implementar en el conjunto de datos, debido a que el controlador original no la soportaba. Esta función de referencia se basa en la función Lemniscata para los ejes x e y, mientras que en z se realiza el seguimiento de una función rampa. Esta señal de referencia se define en las ecuaciones 9.1, 9.2 y 9.3.

$$r_x = d\frac{cos(2\pi ft + \Phi)}{sin(2\pi ft + \Phi)^2 + 1} \tag{9.1}$$

$$r_y = r_x sin(2\pi ft + \Phi) \tag{9.2}$$

$$r_z = mt \tag{9.3}$$

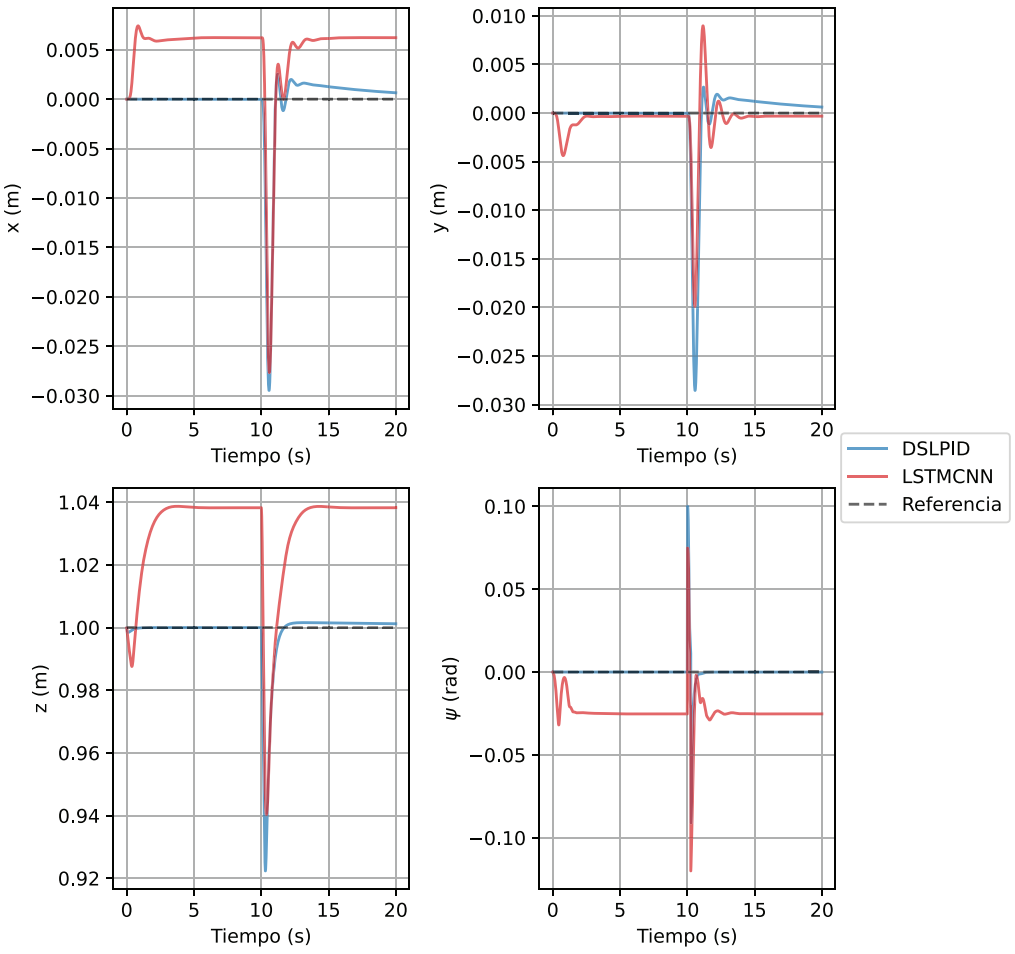


FIGURA 56: Respuesta ante perturbación para LSTMCNN vs DSLPID

Para este caso, $d = 9m$, $f = 0.4Hz$, $\Phi = (3/4)\pi$ y $m = 0.15$.

9.4.0.7. Despegue-Aterrizaje

Como siguiente experimento se propone evaluar el aterrizaje y el despegue del controlador LSTMCNN en comparacion con el controlador base, de la forma en que se presenta en la figura 62.

Para este caso se observa como el controlador LSTMCNN ofrece una respuesta similar en el despegue con un error mayor al del DSLPID en estado estacionario y realizando los cambios en posición a una velocidad menor que el DSLPID. Encontrando que para este escenario el controlador neuronal propuesto abstrae el tipo de respuesta deseado a pesar de que posee un error al momento de seguir la altura especificada.

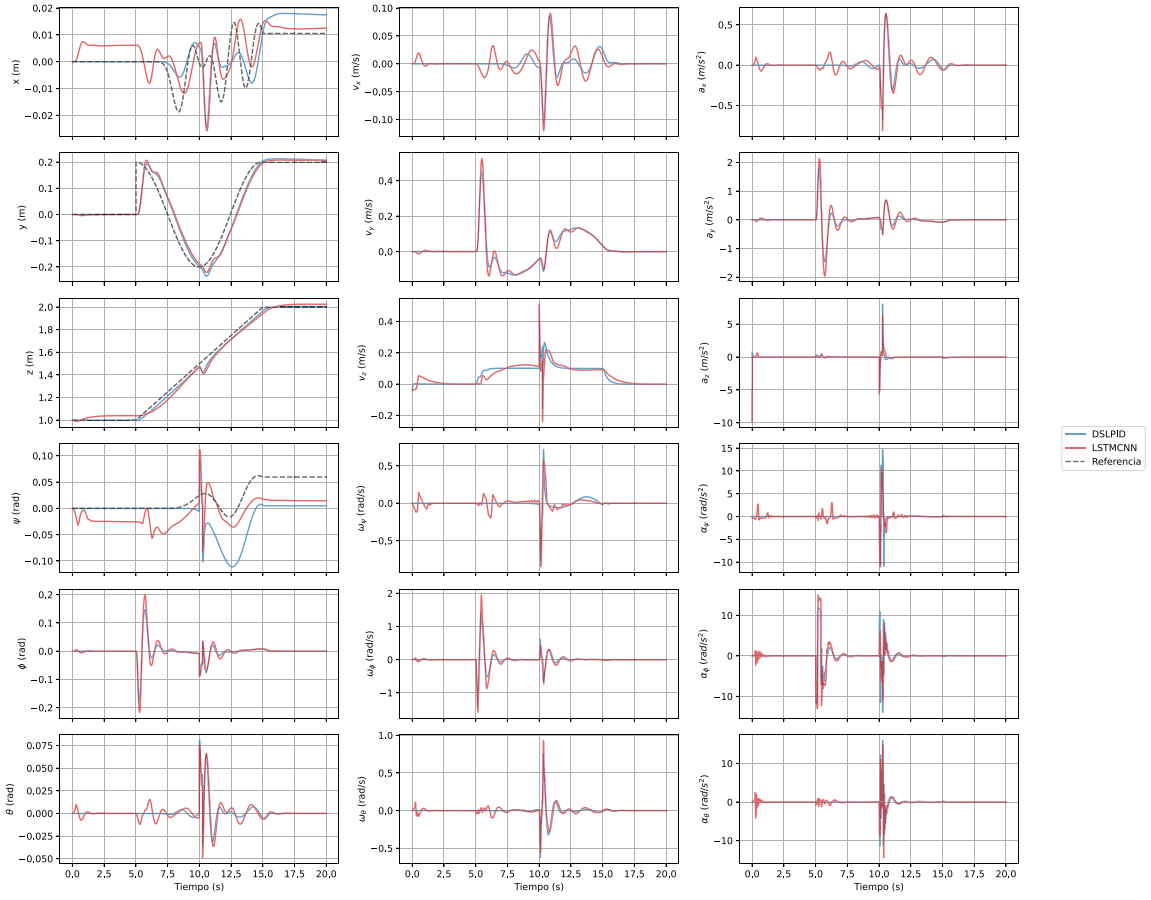
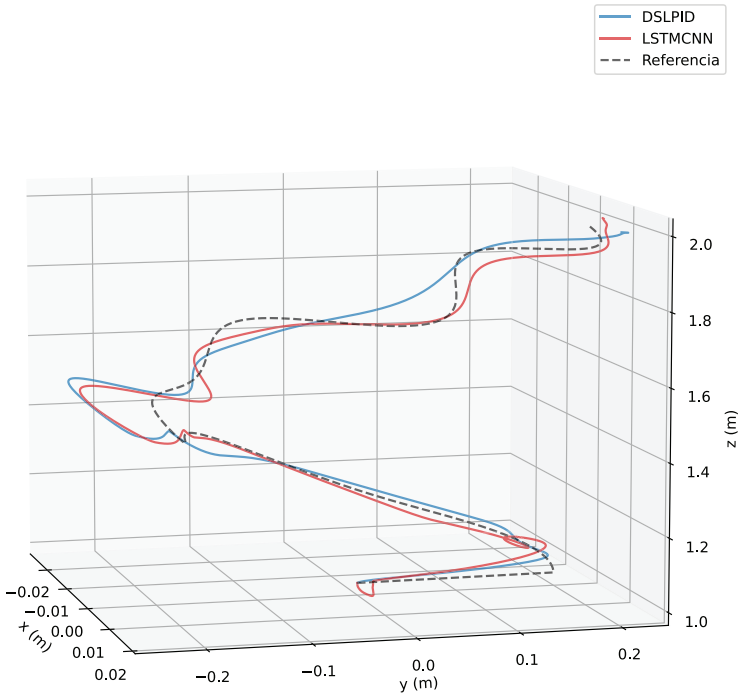


FIGURA 57: Estados de la respuesta a señales varias para LSTMCNN vs DSLPID

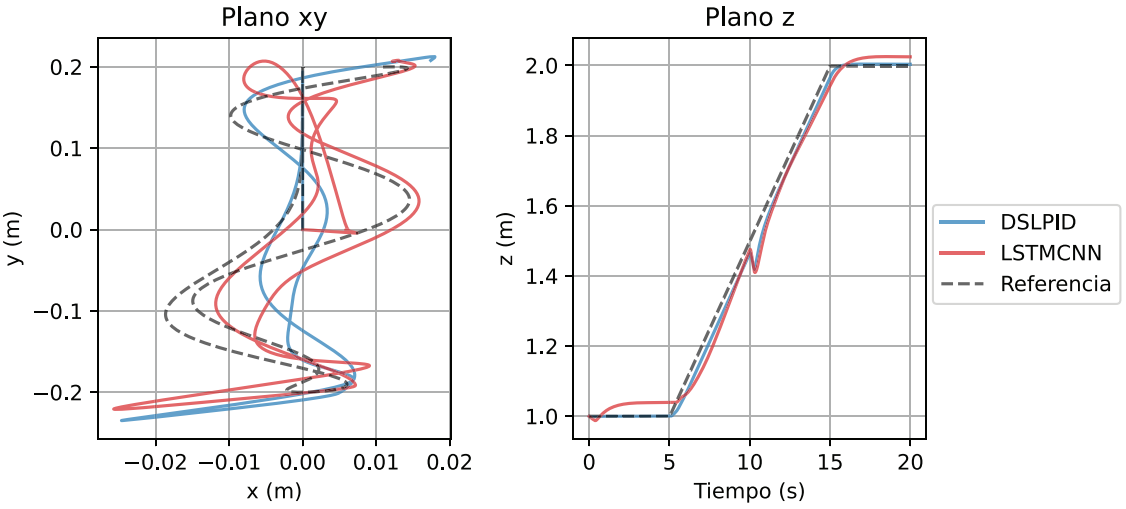
9.4.0.8. Vuelo en Altitud

Esta prueba se propone con el fin de observar el comportamiento del dron, dentro de uno de los estados menos visitados según el análisis realizado al conjunto de datos propuesto, como lo es el vuelo a gran altura, la respuesta a este escenario por parte de la red propuesta se observa en la figura 63.

Donde se observa que a medida que el dron aumenta la altura empieza a desestabilizar el vuelo a partir de 4m. Dada esta respuesta se encuentra que el la red consigue generalizar el vuelo a una altura superior a la mayoría de ejemplos en entrenamiento, pero solo 2m por encima de estos. Encontrando que para este caso el entrenamiento de la red necesitaría mas muestras en vuelos realizados a una altura superior a 4m, para encontrar una mejor dinámica de vuelo con respecto al controlador base.



(A) Vista 3D



(B) Vista Planos XY, Z

FIGURA 58: Vista complementaria respuesta a señales varias para LSTMCNN vs DSLPID

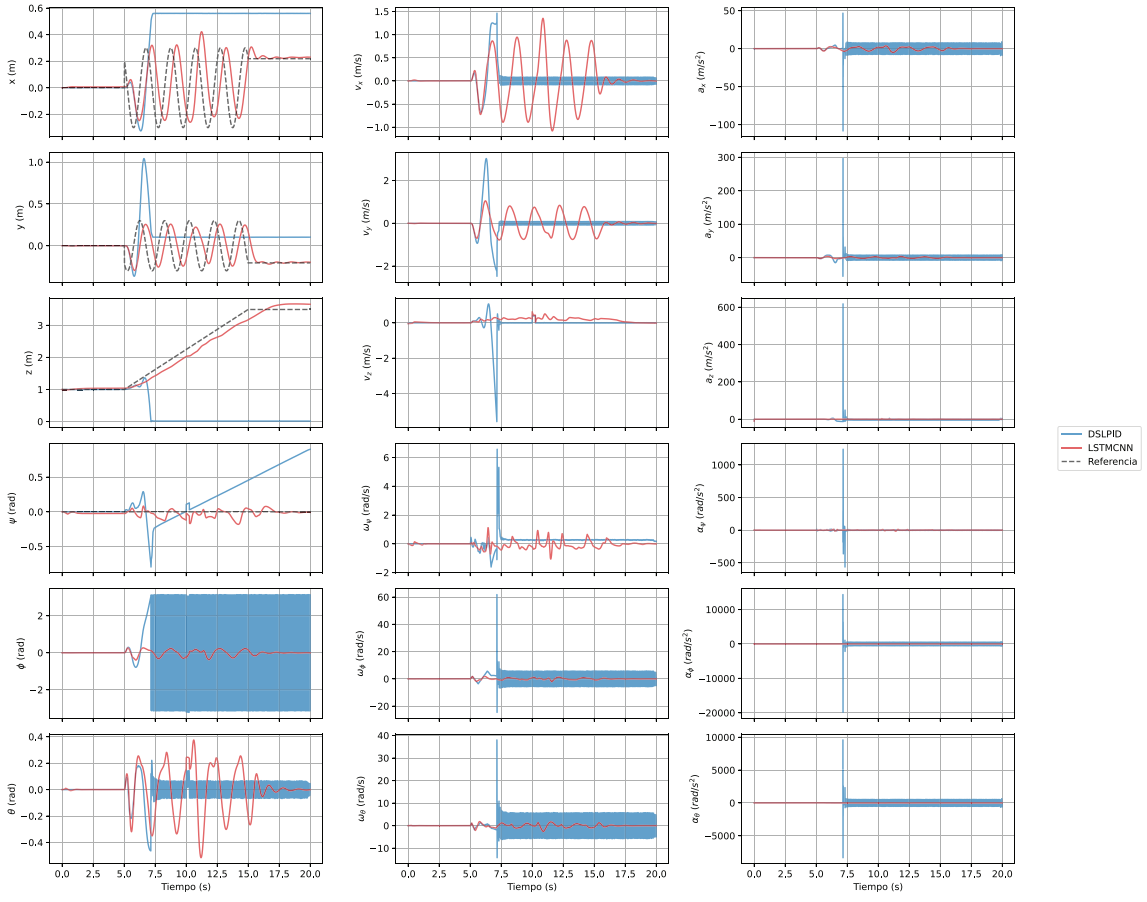
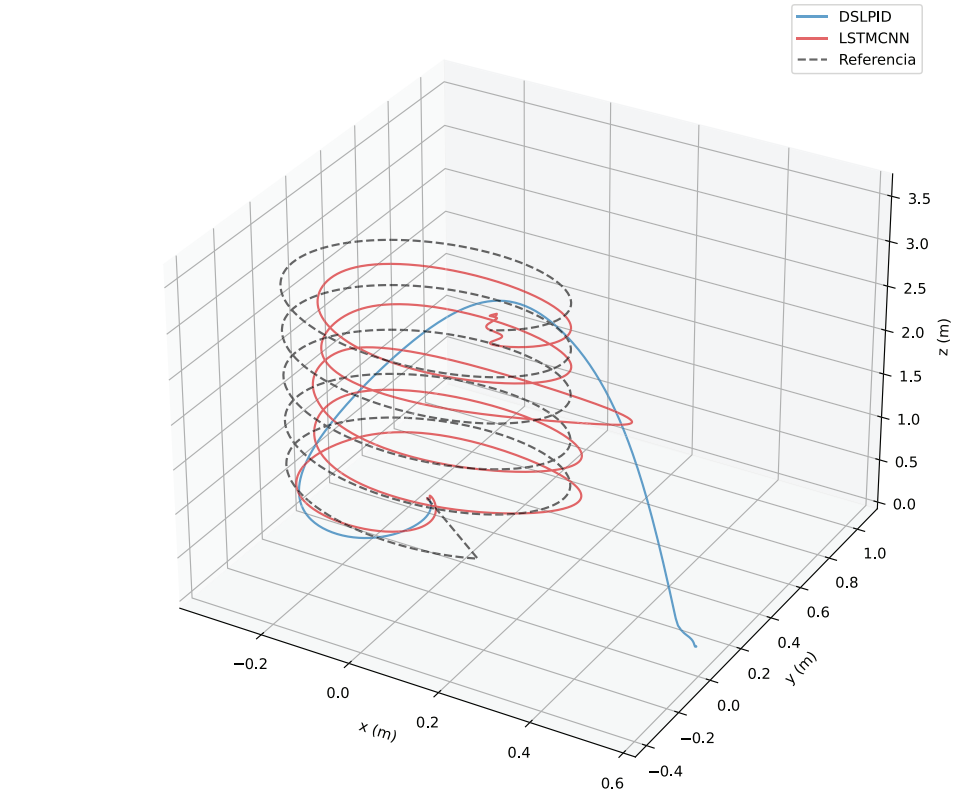


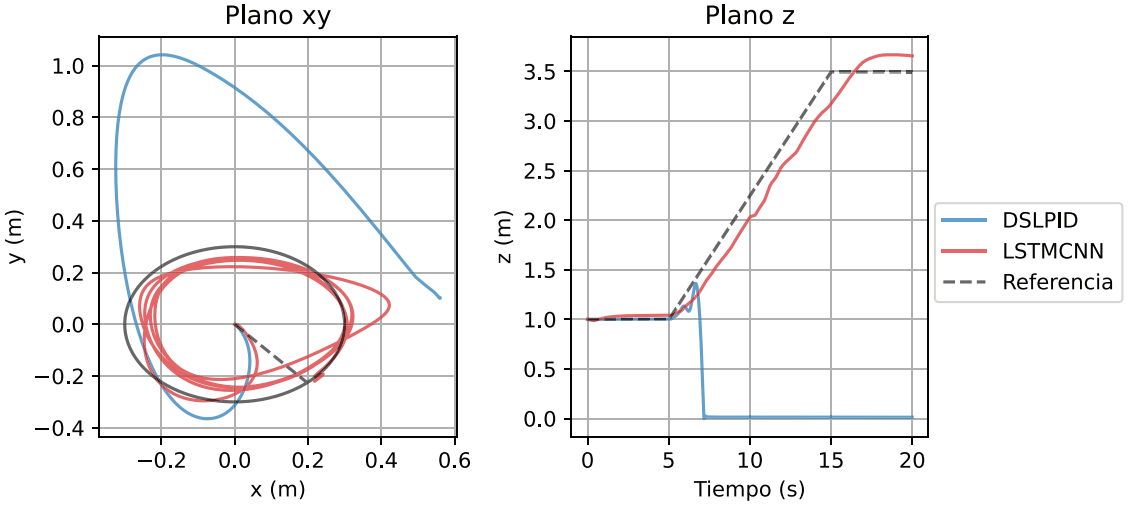
FIGURA 59: Estados de la respuesta a espiral para LSTMCNN vs DSLPID

9.4.0.9. Flujo Inducido (*Downwash)*

Debido a que el modelo entrenado se realizó con el motor de físicas básicas de *Pybullet* (Pyb) en un entorno ideal y con un modelo lineal, en el conjunto de datos no se refleja ningún efecto aerodinámico como el efecto suelo (G_i), arrastre (D) o flujo inducido W (ver sección 6.1.3.4). Por medio de este experimento, se quiere probar si el controlador neuronal tiene la capacidad de inferir comportamientos aerodinámicos externos. Para ello se ubica el controlador DSLPID en una posición inicial $x_0 = -0.1m$, $y_0 = 0m$, $z_0 = 1.5m$ y el controlador LSTMCNN $x_0 = 0.1m$, $y_0 = 0m$, $z_0 = 1m$. Posteriormente se les ingresa una señal sinusoidal de amplitud $A = 0.5m$ y frecuencia $f = 0.2Hz$ como referencia en x, pero desfasadas $\Psi = \pi rad$ (180 grados). De esta manera, cuando se crucen los caminos, el dron más cercano al suelo sufrirá una reducción en la fuerza de sustentación. En las figuras 65 y 64 se muestran los resultados del experimento.

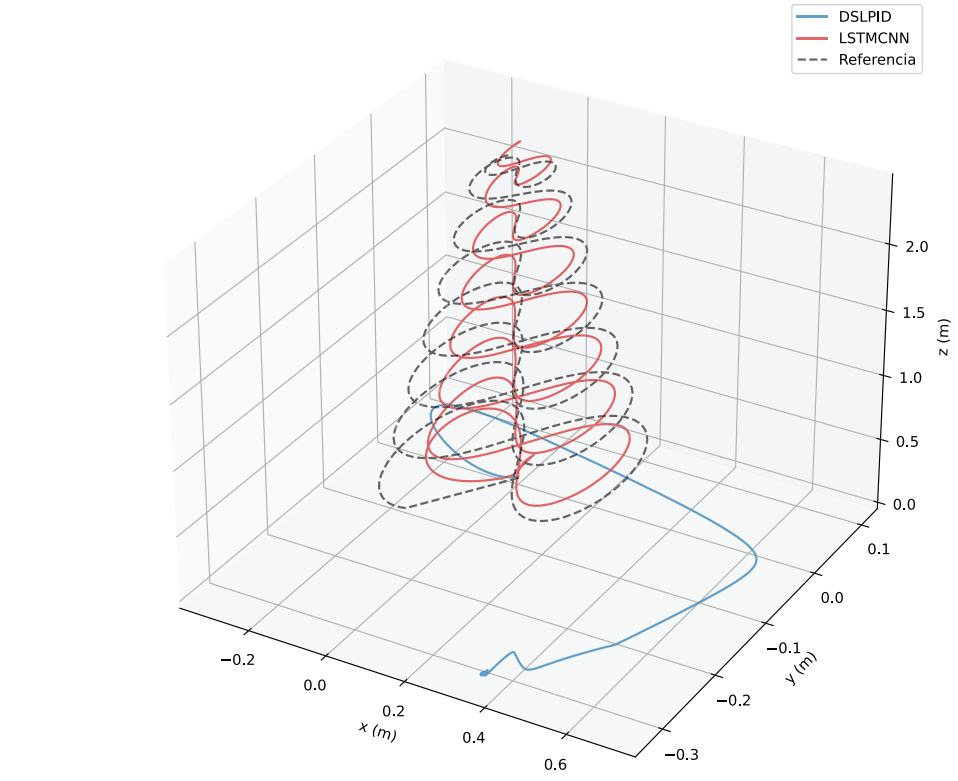


(A) Vista 3D

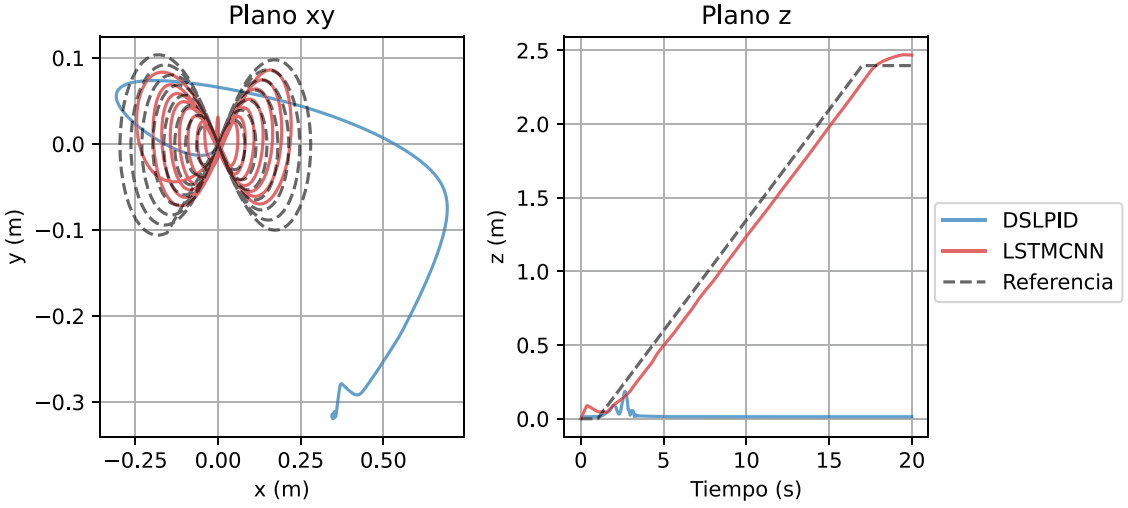


(B) Vista Planos XY, Z

FIGURA 60: Vista complementaria respuesta a espiral para LSTMCNN vs DSLPID



(A) Vista 3D



(B) Vista Planos XY, Z

FIGURA 61: Respuesta a función Lemniscate para LSTMCNN vs DSLPID

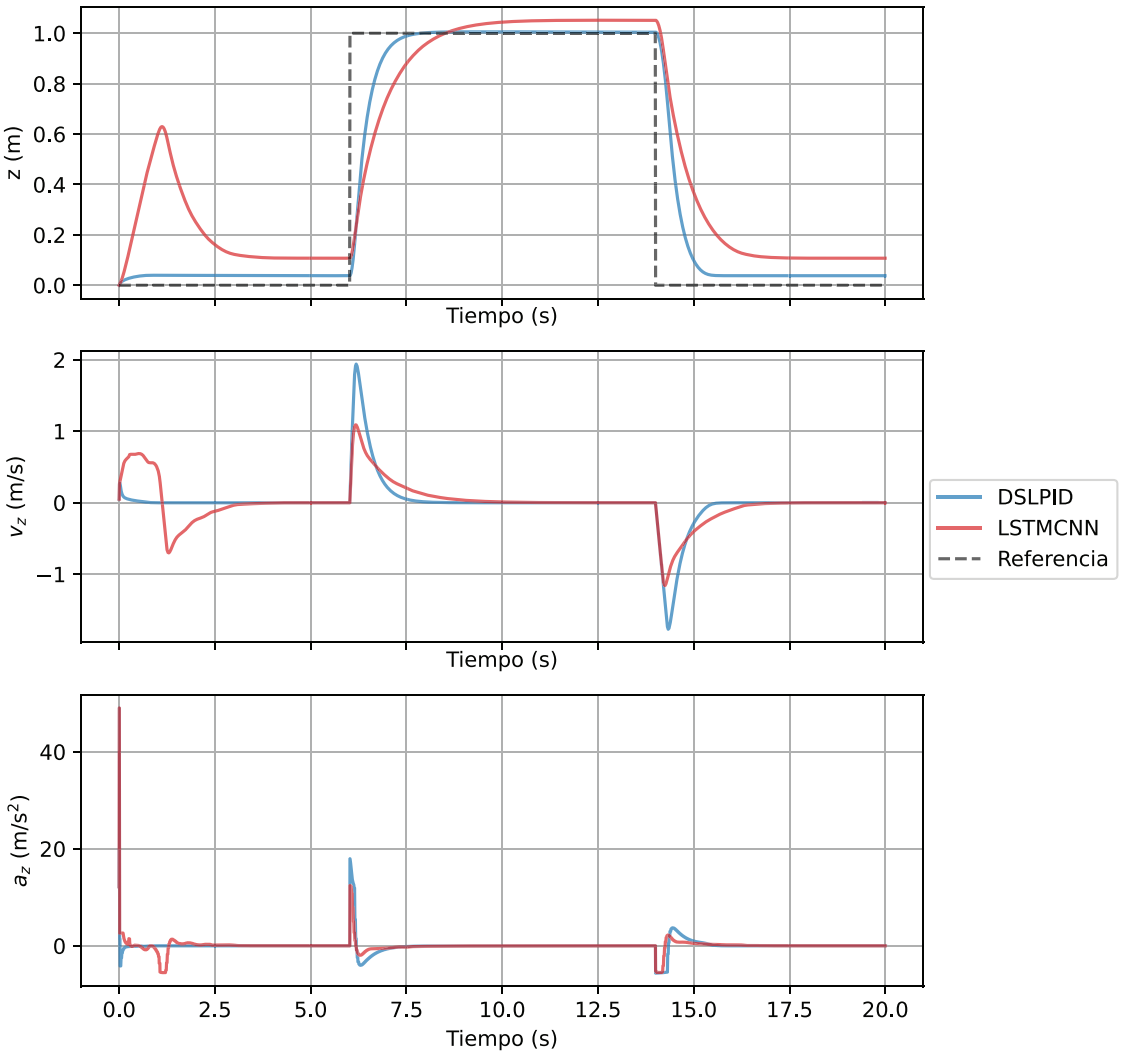


FIGURA 62: Despegue y aterrizaje LSTMCNN vs DSLPID

De la figura 64 se observa que el controlador es capaz de mantenerse estable siguiendo la trayectoria en x y recuperando su posición en y y z después de las perturbaciones provocadas cuando los drones cruzaban sus trayectorias a diferentes alturas. El controlador LSTMCNN pierde altura (aprox 1m), pero recupera el vuelo. El controlador DSLPID no sufre ninguna perturbación.

Se procede a comparar el rendimiento aplicando el efecto para el controlador DSLPID (se intercambia la posición inicial). En la figura 66 se muestran los resultados del experimento para el controlador DSLPID.

Para el controlador DSLPID se observa en la figura **??** que el dron sufre perturbación en el eje z debido al flujo inducido, sin embargo, la atenuación es menor y se recupera fácilmente. La atenuación para el

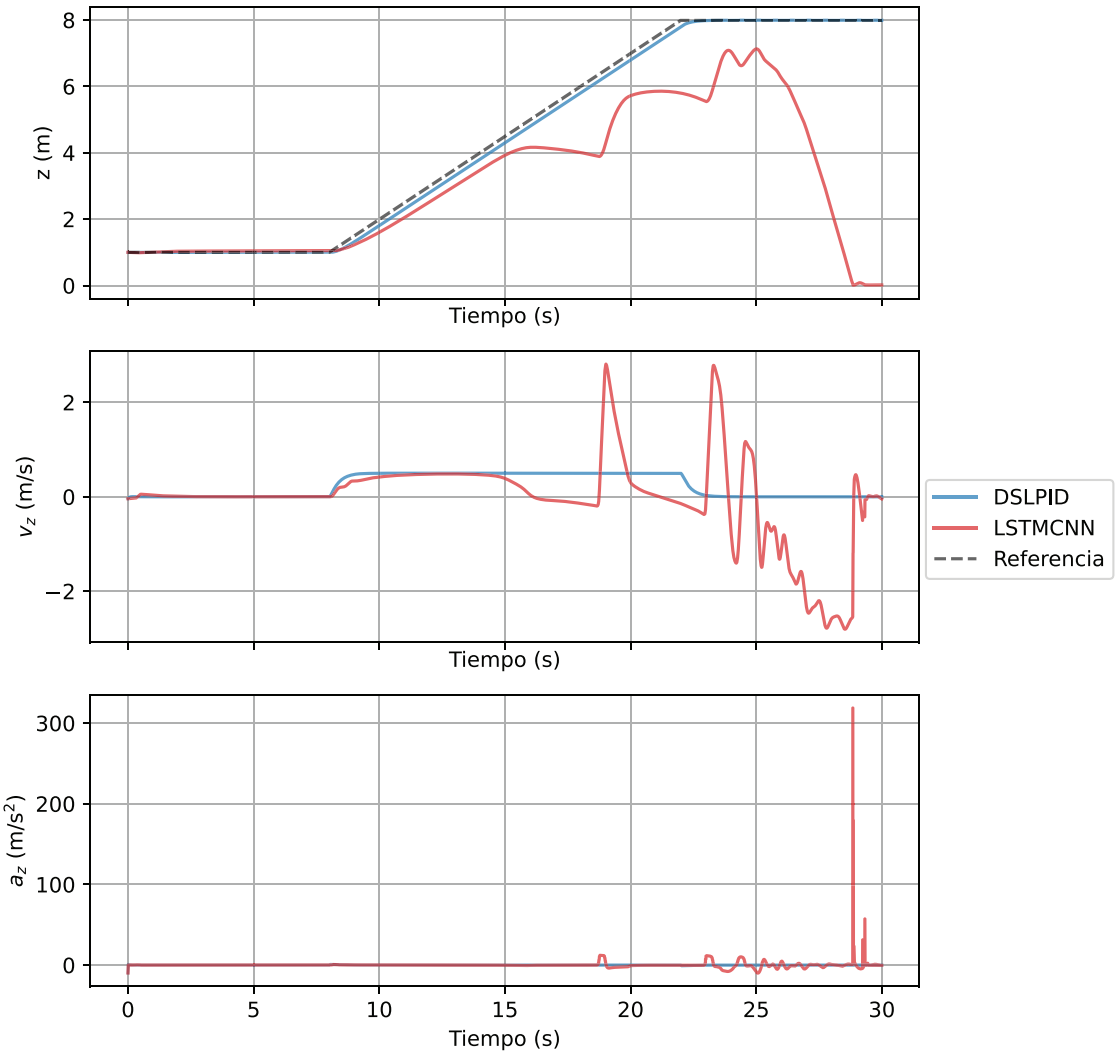


FIGURA 63: Despegue y aterrizaje LSTMCNN vs DSLPID

controlador DSLPID es de 0.03m en z y para el controlador LSTMCNN de 0.95m.

El controlador LSTMCNN fue capaz de generalizar comportamientos fuera del rango de operación del DSLPID, sin embargo no generaliza comportamientos de dinámicas no tenidas en cuenta en la simulación. Si bien, el controlador DSLPID fue sintonizado teniendo en cuenta estos efectos aerodinámicos, no se reflejan en el conjunto de datos debido a la simulación realizada con el motor de físicas básico de *Pybullet* (*Pyb*).

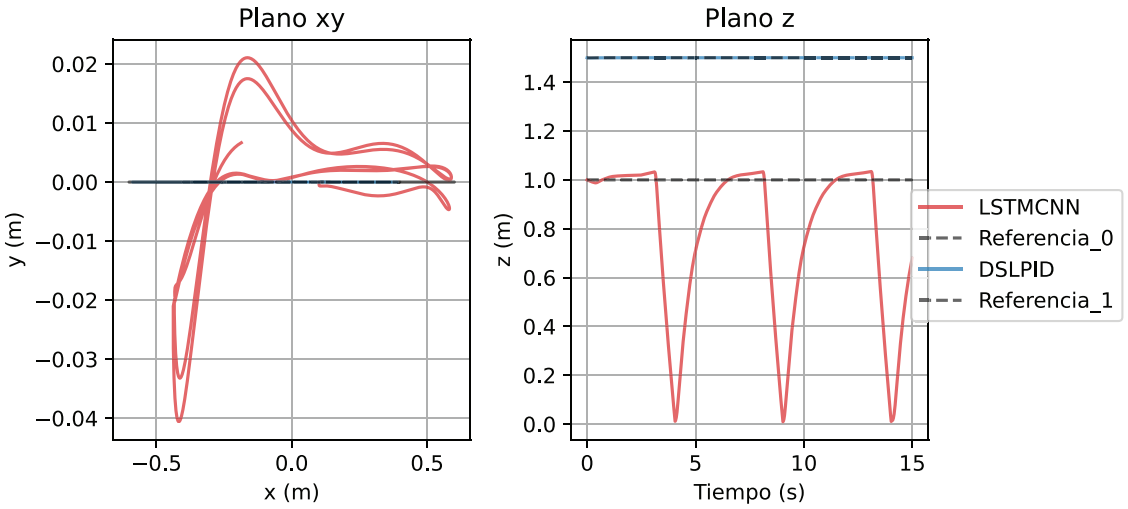


FIGURA 64: Vista flujo inducido aplicado sobre el controlador LSTMCNN

9.4.0.10. Planos Fase

Para este experimento se evaluara las respuestas para diferentes señales de referencia (Sinusoidal, Paso, y vuelo con perturbaciones) en posición contra velocidad para cada eje, esto permitirá comparar las soluciones de nuestro controlador y el controlador PID de referencia dentro de un plano de fase.

De la figura 67, se puede observar que para los planos X e Y se obtienen soluciones similares en ambos controladores, también que dentro de todas las trayectorias planteadas para los diferentes ejes el controlador neuronal converge en dirección al punto de referencia.

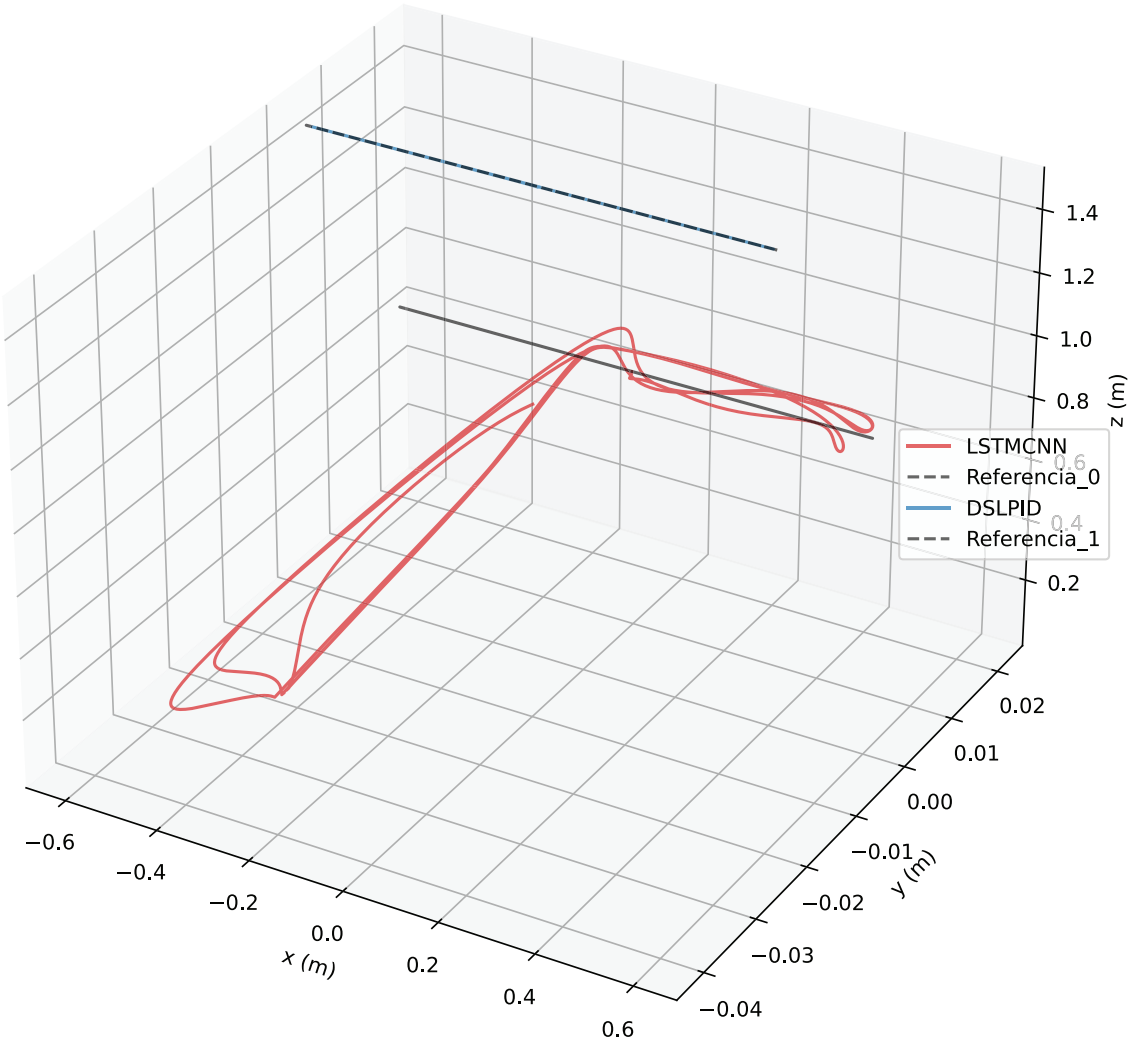


FIGURA 65: Vista planos XY, Z

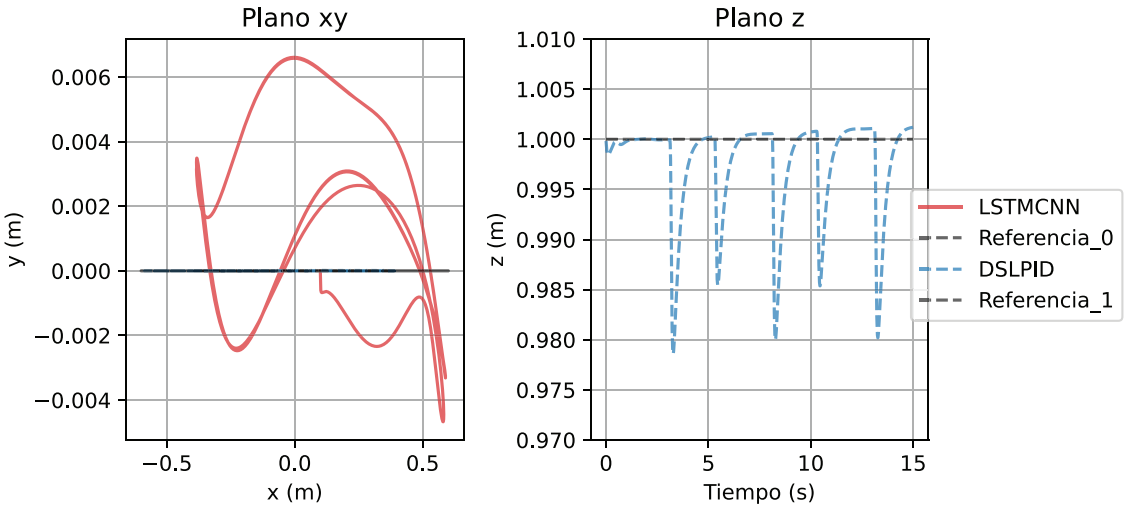
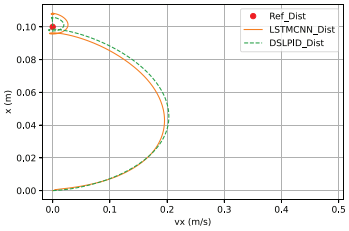
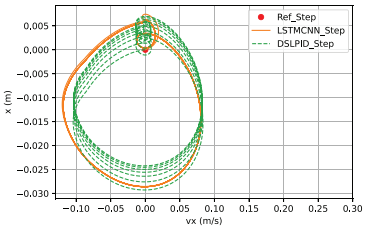


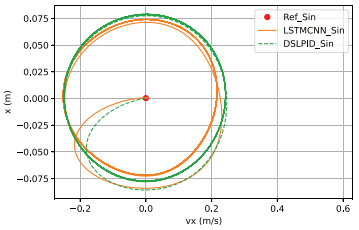
FIGURA 66: Vista flujo inducido aplicado sobre el controlador DSLPÌD



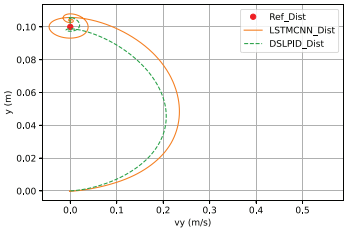
(A) [Trayectoria de Perturbaciones Plano x-vx.



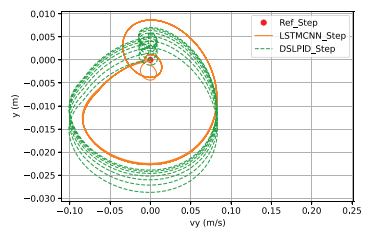
(B) Respuesta al Escalón Plano x-vx.



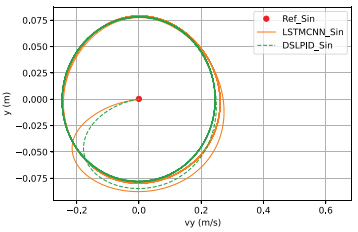
(C) Trayectoria Sinusoidal Plano x-vx.



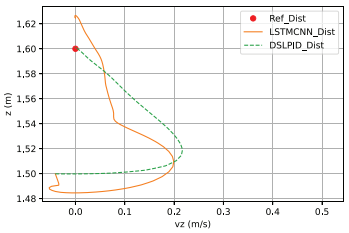
(D) [Trayectoria de Perturbaciones Plano y-vy.



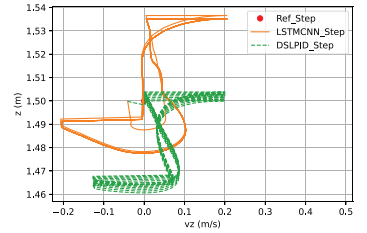
(E) Respuesta al Escalón Plano y-vy.



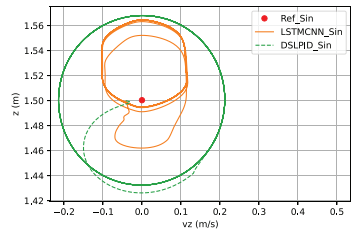
(F) Trayectoria Sinusoidal Plano y-vy.



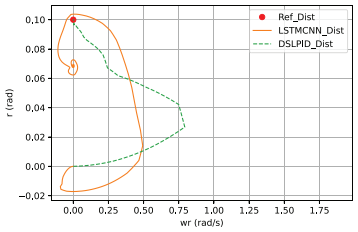
(G) Trayectoria de Perturbaciones Plano z-vz.



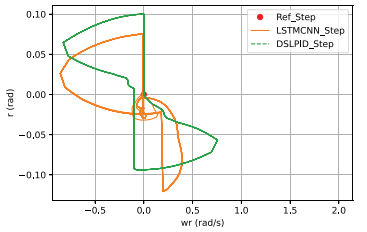
(H) Respuesta al Escalón Plano z-vz.



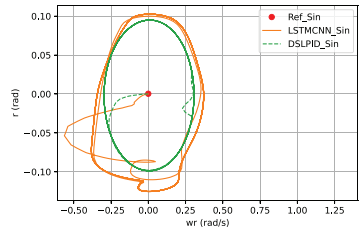
(I) Trayectoria Sinusoidal Plano z-vz.



(J) Trayectoria de Perturbaciones Planoo r-wr.



(K) Respuesta al Escalón Plano r-wr.



(L) Trayectoria Sinusoidal Plano r-wr.

FIGURA 67: Evaluación del plano-fase.

Capítulo 10

Impacto Social

Con este proyecto se propone aportar una base con respecto a la investigación y desarrollo de sistemas de control para vehículos aéreos multi-rotor con técnicas de aprendizaje profundo. El desarrollo planteado podría utilizarse para mejorar el rendimiento de procesos en múltiples aplicaciones donde los UAVs ya operan, como la seguridad, fotografía, agricultura, entretenimiento, gastronomía, mantenimiento, investigación y cualquier otro enfoque donde se requiera un vuelo autónomo y estable para capturar o entregar información.

Capítulo 11

Conclusiones

Se implementó un controlador basado en una red neuronal profunda recurrente (LSTM-CNN) que permite controlar la posición en los ejes x, y, z y ψ de un UAV en *Pybullet* para el dron *Crazyflie 2.0*, mediante el planteamiento de un problema de regresión, utilizando técnicas de aprendizaje supervisado. La metodología propuesta se puede aplicar para otros modelos, controladores, entornos e incluso otros sistemas de control.

El controlador seleccionado como base del sistema fue el $DSLPID$, el cual es un controlador que respondía en menor tiempo, con menor error y demostraba un mayor rango de estabilidad que el controlador $SimplePID$, sin embargo es un controlador lineal que opera sobre el rango de operación [$x = 0, y = 0, z = 1, p = 0, q = 0, r = 0$].

Dentro del conjunto de datos generado para el entrenamiento del controlador, se logró explorar una gran variedad de estados y obtener trayectorias con diferentes componentes frecuenciales. Las trayectorias que mayores componen armónicos aportan son las señales nulas con perturbaciones, ruido de baja frecuencia y señales escalonadas aleatorias. Éstas últimas se encontraron fundamentales para que el controlador adquiriera información sobre el estado estable. Sin embargo, no se logró generalizar los efectos aerodinámicos no evidenciados en el conjunto de datos debido a que se construyó con el motor de físicas básico (PYB).

Se exploró distintas arquitecturas de redes neuronales (*ANN*, *ANN realimentada*, *LSTM*, *CLSTM*, *LSTM-CNN*), encontrándose que una mejor predicción para la arquitectura *LSTM-CNN*. El uso de algoritmo *Hyperband* ayudó a encontrar la arquitectura que mejor se ajustaba a los datos. El tamaño de la ventana se determinó utilizando métodos empíricos y de autocorrelación parcial. El método empírico permitió obtener un balance entre la precisión y el tiempo de inferencia del modelo. La autocorrelación parcial permitió determinar el número de muestras en donde se alcanzaba mayor precisión.

Para la evaluación del sistema se propuso distintos experimentos, los cuales incluían respuesta al escalón, señal de control fuera de los límites de estabilidad del controlador base, señal con

perturbaciones o trayectorias agresivas. Debido a la metodología propuesta, el rendimiento del controlador entrenado depende del controlador base y será máximo igual o peor que éste dentro de su rango de operación, tal como se observo en las pruebas planteadas donde la red obtuvo un mayor error acumulado en comparación con el controlador base, esto gracias a que el error con respecto al sistema de referencia es una magnitud positiva $(MSE >= 0)$. No obstante, se observó que el modelo basado en redes neuronales adquiere un mejor rendimiento y estabilidad que el controlador base cuando se aleja de dicho punto de operación, mostrando capacidades de generalización, especialmente con trayectorias agresivas.

Capítulo 12

Recomendaciones y trabajo futuro

Como continuación de este proyecto se recomienda, obtener un conjunto de datos en el cual se añadan trayectorias enfocadas a una caracterización más compleja del comportamiento aerodinámico del dron, e mediante el uso de un controlador con mejor desempeño y la generación de un conjunto de datos con un simulador que maneje un motor de físicas con un modelo dinámico, sensores o mediante la generación del conjunto de datos en un entorno real.

Para obtener un mejor desempeño del controlador propuesto, se propone transferir los pesos del controlador neuronal al actor dentro de un modelo de aprendizaje por refuerzo, por ejemplo un optimizador de políticas próximas (PPO). Esto con base en 3 observaciones especificas:

1. El desempeño de la red presentada bajo el modelo de aprendizaje supervisado, se encuentra limitado al comportamiento del controlador.
2. Aunque el modelo de aprendizaje por refuerzo toma demasiado tiempo en converger para el problema planteado, al tomar la red presentada en este proyecto como base se puede reducir el tiempo de entrenamiento, iterando de forma mas sencilla sobre los diferentes hiper-parámetros del algoritmo.
3. El algoritmo PPO bajo el modelo de aprendizaje por refuerzo ofrece los mejores resultados en cuanto a la aplicación del control de vuelo, como es descrito en la sección 3.

Para este modelo de aprendizaje por refuerzo adicionalmente es recomendado aplicar una variación de la función F_1 presentada en este proyecto, como función de recompensa. Esto con base en los buenos resultados obtenidos por parte de esta función para la sintonización del controlador de vuelo propuesto por MATLAB/Simulink para el eje ϕ en el Parrot Mambo (Ver sección 7).

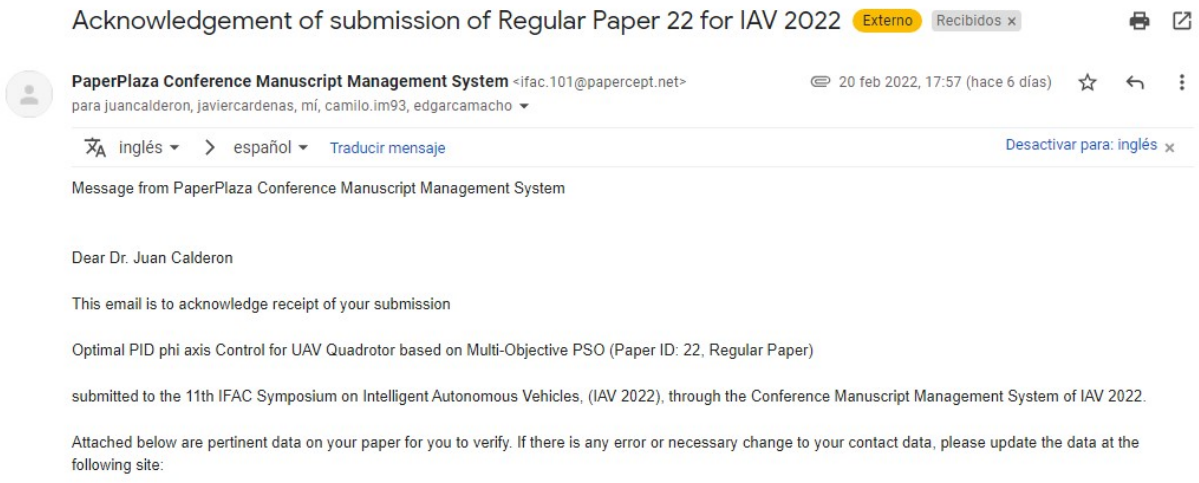
Finalmente se propone evaluar el desempeño en un entorno físico e identificar las dinámicas que no se tienen en cuenta dentro de la simulación. De esta forma generar un conjunto de datos que caracterice

mejor el vuelo del dron. Para dicha implementación se recomienda tener en cuenta el tiempo de inferencia del modelo (este caso es $130ms$), ya que no puede superar el tiempo máximo de respuesta de control (para este caso $20.8ms$). La disminución en dicho tiempo de inferencia se podría realizar mediante la técnica de *pruning* en la cual se reducen los pesos mas bajos de la red neuronal presentada, consiguiendo un modelo con menos parámetros y mayor eficiencia computacional. Así mismo, se recomienda el uso de hardware especializado en la inferencia de redes neuronales y de la implementación en un sistema operativo en tiempo real (RTOS).

Capítulo 13

Trabajos relacionados

En consecuencia del desarrollo de este trabajo de grado se encuentra bajo evaluación de La Federación de Control Automático (IFAC) un articulo titulado ″Optimal PID ϕ axis Control for UAV Quadrotor based on Multi-Objective PSO″ . Para participar en la conferencia de Vehículos Autónomos Inteligentes IAV 2022.



Acknowledgement of submission of Regular Paper 22 for IAV 2022 Externo Recibidos ×

PaperPlaza Conference Manuscript Management System <ifac.101@papercept.net> 20 feb 2022, 17:57 (hace 6 días)

para juancalderon, javiercardenas, mí, camilo.im93, edgarcamacho

inglés > español Traducir mensaje Desactivar para: inglés ×

Message from PaperPlaza Conference Manuscript Management System

Dear Dr. Juan Calderon

This email is to acknowledge receipt of your submission

Optimal PID phi axis Control for UAV Quadrotor based on Multi-Objective PSO (Paper ID: 22, Regular Paper)

submitted to the 11th IFAC Symposium on Intelligent Autonomous Vehicles, (IAV 2022), through the Conference Manuscript Management System of IAV 2022.

Attached below are pertinent data on your paper for you to verify. If there is any error or necessary change to your contact data, please update the data at the following site:

Optimal PID ϕ axis Control for UAV Quadrotor based on Multi-Objective PSO

Javier Alexis Cárdenas * **Uriel Eduardo Carrero** *
Edgar Camilo Camacho * **Juan Manuel Calderón** **

* *Facultad de Ingeniería Electrónica, Universidad Santo Tomás, Bogotá, Colombia (e-mail: {javiercardenas, urielcarrero, juancalderon}@usantotomas.edu.co)*
** *Department of Computer Science and Engineering, Bethune Cookman University, Daytona FL, (e-mail: calderonj@cookman.edu), Universidad Santo Tomás, Bogotá, Colombia (e-mail: juancalderon@usantotomas.edu.co)*

Abstract: In this paper, we tackle the problem of PID Control tuning for a Quadrotor. This task is vital given the several applications that a UAV quadrotor can perform. This paper focuses on the optimal tuning of a PID controller in ϕ axes. We propose using a Multi-objective Particle Swarm Optimization (PSO) algorithm for PID performance improvement. The proposed system is evaluated in a Parrot Mambo FPV UAV Quadrotor virtual model using MATLAB Simulink environment. We compare four different cost functions for the multi-objective optimization approach. Those cost functions focus on improving the UAV performance based on the settling time, overshoot, steady-state error, and control effort. The evaluation system shows that the best performance is reached in combining the different factors through the Multi-objective PSO algorithm.

Keywords: UAV Quadrotor, PID Optimal Control, PSO, Multi-objective optimization

Arquitecturas de Redes Neuronales

ANN

TABLA 42: Arquitectura ANN 1

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	320	relu	450K
FC_2	704	relu	225K
FC_3	192	relu	135K
FC_4	704	relu	135K
FC_5	704	relu	496K
FC_6	4	Lineal	2.8K
Total Parámetros		1.4M	

TABLA 43: Arquitectura ANN 2

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	32	relu	45K
FC_2	512	relu	16K
FC_3	32	relu	16K
FC_4	512	relu	16K
FC_5	704	relu	361K
FC_6	4	Lineal	2.8K
Total Parámetros		459K	

TABLA 44: Arquitectura ANN 3

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	96	relu	135K
FC_2	256	relu	24K
FC_3	96	relu	24K
FC_4	64	relu	6K
FC_5	704	relu	45K
FC_6	4	Lineal	2.8K
Total Parámetros		239K	

ANN Feedback

TABLA 45: Arquitectura ANN 4

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	256	relu	360K
FC_2	704	relu	180K
FC_3	192	relu	135K
FC_4	704	relu	135K
FC_5	256	relu	180K
FC_6	4	Lineal	1K
Total Parámetros		994K	

TABLA 46: Arquitectura ANN 5

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	512	relu	721K
FC_2	320	relu	164K
FC_3	192	relu	61K
FC_4	96	relu	18K
FC_5	64	relu	6.2K
FC_6	4	Lineal	206
Total Parámetros		972K	

TABLA 47: Arquitectura ANN Feedback 1

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	96	relu	159K
FC_2	96	relu	9K
FC_3	512	relu	49K
FC_4	320	relu	164K
FC_5	64	relu	20K
FC_6	4	Lineal	260
Total Parámetros		403K	

TABLA 48: Arquitectura ANN Feedback 2

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	256	relu	426K
FC_2	32	relu	8K
FC_3	192	relu	6K
FC_4	96	relu	18K
FC_5	64	relu	6K
FC_6	4	Lineal	260
Total Parámetros		465K	

LSTM

LSTMCNN

TABLA 49: Arquitectura ANN Feedback 3

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	32	relu	53K
FC_2	32	relu	2K
FC_3	96	relu	3K
FC_4	96	relu	9K
FC_5	96	relu	9K
FC_6	4	Lineal	388
Total Parámetros		76K	

TABLA 50: Arquitectura ANN Feedback 4

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	192	relu	319K
FC_2	256	relu	49K
FC_3	96	relu	24K
FC_4	256	relu	24K
FC_5	192	relu	49K
FC_6	4	Lineal	772
Total Parámetros		994K	

TABLA 51: Arquitectura ANN Feedback 5

Capa	**Neuronas**	**Activación**	# Parámetros
FC_1	320	relu	532K
FC_2	256	relu	82K
FC_3	256	relu	65K
FC_4	64	relu	16K
FC_5	512	relu	33K
FC_6	4	Lineal	2K
Total Parámetros		732K	

CLSTM

TABLA 52: Arquitectura LSTM 1

Capa	Neuronas	Activación	# Parámetros
LSTM_1	224	tanh	221K
LSTM_2	192	tanh	320K
LSTM_3	128	tanh	164K
LSTM_4	192	tanh	246K
LSTM_5	96	tanh	110K
FC_1	448	tanh	43K
FC_2	192	tanh	86K
FC_3	224	tanh	43K
FC_4	224	tanh	50K
FC_5	512	tanh	1.1M
FC_6	4	Lineal	2052
Total Parámetros		1.4M	

TABLA 53: Arquitectura LSTM 2

Capa	Neuronas	Activación	# Parámetros
LSTM_1	64	tanh	22K
FC_1	192	relu	12K
FC_2	416	relu	80K
FC_3	192	relu	80K
FC_4	416	relu	80K
FC_5	256	relu	106K
FC_6	4	Lineal	1028
Total Parámetros		383K	

TABLA 54: Arquitectura LSTM 3

Capa	Neuronas	Activación	# Parámetros
LSTM_1	224	tanh	221K
LSTM_2	192	tanh	320K
LSTM_3	128	tanh	164K
LSTM_4	192	tanh	246K
LSTM_5	196	tanh	304K
FC_1	448	tanh	88K
FC_2	192	tanh	86K
FC_3	224	tanh	43K
FC_4	4	Lineal	900
Total Parámetros		1.4M	

TABLA 55: Arquitectura LSTM 4

Capa	Neuronas	Activación	# Parámetros
LSTM_1	320	tanh	439K
LSTM_2	128	tanh	229K
LSTM_3	128	tanh	131K
FC_1	64	relu	8K
FC_2	96	relu	6K
FC_3	320	relu	31K
FC_4	192	relu	61K
FC_5	4	Lineal	772
Total Parámetros		908K	

TABLA 56: Arquitectura LSTM 5

Capa	Neuronas	Activación	# Parámetros
LSTM_1	416	tanh	730K
LSTM_2	288	tanh	812K
LSTM_3	384	tanh	1M
LSTM_4	288	tanh	775K
FC_1	448	tanh	129K
FC_2	192	tanh	86K
FC_3	224	tanh	43K
FC_4	224	tanh	50K
FC_5	512	tanh	115K
FC_6	4	Lineal	2k
Total Parámetros		3.7M	

TABLA 57: Arquitectura LSTMCNN 1

Capa	Neuronas/Filtros	Activación	# Parámetros
LSTM_1	320	tanh	439K
CONV1D_1	480	relu	461K
LSTM_2	128	tanh	311K
CONV1D_2	288	relu	110K
LSTM_3	128	tanh	213K
FC_1	64	relu	8K
FC_2	96	relu	6K
FC_3	320	relu	31K
FC_4	192	relu	61K
FC_5	4	Lineal	772
Total Parámetros		1.6M	

TABLA 58: Arquitectura LSTMCNN 2

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
LSTM_1	320	tanh	439K
CONV1D_1	320	relu	307K
LSTM_2	32	tanh	45K
CONV1D_2	288	relu	28K
LSTM_3	128	tanh	213K
FC_1	32	relu	4K
FC_2	416	relu	137K
FC_3	288	relu	120K
FC_4	4	Lineal	1.1k
Total Parámetros		1.1M	

TABLA 59: Arquitectura LSTMCNN 3

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
LSTM_1	288	tanh	358K
CONV1D_1	96	relu	147K
LSTM_2	192	tanh	221K
FC_1	128	relu	24K
FC_2	192	relu	24K
FC_4	4	Lineal	772
Total Parámetros		778K	

TABLA 60: Arquitectura LSTMCNN 4

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
LSTM_1	192	tanh	165K
CONV1D_1	128	relu	164K
LSTM_2	96	tanh	86K
CONV1D_2	128	relu	115K
LSTM_3	96	tanh	86K
FC_1	416	relu	40K
FC_2	128	relu	53K
FC_4	4	Lineal	516
Total Parámetros		711K	

TABLA 61: Arquitectura LSTMCNN 5

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
LSTM_1	320	tanh	439K
CONV1D_1	480	relu	461K
LSTM_2	128	tanh	311K
CONV1D_2	288	relu	110K
LSTM_3	128	tanh	213K
FC_1	64	relu	8K
FC_2	96	relu	6K
FC_3	320	relu	31K
FC_4	192	relu	61K
FC_5	4	Lineal	772
Total Parámetros		1.6M	

TABLA 62: Arquitectura CLSTM 1

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
CONV1D_1	480	relu	32K
CONV1D_2	288	relu	411K
LSTM_1	320	tanh	779K
LSTM_2	128	tanh	229K
LSTM_3	128	tanh	131K
FC_1	64	relu	8K
FC_2	96	relu	6K
FC_3	320	relu	31K
FC_4	192	relu	61K
FC_5	4	Lineal	772
Total Parámetros		1.7M	

TABLA 63: Arquitectura CLSTM 2

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
CONV_1	224	relu	307K
MaxPooling1d	224		0
CONV_2	224	relu	307K
MaxPooling1d	224		0
CONV_3	416	relu	307K
MaxPooling1d	416		0
LSTM_1	224	tanh	439K
LSTM_2	192	tanh	45K
LSTM_3	128	tanh	213K
LSTM_4	192	tanh	213K
LSTM_3	96	tanh	213K
FC_1	448	relu	4K
FC_2	192	relu	137K
FC_3	224	relu	120K
FC_4	4	Lineal	1.1k
Total Parámetros		2M	

TABLA 64: Arquitectura CLSTM 3

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
CONV_1	160	relu	10K
MaxPooling1d	160		0
CONV_2	288	relu	138K
MaxPooling1d	288		0
LSTM_1	192	tanh	369K
LSTM_2	96	tanh	110K
FC_1	96	relu	9K
FC_2	480	relu	46K
FC_3	64	relu	30K
FC_4	256	relu	16K
FC_5	4	Lineal	1k
Total Parámetros		733K	

TABLA 65: Arquitectura CLSTM 4

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
LSTM_1	192	tanh	165K
CONV_1	128	relu	164K
LSTM_2	96	tanh	86K
CONV_2	128	relu	115K
LSTM_3	96	tanh	86K
FC_1	416	relu	40K
FC_2	128	relu	53K
FC_4	4	Lineal	516
Total Parámetros		711K	

TABLA 66: Arquitectura CLSTM 5

Capa	**Neuronas/Filtros**	**Activación**	**# Parámetros**
CONV_1	224	relu	13K
MaxPooling1d	224		0
CONV_2	224	relu	15K
MaxPooling1d	224		0
CONV_3	416	relu	279K
MaxPooling1d			0
LSTM_1	224	tanh	574K
LSTM_2	192	tanh	320K
LSTM_3	128	tanh	164K
LSTM_4	192	tanh	246K
LSTM_5	96	tanh	110K
FC_1	448	relu	43K
FC_2	192	relu	86K
FC_3	224	relu	43K
FC_4	4	lineal	900

Bibliografía

[1] Kavita Gupta, Sandhya Bansal y Rajiv Goel. «Uses of Drones In Fighting COVID-19 Pandemic». En: *2021 10th International Conference on System Modeling Advancement in Research Trends (SMART)*. 2021, págs. 651-655.

[2] M. A. Boon, A. P. Drijfhout y S. Tesfamichael. «Comparison of a fixed-wing and multi-rotor UAV for environmental mapping applications: A case study». En: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives* 42 (2W6 ago. de 2017), págs. 47-54. ISSN: 16821750. DOI: 10.5194/ISPRS-ARCHIVES-XLII-2-W6-47-2017.

[3] Jianhai Zhang y col. «Approximation Capability of a Novel Neural Network Model for Dynamic Systems». En: *2009 Second International Conference on Intelligent Computation Technology and Automation*. Vol. 1. 2009, págs. 59-62. DOI: 10.1109/ICICTA.2009.23.

[4] Gang Shi y Shuxing Yang. «Intelligent control of UAV with neuron-fuzzy approach under hierarchical architecture». En: *2008 7th World Congress on Intelligent Control and Automation*. 2008, págs. 5238-5243. DOI: 10.1109/WCICA.2008.4594539.

[5] Ashvin Nair y col. *AWAC: Accelerating Online Reinforcement Learning with Offline Datasets*. 2021. arXiv: 2006.09359 [cs.LG].

[6] Saith Rodríguez y col. «Fast path planning algorithm for the robocup small size league». En: *Robot Soccer World Cup*. Springer. 2014, págs. 407-418.

[7] Saith Rodríguez y col. «STOx's 2016 Team description paper». En: (2013).

[8] Gustavo A Cardona y Juan M Calderon. «Robot swarm navigation and victim detection using rendezvous consensus in search and rescue operations». En: *Applied Sciences* 9.8 (2019), pág. 1702.

[9] Jose León y col. «Robot swarms theory applicable to seek and rescue operation». En: *International Conference on Intelligent Systems Design and Applications*. Springer. 2016, págs. 1061-1070.

[10] Wilson O Quesada y col. «Leader-follower formation for uav robot swarm based on fuzzy logic theory». En: *International Conference on Artificial Intelligence and Soft Computing*. Springer. 2018, págs. 740-751.

[11] Juan D Pabon y col. «Event-Triggered Control for Weight-Unbalanced Directed Robot Networks». En: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, págs. 5831-5836.

[12] Gustavo A Cardona y col. «Robust adaptive synchronization of interconnected heterogeneous quadrotors transporting a cable-suspended load». En: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, págs. 31-37.

[13] Gustavo A Cardona y col. «Adaptive Multi-Quadrotor Control for Cooperative Transportation of a Cable-Suspended Load». En: *2021 European Control Conference (ECC)*. IEEE. 2021, págs. 696-701.

[14] Nestor I Ospina y col. «Argrohbots: An affordable and replicable ground homogeneous robot swarm testbed». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 256-261.

[15] Edgar C Camacho, Nestor I Ospina y Juan M Calderón. «COVID-Bot: UV-C Based Autonomous Sanitizing Robotic Platform for COVID-19». En: *Ifac-papersonline* 54.13 (2021), págs. 317-322.

[16] Edgar C Camacho, Jose Guillermo Guarnizo, Juan M Calderon y col. «Design and Construction of a Cost-Oriented Mobile Robot for Domestic Assistance». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 293-298.

[17] GA Cardona y col. «Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning». En: *2019 SoutheastCon*. IEEE. 2019, págs. 1-7.

[18] Nicolás Gómez y col. «Leader-follower Behavior in Multi-agent Systems for Search and Rescue Based on PSO Approach». En: *SoutheastCon 2022*. IEEE. 2022, págs. 413-420.

[19] Laura J Padilla Reyes y col. «Adaptable Recommendation System for Outfit Selection with Deep Learning Approach». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 605-610.

[20] Daniel A Rincón-Riveros y col. «Automation System Based on NLP for Legal Clinic Assistance». En: *IFAC-PapersOnLine* 54.13 (2021), págs. 283-288.

[21] *Flying High: Drone Use in Fisheries Research - FISHBIO Fisheries Research, Monitoring, and Conservation*. URL: `https://fishbio.com/field-notes/the-fish-report/flying-high-drone-use-fisheries-research` (visitado 10-09-2020).

[22] Juan Lastra. «Diseño de un dron programable de bajo costo». En: *Repositorio Universidad de Cantabria* (ago. de 2017), págs. 1-132. URL: http://hdl.handle.net/10902/12091.

[23] *Wissenschaft und Forschung*. URL: https://www.microdrones.com/de/anwendungen/wissenschaft-und-forschung/ (visitado 10-09-2020).

[24] Francesco Rodella. *El desafío de hacer volar drones sin ayuda humana | Tecnología | EL PAÍS*. Oct. de 2018. URL: https://elpais.com/tecnologia/2018/10/18/actualidad/1539884534_864286.html (visitado 29-09-2020).

[25] Bello Guisado Ángel. «Diseño de controladores de vuelo para un dron modelo PARROT Mambo Minidrone». Feb. de 2019. URL: http://hdl.handle.net/10251/117441.

[26] By Hyon Lim y col. «Open-Source Projects on Unmanned Aerial Vehicles». En: *IEEE Robotics and Automation Magazine* SEPTEMBER 2012 (2012), págs. 33-45. DOI: 10.1109/MRA.2012.2205629.

[27] Madhu Babu Vankadari y col. «A Reinforcement Learning Approach for Autonomous Control and Landing of a Quadrotor». En: *2018 International Conference on Unmanned Aircraft Systems, ICUAS 2018*. Institute of Electrical y Electronics Engineers Inc., ago. de 2018, págs. 676-683. ISBN: 9781538613535. DOI: 10.1109/ICUAS.2018.8453468.

[28] Hugo Manuel Romero Pineda y Daniel Felipe Torres Cardozo. «Análisis Dinámico del Fallo de Rotores en un Hexacóptero». Tesis doct. Bogotá D.C.: Universidad Distrital Francisco José de Caldas, jun. de 2018, pág. 83. URL: http://hdl.handle.net/11349/13866.

[29] P. Castillo, R. Lozano y A. Dzul. «Stabilization of a mini rotorcraft with four rotors». En: *IEEE Control Systems Magazine* 25.6 (2005), págs. 45-55.

[30] Gabriel M. Hoffmann y col. «Quadrotor helicopter flight dynamics and control: Theory and experiment». En: *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference 2007*. Vol. 2. 2007, págs. 1670-1689. ISBN: 1563479044. DOI: 10.2514/6.2007-6461.

[31] S. Norouzi Ghazbi y col. «Quadrotors unmanned aerial vehicles: A review». En: *International Journal on Smart Sensing and Intelligent Systems* 9.1 (mar. de 2016), págs. 309-333. ISSN: 11785608. DOI: 10.21307/ijssis-2017-872.

[32] Amine Abadi y col. «Sliding mode control of quadrotor based on differential flatness». En: *2018 International Conference on Control, Automation and Diagnosis, ICCAD 2018*. Institute of Electrical y Electronics Engineers Inc., mar. de 2018. ISBN: 9781538654071. DOI: 10.1109/CADIAG.2018.8751334.

[33] T. Lee, M. Leok y N. H. McClamroch. «Geometric tracking control of a quadrotor UAV on SE(3)». En: *49th IEEE Conference on Decision and Control (CDC)*. 2010, págs. 5420-5425.

[34] Vemema Kangunde, Rodrigo S. Jamisola y Emmanuel K. Theophilus. «A review on drones controlled in real-time». En: *International Journal of Dynamics and Control* 9.4 (dic. de 2021), págs. 1832-1846. ISSN: 2195-2698.

[35] Ruiz D. y Bravo M. «Navegación autónoma y evasión de obstáculos en UAV usando aprendizaje por refuerzo». En: *Universidad Santo Tomas, Bogotá* (2019), págs. 1-95. URL: http://hdl.handle.net/11634/19029.

[36] Olov Andersson, Mariusz Wzorek y Patrick Doherty. «Deep learning quadcopter control via risk-aware active learning». En: *31st AAAI Conference on Artificial Intelligence, AAAI 2017* (2017), págs. 3812-3818.

[37] Lei Tai, Giuseppe Paolo y Ming Liu. «Virtual-to-real Deep Reinforcement Learning: Continuous control of mobile robots for mapless navigation». En: *arXiv* (2017), págs. 1-6.

[38] Garrett Dale Mckinnon. *The Birth of a Drone Nation: American Unmanned Aerial Vehicles Since 1917*. Inf. téc. 2014. URL: https://digitalcommons.lsu.edu/gradschool_theses/403.

[39] J.D. Blom. *Occasional Paper 37 Unmanned Aerial Systems: A Historical Perspective*. BiblioBazaar, 2012, pág. 153. ISBN: 9781249426707. URL: https://books.google.com.co/books?id=v-i5NAEACAAJ.

[40] Victor Delgado Hernando. *Historia de los drones - El Drone*. 2016. URL: http://eldrone.es/historia-de-los-drones/ (visitado 31-08-2020).

[41] Pablo Guimon. *EE UU mata al poderoso general iraní Soleimani en un ataque con drones en el aeropuerto de Bagdad*. Washington, ene. de 2020. URL: https://elpais.com/internacional/2020/01/03/actualidad/1578010671_559662.html (visitado 29-09-2020).

[42] Nikolaos Passalis y Anastasios Tefas. «Deep reinforcement learning for controlling frontal person close-up shooting». En: *Neurocomputing* 335 (mar. de 2019), págs. 37-47. ISSN: 18728286. DOI: 10.1016/j.neucom.2019.01.046.

[43] Chan Chun y col. «Drone Noise Reduction using Deep Convolutional Autoencoder for UAV Acoustic Sensor Networks». En: *Proceedings - 2019 IEEE 16th International Conference on Mobile Ad Hoc and S03t Systems Workshops, MASSW 2019*. Institute of Electrical y Electronics Engineers Inc., nov. de 2019, págs. 168-169. ISBN: 9781728141213. DOI: 10.1109/MASSW.2019.00043.

[44] L.S. Casey. *Curtiss, the Hammondsport Era, 1907-1915*. Crown Publishers, 1981, págs. 12-15. ISBN: 9780517543269. URL: https://www.amazon.com/Curtiss-Hammondsport-1907-1915-Louis-Casey/dp/0517545659.

[45] Karl Johan Åström. «Process Control—Past, Present, and Future». En: *IEEE Control Systems Magazine* 5.3 (1985), págs. 3-10. ISSN: 02721708. DOI: 10.1109/MCS.1985.1104958.

[46] Quan Quan. *Introduction to multicopter design and control*. Springer Singapore, jun. de 2017, págs. 1-384. ISBN: 9789811033827. DOI: 10.1007/978-981-10-3382-7.

[47] Daler Sharipov y col. «Implementation of a mathematical model of a hexacopter control system». En: *International Conference on Information Science and Communications Technologies: Applications, Trends and Opportunities, ICISCT 2019*. Institute of Electrical y Electronics Engineers Inc., nov. de 2019. ISBN: 9781728125640. DOI: 10.1109/ICISCT47635.2019.9011842.

[48] Paúl Sebastián y Dávila Aldás. «Diseño, construcción y control de un hexacóptero de monitoreo». Jun. de 2015. URL: http://bibdigital.epn.edu.ec/handle/15000/10924.

[49] Daniel Torres. «Analisis Dinamico del Fallo de Rotores en un Hexacoptero». En: *Journal of Chemical Information and Modeling* 53.9 (2019), págs. 1689-1699. ISSN: 1098-6596. URL: http://hdl.handle.net/11349/13866.

[50] Hyunsoo Yang y col. *Multi-rotor drone tutorial: systems, mechanics, control and state estimation*. Abr. de 2017. DOI: 10.1007/s11370-017-0224-y.

[51] Zhongqiu Zhang. «Application of PID Simulation Control Mode in Quadrotor Aircraft». En: *Proceedings - 2020 International Conference on Computer Engineering and Application, ICCEA 2020*. Institute of Electrical y Electronics Engineers Inc., mar. de 2020, págs. 826-829. ISBN: 9781728159041. DOI: 10.1109/ICCEA50009.2020.00181.

[52] Nurul Dayana Salim y col. «PID plus LQR attitude control for hexarotor MAV in indoor environments». En: *Proceedings of the IEEE International Conference on Industrial Technology* (2014), págs. 85-90. DOI: 10.1109/ICIT.2014.6894977.

[53] Oualid Araar y Nabil Aouf. «Full linear control of a quadrotor UAV, LQ vs H∞». En: *2014 UKACC International Conference on Control, CONTROL 2014 - Proceedings*. Institute of Electrical y Electronics Engineers Inc., oct. de 2014, págs. 133-138. ISBN: 9781479950119. DOI: 10.1109/CONTROL.2014.6915128.

[54] E. Rogers y col. «Lyapunov stability theory for linear repetitive processes - The 1D equation approach». En: *European Control Conference, ECC 1999 - Conference Proceedings* (2015), págs. 4774-4779. DOI: 10.23919/ecc.1999.7100090.

[55] Dongbin Lee y Timothy C. Burg. «Lyapunov-based control of unmanned aerial vehicle designed via stability analysis». En: *Control Theory: Perspectives, Applications and Developments*. Nova Science Publishers, Inc., abr. de 2015, págs. 277-297. ISBN: 9781634827300. DOI: 10.4155/9781634827300.ch12.

[56] Teresa Guarda y col. *Territorial Intelligence in the Impulse of Economic Development Initiatives for Artisanal Fishing Cooperatives Teresa*. Vol. 152. Micrads. Springer Singapore, 2020, págs. 119-132. DOI: 10 . 1007 / 978 - 981 - 13 - 9155 - 2. URL: http://link.springer.com/10.1007/978-981-13-9155-2.

[57] R. Analia, Susanto y K. Song. «Fuzzy+PID attitude control of a co-axial octocopter». En: *2016 IEEE International Conference on Industrial Technology (ICIT)*. 2016, págs. 1494-1499. DOI: 10.1109/ICIT.2016.7474981.

[58] Yahui Li y col. «Robust and adaptive backstepping control for nonlinear systems using RBF neural networks». En: *IEEE Transactions on Neural Networks* 15.3 (mayo de 2004), págs. 693-701. ISSN: 10459227. DOI: 10.1109/TNN.2004.826215.

[59] Fendy Santoso y col. «Robust Hybrid Nonlinear Control Systems for the Dynamics of a Quadcopter Drone». En: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50.8 (ago. de 2020), págs. 3059-3071. ISSN: 21682232. DOI: 10.1109/TSMC.2018.2836922.

[60] Siddharth Patel y col. «An Intelligent Hybrid Artificial Neural Network-Based Approach for Control of Aerial Robots». En: *Journal of Intelligent and Robotic Systems: Theory and Applications* 97.2 (feb. de 2020), págs. 387-398. ISSN: 15730409. DOI: 10.1007/s10846-019-01031-z.

[61] Colin Greatwood y Arthur G. Richards. «Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control». En: *Autonomous Robots* 43.7 (oct. de 2019), págs. 1681-1693. ISSN: 15737527. DOI: 10 . 1007 / s10514 - 019 - 09829-4. URL: https://doi.org/10.1007/s10514-019-09829-4.

[62] V. Madhu Babu, Kaushik Das y Swagat Ku. «Designing of self tuning PID controller for AR drone quadrotor». En: *2017 18th International Conference on Advanced Robotics, ICAR 2017*. Institute of Electrical y Electronics Engineers Inc., ago. de 2017, págs. 167-172. ISBN: 9781538631577. DOI: 10.1109/ICAR.2017.8023513.

[63] Shingo Kase y Masahiro Oya. «Adaptive tracking controller for hexacopters with a wind disturbance». En: *Artificial Life and Robotics* 25.2 (mayo de 2020), págs. 322-327. ISSN: 16147456. DOI: 10.1007/s10015-020-00586-7. URL: https://link.springer.com/article/10.1007/s10015-020-00586-7.

[64] Claudio Rosales y col. «Neural control of a Quadrotor: A state-observer based approach». En: *2018 International Conference on Unmanned Aircraft Systems, ICUAS 2018*. Institute of Electrical y Electronics Engineers Inc., ago. de 2018, págs. 647-653. ISBN: 9781538613535. DOI: 10.1109/ICUAS.2018.8453303.

[65] Yuanda Wang y col. «Compensator for Robust Quadrotor Control». En: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2019), págs. 1-13. DOI: 10.1109/TSMC.2018.2884725.

[66] Kai Arulkumaran y col. «Deep reinforcement learning: A brief survey». En: *IEEE Signal Processing Magazine* 34.6 (2017), págs. 26-38. ISSN: 10535888. DOI: 10.1109/MSP.2017.2743240. arXiv: arXiv:1708.05866v2.

[67] Volodymyr Mnih y col. «Asynchronous methods for deep reinforcement learning». En: *33rd International Conference on Machine Learning, ICML 2016* 4 (2016), págs. 2850-2869. arXiv: 1602.01783.

[68] Covid Tesau y Gerald Tesau. «Temporal Difference Learning and TD-Gammon». En: *Communications of the ACM* 38.3 (1995), págs. 58-68. ISSN: 15577317. DOI: 10.1145/203330.203343.

[69] Eduardo F. Morales y Julio H. Zaragoza. «An introduction to reinforcement learning». En: *Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions* (2011), págs. 63-80. DOI: 10.4018/978-1-60960-165-2.ch004.

[70] Tom Schaul y col. «Prioritized experience replay». En: *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* (2016), págs. 1-21. arXiv: 1511.05952.

[71] Nomi Ringach y Megumi Sano. «Prioritized Experience Replay via Learnability Approximation». En: (2018), págs. 1-7.

[72] Yanhua Huang. «Deep Q-networks». En: *Deep Reinforcement Learning: Fundamentals, Research and Applications* (2020), págs. 135-160. DOI: 10.1007/978-981-15-4095-0_4.

[73] Volodymyr Mnih y col. «Human-level control through deep reinforcement learning». En: *Nature* 518.7540 (2015), págs. 529-533. ISSN: 14764687. DOI: 10.1038/nature14236.

[74] Ariel Matías y col. «Desplazamiento de un hexápodo utilizando técnicas de aprendizaje por refuerzo». En: nov. de 2014, págs. 1-4.

[75] Sang Yun Shin, Yong Won Kang y Yong Guk Kim. «Automatic drone navigation in realistic 3d landscapes using deep reinforcement learning». En: *2019 6th International Conference on Control, Decision and Information Technologies, CoDIT 2019*. Institute of Electrical y Electronics Engineers Inc., abr. de 2019, págs. 1072-1077. ISBN: 9781728105215. DOI: 10.1109/CoDIT.2019.8820322.

[76] Guangcun Shan y col. «Control of Quadrotor Drone with Partial State Observation via Reinforcement Learning». En: *Proceedings - 2019 Chinese Automation Congress, CAC 2019*. Institute of Electrical y Electronics Engineers Inc., nov. de 2019, págs. 1965-1968. ISBN: 9781728140940. DOI: 10.1109/CAC48633.2019.8996394.

[77] Victoria J. Hodge, Richard Hawkins y Rob Alexander. «Deep reinforcement learning for drone navigation using sensor data». En: *Neural Computing and Applications* 0123456789 (2020). ISSN: 14333058. DOI: 10.1007/s00521-020-05097-x. URL: https://doi.org/10.1007/s00521-020-05097-x.

[78] William Koch y col. «Reinforcement Learning for UAV Attitude Control». En: *CoRR* abs/1804.04154 (2018). arXiv: 1804.04154. URL: http://arxiv.org/abs/1804.04154.

[79] SPARC. *What is SPARC? The Partnership for Robotics in Europe*. URL: https://www.eu-robotics.net/sparc/about/index.html (visitado 05-09-2020).

[80] AISBL y SPARC. *Robotics 2020 Multi-Annual Roadmap MAR ICT-24 ii for Robotics in Europe, Call 1 ICT23–Horizon 2020*. 01. Feb. de 2014, pág. 308. URL: https://www.eu-robotics.net/cms/upload/downloads/ppp-documents/Multi-Annual_Roadmap2020_ICT-24_Rev_B_full.pdf.

[81] Markus Christen y col. *Zivile Drohnen - Herausforderungen und Perspektiven*. 1.ª ed. Vol. 66. Zürcher Hochschule für Angewandte Wissenschaften, 2018, pág. 252. ISBN: 9783728138934. DOI: 10.3218/3894-1. URL: https://digitalcollection.zhaw.ch/handle/11475/10584?locale=de.

[82] María Jesús Guerrero Lebrón. *La regulación transitoria de los operadores de aeronaves civiles pilotadas por control remoto*. Inf. téc. Sep. de 2014, págs. 12-29. URL: https://dialnet.unirioja.es/servlet/revista?codigo=21839.

[83] Redacción ImpactoTIC. *Las tecnologías 'vuelan' en el posconflicto: drones, radares y contenidos*. Jun. de 2018. URL: https://impactotic.co/tecnologia-posconflicto-en-colombia/ (visitado 15-09-2020).

[84] Luz Vallejo Mejía y Sandra Milena Agudelo - Londoño. «El glifosato alza el vuelo. Análisis retórico del discurso en la prensa nacional de Colombia (2018-2019)». En: *Signo y Pensamiento* 38.75 (nov. de 2019), págs. 1-16. ISSN: 2027-2731. DOI: 10.11144/Javeriana.syp38-75.gava. URL: https://revistas.javeriana.edu.co/index.php/signoypensamiento/article/view/27958.

[85] David Mayorga Perdomo. «Un dron para desminar el país». En: *Perquisa* 34 (nov. de 2015), págs. 6-7. URL: https://www.javeriana.edu.co/pesquisa/un-dron-para-desminar-el-pais/.

[86] Juan Manuel Garzón y Federico Luqe. «Implementación de drones para incrementar la productividad en el agro colombiano». Colegio de Estudios Superiores de Administración - CESA, 2018, pág. 71. URL: http://hdl.handle.net/10726/2302.

[87] Unidad Administrativa Especial de Aeronáutica Civil. *Resolución número 04201 de 2018*. Dic. de 2018. URL: https://diario-oficial.vlex.com.co/vid/resolucion-numero-04201-2018-763853657 (visitado 31-08-2020).

[88] *Drones: Reporting for Work*. URL: https://www.goldmansachs.com/insights/technology-driving-innovation/drones/ (visitado 09-09-2020).

[89] Francesco Sabatino. «Quadrotor control: modeling, nonlinearcontrol design, and simulation». Tesis de mtría. KTH, Automatic Control, 2015, pág. 61.

[90] Moti Ben-Ari. *A Tutorial on Euler Angles and Quaternions*. Wiezmann Institute of Science, 2014, págs. 1-22. URL: http://www.weizmann.ac.il/sci-tea/benari/%20https://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/softwareAndLearningMaterials/quaternion-tutorial-2-0-1.pdf.

[91] *The Physics of How Drones Fly | WIRED*. 2021. URL: https://www.wired.com/2017/05/the-physics-of-drones/.

[92] Tommaso Bresciani. «Modelling, Identification and Control of a Quadrotor Helicopter». Lund University, ago. de 2008, págs. 1-184. URL: http://www.control.lth.se/publications/%20http://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8847641&fileOId=8859343.

[93] Shashi Poddar, Rahul Kottath y Vinod Karar. «Evolution of Visual Odometry Techniques». En: *CoRR* abs/1804.11142 (2018). arXiv: 1804.11142. URL: http://arxiv.org/abs/1804.11142.

[94] Guanya Shi y col. «Neural Lander: Stable Drone Landing Control Using Learned Dynamics». En: *2019 International Conference on Robotics and Automation (ICRA)* (mayo de 2019). DOI: 10.1109/icra.2019.8794351. URL: http://dx.doi.org/10.1109/ICRA.2019.8794351.

[95] Jacopo Panerati y col. *Learning to Fly – a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control*. 2021. arXiv: 2103.02142 [cs.RO].

[96] Julian Förster, Bachelor Thesis y Michael D Hamer Raffaello. «System Identification of the Crazyflie 2.0 Nano Quadrocopter». En: (2015). DOI: 10.3929/ETHZ-B-000214143.

[97] William Neeley. «Design and Development of a High-Performance Quadrotor Control Architecture Based on Feedback Linearization». En: *Electrical and Computer Engineering ETDs* (feb. de 2016). URL: https://digitalrepository.unm.edu/ece_etds/190.

[98] Charles H. Holbrow y col. «Some Physics You Need to Know». En: *Modern Introductory Physics* (2009), págs. 13-62. DOI: 10.1007/978-0-387-79080-0_2.

[99] *Bullet Real-Time Physics Simulation | Home of Bullet and PyBullet: physics simulation for games, visual effects, robotics and reinforcement learning.* URL: https://pybullet.org/wordpress/.

[100] Jacopo Panerati. *gym-pybullet-drones*. 2022. URL: https://github.com/utiasDSL/gym-pybullet-drones (visitado 01-02-2022).

[101] Nathan Michael y col. «The GRASP Multiple Micro-UAV Testbed». En: *IEEE Robotics Automation Magazine* 17.3 (2010), págs. 56-65. DOI: 10.1109/MRA.2010.937855.

[102] *Crazyflie 2.0 – Bitcraze Store*. URL: https://store.bitcraze.io/products/crazyflie-2-0.

[103] Jérôme Le Ny Carlos Luis. «Design of a Trajectory Tracking Controller for a Nanoquadcopter». En: (ago. de 2016). URL: https://arxiv.org/abs/1608.05786.

[104] Rufus Fraanje. *quadcopter_sim*. 2017. URL: https://github.com/prfraanje/quadcopter_sim (visitado 19-11-2021).

[105] «Control System». En: *CIRP Encyclopedia of Production Engineering*. Ed. por Sami Chatti y col. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, págs. 359-359. ISBN: 978-3-662-53120-4. DOI: 10.1007/978-3-662-53120-4_300118. URL: https://doi.org/10.1007/978-3-662-53120-4_300118.

[106] J. Astrom Karl y Tore Hagglund. *PID Controllers, Theory, Design and Tuning*. Second Edition. United States Of America, 1988.

[107] Fethi Tekyaygil. *Keras Model Wars: Sequential vs Functional*. 2020. URL: https://www.datasciencearth.com/keras-model-wars-sequential-vs-functional/ (visitado 23-06-2021).

[108] IBM Cloud Educationl. *What are Neural Networks?* 2020. URL: https://www.ibm.com/cloud/learn/neural-networks (visitado 23-06-2021).

[109] Lisha Li y col. «Hyperband: A novel bandit-based approach to hyperparameter optimization». En: *Journal of Machine Learning Research* 18 (2018), págs. 1-52. ISSN: 15337928. arXiv: 1603.06560.

[110] Xue Ying. «An Overview of Overfitting and its Solutions». En: *Journal of Physics: Conference Series* (feb. de 2019), págs. 256-272. DOI: 10.1088/1742-6596/1168/2/022022. URL: https://www.researchgate.net/publication/331677125_An_Overview_of_Overfitting_and_its_Solutions.

[111] J. Kennedy y R. Eberhart. «Particle swarm optimization». En: *Proceedings of ICNN'95 - International Conference on Neural Networks*. 1995, 1942-1948 vol.4.

[112] Riccardo Poli, James Kennedy y Tim Blackwell. «Particle swarm optimization». En: *Swarm Intelligence* 1.1 (jun. de 2007), págs. 33-57. ISSN: 1935-3820. DOI: 10.1007/s11721-007-0002-0. URL: https://doi.org/10.1007/s11721-007-0002-0.

[113] Erik Cuevas, Alonso Echavarría y Marte A. Ramírez-Ortegón. «An optimization algorithm inspired by the States of Matter that improves the balance between exploration and exploitation». En: *Applied Intelligence* 40.2 (mar. de 2014), págs. 256-272. ISSN: 1573-7497. DOI: 10.1007/s10489-013-0458-0. URL: https://doi.org/10.1007/s10489-013-0458-0.

[114] Michał Bartyś y Bartłomiej Hryniewicki. «The Trade-Off between the Controller Effort and Control Quality on Example of an Electro-Pneumatic Final Control Element». En: *Actuators* 8.1 (2019). ISSN: 2076-0825.

[115] Mathworks. *Quadcopter Project, MATLAB & Simulink*. 2018. URL: https://de.mathworks.com/help/aeroblks/quadcopter-project.html.

[116] G M Tinungki. «The analysis of partial autocorrelation function in predicting maximum wind speed». En: *IOP Conference Series: Earth and Environmental Science* 235 (feb. de 2019), pág. 012097. DOI: 10.1088/1755-1315/235/1/012097. URL: https://doi.org/10.1088/1755-1315/235/1/012097.