Seguimiento de Objetos Móviles a través de un UAV utilizando DNN

por

Erik Francisco Agustín

Tesis sometida como requisito parcial para obtener el grado de

MAESTRÍA EN CIENCIA Y TECNOLOGÍA DEL ESPACIO

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica
Junio 2024
Tonantzintla, Puebla

Bajo la supervisión de:

Dr. Gustavo Rodríguez Gómez
Investigador Titular INAOE
Dr. José Martínez Carranza
Investigador Titular INAOE

Agradecimientos

Agradezco profundamente a mi familia por su apoyo constante e incondicional, por confiar en mi y estar siempre a mi lado mientras persigo mis metas.

Quiero expresar mis agradecimientos al Dr. Gustavo Rodríguez Gómez por ser una gran persona. Aprecio su disponibilidad, apoyo y orientación, no solo en el ámbito académico, sino también en el humano. Sus consejos, confianza y enseñanzas de vida hicieron que el posgrado fuera más ameno, no solo en términos profesionales, sino también en mi crecimiento personal.

Asimismo, agradezco al Dr. José Martínez Carranza por su confianza, apoyo, disponibilidad, orientación y seguimiento en el desarrollo de esta investigación.

Quiero extender mi gratitud al comité de evaluación, conformado por la Dra. Raquel Díaz Hernández, el Dr. Leopoldo Altamirano Robles y el Dr. José Eduardo Mendoza Torres, por su tiempo, consideración y por sus valiosos comentarios y sugerencias, que sin lugar a duda contribuyeron a realizar un mejor trabajo.

A mi pareja sentimental, le agradezco su amor incondicional, compartirme sus ganas de comerse al mundo, paciencia, apoyo constante y comprensión durante el camino hacia mis metas profesionales y personales.

Quiero extender mi gratitud a mis amigos que me motivaron a emprender este posgrado, y a quienes conocí durante el mismo, por su apoyo y, sobre todo, por su motivación para que continúe preparándome y persiga mis sueños.

A David, por su paciencia, disponibilidad, compartir sus conocimientos conmigo y apoyarme en el laboratorio de robótica.

Al Consejo Nacional de Humanidades, Ciencia y Tecnología CONHACYT por el apoyo económico para la realización de esta investigación (1128882).

Abstract

The results of various missions to Mars, which have involved sending probes, rovers, and recently the helicopter Ingenuity, have highlighted the growing need to develop technologies for autonomous navigation. The powered flight capability of Unmanned Aerial Vehicles (UAVs) offers immense applications in planetary exploration. Detecting and tracking moving objects with unknown dynamics is a current challenge, especially in planetary exploration. The existing algorithms for target detection, combined with trajectory tracking control, are computationally expensive and face difficulties in meeting real-time requirements. Although classical techniques for trajectory tracking exist, they present limitations in the design of the UAV's dynamic model. Depending on the model, singularities may arise for certain states, and in practice, it is challenging for the model to encompass all states of the UAV. As a result, control may fail to achieve its objective. This work proposes an architecture of a pre-trained Deep Neural Network (DNN) with Model Predictive Control (MPC) for trajectory tracking. The proposal aims to reduce computational cost, maintain the precision of optimal control in tracking, and increase processing speed compared to MPC. The control commands are derived from visual information and UAV states. Experimental tests were conducted in simulation using the Gazebo program, with the Parrot Bebop 2.0 and the Husky robot. Physical implementation was carried out with the Bebop 2.0 and a differential robot.

Keywords: Control, DNN, Dynamic Model, MPC, UAV.

Resumen

Los resultados de diversas misiones a Marte, en las que se han enviado sondas, rovers y recientemente el helicóptero Ingenuity, han mostrado la creciente necesidad de desarrollar tecnologías para la navegación autónoma. La capacidad de vuelo propulsado de los Vehículos Aéreos No Tripulados (UAV) ofrece aplicaciones inmensas en la exploración planetaria. Detectar y seguir objetos en movimiento cuyas dinámicas son desconocidas es un desafío actual, especialmente en la exploración planetaria. Los algoritmos actuales para detección de objetivos en conjunto con el control de seguimiento de trayectorias, son costosos computacionalmente y enfrentan dificultades en cumplir con requisitos de tiempo real. Aunque existen técnicas clásicas para el seguimiento de trayectorias, presentan limitaciones en el diseño del modelo dinámico del UAV. Dependiendo del modelo, pueden surgir singularidades para ciertos estados, y en la práctica, es difícil que el modelo abarque todos los estados del UAV. Como resultado, el control puede fallar en el logro de su objetivo. Este trabajo propone una arquitectura de Red Neuronal Profunda (DNN) preentrenada con un Control de Modelo Predictivo (MPC) para el seguimiento de trayectorias. La propuesta busca reducir el costo computacional, mantener la presición del control óptimo en el seguimiento e incrementar la velocidad de procesamiento en comparación con el MPC. Los comandos de control se derivan de información visual y estados del UAV. Las pruebas experimentales se llevaron a cabo en simulación en el programa Gazebo, con el Bebop 2.0 de Parrot y el robot Husky. Mientras que, la implementación física se realizó con el Bebop 2.0 y un robot diferencial.

Palabras Clave: Control, DNN, Modelo dinámico, MPC, UAV.

Índice general

Agradecimientos II

Abstract III

Resumen IV

Índice general V

Acrónimos VIII

Índice de figuras 1

Índice de Tablas 5

1. Introducción 7
- 1.1. Motivación . . . 9
- 1.2. Justificación . . . 9
- 1.3. Objetivos del trabajo de tesis . . . 10
  - 1.3.1. Objetivo General . . . 10
  - 1.3.2. Objetivos Específicos . . . 10
- 1.4. Metodología . . . 11
- 1.5. Contribuciones . . . 12
- 1.6. Estructura de la Tesis . . . 13

2. Estado del Arte 14
- 2.1. Detección . . . 14
- 2.2. Planeación . . . 18
- 2.3. Seguimiento . . . 20

3. Marco Teórico 24
- 3.1. Misión Mars 2020 . . . 24
  - 3.1.1. Perseverance . . . 24
  - 3.1.2. Ingenuity . . . 25
- 3.2. Plataformas robóticas . . . 27

3.2.1. UAV . . . 27
3.2.2. Robot terrestre Husky . . . 27
3.3. Entorno de Software . . . 29
3.3.1. ROS . . . 29
3.3.2. bebop_autonomy . . . 30
3.3.3. Gazebo . . . 30
3.3.4. Pytorch . . . 31
3.4. Deep Learning . . . 32
3.4.1. MLP . . . 33
3.4.2. Entrenamiento . . . 34
3.4.3. Hiperparámetros . . . 35
3.4.4. YOLOv8 . . . 36
3.5. Control . . . 36
3.5.1. Control PID . . . 37
3.5.2. MPC . . . 37
3.5.3. Control Neuronal . . . 40
3.6. Filtro de Kalman . . . 41
3.6.1. El filtro discreto de Kalman . . . 41

4. Metodología **44**
4.1. Percepción del Objetivo . . . 44
4.2. Definición del sistema de coordenadas . . . 45
4.3. Estimación de posición . . . 46
4.4. Identificación de la planta . . . 48
4.5. Filtro de Kalman . . . 50
4.6. Diseño de Controles . . . 50
4.6.1. Control PID . . . 51
4.6.2. MPC . . . 52
4.6.3. Diseño de trayectorias de validación de controles PID y MPC . 54
4.6.4. Validación de los controles PID y MPC . . . 54
4.6.5. Control Neuronal . . . 58
4.7. Validación de los controles N-PID y N-MPC . . . 60

5. Experimentos y Resultados **61**
5.1. Percepción del objetivo . . . 61
5.2. Estimación de posición . . . 61
5.3. Identificación de la planta . . . 64
5.4. Filtro de Kalman . . . 66
5.5. Diseño de controles . . . 69
5.5.1. Control PID . . . 69
5.5.2. MPC . . . 70
5.5.3. Diseño de trayectorias de validación de controles PID y MPC . 71
5.5.4. Validación de los controles PID y MPC . . . 73

5.5.5. Control Neuronal 77
5.5.6. Diseño de trayectorias de validación de controles neuronales . . 80
5.6. Validación de los controles N-PID y N-MPC 81
5.6.1. Análisis y Discusión de Resultados 82
5.6.2. Desempeño del N-MPC para otros escenarios 87
5.7. Implementación Física 90

6. Conclusiones y Trabajo Futuro **95**

Acrónimos

DL	Aprendizaje Profundo
CNN	Red Neuronal Convolucional
DNN	Red Neuronal Profunda
FoV	Campo de Visión
FPS	Cuadros por Segundo
GDL	Grados De Libertad
GNC	Guía Navegación y Control
GPS	Sistema de Posicionamiento Global
IMU	Unidad de Medición Inercial
JPL	Jet Propulsion Laboratory
KCF	Kalman Correlational Filter
KF	Filtro de Kalman
LIDAR	Light Detection And Ranging
LRF	Telemetro Laser
LSTM	Memoria a Corto y Largo Plazo
MAVeN	Movement Analysis Vehicle Estimation Network
ML	Aprendizaje Máquina
MLP	Perceptrón Multicapa
MORSE	Modular Open Robots Simulation Engine
MPC	Control Predictivo basado en Modelo

MTE	Error Medio de Seguimiento
PID	Control Proporcional, Integral y Derivativo
PNP	Perspectiva N Puntos
RMSE	Raíz del Error Cuadratico Medio
RL	Aprendizaje por Refuerzo
RN	Red Neuronal
ROS	Sistema Operativo de Robots
Rover	Explorador
SBAS	Sistema de Aumento Basado en Satélites
SDK	Kit de Desarrollo de Software
SAR	Búsqueda y Rescate
SSD	Single Shot Detector
SSE	Error en Estado Estacionario
TSR	Tasa de Éxito de Seguimiento
UAV	Vehículo Aéreo no Tripulado
YOLO	You Only Look Once

Índice de figuras

2.1. Sistema de percepción y control utilizada en (Wu et al., 2022). 15
2.2. Diagrama conceptual de Perspective-n-point (PNP). Imagen tomada de (Li et al., 2021). 15
2.3. Diagrama de estimación de distancia con DNN a través de una cámara monocular utilizada en (Huang y Lai, 2020). 17
2.4. Sistema de percepción y control utilizado en (Muller et al., 2019). . . . 18
2.5. Sistema *end to end* utilizado en (Cai et al., 2019). 18
2.6. Estructura simplificada del sistema de navegación autónoma propuesta por (Zhilenkov y Epifantsev, 2018). 20
2.7. Estrategia híbrida de los módulos de seguimiento y aterrizaje utilizada en (Xie et al., 2020). 21
2.8. Resultados del seguimiento de una plataforma móvil obtenidos en (Xie et al., 2020). 21
2.9. Sistema de percepción y control propuesto en (Li et al., 2021). 21
2.10. Errores de seguimiento entre el UAV y el objetivo moviéndose en linea recta con diferentes velocidades reportados en (Li et al., 2021). 22

3.1. Ilustración del rover Perseverance operando en la superficie de Marte. Tomada de NASA/JPL-Caltech. 25
3.2. Ilustración del helicóptero Ingenuity en la superficie de Marte. Tomada de NASA/JPL-Caltech. 25
3.3. Plataforma del UAV bebop 2.0 con sus 6 GDL. 28
3.4. Plataforma del robot HUSKY con sus 3 GDL. 28
3.5. Representación de la interacción de nodos, mensajes, tópicos y servicios en ROS. 29
3.6. Estructura general de los componentes en Gazebo, imagen tomada de (Koenig y Howard, 2004) 31
3.7. Clasificación del Machine Learning, extraída de(Patterson y Gibson, 2017). 32
3.8. Arquitectura del MLP, imagen tomada de (Naskath et al., 2023). 34
3.9. Tareas que se pueden realizar en YOLOv8 (imagen tomada del repositorio de github de ultralitycs). 36
3.10. Principio del control predictivo tomada de (Szczerbicki, 2009). 38

3.11. Diagrama de bloques del funcionamiento del MPC tomada de (Okasha et al., 2022). 40
3.12. Esquema de Red Neuronal Predictiva para la navegación de un robot móvil, imagen tomada de (Camacho y Bordons, 2007). 41
3.13. Diagrama de operación del filtro de Kalman. 43

4.1. Metodología general empleada para el seguimiento de objetos móviles a través de un UAV utilizando DNN. 44
4.2. Sistema de coordenadas global "G" y local "B" utilizado en el desarrollo de este trabajo. 45
4.3. Detección del "Objetivo" con YOLOv8 donde se muestra el *bounding box*, el pixel central y la confianza de detección. 46
4.4. Diagrama geométrico para calcular la estimación de posición del "Objetivo" detectado desde la cámara a bordo del UAV. 46
4.5. Esquema de control PID implementado para el Bebop 2.0. 52
4.6. Esquema de control MPC implementado para el Bebop 2.0 54
4.7. Trayectorias utilizadas para la validación de los controladores PID y MPC con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: - - la trayectoria, $\square$ su inicio , $\diamond$ fin y $*$ pausas a lo largo de ella. 55
4.8. Arquitectura del MLP desarrollado para la predicción de señales de control. 58
4.9. Esquema de control híbrido implementado para el Bebop 2.0. Las visualizaciones muestran a) N-PID y b) N-MPC. 59
4.10. Trayectorias utilizadas para la validación de los controles neuronales, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: - - la trayectoria, $\square$ su inicio , $\diamond$ fin y $*$ pausas a lo largo de ella. 60

5.1. Área percibida desde la cámara del UAV y comparación entre la posición del "Objetivo" obtenida del simulador contra la posición estimada por la cámara del UAV para una trayectoria de Lemniscata a alturas de (a) 3 m, (b) 4 m y (c) 5 m. 62
5.2. Comparación entre la posición del "Objetivo" medida del simulador y su posición estimada a través de la cámara del UAV durante el Experimento error en zonas de Observación en una circunferencia con radios de 2 m, 3 m y 4 m. 64
5.3. Señal de control u_{ide} para la identificación de la planta donde $u_{vx} = u_{vy} = u_{vz} = u_{ide}$ aplicado como $u_c = (u_{vx}, 0, 0, 0)$, $u_c = (0, u_{vy}, 0, 0)$ y $u_c = (0, 0, u_{vz}, 0)$ de forma independiente. 65
5.4. Señal de control aplicada al UAV y velocidad lineal obtenida para el Bebop 2.0 para el movimiento traslacional en X. 65

5.5. Estimación de velocidad obtenida con el filtro de Kalman contra mediciones del simulador para los datos obtenidos en la sección 5.3 para a) v_x, b) v_y y c) v_z. 68
5.6. Respuesta del control PID sintonizado. 70
5.7. Respuesta del control MPC sintonizado. 71
5.8. Trayectoria original y estimada, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: □ inicio , ⋄ fin y ∗ pausas a lo largo la trayectoria "D", "E" y "F". Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom. 74
5.9. Seguimiento de la trayectoria "D", "E" y "F" utilizando los controladores PID y MPC a) Trayectoria completa b) Zona de Zoom. 76
5.10. Curva de aprendizaje durante 300 épocas de los modelos a) N-PID y b) N-MPC. 78
5.11. Comparación entre las señales de control u_{vx} predichas y el "ground truth" para una muestra de 150 datos del conjunto de validación para los modelos a) N-PID y b) N-MPC. 78
5.12. Comparación entre las señales de control u_{vy} predichas y el "ground truth" para una muestra de 150 datos del conjunto de validación para los modelos a) N-PID y b) N-MPC. 79
5.13. Comparación entre las señales de control $u_{\omega z}$ predichas y el "ground truth" para una muestra de 150 datos del conjunto de validación para N-PID. 79
5.14. Trayectoria original y estimada, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: □ inicio , ⋄ fin y ∗ pausas a lo largo la trayectoria "I". Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom. 80
5.15. Seguimiento de la trayectoria "G", "H" e "I" utilizando los controladores PID, MPC, N-PID, N-MPC. Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom. 83
5.16. RMSE obtenido entre las posiciones de referencia y las alcanzadas con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. Las visualizaciones muestran (a) RMSEX y (b) RMSEY. 84
5.17. Error en Estado Estacionario promedio obtenido con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. 84
5.18. TSR obtenido para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I para un umbral de 2.5 m. 85

5.19. Comparación de seguimiento con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de la trayectoria G. Las visualizaciones muestran a) El seguimiento de la trayectoria y b) Una zona con Zoom donde △ denota la posición del UAV en el mismo instante para cada controlador. 85
5.20. Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria G. 86
5.21. Frecuencia de operación promedio de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. . . . 87
5.22. Seguimiento de las trayectorias J y K utilizando los controladores PID, MPC, N-PID, N-MPC. Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom. 88
5.23. Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria (a) J y (b) K. 90
5.24. Configuración del sistema para la implementación física de los controladores. 91
5.25. Trayectorias L, M , N y O utilizadas para el entrenamiento de los modelos N-PID y N-MPC. T rayectorias P y Q utilizadas para la validación del seguimiento. Se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella. 92
5.26. Seguimiento de las trayectorias de validación utilizando los controladores PID, MPC, N-PID y N-MPC. Se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella. 93
5.27. Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria (a) P y (b) Q. 94

Índice de Tablas

3.1. Ejemplo de funciones de activación utilizadas en Redes Neuronales. . . 34

4.1. Parámetros intrínsecos de la cámara inferior del Bebop 2.0 en Gazebo. . 47
4.2. Datos numéricos almacenados en los archivos ".txt", generados durante el seguimiento del "Objetivo" para la validación de los controles y el entrenamiento de sus versiones neuronales. Se utiliza ✓para indicar que el dato es una entrada de la red para la predicción de señales de control, mientras que las salidas de la red se indican con •. 56
4.3. Datos numéricos almacenados en los archivos ".txt", generados durante el seguimiento del "Objetivo" para la validación de los controladores neuronales. 60

5.1. Porcentaje de confianza promedio obtenida en la detección de la clase "Objetivo" desde distintas alturas. 61
5.2. RMSE y área total percibida para el **Experimento error de estimación.** 62
5.3. RMSE obtenido para el **Experimento error en zonas de observación**. 63
5.4. Parámetros de la dinámica lineal del UAV Parrot Bebop 2.0 estimados con MATLAB®. 66
5.5. RMSE obtenido entre las mediciones del simulador y las predicciones de velocidad utilizando las matrices sintonizadas Q_{kalman} y R_{kalman}. . . 67
5.6. Ganancias sintonizadas para el control de seguimiento PID para el UAV bebop 2.0. 70
5.7. Datos numéricos almacenados en los archivos ".txt", generados durante el seguimiento del "Objetivo" para la estimación de trayectorias de validación de los controladores PID y MPC. 73
5.8. Longitud de las trayectorias estimadas y RMSE obtenido. 73
5.9. RMSE en X, Y y Z para los controladores PID y MPC durante el seguimiento de las trayectorias A,B,C,D,E Y F. 75
5.10. TSR, velocidad y frecuencia de operación de los controladores PID y MPC durante el seguimiento de las trayectorias A,B,C,D,E Y F. 77
5.11. RMSE obtenido entre las predicciones y los el "ground thruth" del conjunto de validación para los modelos N-PID y N-MPC. 79
5.12. Longitud de las trayectorias estimadas y RMSE obtenido. 80

5.13. Resultados obtenidos de error de posición y error de posición en estado estacionario para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. 81
5.14. Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. 82
5.15. Longitud de las trayectorias estimadas y RMSE obtenido. 88
5.16. Resultados obtenidos de error de posición y error de posición en estado estacionario para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias J y K 89
5.17. Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias J y K. 89
5.18. Longitud de las trayectorias empleadas en la implementación física. Se utiliza el símbolo • para indicar que la trayectoria fue utilizada en el entrenamiento del N-PID y N-MPC y el símbolo ✓ para indicar que la trayectoria fue utiliza en la validación del seguimiento utilizando estos modelos. 92
5.19. Resultados obtenidos de RMSEX, RMSEY y SSE para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias P y Q. 93
5.20. Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias P y Q. 94

Capítulo 1

Introducción

El 30 de julio de 2020, desde la Estación de la Fuerza Aérea en Cabo Cañaveral, Florida, se realizó el lanzamiento de la misión Mars 2020, que llevaría al rover Perseverance y al helicóptero Ingenuity a Marte. Convirtiéndose en el primer Vehículo Aéreo no Tripulado (UAV) enviado a otro planeta con el objetivo de demostrar la tecnología del primer vuelo propulsado en Marte. Ingenuity realizó su primer vuelo el 19 de abril de 2021, comenzando así su misión de exploración.[1](von Ehrenfried, 2022).

Ingenuity realizó una serie de vuelos de referencia, con la finalidad de recabar información que permita mejorar el diseño para misiones con UAVs, destacando las siguientes áreas (Balaram et al., 2021):

- Desempeño Aerodinámico y de Control.
- Desempeño de Navegación.
- Desempeño Térmico/Energético.

Por primera vez, el proyecto Ingenuity probó nuevas características que le dan autonomía, sin la intervención directa de los controladores de vuelo del JPL (Jet Propulsion Laboratory); no se usó un joystick para manejar el helicóptero.

Las demoras en las comunicaciones son una parte intrínseca del trabajo con vehículos espaciales en distancias interplanetarias. Por esta razón, las instrucciones se enviaban con anticipación y los datos de ingeniería de la nave regresaban mucho después de que cada vuelo se llevaba a cabo (von Ehrenfried, 2022)[2]. Ingenuity fue diseñado para volar de manera autónoma, tomando decisiones sobre cómo volar hacia un punto específico y mantenerse en condiciones óptimas de temperatura. Surgen nuevos desafíos de acuerdo a las necesidades identificadas en este contexto.

Las aplicaciones de un UAV como robot de exploración son inmensas, gracias a las características propias de su naturaleza, amplían las posibilidades de tareas de exploración, ejemplo de algunas aplicaciones se listan a continuación:

[1]https://mars.nasa.gov/technology/helicopter/

[2]https://ciencia.nasa.gov/seis-cosas-helicoptero-ingenuity

- Seguimiento y aterrizaje del UAV sobre una plataforma móvil: En esta aplicación el UAV podría aterrizar sobre un rover (explorador) de locomoción de ruedas que se encuentre en movimiento, con el objetivo de que ambos estén juntos durante la etapa en la que el UAV recarga su batería, mientras sigue avanzando con ayuda del rover terrestre.
- Despliegue de instrumentos de contacto y entrega o recuperación de muestras para su análisis (Balaram et al., 2021).
- Seguimiento de algún objeto móvil para brindar apoyo visual de su trayectoria en el caso de un trabajo colaborativo entre ambos rovers.
- Seguir un objetivo y observarlo a lo largo del tiempo.
- Las imágenes aéreas proporcionarán una visión más completa de la geología y permitirán la observación de zonas que resultarían inaccesibles para un rover debido a su pronunciada pendiente o superficie resbaladiza proporcionando apoyo estratégico para la travesía del rover terrestre [3](Balaram et al., 2021; Tzanetos et al., 2022).
- Exploradores aéreos para vigilar la superficie del planeta desde arriba, proporcionando reconocimiento preliminar sobre objetivos científicos y de exploración [3], (Balaram et al., 2021).
- Brindar apoyo a astronautas durante futuras misiones para explorar Marte [3], (Balaram et al., 2021).
- Transporte de carga útil de un lugar a otro [3].

Los vuelos realizados y la capacidad demostrada para maniobrar en la atmósfera de Marte indican que las tecnologías de vuelo pueden utilizarse con éxito en futuras misiones a Marte (Green, 2021). Sin embargo, actualmente, el desafío con los seguidores radica en su procesamiento computacional intensivo y en la dificultad para cumplir con los requisitos de tiempo real.

De acuerdo con el trabajo desarrollado por (Li et al., 2021) se ha observado un aumento en la popularidad de la investigación sobre la aplicación de sistemas de Guía, Navegación y Control (GNC) en UAVs. En este contexto, la planificación de rutas, estimación de estados y control de seguimiento de trayectorias desempeña un papel crucial en estos sistemas.

Según señala (Tian et al., 2022), los algoritmos existentes destinados al reconocimiento y seguimiento enfrentan desafíos relacionados con la velocidad, precisión, robustez e inteligencia insuficiente. En línea con esta perspectiva, (Muller et al., 2019) propone la utilización de una Red Neuronal Profunda (DNN) como un sistema de control inteligente. Esta aproximación permite abordar un espectro más amplio de estados en comparación con los controladores clásicos, superando así las limitaciones inherentes a un único controlador. Utilizar técnicas de aprendizaje computacional para un sistema de control resulta en costos computacionales más bajos en comparación con las técnicas de control de trayectoria, como el Control Predictivo basado en Modelo

[3] https://mars.nasa.gov/files/mars2020/MarsHelicopterIngenuity_FactSheet.pdf

(MPC), especialmente en entornos cambiantes donde la planificación se realiza en cada momento, lo que aumenta significativamente el costo computacional.

Detectar y seguir un objeto en movimiento cuya dinámica es desconocida representa un desafío significativo en diversas aplicaciones, incluyendo la exploración planetaria. Este estudio se enfoca en abordar este desafío mediante el uso de técnicas de aprendizaje computacional (ML) para la detección de objetivos y el seguimiento de trayectorias de manera autónoma. Los algoritmos desarrollados en la presente tesis se implementan en Python y se integran en el simulador Gazebo, en conjunto con librerías del Framework ROS (Robot Operating System). Los resultados obtenidos se validan utilizando el UAV Bebop 2.0 de la marca Parrot.

1.1. Motivación

Los sistemas autónomos de navegación en aplicaciones de exploración planetaria constituyen un área en constante desarrollo. La integración de la inteligencia artificial, junto con la fusión de sensores y, en particular, las técnicas de visión por computadora, posibilita la realización de tareas sorprendentes en el ámbito de la exploración planetaria. El objetivo primordial es explorar la posibilidad de vida en otros planetas, comprender su evolución y contribuir a la investigación científica.

En este contexto, surgen diversas necesidades y se identifican nuevos desafíos al navegar en entornos desconocidos. La solución a estos problemas se centra en los sistemas de navegación, cuya optimización podría prolongar la vida útil de los rovers. Es esencial desarrollar sistemas de navegación autónoma capaces de detectar objetos de interés y determinar su ubicación tridimensional utilizando la información visual captada por las cámaras a bordo. Esto permite minimizar la cantidad de sensores necesarios y, por ende, reducir el peso de la carga útil de los UAV, un aspecto crucial en las misiones de exploración planetaria.

1.2. Justificación

Actualmente, el helicóptero robótico Ingenuity ha concluido sus vuelos en Marte. Sin embargo, ha demostrado con éxito la capacidad de volar y maniobrar en la atmósfera marciana. Esto sugiere que la tecnología de vuelo puede ser implementada con éxito en futuras misiones en la superficie de Marte (Green, 2021).

Los sistemas de seguimiento autónomo de objetivos mediante UAVs son viables al emplear exclusivamente la información visual proveniente de sus cámaras a bordo, replicando así la acción de un piloto real que persigue un objetivo. Sin embargo, la limitada capacidad computacional y carga útil restringida de estos vehículos han impedido que los resultados alcancen el nivel de eficiencia logrado por un piloto profesional. Esto subraya la necesidad de optimizar los recursos disponibles a bordo del UAV, desarrollando algoritmos de navegación que logren con eficacia tareas de percepción,

planificación de trayectorias, evasión de obstáculos y control, permitiendo la ejecución de maniobras más complejas.

En este contexto, abordar el problema mediante el desarrollo de sistemas de navegación autónoma que optimicen la información visual de la cámara del UAV para estimar la posición del objetivo, junto con la aplicación de técnicas de aprendizaje computacional para el diseño del sistema de control, emerge como una estrategia clave. Esta aproximación ofrece eficiencia y adaptabilidad en entornos dinámicos, presentando una alternativa importante a los enfoques tradicionales de control de trayectoria, como el MPC. La implementación de algoritmos de aprendizaje computacional no solo supera limitaciones técnicas, sino que también permite mejorar continuamente el rendimiento del UAV en tareas complejas, garantizando mayor eficacia y autonomía en la consecución de sus objetivos.

1.3. Objetivos del trabajo de tesis

1.3.1. Objetivo General

Diseñar un sistema autónomo para seguimiento de objetos móviles terrestres con estructura rígida (no deformable), por medio de técnicas de aprendizaje computacional.

1.3.2. Objetivos Específicos

- Seleccionar un detector de objetos en imágenes y entrenarlo para detectar el objeto de interés.
- Desarrollar un modulo de estimación de posición 3D del Objetivo.
- Diseñar dos técnicas de control de seguimiento de trayectorias.
- Diseñar una arquitectura de Red Neuronal (RN) para el control de seguimiento de trayectoria, con el objetivo de obtener los comandos de control de vuelo de Alabeo, Cabeceo y/o Guiñada.
- Implementar los sistemas propuestos en un simulador.
- Validar el comportamiento de los controles.
- Evaluar los resultados obtenidos del desempeño del control al utilizar técnicas de aprendizaje computacional contra los obtenidos con técnicas de control.
- Implementar los sistemas de control propuestos en un UAV físico para el seguimiento de un vehículo de locomoción terrestre.
- Evaluar los resultados obtenidos del desempeño del control al utilizar técnicas de aprendizaje computacional contra los obtenidos con técnicas de control, tanto en los resultados de simulación y en la plataforma física.

1.4. Metodología

La metodología propuesta se muestra a continuación:

1. **Creación del ambiente de simulación:** El entorno de simulación se diseñará en el software Gazebo e incluirá lo siguiente:
   - Una superficie terrestre de exploración.
   - El UAV Bebop 2.0 de Parrot (como seguidor)
   - El vehículo terrestre Husky (Objetivo).
2. **Creación del conjunto de imágenes de entrenamiento para detección del objetivo:** En el entorno de simulación del paso 1, se realizarán vuelos manuales del UAV sobre el vehículo terrestre a tres alturas diferentes. Se capturarán imágenes desde diversos ángulos del objetivo que serán utilizadas como conjunto de entrenamiento y validación para el módulo de percepción del objetivo.
3. **Implementación de la red de detección de objetos para el modulo de percepción:** Por medio de una CNN como YOLO (*You Only Look Once*) se detectará el objetivo tras haberla entrenado con el conjunto de imágenes del punto 2.
4. **Creación del módulo de estimación de posición del objetivo:** A través del algoritmo propuesto en (Martinez-Carranza y Rojas-Perez, 2022) y de la detección obtenida por el módulo de percepción se estimará la posición 3D del objetivo.
5. **Diseño del controlador Proporcional Integral Derivativo (PID) para seguimiento de trayectorias:** Se diseñará y sintonizará un control PID para el seguimiento de trayectorias para el UAV Bebop 2.0. Este controlador en conjunto con los módulos anteriores servirá para generar las trayectorias que se utilizarán para validar y comparar el desempeño de los controles PID, MPC, y sus versiones neuronales.
6. **Diseño del controlador de seguimiento de trayectorias utilizando un Control Predictivo basado en el Modelo (MPC):** Se diseñará y sintonizará el MPC para el seguimiento de trayectorias para el UAV Bebop 2.0.
7. **Diseño de las trayectorias de validación de los controles PID, MPC y sus versiones neuronales:** Se conducirá de forma manual el robot terrestre HUSKY con la finalidad de crear trayectorias con transiciones entre curvas, tramos rectos y pausas. Durante este proceso el UAV seguirá al objetivo con el control PID utilizando la posición obtenida del modulo de estimación de posición. Las posiciones estimadas durante el seguimiento se obtendrán en lapsos de tiempo constantes, obteniendo así un conjunto de trayectorias que se utilizarán para la validación de los controladores PID, MPC, y de sus versiones neuronales.
8. **Validación de los controles PID y MPC:** Se realizará el seguimiento de las trayectorias obtenidas en el punto 7 a través del UAV utilizando los controles PID y MPC. Durante el seguimiento de las trayectorias anteriores se almacenarán a una frecuencia constante las acciones de control enviadas al UAV $(u_x, u_y, u_z, u_{\omega z})$ y algunos estados como altitud, posición actual, etc.

Con esta información, se calcularán diversas métricas que se utilizarán para la validación del desempeño de los controladores proporcionando observaciones precisas sobre su eficacia y posibles áreas de mejora.

9. **Diseñar las arquitecturas de red neuronal para el control de seguimiento:** Estos módulos tomarán como entrada los últimos 5 puntos guía de la trayectoria del objetivo, proporcionados por el módulo de estimación de posición, junto con ciertos estados del UAV. Utilizando esta información, la red calculará los valores de control de alabeo, cabeceo,y guiñada para seguir el objetivo. Las arquitecturas estarán basadas en las previamente desarrolladas en (Cai et al., 2019; Li et al., 2021; Muller et al., 2019; Xie et al., 2020).
10. **Creación del conjunto de entrenamiento y validación de los controles neuronales:** El conjunto de datos recopilado en el punto 8 será empleado en la fase de entrenamiento y validación de los controladores neuronales PID y MPC.
11. **Validación de los controles neuronales:** Se entrenarán las arquitecturas desarrolladas utilizando el conjunto de datos del punto 10 y se realizará el seguimiento de un conjunto de trayectorias generadas en el punto 7, con el propósito de validar los controladores neuronales. Este proceso se realizará mediante un procedimiento similar al descrito en el punto 8.
12. **Evaluar los resultados de los controladores contra sus versiones neuronales:** Se realizará el seguimiento de las trayectorias utilizadas en el punto 11, ahora con los controladores PID y MPC con el objetivo de realizar un análisis de los resultados obtenidos y contrastarlos con los obtenidos a través de los controles neuronales.
13. **Implementación de los controladores en el UAV físico:** Implementar los algoritmos de control desarrollados en el UAV Bebop 2.0 para realizar el seguimiento de un robot móvil terrestre de locomoción con ruedas con sistema motriz diferencial.
14. **Analizar y comparar los resultados:** Con los resultados obtenidos de los controladores implementados de forma real en el UAV Bebop 2.0 de Parrot vs los resultados obtenidos a través de simulación para cada controlador respectivamente.

1.5. Contribuciones

Las principales contribuciones de la tesis son:

- Inspirados en investigaciones previas (Cai et al., 2019; Muller et al., 2019; Xie et al., 2020), introducimos una arquitectura de control neuronal innovadora entrenada a partir de un controlador óptimo. Esta arquitectura realiza un seguimiento de un objetivo móvil utilizando el historial de los últimos 5 puntos guía que describen su movimiento. Estos puntos guía representan el historial de la posición tridimensional del objetivo con respecto a un sistema global de coordenadas, estimados mediante una cámara monocular a bordo del UAV.

- A diferencia de los enfoques anteriores de (Cai et al., 2019; Muller et al., 2019), donde el UAV se mantiene en movimiento constantemente, nuestro trabajo aborda la posibilidad de que el UAV se detenga y mantenga un vuelo estacionario cuando el objetivo también se detiene. Posteriormente, el UAV puede reanudar el seguimiento una vez que el objetivo comienza a moverse nuevamente.
- En contraste con el enfoque propuesto por Sa et al., 2017, que emplea un controlador MPC para gestionar la posición en X e Y, y controladores PID para la altitud y orientación con el ángulo de guiñada, nuestra investigación opta por una variante neuronal del MPC para el control de posición, mientras que mantiene el control de altitud y guiñada mediante un controlador PID.
- Se demuestra la viabilidad de obtener un controlador neuronal con un rendimiento comparable al de un control MPC, optimizando su tiempo de ejecución para su implementación en sistemas con recursos computacionales limitados, no restringiéndose únicamente a procesos lentos para los cuales fue diseñado el MPC.

1.6. Estructura de la Tesis

La estructura del trabajo desarrollado se muestra a continuación. En el capítulo 2 se muestra una investigación sobre el estado del arte, con los trabajos mas relevantes relacionados al trabajo de tesis propuesto, pasando al capítulo 3 donde se muestra el marco teórico sobre los principales conceptos necesarios para comprender e implementar el seguimiento de objetivos móviles a través de un UAV por medio de DNN. En el capítulo 4 se muestra la metodología seguida para el desarrollo de este trabajo de investigación, pasando a la sección 5 donde se muestran los experimentos realizados y los resultados obtenidos, finalmente en el capítulo 6 se presentan las conclusiones obtenidas a partir de los resultados experimentales obtenidos y se plantea el trabajo futuro propuesto que podría complementar el trabajo desarrollado.

Capítulo 2

Estado del Arte

Este capítulo proporciona una revisión de la literatura que aborda el seguimiento de objetivos móviles, centrándose en el uso de técnicas clásicas de control y de aprendizaje computacional aplicadas en vehículos autónomos. La investigación se enfoca principalmente en los desarrollos relacionados con la percepción, la planificación de trayectorias y el control de seguimiento.

2.1. Detección

La cámara es uno de los sensores principales contenidos en un UAV, a través de ella el UAV puede percibir el entorno en el que se encuentra. Este dispositivo es utilizado principalmente para conocer visualmente el entorno y navegar a través de él. Esto se logra principalmente con técnicas de visión por computadora para detectar algún objeto dentro del Campo de Visión (FoV, por sus siglas en inglés) del sensor y poder evadir obstáculos. Aunque la detección de objetos puede lograrse mediante la combinación de información de diversos sensores, como se discute en (Wu et al., 2022), el enfoque basado en visión destaca por su resistencia a la interferencia, rapidez, precisión y aplicabilidad a diversos escenarios, en comparación con métodos como la adquisición de posición mediante sistemas como el Sistema de Posicionamiento Global (GPS) o el Sistema de Aumentación Basado en Satélites (SBAS). En este contexto, se describen a continuación algunos trabajos relevantes que emplean sistemas de visión para la estimación de la posición del objetivo.

En (Wu et al., 2022) utilizan una cámara monocular y YOLOv5 para detectar un marcador con un símbolo H como el objetivo, de la detección obtienen la posición del píxel del objetivo en la imagen y, junto con la información conocida de su tamaño real, estiman la posición del UAV, relativa al objetivo a través de una transformación de un sistema de coordenadas. En la figura 2.1. se observa su sistema utilizado.

Por otro lado, en (Li et al., 2021) se realiza el seguimiento de un objetivo en movimiento a través de un UAV, utilizan una cámara airbone para tomar imágenes que son utilizadas como entrada del módulo de percepción para obtener como salida la

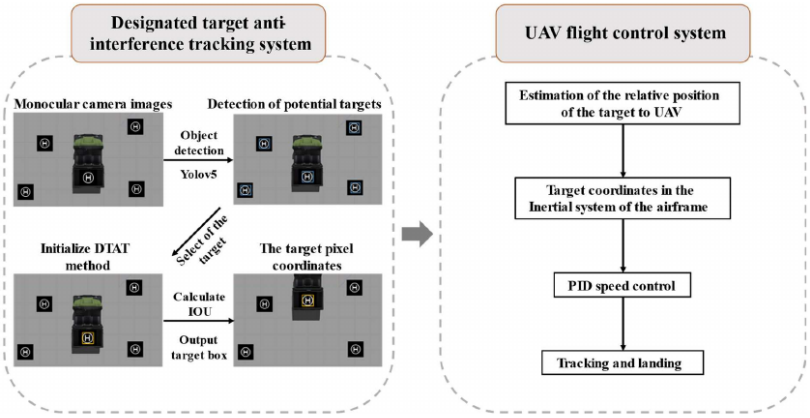


Figura 2.1: Sistema de percepción y control utilizada en (Wu et al., 2022).

distancia relativa entre el UAV y el objetivo para cada una de las imágenes. El módulo se encarga de la detección del objetivo con YOLOv3 entrenándola con un conjunto de 400 imágenes para el objetivo de interés del cual, el 80 % se utilizó para entretenimiento y el 20 % para las pruebas. A través del proceso de detección, se extrae la posición y tamaño del objetivo en la imagen. Esta información se utiliza para estimar la posición, abordando el problema de *perspective-n-point* (PNP), como se ilustra en la figura 2.2. En este trabajo, se mantiene constante la altura del UAV.

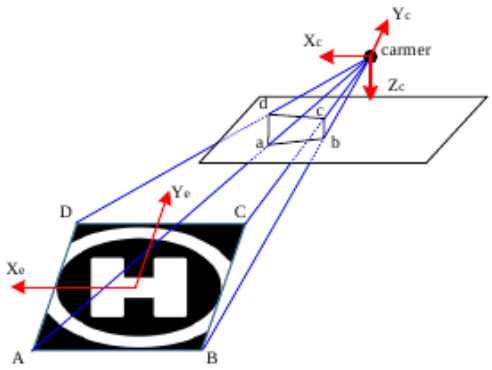


Figura 2.2: Diagrama conceptual de Perspective-n-point (PNP). Imagen tomada de (Li et al., 2021).

En (Tian et al., 2022) se realiza el seguimiento de objetos móviles pequeños enfocado al seguimiento de peatones, realizan una mejora de YOLOv4 adaptándola a objetivos pequeños desde la perspectiva del UAV evaluándola con el conjunto de datos de VisDrone, el modelo obtenido se recorta, comprime y se combina con *Kalman Correlational Filter* (KCF) para realizar el reconocimiento y seguimiento en tiempo real. En sus experimentos, el UAV se encuentra a una altura de 16 metros y realiza el seguimiento a lo largo de 493 metros en 7 minutos y 13 segundos, equivalente a una

velocidad promedio de 1.14 m/s. Su algoritmo mejorado puede realizar el seguimiento del objetivo incluso aunque exista oclusión parcial del mismo. Cuando esto ocurre, el algoritmo compara la posición del objeto antes y después de desaparecer. Si la diferencia de posición se encuentra dentro de un umbral, el algoritmo continua con el seguimiento. En caso contrario, el algoritmo se reinicia. De esta manera el algoritmo puede realizar el reconocimiento, inicialización automática y un seguimiento robusto.

La estimación de la distancia se lleva a cabo según la posición del objeto, representada por las coordenadas (x, y) y las dimensiones (w y h) correspondientes al ancho y alto de la detección realizada. Posteriormente, se estima la posición relativa del objeto respecto al centro de las coordenadas en píxeles de la imagen. Las imágenes capturadas tienen una resolución de 1920x1080 píxeles, donde el objetivo ocupa únicamente entre 69x63 píxeles. Para mitigar la interferencia de objetivos similares, se implementa un algoritmo de seguimiento *KCF*.

En (Mercado-Ravell et al., 2019) se realiza la detección y persecución de un objeto con dinámica desconocida, en particular de un rostro humano. La detección del objeto se logra usando un clasificador Haar basado en características en cascada, el nodo de detección visual provee la posición del objetivo a seguir, primero se estima la posición relativa del objetivo con respecto al UAV transformando las coordenadas del espacio de imagen al mundo real, donde la profundidad es estimada usando a priori el conocimiento del tamaño del objeto en el mundo real.

Realizan la implementación en el A.R.Drone Parrot y ejecutan los nodos previamente mencionados en ROS sobre una estación terrena para lograr la detección de objetivos y seguimiento con el UAV. A través de *AR Drone Driver* pueden enviar las señales de control, obtener la información de los sensores embebidos del UAV y el vídeo de ambas cámaras a una tasa de 30 Hz.

Por otro lado, en (Zhang et al., 2020) proponen un *Framework* de seguimiento visual de objetos llamado DyWinSiam que puede acelerar el proceso de seguimiento significativamente comparado con el estado de arte de seguidores basados en redes neuronales, el cual puede seguir un objetivo como personas, bicis, carros etc., de una secuencia de vídeo capturada en tiempo real. Resultados experimentales muestran que su seguidor propuesto puede correr a una velocidad de 98.78 Cuadros Por Segundo (FPS), que es 3.92/1.78 veces comparada con SiamRPN y DaSiamRPN, respectivamente, manteniendo su precisión alrededor del 79 %.

En (Huang y Lai, 2020) se detecta y estima la distancia de un UAV objetivo de ala fija entrando en el campo de visión de una cámara monocular a bordo de un UAV. El diagrama del sistema que utilizan para detectar al objetivo se muestra en la figura 2.3, a través de YOLO preentrenado con COCO (Lin et al., 2014), se realiza la detección del objetivo, una vez teniendo el *bounding box* se recorta el objetivo en una ventana de tamaño de 100x100 pixeles, la imagen entra a la Red Neuronal Convolucional (CNN) que cuenta con dos etapas, una red de extracción de características basada en VGG16 y posteriormente la etapa de estimación de la posición.

Se utiliza el filtro de Kalman para mejorar el seguimiento del objetivo cuando YOLO no es capaz de detectar al objetivo y a su vez suavizar la distancia estimada

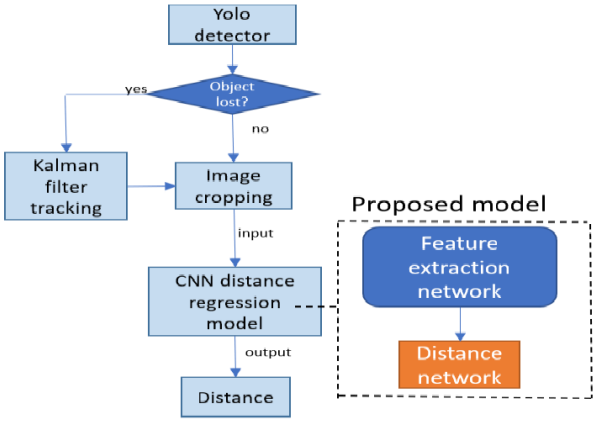


Figura 2.3: Diagrama de estimación de distancia con DNN a través de una cámara monocular utilizada en (Huang y Lai, 2020).

por la red. Para la etapa de estimación de posición de su sistema de aprendizaje con Aprendizaje Profundo (DL) utilizan un conjunto de datos de 10,000 imágenes sintéticas recortadas de 100x100 pixeles obtenidas de blender (software de animación) con distancias entre 30 m a 95 m del objetivo a diferentes valores de actitud: Alabeo y Cabeceo en un rango de [-15°, 15°] y Guiñada en un rango de [-75°, 75°]. La arquitectura de su red de estimación consta de 4 capas totalmente conectadas seguidas por un *batch* de normalización y activación Relu.

La evaluación de sus resultados se realizó con vídeos sintéticos y reales con una velocidad de vuelo del objetivo constante y se determinó el RMSE antes y después de aplicar el filtro de Kalman. Para un caso de vídeo sintético a una distancia de 30 a 60 metros, se obtuvo un RMSE de 1.580 m antes de aplicar el filtro de Kalman y de 0.651 m después de aplicarlo. Para las pruebas reales utilizaron un UAV de ala fija como el objetivo, la posición se determina través del Sistema de Posicionamiento Global (GPS) a 5 Hz. Mientras que, el vídeo se toma a 30 FPS con una resolución de vídeo de 1920x1080 pixeles obteniendo un RMSE de 3.369 m a 12 diferentes puntos de captura de entre 20 a 60 metros de distancia.

En (Muller et al., 2019) utilizan redes neuronales para la percepción y el control, la red de percepción recibe como entrada una imagen monocular RGB a una tasa de 60 FPS para obtener como salida una trayectoria relativa a la posición actual del UAV. En su trabajo se obtienen 5 puntos uniformemente muestreados que representan la trayectoria a seguir, dicha red consta de 8 capas, 5 capas convolucionales con 20, 24, 28, 30 y 32 filtros y 3 capas completamente conectadas con 1800, 800 y 100 unidades ocultas respectivamente, utilizan una función de perdida L2 y un *dropout* de 0.5 en las capas totalmente conectadas, su sistema fue implementado en TensorFlow con una tasa de aprendizaje de 5×10^{-4} con un optimizador Adam. Por otro lado la red de control calcula señales de control de bajo nivel tomando la trayectoria y los estados del UAV como entrada como se observa en la figura 2.4.

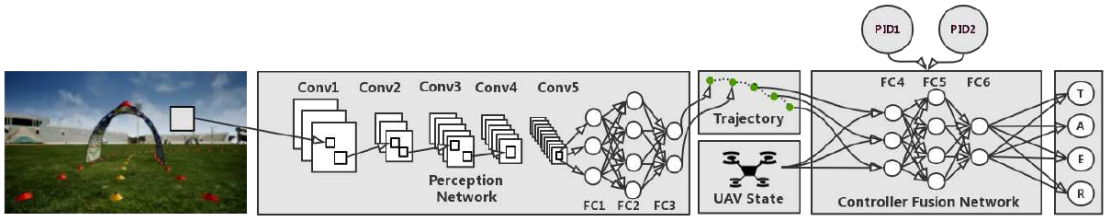


Figura 2.4: Sistema de percepción y control utilizado en (Muller et al., 2019).

Finalmente en (Gama et al., 2022) desarrollaron un método *end to end* para la detección y seguimiento de personas utilizando DNN, la etapa de detección consiste en un modelo basado en *Single Shot Detector* (SSD) con *MobileNet* al que llaman *MobilNet SSD* que reduce la potencia consumida por las redes neuronales y de esta manera poder ser utilizada en dispositivos de gama baja e implementarse de manera real en un cuadricóptero de pequeña escala. Su trabajo se enfoca en la detección de una sola clase, utilizaron un conjunto de datos de 60,000 imágenes de 32x32 píxeles separados en 10 grupos para el entrenamiento del sistema de detección, la salida de la red es la imagen con un *boundary box* sobre el objeto detectado.

2.2. Planeación

Existen trabajos en los que la la etapa de detección, planeación y control se realizan en un mismo módulo, llamados métodos *end to end*. En el trabajo desarrollado por (Cai et al., 2019) los módulos de percepción, capas de comportamiento y planificación del movimiento trabajan juntos en un método *end to end* que recibe un comando de alto nivel e imágenes para la navegación, aplicado a un vehículo terrestre autónomo. El comportamiento total del sistema se muestra en la figura 2.5.

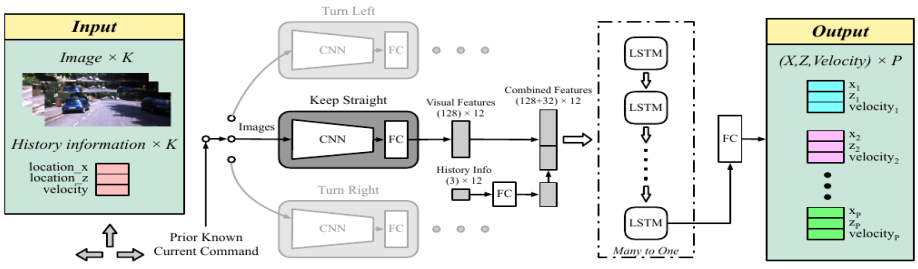


Figura 2.5: Sistema *end to end* utilizado en (Cai et al., 2019).

Su algoritmo usa imágenes de la vista frontal e información de los estados de los pasados 1.5 segundos como entrada para calcular una posible trayectoria sin colisiones, esta trayectoria contiene velocidad y posición lateral/longitudinal 3 segundos en el futuro. Con una CNN extraen las características visuales del entorno y a través de una *Long-Short-Term Memory* (LSTM) consideran las dependencias temporales.

Su funcionamiento se basa en 3 comandos de manejo: gira izquierda, gira derecha o ve de frente, con el cual se selecciona la subred adecuada donde se reciben imágenes con una resolución de 244x244 píxeles y obtiene a la salida un vector de características de 128. Su estructura esta conformada por 4 capas convolucionales seguido de un Perceptrón MultiCapa (MLP) con tres capas totalmente conectadas, la activación de las capas convoluciones se realiza con ReLU, mientras que para el modulo de LSTM el numero de características en estados ocultos es de 512 con 3 capas recurrentes, el sistema fue implementado en PyTorch y utilizan el optimizador Adam con una tasa de aprendizaje de 0.001 y un tamaño de *batch* de 32. Cada una de las subredes tienen la misma arquitectura solo difieren en los conjuntos de datos con las que fueron entrenadas.

Cada red aprende de trayectorias de ejemplo, tomando como entrada imágenes y estados pasados de movimiento del carro. La salida es una trayectoria libre de colisiones de 3 segundos en el futuro, utilizan información de un conjunto de datos de Robotcar del cual extrajeron 52,300 imágenes y las etiquetaron con diferentes comandos indicando al vehículo a donde ir basado en la trayectoria *ground truth*, en este caso no utilizaron comandos en los cuales el auto quedaba en alto total como en señales de alto o semáforo en rojo, es decir; el vehículo siempre se encuentra en movimiento. Posteriormente se realizo un procesamiento de cada una de las 52,300 imágenes equipándolas con la trayectoria previa en los pasados 1.5 segundos y con una interpolación de la trayectoria en los próximos 3 segundos con posición y velocidad a 15 Hz en un sistema de coordenadas local, para evaluar su método miden el promedio del error lateral, longitudinal y de posición final entre todos los puntos planeados y los puntos reales de la trayectoria.

A diferencia del trabajo desarrollado por (Cai et al., 2019), donde se logra la percepción, y planeación en un método *end to end*, en (Zhilenkov y Epifantsev, 2018) proponen un sistema de 4 etapas para realizar la percepción, planeación y control de seguimiento con los siguientes módulos:

- Un sistema de vídeo panorámico.
- Un sistema de unidades de procesamiento en paralelo.
- Un sistema de reconocimiento y clasificación.
- Un sistema de decisión.

Las primeras dos etapas consisten en adquirir las imágenes para después recopilar y extraer información importante como bordes, pasando a la tercera etapa en la que se realiza la clasificación, a la salida de la red neuronal se obtienen los 3 valores que reflejan la probabilidad de que la entrada pertenezca a alguna de las siguientes clases: girar derecha, girar a la izquierda o moverse hacia adelante. El bloque 4 contiene un sistema de decisión y un sistema de análisis de trayectorias, el primero de ellos se encarga de analizar las decisiones entrantes basadas en la información sobre la trayectoria previa del trafico y el mapa del terreno.

También proponen utilizar métodos de clasificación y toma de decisiones con redes neuronales artificiales y lógica difusa. La estructura del sistema desarrollado se muestra en la figura 2.6.

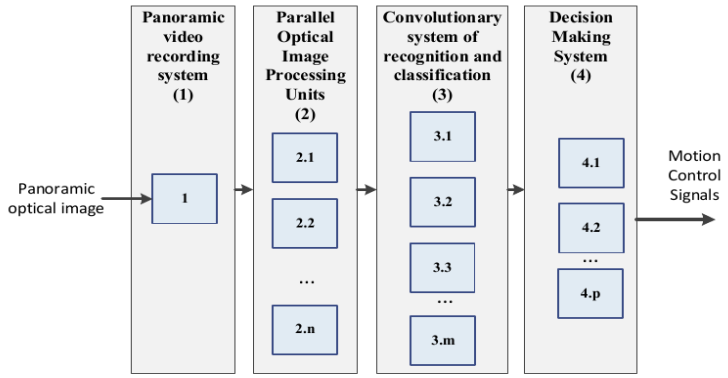


Figura 2.6: Estructura simplificada del sistema de navegación autónoma propuesta por (Zhilenkov y Epifantsev, 2018).

2.3. Seguimiento

El trabajo desarrollado por (Xie et al., 2020) se enfoca en seguir una plataforma de aterrizaje móvil a través de un UAV utilizando Aprendizaje por Refuerzo (RL) y una red neuronal en un método *end to end*, consiste en el desarrollo de una estrategia híbrida que consta de dos partes, una etapa de seguimiento y una de aterrizaje, la etapa de seguimiento utiliza un algoritmo de aprendizaje por refuerzo para ajustar la velocidad del UAV en el plano horizontal y lograr un seguimiento estable de la plataforma en movimiento (vehículo terrestre con ruedas), el aterrizaje ajusta la altura del UAV en la dirección vertical basada en reglas heurísticas. En la figura 2.7 se muestra la estructura híbrida utilizada. El método propuesto no requiere información previa del movimiento de la plataforma y el UAV puede trabajar bien incluso con medidas intermitentes y ruidosas.

Como se muestra en la figura 2.7, la red consta de tres capas completamente conectadas, las capas FC1 Y FC3 son activadas con la función Relu con 30 neuronas ocultas cada una, y la capa FC2 se activa con la función Tanh con 2 neuronas, teniendo como entrada los estados del UAV y a la salida las acciones a realizar.

Las pruebas en simulación se realizaron en el *Modular Open Robots Simulation Engine* (MORSE) en un área de 37 x 85 metros y la frecuencia del controlador del UAV fue de 20 Hz, limitaron la velocidad horizontal por debajo de 10 m/s y la velocidad vertical por debajo de 3 m/s . La evaluación se llevo a cabo calculando el RMSE y Tasa de Éxito de Seguimiento (TSR) definida como la distancia horizonte entre el UAV y la plataforma móvil (ecuación 2.1). Los resultados de su trabajo se muestran en la figura 2.8, para la etapa de seguimiento se comparan los resultados obtenidos con un control PID sintonizado con el método de proporción critica y la experiencia en ingeniería.

$$TSR = \sum_{i=0}^{N} \frac{D_i}{N} \text{x} 100\,\%, D_i = \begin{cases} 1 & dist < 3 \\ 0 & dist \geq 3 \end{cases} \tag{2.1}$$

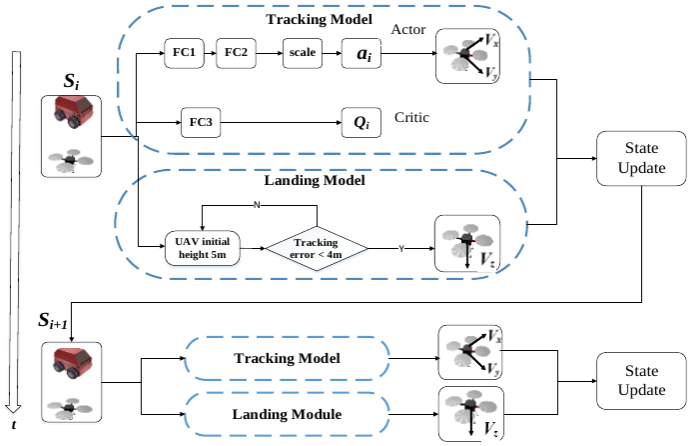


Figura 2.7: Estrategia híbrida de los módulos de seguimiento y aterrizaje utilizada en (Xie et al., 2020).

Controller	PID Method				Proposed Method				Units
Movement	Linear	Circular	Random1	Random2	line	Circular	Random1	Random2	
x axis (RMSE)	0.2969	2.4204	3.3479	3.7198	0.3896	1.7466	2.8444	2.6720	m
y axis (RMSE)	0.8809	1.9919	2.7271	3.6738	1.1834	2.0197	1.9053	2.0395	m
TSR (%)	95.0	65.5	27.5	20.3	94.8	63.8	45.3	37.2	–

Figura 2.8: Resultados del seguimiento de una plataforma móvil obtenidos en (Xie et al., 2020).

En el trabajo desarrollado por (Li et al., 2021), para la tarea de seguimiento se aplica un método de RL para entrenar una red neuronal Poly para control de las decisiones. Utilizan dos módulos, uno de percepción y otro de *Guidance Control*, los resultados del módulo de percepción son la entrada del módulo de *Guidance Control* de tal manera que a la entrada se reciben 5 distancias relativas entre el UAV y el objetivo en los 5 instantes de tiempo previos, con esta información se obtiene a la salida la velocidad calculada del UAV en x y y. En la figura 2.9. Se observa el sistema general propuesto en su trabajo.

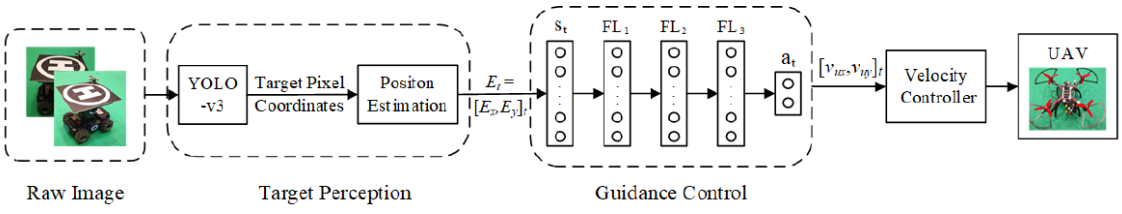


Figura 2.9: Sistema de percepción y control propuesto en (Li et al., 2021).

Los nodos de las capas ocultas son [128 128 64] la función de activación de las capas ocultas es la función Relu y la función de la capa de salida es la función Tanh. Los resultados también fueron validados con una plataforma física utilizando un

Velocity	0.5 m/s	Velocity	0.5 m/s
Proportional Integral Derivative (PID) Controller			
MTE of X (m)	0.55	1.14	
MTE of Y (m)	0.12	0.16	Failed
SSE (m)	0.52	1.53	
Neural Network Controller			
MTE of X (m)	0.46	0.80	0.83
MTE of Y (m)	0.02	0.07	0.08
SSE (m)	0.41	0.85	0.95

Figura 2.10: Errores de seguimiento entre el UAV y el objetivo moviéndose en linea recta con diferentes velocidades reportados en (Li et al., 2021).

cuadricóptero ensamblado por ellos mismos y el vehículo de aterrizaje corresponde al DJI robotmaster s1. Su plataforma experimental es capaz de realizar el seguimiento del objetivo a una velocidad de hasta 1.2 m/s. Concluyen que la velocidad máxima de seguimiento esta relacionada al FoV de la cámara y a las características inherentes del UAV. Las pruebas fueron realizadas en un ambiente de simulación, en este entorno el UAV y el objetivo pueden moverse libremente.

El seguimiento en simulación mantiene en todo momento al objetivo en el FoV, cuando el objetivo escapa del campo de visión, el UAV se detiene y el proceso de seguimiento falla. La variación de velocidad cambia aleatoriamente cada 0.5 segundos entre 0.5 y 1 m/s, los resultados fueron comparados con un control PID para contrastar los resultados obtenidos con su control neuronal, el desempeño se midió calculando el Error Medio de Seguimiento (MTE) para el eje X y Y y el Error en Estado Estacionario (SSE) obteniendo mejores resultados para el control neuronal al seguir mas rápido al objetivo que el PID. Se concluye que la velocidad del objetivo para cuando el UAV vuela a 3m de altura debe ser de 1.2 m/s, los resultados reportados se muestran en la figura 2.10.

En (Mercado-Ravell et al., 2019) los autores utilizaron el filtro de Kalman para el seguimiento y estimar la posición relativa del movimiento del objeto con respecto a la cámara del UAV, el Filtro de Kalman (KF) para el seguimiento de objetivos en movimiento hace al sistema mas robusto a falsos positivos mientras existan otros objetivos en la escena, los resultados son una estimación mas suave donde el KF es capaz de manejar detecciones erróneas. El filtro de Kalman es una técnica poderosa para la estimación de estados óptimos y filtrado de sistemas lineales perturbados con ruido Gaussiano. En este caso ellos aplican una versión discreta del KF al modelo cinemático del objetivo detectado.

Proponen un controlador de posición relativa, después de calcular la posición relativa entre el UAV y el humano objetivo en el marco de referencia, donde el UAV es controlado como un robot omnidireccional realizando movimientos laterales a través del Alabeo y Cabeceo en lugar de la Guiñada, el seguimiento se realiza a través de un controlador PD lineal en cascada para mantenerse siempre a 2 metros del objetivo, todos los cálculos son realizados *off board*, enviando las medidas de los sensores a una estación terrestre a una frecuencia de 200 Hz.

El procesamiento de imágenes y los algoritmos de control se realizan en tiempo real en ROS a una tasa de 30 Hz. Se implementó un tercer nodo para el control de posición relativa y un controlador PD se utiliza para mantener cierta distancia entre el objetivo y el UAV. El RMSE obtenido para la estimación de posición relativa fue de 0.1294 m para x, 0.2392 m para y, y 0.1802 m en z.

En (Muller et al., 2019) se propone que una DNN también puede ser aplicada para el control permitiendo que la dinámica del UAV pueda ser inherentemente aprendida. Básicamente se propone un controlador de aprendizaje optimizando una DNN a la que llaman *Controll Fusion Network* (CFN) (figura 2.4) donde la red aprende de un controlador robusto con trayectoria filtrada, que suprime las trayectorias ruidosas y las imperfecciones de controladores individuales. CFN toma los puntos de la trayectoria predicha de la red de percepción y junto con los estados del UAV (orientación y velocidad) como entradas predicen las 4 señales del control del UAV Impulso, Cabeceo, Alabeo y Guiñada.

La DNN consta de 3 capas completamente conectadas con 64, 32, 16 unidades ocultas con un *dropout* de 0.5 en la segunda capa, optimizador Adam y una tasa de aprendizaje de 1×10^{-3}. No se reportan resultados numéricos.

Capítulo 3

Marco Teórico

El siguiente capítulo aborda conceptos fundamentales para el desarrollo del presente trabajo, centrándose específicamente en la tarea de seguimiento de objetivos móviles en aplicaciones de exploración planetaria. Se analiza información relevante proveniente de los rovers de la misión Mars 2020, con un enfoque principal en Ingenuity. Estos datos son vitales para establecer criterios de diseño que aseguren la implementación eficiente y efectiva de las técnicas de seguimiento en el contexto de la exploración planetaria.

Además, se presentan las plataformas robóticas utilizadas, el control PID, MPC, control neuronal con Perceptrón MultiCapa (MLP), el filtro de Kalman y una descripción detallada de los *frameworks* empleados, todos orientados a lograr los objetivos establecidos en la sección 1.3.

3.1. Misión Mars 2020

El 18 de febrero de 2021, Perseverance aterrizó con éxito en el cráter Jezero en Marte, llevando consigo el helicóptero Ingenuity. La misión principal de Perseverance es buscar signos directos de vida y preparar muestras geológicas cruciales para un eventual retorno a la Tierra. Mientras tanto, el principal objetivo de Ingenuity fue demostrar la tecnología del primer vuelo propulsado en otro planeta. Aunque actualmente Ingenuity ha concluido sus vuelos en Marte, ha demostrado con éxito la capacidad de volar y maniobrar en la atmósfera marciana. Esto sugiere que la tecnología de vuelo puede ser implementada con éxito en futuras misiones en la superficie de Marte (Green, 2021).

3.1.1. Perseverance

El rover Perseverance (figura 3.1) tiene una velocidad promedio de 4.2 cm/s, aunque su velocidad puede parecer lenta en comparación con la de los vehículos terrestres, es importante destacar que dentro de los estándares establecidos para vehículos marcianos, el rover es notable por su capacidad para transitar terrenos planos y difíciles (von Ehrenfried, 2022).

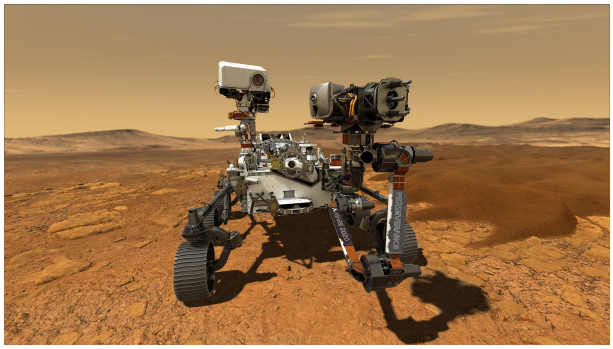


Figura 3.1: Ilustración del rover Perseverance operando en la superficie de Marte. Tomada de NASA/JPL-Caltech.

En la planificación de las trayectorias del rover, se ha implementado un algoritmo que incluye una pausa planificada después de un intervalo de tiempo específico. Esta estrategia se ha diseñado para asegurar un desplazamiento seguro y eficiente del rover en Marte. El patrón de movimiento del rover consiste en conducir durante aproximadamente 10 segundos, detenerse para observar y comprender el terreno circundante durante 20 segundos, y luego continuar avanzando de manera segura durante otros 10 segundos. Este enfoque contribuye a una exploración cuidadosa y detallada del entorno marciano[1].

3.1.2. Ingenuity

Ingenuity está compuesto por diversos subsistemas que incluyen la interfaz de mástil, palas del rotor, actuadores, sistema de aterrizaje, energía, fuselaje y térmica, telecomunicaciones, sensores y cámaras, así como el procesamiento de aviónica (Balaram et al., 2021). La figura 3.2 muestra una ilustración del Ingenuity.

Figura 3.2: Ilustración del helicóptero Ingenuity en la superficie de Marte. Tomada de NASA/JPL-Caltech.

[1] https://mars.nasa.gov/mer/mission/rover/wheels-and-legs/

El sistema de sensores de navegación de Ingenuity comprende componentes esenciales, entre ellos, la unidad de medición inercial (IMU), una cámara monocromática orientada hacia el nadir y otra cámara a color orientada hacia el exterior, además de un telémetro láser (LRF) y un inclinómetro (Tzanetos et al., 2022). La cámara inferior respalda la odometría visual, proporcionando información precisa sobre la posición, velocidad y altitud del helicóptero, datos esenciales para la navegación (von Ehrenfried, 2022).

El helicóptero Ingenuity utiliza sistemas avanzados de percepción y navegación durante sus operaciones en Marte. Su sistema de navegación se fundamenta en el algoritmo de Estimación Visual de Navegación del Helicóptero en Marte (MAVeN), el cual le permite determinar su posición y movimiento en vuelo mediante la combinación de detección y seguimiento de características visuales, mediciones inerciales y distancias al suelo (Grip et al., 2022).

Las imágenes capturadas por la cámara de navegación se procesan a una velocidad de 30 Hz para detectar y seguir características visuales en cada cuadro. Estas características, junto con las mediciones del telémetro láser (LRF) a 50 Hz, se utilizan para corregir la solución mediante un filtro de Kalman extendido (Tzanetos et al., 2022).

El sistema de Guía, Navegación y Control (GNC) del Ingenuity desempeña un papel crucial en su operación exitosa en Marte, a continuación se muestra una breve descripción de cada modulo:

- **Guía**: Este subsistema genera trayectorias deseadas basadas en comandos enviados desde el control terrestre, estableciendo la dirección y el camino que el helicóptero debe seguir durante el vuelo.
- **Navegación**: Determina la posición y movimiento del helicóptero en vuelo mediante detección y seguimiento de características visuales, mediciones inerciales y distancias al suelo, permitiendo que el Ingenuity mantenga su curso y realice maniobras precisas.
- **Control**: Se encarga de minimizar la diferencia entre el movimiento deseado y el estimado del helicóptero mediante el comando de los actuadores, controlando la estabilidad, altitud y dirección del vuelo para un rendimiento óptimo durante las operaciones.
- **Comandante de Modo**: Coordina acciones asociadas con diversas fases de vuelo, como la inicialización, despegue y aterrizaje, garantizando transiciones suaves entre diferentes estados de vuelo.

Este conjunto integral de sistemas garantiza que el Ingenuity pueda realizar operaciones seguras y controladas en el entorno desafiante de Marte, permitiendo el éxito de sus misiones en el planeta rojo.

En términos de velocidad, el helicóptero tiene la capacidad de ascender a una velocidad de 1 m/s, y su altímetro proporciona mediciones confiables hasta aproximadamente 10 metros sobre la superficie marciana. No obstante, se impone una restricción

en la velocidad lateral para asegurar una superposición adecuada de imágenes de navegación entre cuadros sucesivos, lo que facilita el cálculo preciso de la velocidad lateral del vehículo. A una altura de 5 metros, la velocidad lateral se mantiene en alrededor de 2 m/s. Cabe señalar que, a altitudes mayores, se permiten velocidades laterales más elevadas (Balaram et al., 2021; von Ehrenfried, 2022).

En resumen, la misión conjunta de Perseverance e Ingenuity ha sido un logro significativo en la exploración de Marte, demostrando tecnologías avanzadas y proporcionando valiosa información sobre la superficie del planeta rojo (Grip et al., 2022).

3.2. Plataformas robóticas

Las plataformas robóticas desempeñan un papel esencial en el diseño y desarrollo de robots. Estas plataformas, al poseer capacidades específicas y configuraciones mecánicas únicas, proporcionan un sólido marco sobre el cual podemos programar algoritmos para una amplia gama de aplicaciones. Al dotar a estas plataformas con habilidades y sensores especializados se pueden utilizar para la validación de algoritmos en distintas aplicaciones, como tareas de exploración, fabricación, asistencia, entre otras. La versatilidad de las plataformas robóticas las convierte en herramientas fundamentales para impulsar avances en la robótica, permitiendo el desarrollo de soluciones innovadoras y adaptativas en diversos campos de aplicación, las plataformas utilizadas se muestran a continuación.

3.2.1. UAV

El UAV utilizado durante este trabajo es el Bebop 2.0 de Parrot, cuenta con 6 grados de libertad (GDL), 3 de traslación y 3 de orientación detallados a continuación:

- Desplazamiento hacia adelante y atrás a lo largo del eje X_{UAV}.
- Desplazamiento hacia la izquierda y derecha a lo largo del eje Y_{UAV}.
- Desplazamiento hacia arriba y abajo a lo largo del eje Z_{UAV}.
- Giro sobre el eje X_{UAV} (Alabeo).
- Giro sobre el eje Y_{UAV} (Cabeceo).
- Giro sobre del eje Z_{UAV} (Guiñada).

La figura 3.3 muestra el UAV y sus 6 GDL descritos anteriormente.

3.2.2. Robot terrestre Husky

El robot terrestre empleado en este trabajo es el HUSKY (figura 3.4), una plataforma de desarrollo robótico que, gracias a sus dimensiones (0,99 x 0,67 x 0,39 m), su potente tren motriz de alto par y su construcción robusta, puede enfrentarse a diversos terrenos y entornos de investigación. La capacidad de ajustar las velocidades de las ruedas motrices en cada lado permite al HUSKY realizar movimientos como giros de 360°, evitar

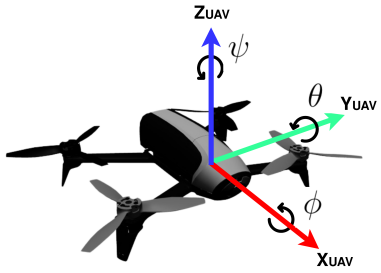


Figura 3.3: Plataforma del UAV bebop 2.0 con sus 6 GDL.

obstáculos y adaptarse a diferentes tipos de terreno. Además, su estructura resistente le posibilita hacer frente a situaciones extremas y desafiantes, como terrenos irregulares, obstáculos y condiciones climáticas adversas[2].

El HUSKY es un robot de locomoción terrestre con configuración diferencial, lo que significa que tiene dos ruedas motrices en cada lado del robot que pueden girar a diferentes velocidades. Este diseño le permite al robot realizar giros y movimientos avanzados mediante la variación de las velocidades de las dos ruedas en cada lado, lo que resulta en que el robot se mueva en sus tres grados de libertad (GDL).

- **Desplazamiento Lineal en el Eje X**: El robot puede moverse hacia adelante o hacia atrás a lo largo del eje X.
- **Desplazamiento Lineal en el Eje Y**: El robot puede moverse lateralmente a lo largo del eje Y.
- **Rotación alrededor del Eje Z** (θ): El robot puede rotar alrededor del eje vertical (Z), lo que le permite cambiar su orientación.

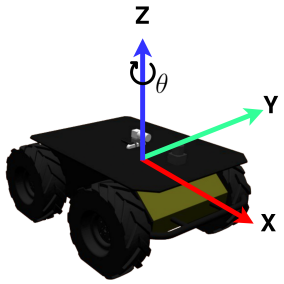


Figura 3.4: Plataforma del robot HUSKY con sus 3 GDL.

Husky cuenta con un sólido respaldo en ROS, con un código de código abierto impulsado por la comunidad. Dispone de paquetes que posibilitan la operación del vehículo y la obtención de lecturas de sus diversos sensores. Un ejemplo de esto se encuentra en el repositorio de HUSKY de Tinker-Twins [3].

[2]https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/
[3]https://github.com/Tinker-Twins/Husky?tab=readme-ov-file

3.3. Entorno de Software

Existe un amplio rango de *Open Source* para el desarrollo e implementación de sistemas robóticos y Deep Learning (DL), a continuación se muestran los *Frameworks* necesarios para el desarrollo de la presente investigación.

3.3.1. ROS

Robot Operating System ROS es un sistema de código abierto que ofrece un conjunto de bibliotecas y software, define los elementos, interfaces y herramientas necesarios para desarrollar aplicaciones de robótica avanzadas. Es compatible con una variedad de dispositivos, desde cámaras y LIDAR hasta controladores de motores, e incluye algoritmos de vanguardia para su implementación. Para mas información puede consultar el sitio web oficial [4].

Los conceptos principales para implementar ROS son mensajes, nodos, tópicos y servicios (Quigley et al., 2009) los cuales se detallan a continuación.

- **Nodo**: Son procesos individuales que se ejecutan de manera independiente comunicándose entre si a través del sistema de mensajes de ROS, se crea una funcionalidad integrada del sistema completo (conformado por todos los nodos).
- **Mensajes**: Se refiere a un tipo de estructura de datos diseñada para facilitar la comunicación entre nodos. Ejemplos de estos tipos de datos incluyen variables como enteros, números de punto flotante y otros formatos que representan la información transmitida entre los diferentes elementos del sistema.
- **Tópico**: Son un canal de publicación o suscripción diseñada para facilitar la comunicación entre nodos en ROS. Estos canales permiten el intercambio eficiente de datos entre nodos y resultan particularmente beneficiosos cuando varios nodos necesitan compartir información, como la proveniente de sensores.
- **Servicio**: Son un mecanismo de comunicación síncrona uno a uno en las que un nodo solicita una acción a otro nodo y espera su respuesta antes de continuar con la ejecución.

En la figura 3.5 se muestra una representación grafica para su mejor comprensión.

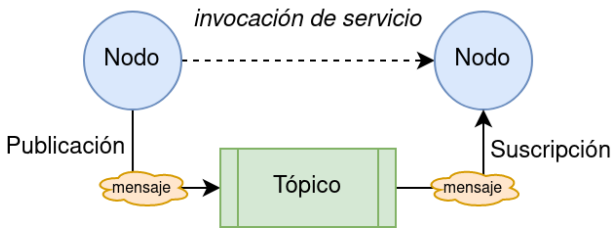


Figura 3.5: Representación de la interacción de nodos, mensajes, tópicos y servicios en ROS.

[4] https://www.ros.org/

3.3.2. bebop_autonomy

Es un *driver* de ROS de código abierto para el Bebop 1.0 y 2.0 de Parrot basado en el Kit de Desarrollo de Software oficial ARDroneSDK3 (SDK por sus siglas en inglés), con el cual podemos establecer la comunicación entre el UAV y ROS habilitando la publicación y suscripción de los tópicos del Bebop, y de esta manera tener acceso a la información de sus sensores y enviar comandos de control (Giernacki et al., 2020).

Una vez que el UAV despega de la superficie, el movimiento de traslación y orientación se logra a través del comando de pilotaje enviando señales de control *u* a través del tópico $bebop/cmd\_vel$:

$$u(t) = [\text{linear.x, linear.y, linear.z, angular.z}]^T$$

Cada una de estas señales se encuentra en un rango de $[-1, 1]$ que el controlador interno convierte en ángulos específicos y velocidades lineales y angulares $[\phi, \theta, v_z, \omega_z]$ respectivamente con la siguiente relación:

roll_degree = linear.y * max_tilt_angle
pitch_degree = linear.x * max_tilt_angle
ver_vel_m_per_s = linear.z * max_vert_speed
rot_vel_deg_per_s = angular.z * max_rot_speed

Donde, el valor del ángulo de $roll$ y $pitch$ depende del valor *PilotingSettingsMaxTiltCurrent*, especificado en grados, mientras que los valores de velocidad lineal y angular depende del *SpeedSettingsMaxVerticalSpeedCurrent* y *SpeedSettingsMaxRotationSpeedCurrent*, especificada en $[m/s]$ y $[^\circ/s]$ respectivamente, cada uno de estos valores máximos es configurable. Una lista detallada de los comandos para envió de comandos y lectura de sensores se encuentra en la pagina web de bebop_autonomy [5]

3.3.3. Gazebo

Gazebo es un simulador de robótica de código abierto que nos permite probar y validar nuestros algoritmos en un entorno virtual realístico 3D para obtener resultados preliminares antes de implementarlos en el mundo real.

Proporciona simulaciones precisas de diversos sensores avanzados incorporados en este simulador, sin embargo también se pueden desarrollar modelos 3D de robots, actuadores, sensores, etc. Gazebo mantiene una API simple para agregar estos objetos y que puedan interactuar con los programas del cliente (Koenig y Howard, 2004). La arquitectura de Gazebo se muestra en la figura 3.6.

[5] https://bebop-autonomy.readthedocs.io/en/latest/index.html

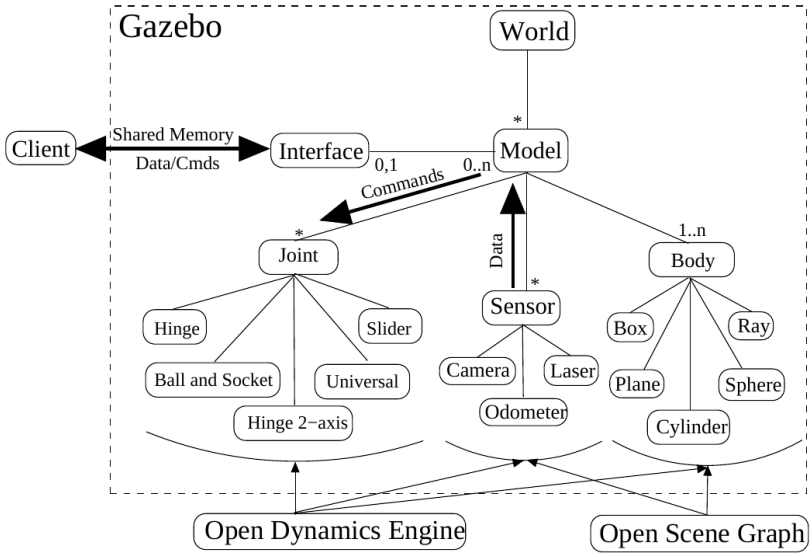


Figura 3.6: Estructura general de los componentes en Gazebo, imagen tomada de (Koenig y Howard, 2004)

Algunos elementos de Gazebo corresponden a:

- **El mundo:** Es el conjunto de todos los elementos estáticos y en movimiento del entorno de simulación que brinda una simulación realista.
- **Modelo del Robot**: Se pueden modelar Robots con gran detalle que incluyen sus estructuras, sensores, actuadores, etc.
- **Sensores**: Simulación precisa de sensores como cámaras, lasers, sonares, para recabar y proporcionar información/ medidas por ejemplo para percepción y mapeo.

3.3.4. Pytorch

Es un marco de aprendizaje profundo de código abierto creado por Facebook y lanzado en 2016, basado en Python que facilita la implementación de *frameworks* de aprendizaje profundo. Emplea una estructura de datos universal llamada "tensor" para gestionar la información a través de la red neuronal. Estos tensores son utilizados para almacenar, representar y transformar datos, constituyendo un elemento fundamental en el desarrollo de redes neuronales en Pytorch (Ketkar y Moolayil, 2021).

El proceso para implementar una red neuronal en este *Framework* podría llevarse a cabo siguiendo los pasos que se enlistan a continuación:

- **Importar bibliotecas:** Se importan las bibliotecas y módulos de programación necesarios para el desarrollo y entrenamiento de la red neuronal.
- **Definir la arquitectura de la red:** Se especifica la estructura de la red neuronal como el numero de capas, su tipo (por ejemplo, capas densas, convolucionales), y las conexiones entre ellas.

- **Configurar hiperparámetros y cargar datos:** Se establece los ajustes que no se aprenden durante el entrenamiento como la tasa de aprendizaje, el numero de épocas. Además se cargan los datos que se utilizarán para entrenar y validar la red.
- **Instanciar la red y definir la función de perdida y el optimizador:** Se crea una instancia (modelo) de la red neuronal con la arquitectura definida anteriormente. También se elige una función de pérdida que cuantifica la discrepancia entre las predicciones y los valores reales. Además, se selecciona un optimizador que ajusta los pesos de la red para minimizar la pérdida durante el entrenamiento.
- **Entrenar la red neuronal:** La red se entrena utilizando los datos cargados. Durante el entrenamiento, los pesos de la red se ajustan iterativamente para reducir la pérdida. Este proceso implica propagar hacia atrás el error (*backpropagation*) y actualizar los pesos utilizando el optimizador elegido. El entrenamiento continúa a lo largo de múltiples épocas hasta que la red alcanza un nivel de rendimiento satisfactorio o converge hacia una solución aceptable.

3.4. Deep Learning

El Aprendizaje Profundo (DL) es una subárea del aprendizaje automático que se fundamenta en el uso de redes neuronales profundas para llevar a cabo tareas más complejas y extraer características más sofisticadas de los datos. Los métodos de DL emplean arquitecturas de redes neuronales con un elevado número de capas y parámetros, por lo tanto, también se les conoce como Redes Neuronales Profundas (DNN) (Shinde y Shah, 2018). Un panorama general de esta clasificación se muestra en la figura 3.10.

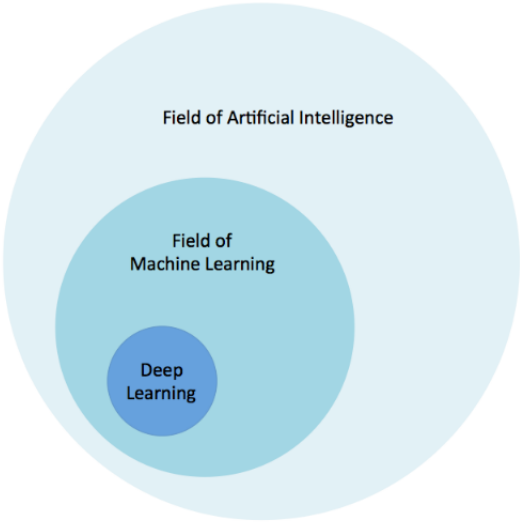


Figura 3.7: Clasificación del Machine Learning, extraída de(Patterson y Gibson, 2017).

Este tipo de redes utiliza operaciones avanzadas en las capas ocultas o múltiples activaciones mas que una simple función de activación, es por esto que DNN permiten ser alimentado con datos crudos de entrada y automáticamente descubre una representación para la correspondiente tarea de aprendizaje (Janiesch et al., 2021).

Tienen características como:

- Emplea una cascada de numerosas capas de unidades de procesamiento no lineal para extraer y transformar características. Cada capa sucesiva utiliza la salida de la capa anterior como entrada. Los algoritmos pueden ser supervisados o no supervisados.
- Están fundamentados en el aprendizaje (no supervisado) de múltiples niveles de características o representaciones de los datos, donde las características de niveles superiores se derivan de aquellas de niveles inferiores, dando lugar a una representación jerárquica.

En una red profunda existen muchas capas entre la entrada y la salida que permite al algoritmo usar múltiples capas de procesamiento compuestas de múltiples transformaciones lineales y no lineales (Ongsulee, 2017).

Finalmente podríamos decir entonces que todo Aprendizaje Profunto es *Machine Learning* pero no todo *Machine Learning* es Aprendizaje Profundo.

3.4.1. MLP

El Perceptrón MultiCapa MLP (por sus siglas en inglés) es un tipo de modelo de Aprendizaje Profundo que consiste en una red neuronal artificial compuesta por al menos tres capas de nodos que son:

- **Capa de entrada:** Se introducen los datos de entrada para ser procesados.
- **Capas ocultas:** Una o mas capas que contienen las neuronas que realizan transformaciones a la entrada.
- **Capa de salida:** Obtiene a su salida el resultado final del modelo, como una clasificación o regresión.

Como se muestra en la figura 3.8. El MLP es una red neuronal *feedforward* en la cual la información fluye unidireccionalmente de la capa de entrada hacia la de salida pasando por las capas ocultas.

En este sistema de neuronas interconectadas, los nodos son conectados por pesos y la señales de salida son una función de las entradas al nodo, modificadas por una función de activación o función de transferencia, esta relación puede expresarse con la ecuación 3.1 para cada neurona.

$$y = f(z), z = \sum_{i=0}^{n} w_i x_i \tag{3.1}$$

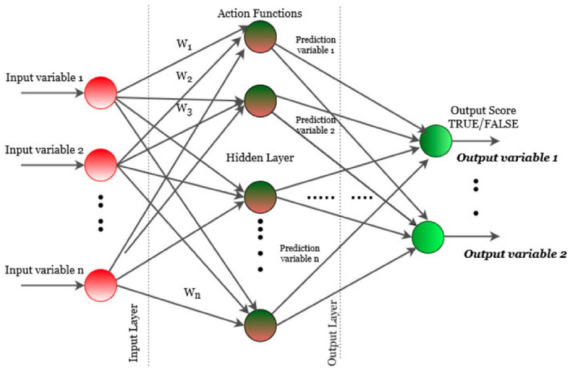


Figura 3.8: Arquitectura del MLP, imagen tomada de (Naskath et al., 2023).

Donde: $x_0 = 1$, w_0 es el *bias* y y la salida de la neurona.

Las funciones de activación determinan la salida de la neurona, introducen no linealidades en el modelo permitiendo así que la red aprenda y represente relaciones complejas en los datos, existe una gran variedad de ellas. Sin embargo, en la Tabla 3.1 solo se muestran las utilizadas en este trabajo.

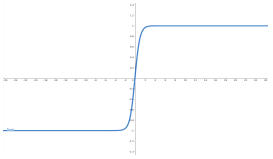
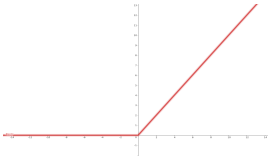
Función de activación	**Ecuación**	**Gráfica 2D**
TanH: Puede ser utilizado en capas ocultas para problemas de clasificación y regresión.	$f(x) = \tanh(x) = \dfrac{(e^x - e^{-x})}{e^x + e^{-x}}$ $f(x) \in (-1, 1)$	
Rectified Linear Unit (ReLU): Actúa como regularizador natural establece a cero todas las entradas negativas.	$f(x) = \text{máx}(0, x)$ $f(x) \in [0, \infty)$	

Tabla 3.1: Ejemplo de funciones de activación utilizadas en Redes Neuronales.

Seleccionando los pesos adecuados y las funciones de transferencia se puede aproximar cualquier función suave y medible entre la entrada y la salida, para su entrenamiento se requiere un conjunto de datos de entrada asociadas a un vector de salida (Gardner y Dorling, 1998).

3.4.2. Entrenamiento

Durante la fase de entrenamiento, la red aprende de manera supervisada, es decir el modelo es alimentado con un conjunto de datos que incluye ejemplos de entradas junto con sus correspondientes salidas deseadas. Durante este proceso la salida actual con la

esperada no son idénticas, es decir existe un error el cual se emplea para ajustar los pesos de la red y reducir así el error del MLP. De esta manera, el modelo adapta sus parámetros internos mediante algoritmos de optimización con el fin de minimizar la discrepancia entre las predicciones y las salidas reales.

Existen diversos métodos como para el entrenamiento de la red, el mas utilizado corresponde al algoritmo de *backpropagation*, donde los pesos pueden ser corregidos propagando los errores capa a capa iniciando desde a capa de salida y propagando hacia la inicial (Taud y Mas, 2018).

El objetivo es encontrar una función desconocida que relacione el vector de entradas en X con un vector de salida Y.

$$Y = f(X) \tag{3.2}$$

Donde $X = [nxk]$, $Y = [nxj]$, n es el número de patrones de entrenamiento, k es el número de nodos de entrada, j es el número de nodos de salida. Durante el entrenamiento la función f es optimizada de tal manera que las salidas de la red de los datos de entrada sean tan parecidas como sea posible a las salidas objetivo, las matrices X y Y representan los datos de entrenamiento, y la función f para una arquitetura de red se determina ajustando los pesos de la red (Gardner y Dorling, 1998).

3.4.3. Hiperparámetros

El desempeño del modelo no solo depende de la elección de las variables, el número de capas ocultas, nodos y datos de entrenamiento sino también de los parámetros de entrenamiento(Taud y Mas, 2018).

Los hiperparámetros de una red neuronal son ajustes que se configuran antes de iniciar el proceso de entrenamiento y que no son aprendidos durante el mismo. Estos parámetros impactan en la estructura general y el comportamiento de la red, y su modificación puede tener un impacto significativo en el rendimiento del modelo. Algunos ejemplos típicos de hiperparámetros en una red neuronal son:

- **Tasa de aprendizaje**: Es la velocidad con la que los pesos de la red se ajustan durante el entrenamiento.
- **Número de capas y unidades por capa**: Define la arquitectura general de la red, especificando cuántas capas tiene y cuántas unidades (neuronas) hay en cada capa.
- **Funciones de activación**: Son funciones matemáticas que determinan la salida de cada neurona.
- **Tamaño del lote (batch size)**: Es el número de ejemplos de entrenamiento utilizados en una iteración del algoritmo de optimización.
- **Épocas**: Representa el número de veces que todo el conjunto de datos de entrenamiento pasa por la red durante el entrenamiento.
- **Optimizador**: Define el algoritmo de optimización utilizado para ajustar los pesos de la red.

3.4.4. YOLOv8

You Only Look Once (YOLO) es un algoritmo de detección de objetos en tiempo real creado por Joseph Redmon y Santosh Divvala en 2016, a lo largo del tiempo se han desarrollado diferentes versiones con mejoras sobre todo en la presición y velocidad llegando actualmente a su versión v8 lanzada por ultralitycs, es una red neuronal convolucional que soporta tareas de visión artificial como la detección, estimación de posición , seguimiento y clasificación mostrados en la figura 3.9, la descripción detallada de cada tarea puede encontrarse en el repositorio de github de ultralitycs [6].

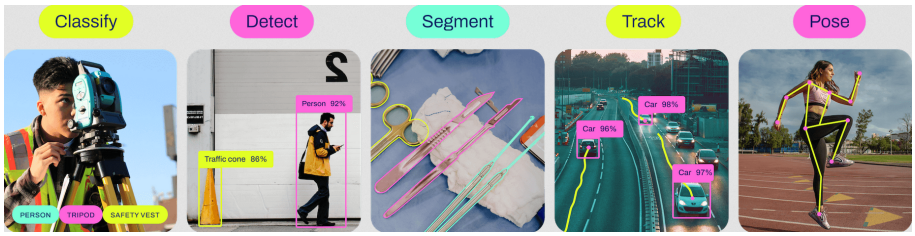


Figura 3.9: Tareas que se pueden realizar en YOLOv8 (imagen tomada del repositorio de github de ultralitycs).

La detección, segmentación y modelos de pose son entrenados utilizando la base de datos de COCO (Lin et al., 2014). Sin embargo, por la aplicación en este trabajo, se aborda únicamente la tarea de detección entrenando a la red con un conjunto de datos personalizado con la finalidad de detectar un objeto sobre una imagen y obtener un *Bounding Box* que es un rectángulo que encierra al objeto de interés para indicar la ubicación y tamaño aproximado del objeto dentro de la imagen. La información de las coordenadas del *Bounding Box* corresponden al borde superior izquierdo (x_1,y_1) y del borde inferior derecho (x_2,y_2) en píxeles de donde es fácil conocer el ancho y alto en píxeles del objeto detectado.

3.5. Control

El control es una disciplina fundamental en ingeniería que se ocupa de la manipulación y regulación de sistemas dinámicos para lograr un comportamiento deseado. En este contexto, se exploran diversas técnicas de control para explorar la precisión y eficiencia en tareas de seguimiento. Entre las estrategias que se abordan se encuentran el control Proporcional-Integral-Derivativo (PID), el Control Predictivo basado en Modelo (MPC) y el control neuronal.

[6] https://github.com/ultralytics/ultralytics

3.5.1. Control PID

Un control PID sin restricciones de la señal de control $u(t)$ se muestra en la ecuación (3.3).

$$u(t) = k_p e(t) + k_i \int_0^{t_h} e(t)dt + k_d \frac{d}{dt} e(t) \tag{3.3}$$

donde $e(t)$ corresponde al error entre la referencia y la salida real de la planta (error de control), k_p, k_i y k_d son las ganancias proporcional, integral y derivativa, respectivamente, mientras que t_h se refiere al horizonte de control. Para el control del UAV es necesario utilizar la forma discreta del control PID descrita por la ecuación (3.4).

$$u_n = k_p.e_n + k_I.\sum_{m=0}^{N_h} e_m + k_D.(e_n - e_{n-1}) \tag{3.4}$$

La señal de control u_n corresponde a la señal de control en el n-$ésimo$ paso de tiempo, N_h es el horizonte de control, e_n y e_{n-1} es el error de control en el instante actual y anterior (Giernacki et al., 2020). Debido a las características del control interno del UAV elegido (Bebop 2.0), esta señal debe estar en un rango de $[-1, 1]$.

3.5.2. MPC

El termino Control Predictivo basado en el Modelo (MPC) no se refiere a una estrategia de control especifica, pero si aun amplio rango de métodos que utilizan directamente el modelo dinámico del proceso a fin de predecir el comportamiento futuro de la salida en instantes de tiempo futuro (horizonte de predicción), obteniendo las señales de control que minimizan una función costo a lo largo de dicho horizonte (Camacho et al., 2007).

En la figura 3.10 se muestra la idea general del MPC. Donde, en cada instante consecutivo de muestreo k se calcula un conjunto de incrementos de control futuros, es decir:

$$\triangle u(k) = \begin{bmatrix} \triangle u(k|k) \\ \vdots \\ \triangle u(k + N_u - 1|k) \end{bmatrix}$$

Se asume que $\triangle u(k + N_u - 1|k) = 0$ para $p \geq N_u$, donde N_u es el horizonte de control.

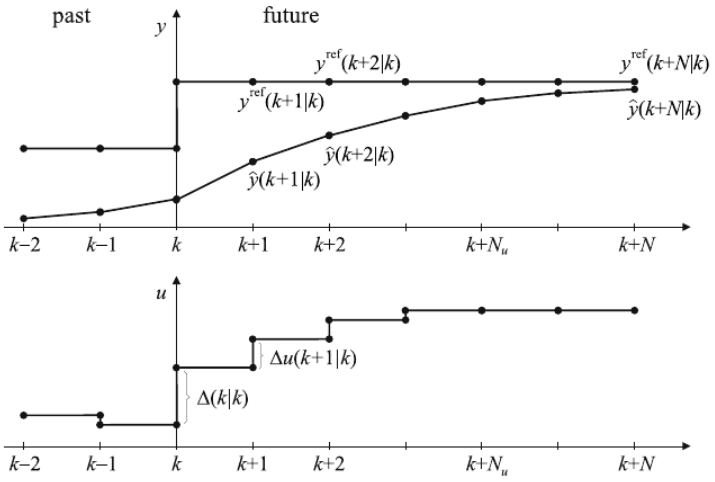


Figura 3.10: Principio del control predictivo tomada de (Szczerbicki, 2009).

De acuerdo con (Okasha et al., 2022), el MPC se puede realizar siguiendo los siguientes pasos:

1) Construcción del modelo discreto en espacio de estados.
2) Sobre un horizonte de predicción predefinido, se predicen los estados futuros del sistema y sus salidas para una señal de control dada para todo el horizonte de predicción.
3) A través de un proceso de optimización se selecciona la secuencia de control que minimiza la función costo en lazo cerrado.
4) Solo la primera señal de control del horizonte se aplica a la planta.
5) El método se repite en un ciclo secuencial desde el paso 2.

Resulta entonces necesario definir el modelo del proceso y la formulación del problema de optimización.

Modelo del proceso

A través del modelo dinámico de la planta se conoce el comportamiento de los estados del UAV como respuesta de una señal de entrada de control dada, este modelo puede obtenerse a través de distintos métodos teóricos como Euler-Lagrange o Newton, o realizando una identificación del sistema. Entre mas preciso sea el modelo al comportamiento de la planta física típicamente implica que la complejidad aumente reflejada en un mayor número de estados y aun mas importante, no linealidades.

Formulación del problema de optimización

Se establece un proceso de optimización basada en una estrategia de control, donde el problema de optimización cuenta con una función objetivo y restricciones basadas en el conocimiento del modelo de proceso.

Función objetivo: La función objetivo definida debe tener como finalidad, general una señal de control de entrada óptima de tal manera que al ser minimizada nuestra meta sea cumplida. De forma general el objetivo principal de la salida $\hat{y}$ en el horizonte seleccionado es que siga una determinada referencia y^{ref} y al mismo tiempo el esfuerzo de control $\triangle u$ necesario para lograr el seguimiento sea penalizado. Una ventaja de los algoritmos de MPC es que pueden ser usados para procesos multivariables, es decir si el proceso tiene n_u entradas y n_y salidas la función objetivo puede ser expresada conforme a la ecuación 3.5.

$$J(k) = \sum_{p=1}^{N} \|y^{ref}(k+p|k) - \hat{y}(k+p|k)\|_Q^2 + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|_R^2 \tag{3.5}$$

Donde $y^{ref}(k+p|k)$ es la trayectoria de referencia y $\hat{y}(k+p|k)$ son las predicciones de la salida sobre el horizonte de predicción $N \geq N_u$.

$$\|M\|_A^2 = M^T AM, \quad Q \geq 0 \quad y \quad R \geq 0$$

Q y R son matrices de pesos de dimensión n_y x n_y y n_u x n_u, ecuación 3.6 y 3.7.

$$Q = diag\,(Q_1, Q_2, Q_3, ...Q_{ny}) \tag{3.6}$$
$$R = diag\,(R_1, R_2, R_3, ...R_{nu}) \tag{3.7}$$

En la ecuación 3.5, el termino de penalización de las señales de control se ve reflejado como una reducción en movimientos innecesarios y minimizar el consumo de energía, dando como resultado un equilibrio entre el consumo de energía y la agresividad de seguimiento de la trayectoria de referencia (Almozel, 2020).

Restricciones: De acuerdo con (Almozel, 2020), podemos identificar restricciones en los actuadores, y en los estados principalmente las cuales se detallan a continuación:

- **Restricciones en los actuadores**: Las señales de control están sujetas a restricciones físicas, en el caso de un cuadricóptero por ejemplo los actuadores tienen una velocidad máxima de rotación y por lo tanto su empuje es limitado, estas restricciones físicas deben ser contempladas por el diseñador para garantizar su funcionamiento en un cierto dominio de operabilidad, las restricciones también pueden expresar como la entrada de control en si misma no pueda cambiar de un valor a otro muy distinto en pasos de tiempo sucesivos.
- **Restricciones en los estados**: El estado del cuadricóptero también está limitado dentro de un rango especificado. Esto puede deberse tanto a que el modelo en sí no modela las dinámicas más allá de ese rango o como medidas de seguridad de acuerdo al diseñador.

Finalmente el problema de optimización se plantea de la siguiente manera:

$$\min_{\triangle u(k|k)...\triangle u(k+N_u-1|k)} J(k) \tag{3.8}$$

Sujeta a las siguientes restricciones:

$$\begin{aligned} u^{min} \leq u(k+p|k) \leq u^{max}\ , p = 0, ..., N_u - 1 \\ -\triangle u^{max} \leq \triangle u(k+p|k) \leq \triangle u^{max}\ , p = 0, ..., N_u - 1 \\ y^{min} \leq \hat{y}(k+p|k) \leq y^{max}\ , p = 1, ..., N \end{aligned}$$

Donde: u_{min}, u_{max}, $\triangle u^{max}$, y^{min} y y^{max} son vectores.

La figura 3.11 muestra el proceso general del MPC con cada una de sus componentes.

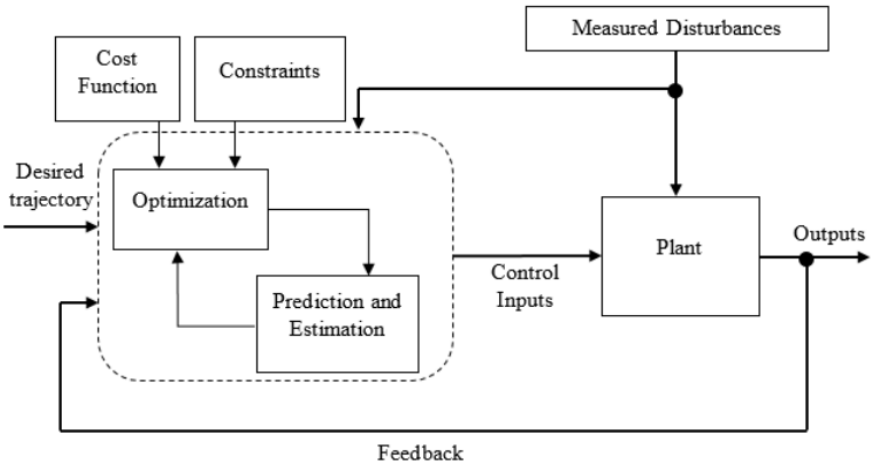


Figura 3.11: Diagrama de bloques del funcionamiento del MPC tomada de (Okasha et al., 2022).

3.5.3. Control Neuronal

"Control Neuronal se refiere a una metodología en la cual el controlador por si mismo es una red neuronal, y a una metodología en la cual los controladores son diseñados basados en un modelo de red neuronal de la planta." (Nguyen et al., 2002). De acuerdo con la definición anterior, los métodos en los que el controlador es una red neuronal se llaman métodos directos, mientras en el caso de métodos indirectos se refiere a que su diseño esta basado en un modelo de red neuronal de la planta que se quiere controlar, es decir que el controlador no es una red neuronal pero se deriva de una planta que se modela a través de una red neuronal.

Exactamente, los métodos indirectos se refieren a técnicas de control en las cuales el diseño del controlador se basa en un modelo de red neuronal de la planta que se quiere controlar. En estos métodos, la red neuronal se utiliza para modelar la dinámica y el comportamiento de la planta en lugar de formar parte directa del controlador.

La figura 3.12 muestra un ejemplo de control Neuronal utilizando un MLP predictivo para la navegación de un robot móvil donde la red aprende de las acciones de control pasadas, referencias futuras e información de los sensores obteniendo a la salida las señales de control correspondientes para alcanzar las futuras referencias (Camacho y Bordons, 2007).

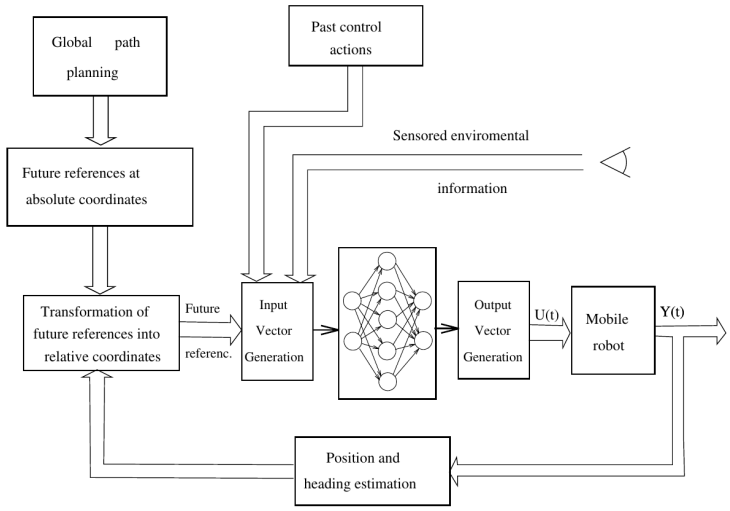


Figura 3.12: Esquema de Red Neuronal Predictiva para la navegación de un robot móvil, imagen tomada de (Camacho y Bordons, 2007).

3.6. Filtro de Kalman

El filtro de Kalman es un algoritmo de estimación utilizado para obtener estimaciones precisas del estado de un sistema dinámico a partir de mediciones a lo largo del tiempo que pueden o no incluir ruido estático (Kim y Bang, 2018). Originalmente formulado en el dominio continuo, el filtro de Kalman continuo es eficaz para sistemas descritos mediante ecuaciones diferenciales. Sin embargo, en la práctica, muchos sistemas son muestreados a intervalos discretos. La transición del filtro de Kalman continuo al filtro de Kalman discreto se vuelve esencial en estos casos.

3.6.1. El filtro discreto de Kalman

Es utilizado para estimar estados de sistemas lineales en el formato de espacio de estados de un sistema discreto en el tiempo donde las mediciones y estimación de estados se realiza en puntos discretos de tiempo (ecuación 3.9 y 3.10).

Ecuación de estado:

$$\mathbf{x}_k = \mathbf{F}x_{k-1} + \mathbf{B}u_{k-1} + \mathbf{w}_{k-1} \tag{3.9}$$

Ecuación de salida:

$$\mathbf{z}_k = \mathbf{H}x_k + \mathbf{v}_k \tag{3.10}$$

Donde:

- $\mathbf{F}$ es la matriz de transición de estados que relaciona el estado en el paso de tiempo anterior $k-1$ con el estado en el paso de tiempo actual k.
- $\mathbf{B}$ es la matriz de control que relaciona la señal de control con el estado x.
- k el el indice de tiempo.
- $\mathbf{x}$ es el vector de estados.
- u es el vector de entradas del sistema.
- $\mathbf{z}$ es el vector de mediciones.
- $\mathbf{H}$ es la matriz de mediciones que relaciona el estado con las mediciones z_k.
- $\mathbf{w}$ es llamado el ruido del proceso.
- $\mathbf{v}$ es el ruido en la medición.

El vector x contiene la información de los estados del sistema, sin embargo, medir directamente cada uno de estos estados resulta complicado, en su lugar es mas fácil medir la salida del sistema, que esta relacionada con x pero afectada por ruido. Teniendo esto en mente el empleo del algoritmo del filtro de Kalman estima los estados actuales del vector de estados utilizando las mediciones actuales asumiendo que el ruido del proceso y de medición tienen una distribución gaussiana con media cero y covarianza Q y R respectivamente y que además no existe correlación entre ellas (Simon, 2001).

$$p(w) \sim N(0, Q).$$
$$p(v) \sim N(0, R).$$

Durante este proceso, el filtro calcula la estimación del estado del sistema en un momento dado y recibe información de retroalimentación en forma de mediciones (que pueden contener ruido), por ende, las ecuaciones del filtro de Kalman se dividen en dos conjuntos.
Ecuaciones de predicción:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}\mathbf{x}_{k-1|k-1} + \mathbf{B}\mathbf{u}_k \tag{3.11}$$

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1|k-1}\mathbf{F}^{\mathsf{T}} + \mathbf{Q} \tag{3.12}$$

Ecuaciones de actualización:

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1}\mathbf{H}_k^{\mathsf{T}}(\mathbf{H}\hat{\mathbf{P}}_{k|k-1}\mathbf{H}^{\mathsf{T}} + \mathbf{R})^{-1} \tag{3.13}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}) \tag{3.14}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\,\hat{\mathbf{P}}_{k|k-1} \tag{3.15}$$

Las ecuaciones de predicción tienen la tarea de proyectar hacia adelante (en el tiempo) las estimaciones actuales del estado y la covarianza del error, generando así las estimaciones a priori para el próximo paso temporal. Por otro lado, las ecuaciones de corrección de medición se encargan de la retroalimentación, es decir, de incorporar una nueva medición en la estimación a priori para producir una estimación a posteriori mejorada. Este proceso iterativo de predicción y corrección se repite continuamente en un proceso cíclico para mejorar la precisión de la estimación del estado del sistema a lo largo del tiempo como se muestra en la figura 3.13 (Welch, Bishop et al., 1995) por eso se dice que estima un proceso utilizando una forma de control realimentado.

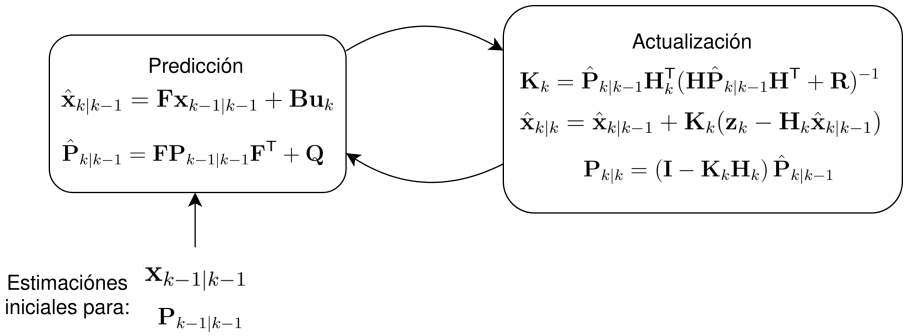


Figura 3.13: Diagrama de operación del filtro de Kalman.

Donde las matrices Q Y R son usualmente usadas como sintonización de parámetros para ajustar el desempeño deseado (Welch, Bishop et al., 1995).

Capítulo 4

Metodología

En este capítulo se detalla la metodología empleada para el seguimiento de objetos móviles a través de un UAV utilizando aprendizaje profundo. La figura 4.1 proporciona un esquema general de las diversas tareas realizadas en esta etapa. Destacan la percepción del objeto, la estimación de posición, la creación de controles PID y MPC, el diseño del filtro de Kalman para la estimación de estados, la arquitectura de la RN para el control del UAV, y finalmente, la implementación de cada uno de estos módulos.

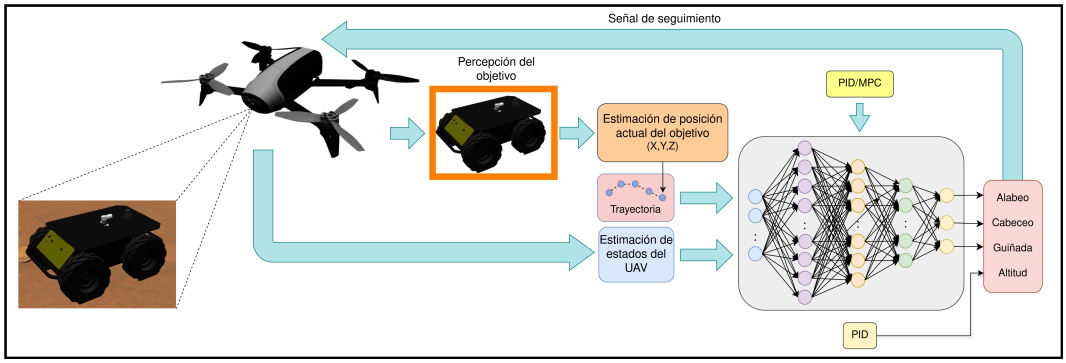


Figura 4.1: Metodología general empleada para el seguimiento de objetos móviles a través de un UAV utilizando DNN.

4.1. Percepción del Objetivo

Se utiliza YOLOv8 para la detección del objetivo a través de la cámara monocular instalada en el UAV. En este estudio, se designó al robot Husky como el objetivo a seguir, al cual nos referiremos de ahora en adelante como “Objetivo”. En primer lugar se generó un conjunto de imágenes capturadas por la cámara del UAV en Gazebo, las cuales fueron obtenidas mediante vuelos manuales del UAV a alturas de 3, 4 y 5 metros sobre la superficie, garantizando que el “Objetivo” se encontrara en distintas partes dentro del campo de visión de la cámara del UAV.

En total, se capturaron 750 imágenes con una resolución de 640x480 píxeles, mostrando el “Objetivo” desde diversas perspectivas, alturas, ángulos y escalas. Para el proceso de entrenamiento, se utilizó el 80 % de estas imágenes, reservando el 20 % restante para la validación. El etiquetado de ambos conjuntos se llevó a cabo a través del sitio web [1].

El entrenamiento de YOLOv8 se realiza utilizando el modelo yolov8n.pt, donde definimos las clases “Fondo” y “Objetivo”. La configuración del proceso de entrenamiento consiste en un tamaño de lote de 8, una tasa de aprendizaje de 0.001 y un total de 100 épocas. Aunque el resto de la configuración se dejó en sus valores por defecto, se puede encontrar más detalle en la página oficial de configuración de YOLOv8 en Ultralytics [2]. Finalmente, obtuvimos el modelo entrenado, el cual nos permite detectar las clases “Fondo” y “Objetivo”.

4.2. Definición del sistema de coordenadas

A continuación, se establece el sistema de coordenadas empleado a lo largo de este trabajo para el desarrollo de los módulos de estimación de posición, identificación de la planta, filtro de Kalman y control. Se definen dos sistemas de coordenadas: el sistema global, denotado como “G”, y el sistema local, denotado como “B”, ambos siguiendo la convención de la mano derecha (consulte la figura 4.2).

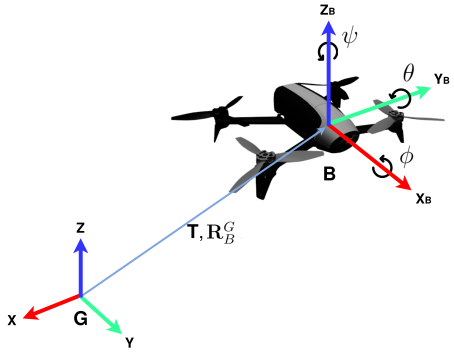


Figura 4.2: Sistema de coordenadas global “G” y local “B” utilizado en el desarrollo de este trabajo.

En el sistema local “B”, situado en el UAV, el eje X_B, Y_B, Z_B indican la dirección hacia adelante, izquierda y hacia arriba del UAV. En la figura también se muestran los ángulos de Euler de alabeo (ϕ), cabeceo (θ) y guiñada (ψ) alrededor de los ejes X_B, Y_B y Z_B, respectivamente. T corresponde al vector de traslación con respecto a “G”, y R_B^G es la matriz de rotación que transforma un vector definido en “B” a un vector con respecto a “G”.

[1] https://www.makesense.ai/

[2] https://docs.ultralytics.com/usage/cfg/

4.3. Estimación de posición

Tras la detección del "Objetivo" con YOLOv8, obtenemos un par de coordenadas en píxeles para el objeto detectado. Estas coordenadas representan la posición de dos esquinas opuestas de un rectángulo que rodea al objeto detectado, como se muestra en la figura 4.3. Utilizamos estas coordenadas para identificar el píxel central del "Objetivo", una parte crucial del proceso, ya que se utiliza para estimar su posición relativa al UAV. Posteriormente, esta posición es referida al sistema de coordenadas global.

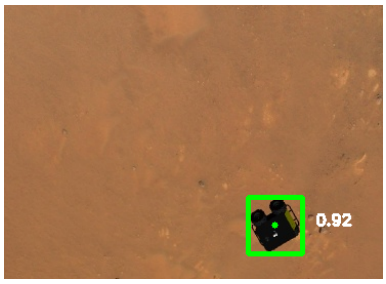


Figura 4.3: Detección del "Objetivo" con YOLOv8 donde se muestra el *bounding box*, el pixel central y la confianza de detección.

De acuerdo con (Martinez-Carranza y Rojas-Perez, 2022), al conocer la posición tridimensional del UAV, su orientación y asumiendo que la superficie es completamente horizontal, es posible calcular la profundidad para cada píxel en una imagen cromática de una cámara. En nuestro contexto, nos enfocamos exclusivamente en determinar el vector de profundidad para el píxel localizado en el centro del "Objetivo". La figura 4.4 ilustra un diagrama geométrico para el cálculo de estimación de posición.

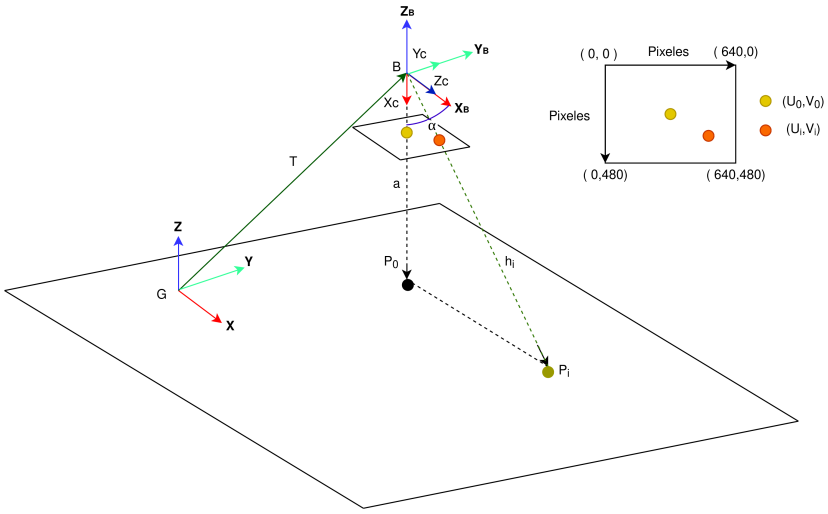


Figura 4.4: Diagrama geométrico para calcular la estimación de posición del "Objetivo" detectado desde la cámara a bordo del UAV.

Donde:

- (X, Y, Z) Representa el sistema de coordenadas global.
- (X_B, Y_B, Z_B) Representa el sistema de coordenadas del UAV.
- (X_C, Y_C, Z_C) Representa el sistema de coordenadas de la cámara del UAV.
- α es el ángulo comprendido entre el eje X_{B} y el eje X_C, medido con respecto al eje X_{B}.

Para obtener la estimación de posición, ubicamos el sistema de coordenadas de la cámara (X_c, Y_c, Z_c) en su centro de proyección. En este trabajo, empleamos la cámara inferior del UAV Bebop 2.0 en Gazebo, donde fijamos α en $90°$. Inicialmente, trazamos un vector $\mathbf{n}_p$ que es paralelo al vector de gravedad y perpendicular al plano de nuestra superficie de exploración (plano XY). La magnitud de este vector, según la configuración geométrica utilizada, corresponde a la altura del UAV, como se muestra en la ecuación 4.1.

$$\mathbf{n}_p = \begin{bmatrix} a & 0 & 0 \end{bmatrix}^T \tag{4.1}$$

Este vector, a su vez representa un punto P_0 en el plano de la superficie de exploración, por lo tanto se puede calcular la intersección de un rayo $\mathbf{h}_i$ (ecuación 4.2) con el plano XY, este rayo parte del centro óptico de proyección de la cámara e intercepta el plano de la imagen de la cámara en las coordenadas de algún pixel denotadas como (ui, vi). La magnitud de el vector $\mathbf{h}_i$ esta en función de algunos parámetros intrínsecos de la cámara como la distancia focal (f_u, f_v) y el centro óptico (u_0, v_0) (tabla 4.1), los cuales se obtienen tras un proceso de calibración.

Parámetros	$\mathbf{f}_u$	$\mathbf{f}_v$	$\mathbf{C}_x$	$\mathbf{C}_y$
Magnitud	198.160589	198.172204	319.179392	239.497296

Tabla 4.1: Parámetros intrínsecos de la cámara inferior del Bebop 2.0 en Gazebo.

$$\mathbf{h}_i = \begin{bmatrix} 1 \\ (u_o - u_i)/f_u \\ (v_0 - v_i)/f_v \end{bmatrix} \tag{4.2}$$

Finalmente aplicando la ecuación vectorial de intersección de una línea con un plano, es posible obtener la estimación de posición $\mathbf{p}_i$ del "Objetivo" (intersección del rayo $\mathbf{h}_i$ con el plano de exploración XY) ecuación 4.3.

$$\mathbf{p}_i = \begin{pmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{pmatrix} = \left(\frac{\mathbf{p}_0^T \mathbf{n}_p}{\mathbf{h}^T \mathbf{n}_p} \right) \mathbf{h}_i \tag{4.3}$$

Donde $\mathbf{p}_i$ representa la estimación de la posición del "Objetivo" con respecto al sistema de coordenada de la cámara con origen en el centro óptico de la cámara del UAV, que asumimos coincide con el centro de masa del UAV. Esta estimación se transforma al sistema de coordenadas global a través de la ecuación 4.4.

$$\mathbf{p}_G = R_{uav}(q)[R_{cam}(\alpha)p_i] + T_{uav} \tag{4.4}$$

Donde T_{uav} representa la posición del UAV con respecto al sistema de coordenadas global, $R_{\text{cam}}(\alpha)$ es la matriz de rotación del sistema de coordenadas de la cámara sobre el eje Y_{uav} con un ángulo α, y $R_{\text{uav}}(q)$ es la matriz de rotación del sistema de coordenadas del UAV dado el cuaternión q con respecto al sistema de coordenadas global.

Por lo tanto, $[R_{\text{cam}}(\alpha)\mathbf{p}_i]$ representa la proyección de la estimación de posición $\mathbf{p}_i$ desde el sistema de la cámara (X_c, Y_c, Z_c) al sistema de coordenadas del UAV (X_B, Y_B, Z_B). En consecuencia, $R_{\text{uav}}(q)[R_{\text{cam}}(\alpha)\mathbf{p}_i]$ representa la proyección de la posición del "Objetivo" ahora en el sistema de coordenadas del UAV con la misma orientación que el sistema global. Finalmente, tras sumarle el vector Tuav, se obtiene el vector $\mathbf{p}_G$. Este último representa la posición del "objetivo" ahora en coordenadas del sistema global (X, Y, Z).

4.4. Identificación de la planta

Implementar un Control Predictivo Basado en el modelo (MPC) implica comprender el comportamiento dinámico de la planta a través de un modelo dinámico. Este modelo puede ser obtenido teóricamente, como los utilizados en (Almozel, 2020; Matus-Vargas et al., 2017), o identificado experimentalmente, como se describe en (Sa et al., 2017).

La planta Parrot Bebop 2.0 presenta una arquitectura cerrada que dificulta el acceso directo al control individual de sus hélices, cuenta con un controlador interno que recibe comandos específicos para el control de actitud y altitud del UAV. Esto complica la integración directa de señales de control derivadas de modelos dinámicos obtenidos a través de Euler-Lagrange o Newton. Sin embargo, utilizar un modelo dinámico de la planta obtenido a través de una aproximación numérica facilita la vinculación de las señales de control derivadas de este modelo con las señales necesarias para el controlador interno, lo que resulta en la integración efectiva de comandos de control.

Por tanto, se realiza la identificación de la planta utilizando el método descrito en (Sa et al., 2017), donde, a través de una aproximación dinámica de actitud y altitud de primer orden logran realizar el control de posición a través de un controlador de bajo nivel para seguir las referencias de movimiento horizontal y vertical. Esta aproximación proporciona la información necesaria al MPC para realizar el control de posición. Nuestra aproximación modela la dinámica del UAV a través de un modelo dinámico lineal, considerando que el vehículo se mantiene alineado constantemente con el sistema de coordenadas global, es decir, $\psi = 0°$. Además, en condiciones cercanas de vuelo

estacionario, en el cual asumimos ángulos pequeños de alabeo y cabeceo (Kamel et al., 2017).

Utilizando el *ToolBox* de sistemas de control de MATLAB®, obtenemos la función de transferencia de primer orden para modelar el movimiento traslacional en los ejes X, Y y Z (véase ecuaciones 4.5 a 4.7).

$$H_{vx}(s) = \frac{v_x(s)}{u_{vx}(s)} = \frac{K_{vx}}{\tau_{vx}s + 1} \tag{4.5}$$

$$H_{vy}(s) = \frac{v_y(s)}{u_{vy}(s)} = \frac{K_{vy}}{\tau_{vy}s + 1} \tag{4.6}$$

$$H_{vz}(s) = \frac{v_z(s)}{u_{vz}(s)} = \frac{K_{vz}}{\tau_{vz}s + 1} \tag{4.7}$$

Este proceso implica recopilar un conjunto de datos de entrada de la planta que corresponden a los comandos de control de velocidad lineal del UAV (u_{vx}, u_{vy}, u_{vz}), así como las salidas (v_x, v_y, v_z), que representan las velocidades lineales, estos datos son recopilados a través de los experimentos mostrados en la sección 5.3.

Posteriormente, estos datos fueron procesados en MATLAB®, utilizando la herramienta de identificación de sistemas para obtener las funciones de transferencia de primer orden correspondientes a cada uno de los ejes X, Y, y Z. Después de aplicar la transformada inversa de Laplace y realizar los despejes necesarios, se obtuvo el siguiente sistema de ecuaciones diferenciales que modelan la dinámica del UAV, utilizando una aproximación lineal del sistema.

$$\begin{aligned}
\dot{v_x}(t) &= \frac{K_{vx}}{\tau_{vx}} u_{vx}(t) - \frac{v_x(t)}{\tau_{vx}} \\
\dot{v_y}(t) &= \frac{K_{vy}}{\tau_{vy}} u_{vy}(t) - \frac{v_y(t)}{\tau_{vy}} \\
\dot{v_z}(t) &= \frac{K_{vz}}{\tau_{vz}} u_{vz}(t) - \frac{v_z(t)}{\tau_{vz}}
\end{aligned}$$

Definiendo los estados $\dot{s} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$ y los controles $u = [u_{vx}, u_{vy}, u_{vz}]^T$ se obtiene su representación en variables de estado expresada en forma matricial $\dot{s}\,(t) = As\,(t) + Bu\,(t)$ como se muestra en la ecuación 4.8:

$$\dot{s}\,(t) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\frac{1}{\tau_{vx}} & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\tau_{vy}} & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_{vz}} \end{bmatrix} s\,(t) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{K_{vx}}{\tau_{vx}} & 0 & 0 \\ 0 & \frac{K_{vy}}{\tau_{vy}} & 0 \\ 0 & 0 & \frac{K_{vz}}{\tau_{vz}} \end{bmatrix} u\,(t) \tag{4.8}$$

4.5. Filtro de Kalman

El Bebop 2.0 en Gazebo está equipado con un sensor de odometría que ofrece información sobre la velocidad lineal en los ejes X, Y, Z y la velocidad angular alrededor de estos ejes. Además, proporciona datos sobre la posición (x, y, z) y los ángulos de inclinación (ϕ, θ, ψ). Esta información resulta esencial para el control de posición del UAV. Sin embargo, la plataforma física carece de sensores con precisión similares, lo que hace necesario implementar un estimador de estados. Para lograr esto, se ha diseñado un filtro de Kalman con el objetivo de estimar los estados de velocidad lineal en X, Y, Z (necesarios para implementar el MPC), utilizando la información de posición x, y, z del UAV.

De acuerdo con la ecuación 3.9, para implementar el filtro de Kalman, es necesario obtener una versión discreta del modelo del sistema, en nuestro caso a 20 Hz. Para lograrlo, se realiza la discretización de las matrices A y B de la ecuación 4.8 utilizando la función `c2d` de MATLAB.

Una vez que el sistema ha sido discretizado, se procede a implementar el algoritmo del filtro discreto de Kalman. En este proceso, se busca sintonizar las matrices P_{kalman}, Q_{kalman} y R_{kalman} (ecuación 4.9 a 4.11) con el objetivo de estimar los estados de velocidad lineal utilizando mediciones de posición como entrada al filtro. Para lograr esto, se emplean los datos de posición y velocidad utilizados en el proceso de identificación de la planta (sección 4.4). Además, la sintonización de estas matrices tiene como meta obtener un RMSE aceptable entre las predicciones de velocidad proporcionadas por el filtro y los valores de velocidad medidos.

$$P_{\text{kalman}} = \text{diag}\left(P_{kx}, P_{ky}, P_{kz}, P_{kvx}, P_{kvy}, P_{kvz}\right) \tag{4.9}$$

$$Q_{\text{kalman}} = \text{diag}\left(Q_{kx}, Q_{ky}, Q_{kz}, Q_{kvx}, Q_{kvy}, Q_{kvz}\right) \tag{4.10}$$

$$R_{\text{kalman}} = \text{diag}\left(R_{kx}, R_{ky}, R_{kz}\right) \tag{4.11}$$

Después de sintonizar las matrices Q y R que nos garantizan la precisión buscada del filtro de Kalman se implemento el algoritmo en ROS y Gazebo para estimar la velocidad del Bebop 2.0 utilizando unicamente lecturas de posición del simulador.

4.6. Diseño de Controles

Debido a la aplicación de la tarea de seguimiento para robots de exploración planetaria y, en un futuro, como asistencia a astronautas, se plantean las siguientes características de diseño:

1) El control debe ser capaz de realizar el seguimiento del objetivo cuando este se mueve entre 0.1 m/s y 1.5 m/s.
2) El controlador debe ser capaz de calcular las señales de control a una frecuencia mínima de 20 Hz.

3) El error de posición en estado estacionario entre el UAV y el objetivo debe ser inferior a 5 cm. En este contexto, el estado estacionario se refiere al momento en que el "Objetivo" se detiene durante al menos 2 s.
4) Durante el seguimiento el UAV debe mantenerse a no mas de 2.5 m del objetivo, esto se cuantificara a través del TSR.
5) El sistema de control debe tener la capacidad de seguir trayectorias con transiciones entre curvas y tramos rectos, incluyendo características como curvas de 90°, cerradas semicirculares, en forma de "S", suaves y en forma de "U". Además, debe ser capaz de realizar pausas en puntos específicos.

Una vez definidas las características de diseño se procede al diseño de los controladores.

4.6.1. Control PID

Se implementa un control de seguimiento de trayectoria el cual genera las señales de control de movimiento del UAV, con el fin de seguir una trayectoria en $\mathbb{R}^3$. Para ello se emplea un control PID utilizando el modelo cinemático simplificado de un UAV utilizado en (Salinas et al., 2014). En este modelo se asume que el movimiento de alabeo y cabeceo es casi nulo (ecuación 4.12).

$$V = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} cos(\psi_B) & -sin(\psi_B) & 0 & 0 \\ sin(\psi_B) & cos(\psi_B) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_{Bx} \\ v_{By} \\ v_{Bz} \\ \omega_{Bz} \end{bmatrix} = Jv. \tag{4.12}$$

Donde J es la matriz Jacobiana que relaciona velocidades lineales y angulares $(v_{Bx}, v_{By}, v_{Bz}, \omega_{Bz})^T$ en un sistema de coordenadas fijo al UAV B con aquellas $(\dot{x}, \dot{y}, \dot{z}, \dot{\psi})^T$ referidas al sistema de coordenadas global G, con ψ_B como el ángulo de Guiñada del UAV.

El Bebop 2.0 se controla a través de comandos de velocidad lineal y angular los cuales denotaremos como $u_c = [u_{vx}, u_{vy}, u_{vz}, u_{wz}]$. A partir de la ecuación 4.12, buscamos obtener las velocidades representadas como v de tal manera que $u_c = v$. El $det[J] \neq 0$ por lo tanto existe su inversa J^{-1}, esto nos conduce a la siguiente expresión:

$$u_c = J^{-1}V \tag{4.13}$$

Considerando que se busca regular la posición del UAV, se propone el siguiente control:

$$u = K_P e_{pos} + K_I \int e_{pos} + K_D \dot{e}_{pos} \tag{4.14}$$

Donde $K_P, K_I, K_D \, \epsilon \mathbb{R}^{mxm}$ son matrices diagonales de constantes proporcionales, integrales y derivativas respectivamente, que limitan el valor máximo de la acción de control para cada estado del UAV, $e_{pos} \, \epsilon \mathbb{R}^{mx1}$ es el vector de errores de posición y orientación del UAV, y m es el número de entradas de control del UAV, es decir:

$$
\begin{aligned}
K_P &= \text{diag}(k_{px}, k_{py}, k_{pz}, k_{p\psi}) \\
K_I &= \text{diag}(k_{ix}, k_{iy}, k_{iz}, k_{i\psi}) \\
K_D &= \text{diag}(k_{dx}, k_{dy}, k_{dz}, k_{d\psi}) \\
e_{\text{pos}} &= (e_x, e_y, e_z, e_\psi)^T
\end{aligned}
$$

Las matrices K_P, K_I, K_D se sintonizaron de partiendo del método de sintonización de Ziegler Nichols conocido también como el método de las oscilaciones criticas.

El vector e_{pos} se mide con respecto al sistema de coordenadas global G. Por lo tanto, el control u corresponde a las velocidades lineales y angulares del UAV con respecto al sistema de coordenadas global, es decir $u = V$. Sustituyendo la expresión 4.14 en 4.13 obtenemos la señal de control deseada u_c:

$$
u_c = J^{-1}[K_P e_{pos}] + J^{-1}[K_I \int e_{pos}] + J^{-1}[K_D \dot{e}_{pos}] \tag{4.15}
$$

Este controlador fue implementado de forma discreta a una frecuencia de 20 Hz[3]. Un diagrama general de como se implementa el control PID con el UAV se muestra en la figura 4.5, donde $s = (x, y, z, \psi)^T$.

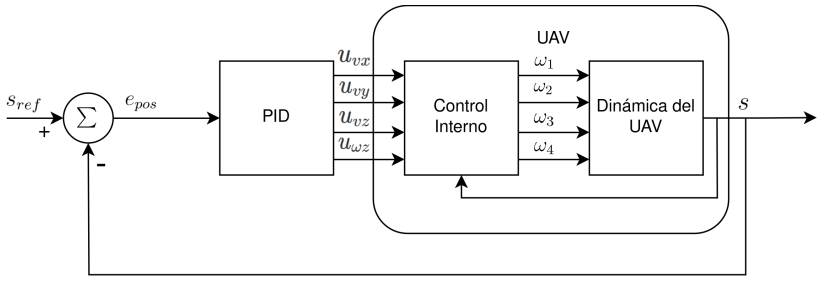


Figura 4.5: Esquema de control PID implementado para el Bebop 2.0.

4.6.2. MPC

El algoritmo de MPC descrito en la sección 3.5.2, genera señales óptimas a lo largo de un horizonte de predicción con la finalidad de realizar una tarea de seguimiento de trayectoria. De acuerdo con (Almozel, 2020), la función costo en este caso se define con la finalidad de minimizar la distancia entre el UAV y la referencia de trayectoria

[3]Este mismo controlador se utilizo para el control de Altitud y Guiñada en la sección controles neuronales.

en cada paso de tiempo, y además, eliminar movimientos innecesarios minimizando el consumo de energía, todo esto a través de una penalización cuadrática a la distancia y entradas de control, el problema de optimización entonces se formula de la siguiente manera (ecuación 4.16).

$$\min_{\triangle u(k|k)...\triangle u(k+N_u-1|k)} \sum_{p=1}^{N} \|y^{ref}(k+p|k) - \hat{y}(k+p|k)\|^2_{Q_{MPC}} + \sum_{p=0}^{N_u-1} \|\triangle u(k+p|k)\|^2_{R_{MPC}} \tag{4.16}$$

Sujeta a las siguientes restricciones:

$$\begin{aligned} u^{min} \leq u(k+p|k) \leq u^{max} \text{ , } p = 0, ..., N_u - 1 \\ -\triangle u^{max} \leq \triangle u(k+p|k) \leq \triangle u^{max} \text{ , } p = 0, ..., N_u - 1 \\ y^{min} \leq \hat{y}(k+p|k) \leq y^{max} \text{ , } p = 1, ..., N \end{aligned}$$

Donde N_u es el horizonte de control, $y^{ref}(k+p|k)$ es la trayectoria de referencia y $\hat{y}(k+p|k)$ son las predicciones de la salida sobre el horizonte de predicción N, note que:

$$\begin{aligned} N \geq N_u \\ \|M\|^2_A = M^T A M \\ Q_{MPC} \geq 0, R_{MPC} \geq 0 \end{aligned}$$

Las matrices $Q_{MPC} \epsilon \mathbb{R}^{nxn}$ y $R_{MPC} \epsilon \mathbb{R}^{mxm}$ (ecuación 4.17 y 4.18) corresponden a las matrices de penalización de estados y entradas de control respectivamente, sintonizadas de manera heurística, siendo n el numero de estados y m el número de estradas de control, note que solo se realiza el control de movimiento en X, Y y Z, manteniendo la orientación como $\psi = 0$ controlada con un control PID:

$$Q_{\text{MPC}} = \text{diag}\left(Q_x, Q_y, Q_z, Q_{vx}, Q_{vy}, Q_{vz}\right) \tag{4.17}$$

$$R_{\text{MPC}} = \text{diag}\left(R_x, R_y, R_z\right) \tag{4.18}$$

Una vez formulado el problema de optimización, se resuelve empleando la biblioteca de Python, CVXPY. En este proceso, el modelo dinámico discretizado, representado por la ecuación 4.8, se utilizó para estimar la secuencia de control a lo largo del horizonte de predicción, que fue elegido como una duración de 10 pasos de tiempo.

Además, se seleccionaron las siguientes restricciones físicas para los estados de velocidad de la planta, tal como se muestra en la ecuación 4.19.

$$\begin{pmatrix} -3\frac{m}{s} & -3\frac{m}{s} & -3\frac{m}{s} \end{pmatrix}^T \leq v \leq \begin{pmatrix} 3\frac{m}{s} & 3\frac{m}{s} & 3\frac{m}{s} \end{pmatrix}^T \tag{4.19}$$

Mientras que la ecuación 4.20 expresa las restricciones para los controles :

$$\begin{pmatrix} -1 & -1 & -1 \end{pmatrix}^T \leq u \leq \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}^T \tag{4.20}$$

Un diagrama general de como se implementa el control MPC con el UAV se muestra en la Figura 4.6.

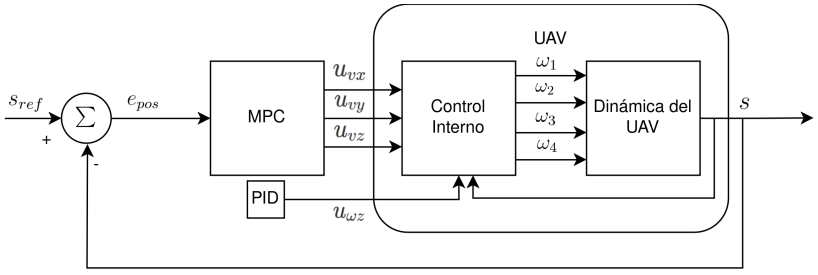


Figura 4.6: Esquema de control MPC implementado para el Bebop 2.0

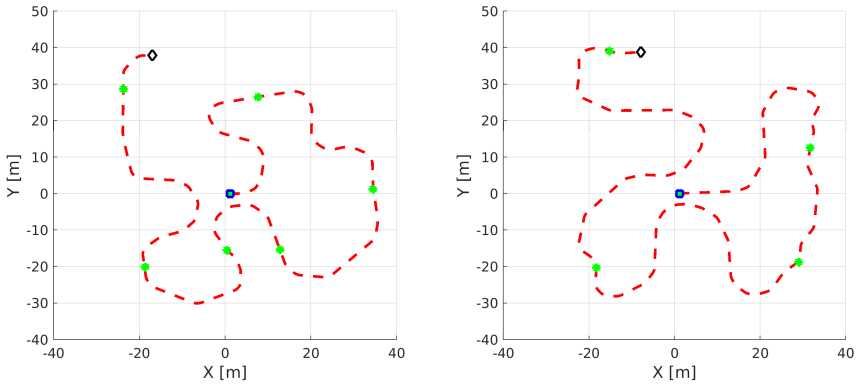
4.6.3. Diseño de trayectorias de validación de controles PID y MPC

Con el propósito de validar los controladores PID y MPC, se obtuvo un conjunto de 6 trayectorias con las características definidas en el punto 5 de la sección 4.6. Las trayectorias “A”, “B”, “C”, “D”, “E” y “F” (figura 4.7), abarcan un área de 50 x 50 m aproximadamente y están conformadas por una serie de puntos estimados desde la cámara del UAV que representan la posición del “Objetivo” mientras este fue conducido de forma manual a una velocidad de 1.5 m/s, los experimentos para obtener cada una de estas trayectorias se detallan en la sección 5.5.3.

4.6.4. Validación de los controles PID y MPC

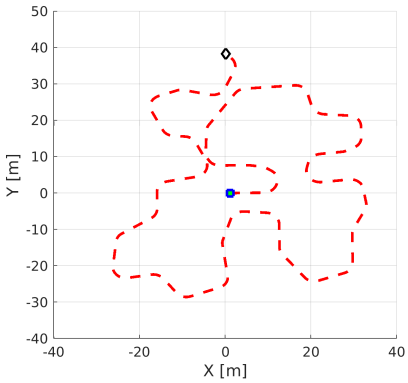
La validación de los controles consiste en el seguimiento de las trayectorias “A”, “B”, “C”, “D”, “E” y “F”. Simultáneamente, se almacena un conjunto de datos en un archivo “.txt” para cada una de ellas. Este conjunto de datos incluye los estados del UAV, el historial de las últimas 4 ubicaciones del “Objetivo” antes de su posición actual, las señales de control, los errores de posición, etc. El conjunto de datos recopilados en cada instante de tiempo se detalla en la tabla 4.2.

La recopilación de este conjunto de datos se llevó a cabo a una frecuencia de 20 Hz durante 4500 iteraciones mientras se efectuaba el seguimiento de cada una de las trayectorias mediante los controladores PID y MPC, generando así un total de 12 archivos “.txt” (6 para cada controlador). Los detalles de los experimentos de validación se encuentran en la sección 5.5.4. A partir de este conjunto de datos, se realizó el cálculo de las métricas establecidas en la sección 4.6.4 con el objetivo de validar los controladores. Los datos recabados también se utilizan para el entrenamiento de las versiones neuronales de los controles PID y MPC.

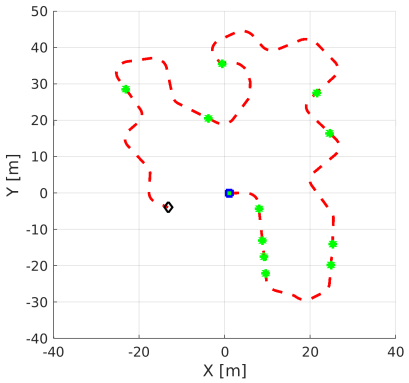


(a) Trayectoria “A”.

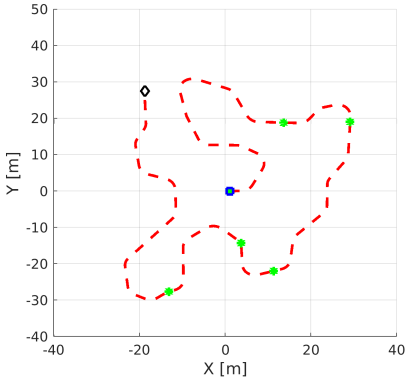
(b) Trayectoria “B”.



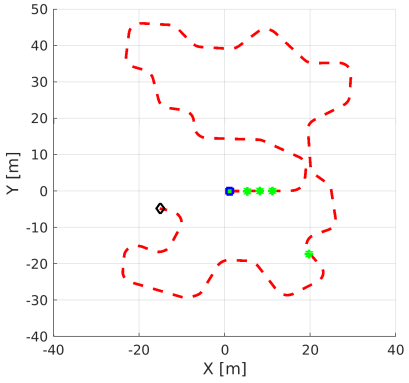
(c) Trayectoria “C”.



(d) Trayectoria “D”.



(e) Trayectoria “E”.



(f) Trayectoria “F”.

Figura 4.7: Trayectorias utilizadas para la validación de los controladores PID y MPC con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella.

Variable	Significado	Unidades	PID	MPC
$\mathbf{x}_{k-4}$	Posición estimada del objetivo en X en el instante $k-4$.	m	✓	✓
$\mathbf{y}_{k-4}$	Posición estimada del objetivo en Y en el instante $k-4$.	m	✓	✓
$\mathbf{x}_{k-3}$	Posición estimada del objetivo en X en el instante $k-3$.	m	✓	✓
$\mathbf{y}_{k-3}$	Posición estimada del objetivo en Y en el instante $k-3$.	m	✓	✓
$\mathbf{x}_{k-2}$	Posición estimada del objetivo en X en el instante $k-2$.	m	✓	✓
$\mathbf{y}_{k-2}$	Posición estimada del objetivo en Y en el instante $k-2$.	m	✓	✓
$\mathbf{x}_{k-1}$	Posición estimada del objetivo en X en el instante $k-1$.	m	✓	✓
$\mathbf{y}_{k-1}$	Posición estimada del objetivo en Y en el instante $k-1$.	m	✓	✓
$\mathbf{x}_k$	Posición estimada del objetivo en X en el instante k.	m	✓	✓
$\mathbf{y}_k$	Posición estimada del objetivo en Y en el instante k.	m	✓	✓
$\boldsymbol{\psi}$	Ángulo *yaw* del UAV en el instante k	rad	✓	
$\mathbf{rot}_e$	Error absoluto entre el angulo de referencia y ψ	rad	✓	
$\mathbf{x}_{uav}$	Posición X del UAV en el instante k	m	✓	✓
$\mathbf{y}_{uav}$	Posición Y del UAV en el instante k	m	✓	✓
$\mathbf{u}_{vx}$	Señal de control u_x	$adim.$	•	•
$\mathbf{u}_{vy}$	Señal de control u_y	$adim.$	•	•
$\mathbf{u}_{\omega z}$	Señal de control $u_{\omega z}$	$adim.$	•	
$\mathbf{V}_x$	Velocidad lineal x del UAV	m/s	✓	✓
$\mathbf{V}_y$	Velocidad lineal y del UAV	m/s	✓	✓
$\boldsymbol{\omega_z}$	Velocidad angular del UAV alrededor de z	rad/s	✓	
$\mathbf{x}_e$	Error de posición en X en el instante k	m	✓	✓
$\mathbf{y}_e$	Error de posición en Y en el instante k	m	✓	✓
dist	Distancia en el instante k entre el UAV y el objetivo.	m	✓	✓
$\mathbf{t}_{uc}$	Tiempo que le toma al controlador calcular u_c	s		

Tabla 4.2: Datos numéricos almacenados en los archivos ".txt", generados durante el seguimiento del "Objetivo" para la validación de los controles y el entrenamiento de sus versiones neuronales. Se utiliza ✓para indicar que el dato es una entrada de la red para la predicción de señales de control, mientras que las salidas de la red se indican con •.

Métricas de validación

Las métricas utilizadas nos proporcionarán información cualitativa y cuantitativa sobre el rendimiento de los controladores para evaluar de manera objetiva el desempeño de los sistemas de control.

- La Raíz del Error Cuadrático Medio (RMSE) en X y Y entre la trayectoria de referencia y la trayectoria seguida por el UAV.
- Velocidad promedio del UAV a lo largo del seguimiento.
- Error de posición en Estado Estacionario (SSE) entre el UAV y el "Objetivo".
- Tasa de Éxito de Seguimiento (TSR) utilizada en (Xie et al., 2020) definida como:

$$TSR = \sum_{i=0}^{N} \frac{D_i}{N} \text{x} 100\,\%, D_i = \begin{cases} 1 & dist < umbral \\ 0 & dist \geq umbral \end{cases}$$

  donde *dist* se refiere a la distancia horizonte entre el UAV y el objetivo y el *umbral* se refiere a la distancia para la cual queremos evaluar el seguimiento.
- Frecuencia de operación promedio y mínima del controlador.

El RMSE entre la trayectoria de referencia y la seguida por el UAV proporciona información valiosa sobre la precisión y similitud de ambas trayectorias. Un valor cercano a cero indica que el UAV sigue de manera precisa la trayectoria objetivo, manteniendo una proximidad constante con la trayectoria de referencia.

En relación a la velocidad promedio del UAV, es fundamental que permanezca cercana a la velocidad objetivo de 1.5 m/s. Una velocidad inferior sugiere un retraso del UAV respecto al objetivo, mientras que un aumento indica que el UAV se está acercando.

El SSE nos proporciona información sobre la capacidad del UAV para aproximarse al punto deseado, especialmente después de que este se ha detenido por al menos dos segundos. Un SSE cercano a cero indica una posición prácticamente sobre el objetivo, evidenciando alta precisión y permitiendo determinar la capacidad del controlador para mantener en vuelo estacionario al UAV en puntos específicos de la trayectoria, en caso contrario indica que el UAV no llega a posicionarse con precisión sobre el "Objetivo" una vez que este se ha detenido.

Según las características de diseño de los controladores, que requieren mantener el objetivo a menos de 2.5 metros, el TSR calcula el porcentaje en el cual el UAV se mantuvo a una distancia menor a este umbral durante el seguimiento. Un valor cercano al 100 % indica un controlador efectivo que mantiene al UAV dentro de este umbral haciendo menos probable que el "Objetivo" escape del FoV de la cámara. Un valor más bajo sugiere dificultades en mantener la proximidad.

La frecuencia promedio de operación refleja el tiempo necesario para que el controlador realice los cálculos de las señales de control. Un valor cercano a cero indica un mayor tiempo requerido para obtener las señales, lo que podría interpretarse como "lentitud" en el desempeño del controlador. Para garantizar un funcionamiento óptimo, es

esencial que tanto la frecuencia promedio como la mínima superen los 20 Hz, evitando retrasos en el sistema y asegurando un seguimiento preciso.

4.6.5. Control Neuronal

De acuerdo con (Nguyen et al., 2002), se puede implementar una red neuronal como controlador, donde la red puede ser entrenada a partir de cierto criterio, ya sea utilizando únicamente datos numéricos de entrada-salida o a través de un modelo matemático de la planta. En este contexto, se desarrolla un MLP como controlador utilizando únicamente datos numéricos de entrada-salida para el entrenamiento y validación de la red, a fin de que la red pueda generar señales de control con información de entrada.

En el estado del arte, algunos trabajos utilizan un MLP como controlador, incorporando 3 capas ocultas (Li et al., 2021; Muller et al., 2019; Xie et al., 2020). Considerando esto, se llevaron a cabo diversas configuraciones, variando el número de nodos, capas y otros parámetros. No obstante, se observó que 3 capas ocultas eran suficientes para abordar el problema en cuestión. La figura 4.8 ilustra la arquitectura diseñada para el control neuronal PID denotado de ahora en adelante como N-PID, que cuenta con 20 neuronas de entrada, 3 capas ocultas con 64, 32 y 16 neuronas respectivamente y 3 neuronas de salida para el calculo de comandos de Alabeo, Cabeceo y Guiñada, mientras que la altitud se controla a través de un control PID. Los detalles de las entradas y salidas de la red se encuentran en la tabla 4.2.

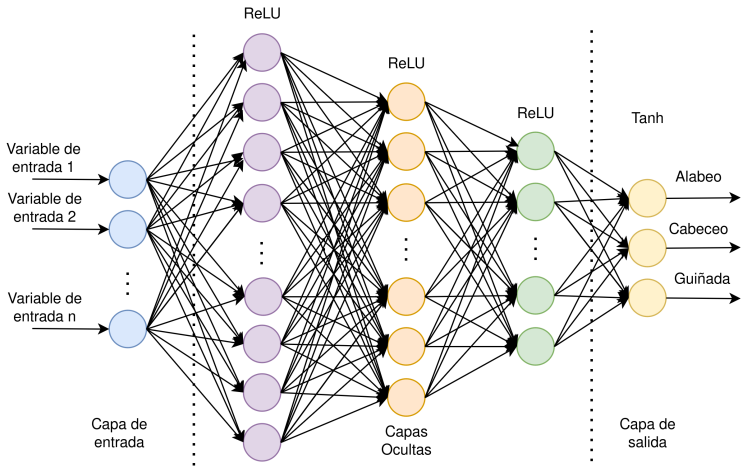


Figura 4.8: Arquitectura del MLP desarrollado para la predicción de señales de control.

Por otro lado, la arquitectura utilizada para el control neuronal MPC denotado de ahora en adelante como N-MPC experimenta ligeras modificaciones en la capa de entrada, con 17 neuronas, y en la capa de salida, con 2 neuronas correspondientes a los comandos de Alabeo y Cabeceo, mientras que los comandos de altura y Guiñada se calculan a través de un control PID. Estas adaptaciones se deben a la naturaleza del MPC desarrollado en la sección 4.6.2, donde la orientación del UAV permanece

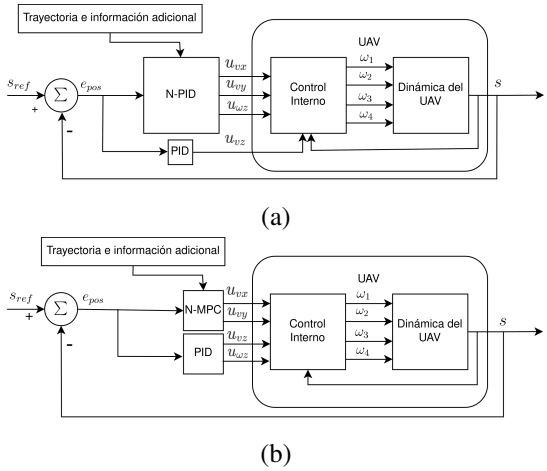


Figura 4.9: Esquema de control híbrido implementado para el Bebop 2.0. Las visualizaciones muestran a) N-PID y b) N-MPC.

constante en todo momento (una señal de control no contemplada en el modelo). Por lo tanto, algunos datos de la tabla 4.2 resultan irrelevantes para el entrenamiento de la red.

Ambas arquitecturas diseñadas se tratan de un MLP con todas sus capas completamente conectadas, donde la función de activación de las tres primeras capas ocultas corresponde a la función ReLU, mientras que la capa de salida a la función Tanh. Al tratarse de un problema de regresión, se decidió dejar la última capa con esta función de activación debido a que la salida de la red (señal de control) puede ser cualquier valor real comprendido entre [-1,1], y la función de activación puede limitar el rango de los valores que la red puede predecir en el rango requerido.

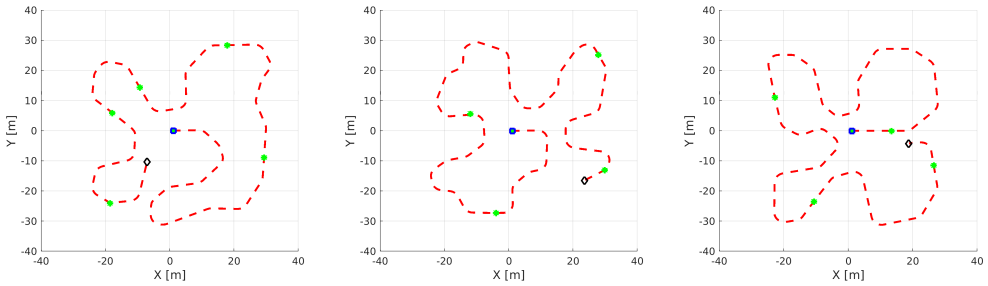
Un diagrama general de como se implementa el control neuronal y PID para control de actitud y altitud para los modelos N-PID y N-MPC con el UAV se muestra en la figura 4.9.

Utilizando los archivos generados en la sección 4.6.4, se lleva a cabo el entrenamiento de las arquitecturas N-PID y N-MPC, empleando el 80 % de los datos para el entrenamiento y el 20 % para la validación en cada caso.

- **Entrenamiento de los controles neuronales:** El entrenamiento se lleva a cabo utilizando el optimizador Adam durante 300 épocas, con una tasa de aprendizaje de 0.001 y un tamaño de lote de 100.
- **Validación de los controles neuronales:** La etapa de validación consiste en utilizar el 20 % de los datos restantes para predecir las señales de control correspondientes y obtener el RMSE entre la predicción obtenida de la red (u_{cr}) con las salidas de control (u_c) de los controles PID y MPC de cada archivo ".txt" respectivamente.

4.7. Validación de los controles N-PID y N-MPC

Una vez entrenados los modelos N-PID y N-MPC para la predicción de señales de control, se diseñaron las trayectorias “G”, “H” y “I” (vea figura 4.10) con el procedimiento descrito en la sección 4.6.3.



(a) Trayectoria “G”. (b) Trayectoria “H”. (c) Trayectoria “I”.

Figura 4.10: Trayectorias utilizadas para la validación de los controles neuronales, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella.

Los experimentos realizados para la validación de los controladores se encuentran detallados en la sección 5.6. En estos experimentos se lleva a cabo el seguimiento del “Objetivo” a lo largo de estas trayectorias, registrando simultáneamente un conjunto de datos (vea tabla 4.3) en un archivo de texto ".txt", con el fin de analizar los resultados obtenidos y validar los controladores a través de las métricas indicadas en la sección 4.6.4.

Variable	Significado	Unidades
$\mathbf{x}_{obj.}$	Posición del objetivo en X medida del simulador.	m
$\mathbf{y}_{obj.}$	Posición del objetivo en Y medida del simulador.	m
$\mathbf{z}_{obj.}$	Posición del objetivo en Z medida del simulador.	m
$\mathbf{t}_{uc.}$	Tiempo que le toma al controlador calcular u_c.	s

Tabla 4.3: Datos numéricos almacenados en los archivos “.txt”, generados durante el seguimiento del “Objetivo” para la validación de los controladores neuronales.

Capítulo 5

Experimentos y Resultados

En este capítulo se muestran los experimentos realizados para la validación de los módulos desarrollados para el seguimiento de objetos móviles a través de un UAV utilizando DNN.

5.1. Percepción del objetivo

Tras entrenar YOLOv8 con nuestro conjunto de imágenes según la configuración detallada en la sección 4.1, logramos un mAP50 de 0.9947 después de 100 épocas. Posteriormente, con el objetivo de determinar el porcentaje de confianza promedio en la detección a diferentes alturas, realizamos vuelos manuales a 3, 4 y 5 metros. Durante estos vuelos, nos aseguramos de mantener constantemente el "Objetivo" dentro del campo de visión de la cámara del UAV. Los resultados se presentan en la tabla 5.1.

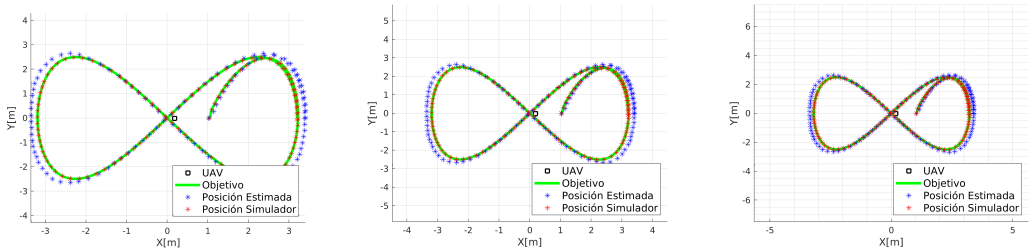
"Objetivo"	**3 m**	**4 m**	**5 m**
% Confianza prom.	91.43	91.21	89.15

Tabla 5.1: Porcentaje de confianza promedio obtenida en la detección de la clase "Objetivo" desde distintas alturas.

Como se observa, el porcentaje de confianza en la detección disminuye a medida que el UAV se eleva. Esta información resulta valiosa para establecer la altura apropiada en la que se logre detectar el objetivo con al menos un 85 % de confianza a lo largo del seguimiento.

5.2. Estimación de posición

A continuación, se presentan los experimentos llevados a cabo con el objetivo de evaluar el error en la estimación de posición obtenida a través de la cámara del UAV desde



(a) (b) (c)

Figura 5.1: Área percibida desde la cámara del UAV y comparación entre la posición del "Objetivo" obtenida del simulador contra la posición estimada por la cámara del UAV para una trayectoria de Lemniscata a alturas de (a) 3 m, (b) 4 m y (c) 5 m.

diferentes alturas. Este análisis busca determinar la altura adecuada para nuestra aplicación, la cual está directamente relacionada con el área de observación alcanzada desde diversas alturas y con el RMSE obtenido para una región específica que consideramos ideal para mantener al "Objetivo" durante el seguimiento.

Experimento error de estimación: Se coloca el UAV a una altura de 3 m en vuelo estacionario sobre el origen del sistema de coordenadas global (0,0,3) con el UAV orientado hacia el eje $X+$, es decir, ($\psi = 0°$). Mientras que, el "Objetivo " se coloca 1 m delante (1,0,0) con la misma orientación. Posteriormente el "Objetivo" comienza a seguir una trayectoria Lemniscata con radio mayor y menor de 3.2 m y 2.5 m sobre los ejes X y Y respectivamente, con centro justo debajo del UAV. La estimación de posición se lleva a cabo en intervalos de tiempo de 0.4 s mientras el "Objetivo" sigue la trayectoria.

Este experimento también se realizó colocando el UAV a alturas de 4 m y 5 m. Al finalizar, se grafican las posiciones reales del "Objetivo" contra la estimación de posición obtenida desde el UAV para las alturas anteriormente mencionadas (vea figura 5.1).

Además, se realiza el cálculo del RMSE entre las posiciones estimadas y las obtenidas a través del simulador en cada experimento (consulte la tabla 5.2). Asimismo, se presenta el área total de observación aproximada alcanzada desde estas alturas. La menor y mayor distancia percibida está asociada con lo alto y ancho de la imagen respectivamente.

Magnitud	**3m**	**4m**	**5m**	**Unidades**
RMSE	0.17981	0.17591	0.15453	m
Área percibida	6.46 x 8.58	8.96 x 11.74	11.3 x 15	m

Tabla 5.2: RMSE y área total percibida para el **Experimento error de estimación.**

De acuerdo con la tabla 5.2, se observa que el RMSE disminuye conforme el UAV se eleva y aumenta en caso contrario. Este comportamiento está relacionado con el área de observación percibida por la cámara. Cuando el UAV se eleva, la trayectoria descrita por el objetivo, se concentra en un área cada vez más compacta en relación con el área total percibida desde esa altura. Esto provoca que el error de estimación sea menor en comparación a cuando el UAV vuela a una altura menor.

Aunque el RMSE nos proporciona un promedio de las diferencias entre las trayectorias estimadas y las reales para cada caso, es evidente que algunos puntos presentan discrepancias mayores o menores a esta cantidad. De la figura 5.1, podemos observar que el error de estimación aumenta conforme el "Objetivo" se aleja del centro de la imagen de la cámara, y disminuye en caso contrario, llegando a alcanzar un valor mínimo cuando el objetivo se encuentra justo debajo del UAV.

Basándonos en los experimentos realizados anteriormente, seleccionamos una altura de seguimiento de 4 metros debido al comportamiento del error de estimación a esta altura, el área total de observación percibida y que nos garantiza un porcentaje de confianza en la detección del " Objetivo" mayor al 85 % desde esta altura (vea tabla 5.1). Esta altura también tiene la ventaja de ser la intermedia entre los experimentos realizados, lo cual facilita la exclusión de resultados erróneos en la estimación de posición en las etapas subsiguientes.

Una vez definida la altura de vuelo, y comprendido el comportamiento del error de estimación, es necesario conocer el error obtenido a diferentes distancias desde el centro de la imagen. Por este motivo, se realiza el siguiente experimento.

Experimento error en zonas de observación: Bajo el mismo escenario del experimento error de estimación, posicionando ahora el UAV en vuelo estacionario a una altura de 4 m, el objetivo realiza el seguimiento de trayectorias circulares centradas en la posición del UAV, con radios de 2, 3 y 4 m. Definiremos el área contenida dentro de estas circunferencias como zonas de observación "A", "B" Y "C" respectivamente. Al finalizar, se grafican las posiciones reales del "Objetivo" contra la estimación de posición obtenida desde el UAV (vea figura 5.2) para las trayectorias anteriormente mencionadas. Se calcula el RMSE entre las posiciones estimadas y las medidas del simulador para las trayectorias del experimento anterior (vea tabla 5.3).

Circunferencia	**r=2 m**	**r=3 m**	**r=4 m**	**Unidades**
RMSE	0.13139	0.17930	0.23115	m

Tabla 5.3: RMSE obtenido para el **Experimento error en zonas de observación**.

El RMSE en la tabla anterior representa el error de estimación máximo alcanzado mientras el "Objetivo" se encuentre dentro de alguna de las zonas "A","B" o "C".

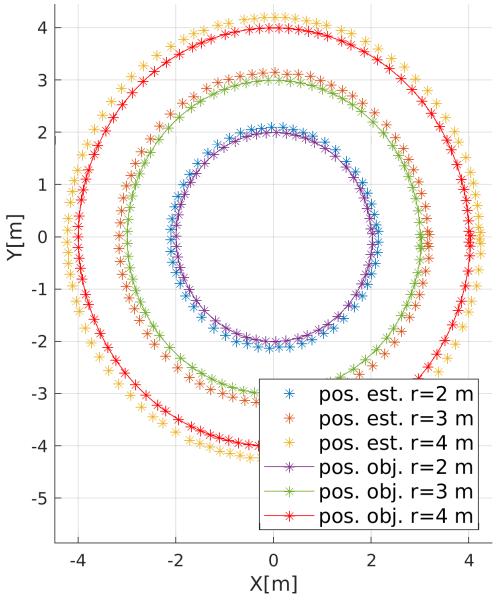


Figura 5.2: Comparación entre la posición del "Objetivo" medida del simulador y su posición estimada a través de la cámara del UAV durante el Experimento error en zonas de Observación en una circunferencia con radios de 2 m, 3 m y 4 m.

5.3. Identificación de la planta

Se llevaron a cabo tres experimentos con el objetivo de determinar la función de transferencia de primer orden de la planta Parrot Bebop 2.0 para el movimiento traslacional en X, Y y Z.

Experimento de Identificación: El UAV fue colocado en vuelo estacionario a una altura de 15 m sobre el origen del sistema de coordenadas global (0,0,15) con el UAV orientado hacia el eje $X+$, es decir, ($\psi = 0°$). Posteriormente, se introdujo al UAV una señal de control $u_c = (u_{vx}, u_{vy}, u_{vz}, u_{\omega z})$ con $u_{v_x} = u_{ide}$, donde u_{ide} es la señal de control mostrada en la figura 5.3, la cual parte desde cero y va aumentando hasta el valor máximo de 1, atravesando valores positivos y negativos en incrementos de 0.1, mientras que, u_{vy}, u_{vz} y $u_{\omega z}$ se mantienen en 0 con la finalidad de hacer que el UAV se mueva únicamente en el eje X. Durante el movimiento del UAV se guardan las mediciones de posición x, velocidad lineal v_x y la señal de control u_{vx} en un archivo ".txt" a una frecuencia de 20 Hz (frecuencia seleccionada debido a que la cámara del Bebop 2.0 trabaja a 30 Hz). El experimento concluye una vez que toda la señal de control ha sido ejecutada por completo.

La figura 5.4 muestra gráficamente los datos de entrada y salida para la velocidad lineal en X, obtenida durante el experimento con los cuales se realiza la identificación de la planta utilizando MATLAB®. Sin embargo, este experimento se realizó también para el movimiento en el eje Y y Z enviando la señal de control u_{ide} a u_{vy} y u_{vz} respectivamente, manteniendo el resto de señales en 0 en cada caso. Tras finalizar estos

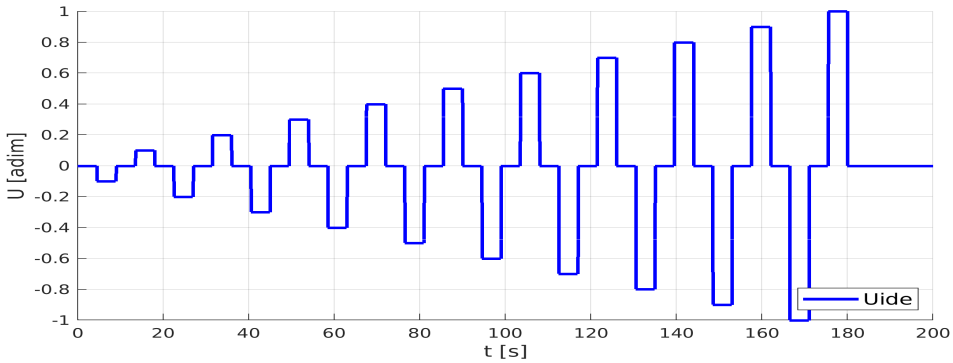


Figura 5.3: Señal de control u_{ide} para la identificación de la planta donde $u_{vx} = u_{vy} = u_{vz} = u_{ide}$ aplicado como $u_c = (u_{vx}, 0, 0, 0)$, $u_c = (0, u_{vy}, 0, 0)$ y $u_c = (0, 0, u_{vz}, 0)$ de forma independiente.

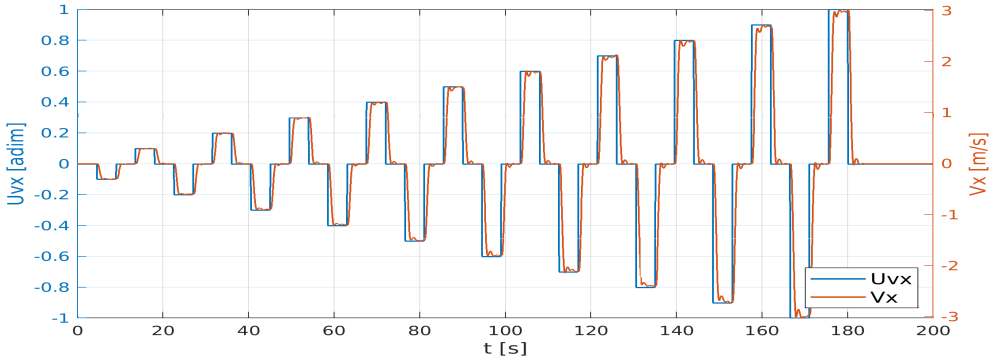


Figura 5.4: Señal de control aplicada al UAV y velocidad lineal obtenida para el Bebop 2.0 para el movimiento traslacional en X.

experimentos se obtuvieron 3 archivos ".txt" (cada uno con información de el valor de la señal de control, posición [1] y velocidad lineal en cada eje).

Las funciones de transferencia obtenidas se muestran en las ecuaciones 5.1, 5.2 y 5.3 para el movimiento de traslación del UAV en X, Y, Z respectivamente.

$$H_{vx}(s) = \frac{v_x(s)}{u_{cvx}(s)} = \frac{3{,}1074}{0{,}7462s + 1} \tag{5.1}$$

$$H_{vy}(s) = \frac{v_y(s)}{u_{cvy}(s)} = \frac{3{,}1085}{0{,}7434s + 1} \tag{5.2}$$

$$H_{vz}(s) = \frac{v_z(s)}{u_{cvz}(s)} = \frac{3{,}1371}{1{,}0699s + 1} \tag{5.3}$$

[1] Los datos de posición (x, y, z) se utilizan para la validación de el módulo de filtro de Kalman mostrado en la siguiente sección.

Cada una de las ganancias K del sistema y las constantes de tiempo τ se muestran en la tabla 5.4 junto con el porcentaje de ajuste del modelo obtenido con respecto a los datos de entrada y salida proporcionados para su estimación.

Parámetro	$\mathbf{H_{vx}(s)}$	$\mathbf{H_{vy}(s)}$	$\mathbf{H_{vz}(s)}$
τ	0.7462	0.7434	1.0699
K	3.1074	3.1085	3.1371
% de ajuste	84.82	84.87	91.44

Tabla 5.4: Parámetros de la dinámica lineal del UAV Parrot Bebop 2.0 estimados con MATLAB®.

Sustituyendo los datos de la tabla 5.4 en la ecuación 4.8, finalmente obtenemos su representación matricial $\dot{s}\left(t\right) = As\left(t\right) + Bu\left(t\right)$, donde $s(t)$ corresponde a los estados del UAV y $u(t)$ a los controles:

$$\dot{s}\left(t\right) = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1{,}34 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1{,}345 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0{,}9347 \end{bmatrix} s\left(t\right) + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 4{,}164 & 0 & 0 \\ 0 & 4{,}181 & 0 \\ 0 & 0 & 2{,}932 \end{bmatrix} u\left(t\right) \tag{5.4}$$

5.4. Filtro de Kalman

La implementación del filtro de Kalman requiere discretizar las matrices A y B de la ecuación 5.4. Estas son discretizadas a 20 Hz a través de la función `c2d` de MATLAB para obtener sus versiones discretas A_d y B_d (ecuación 5.5 y 5.6).

$$A_d = \begin{bmatrix} 1{,}0000 & 0{,}0 & 0{,}0 & 0{,}0484 & 0{,}0 & 0{,}0 \\ 0{,}0 & 1{,}0000 & 0{,}0 & 0{,}0 & 0{,}0484 & 0{,}0 \\ 0{,}0 & 0{,}0 & 1{,}0000 & 0{,}0 & 0{,}0 & 0{,}0488 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}9352 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}9350 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}0 & 0{,}9543 \end{bmatrix} \tag{5.5}$$

$$B_d = \begin{bmatrix} 0{,}0051 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}0051 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}0036 \\ 0{,}2014 & 0{,}0 & 0{,}0 \\ 0{,}0 & 0{,}2022 & 0{,}0 \\ 0{,}0 & 0{,}0 & 0{,}1432 \end{bmatrix} \tag{5.6}$$

A continuación se muestra el experimento realizado para la sintonización de las matrices Q_{kalman} y R_{kalman} para la estimación de estados de velocidad lineal a partir de datos de posición.

Experimento de sintonización de matrices $\mathbf{Q_{kalman}}$ y $\mathbf{R_{kalman}}$: Se propone una aproximación inicial de la matriz de covarianza P_{kalman} del estado estimado (véase ecuación 4.9). A partir de esta aproximación, las matrices Q y R (véase ecuaciones 4.10 y 4.11) se ajustan de manera heurística a lo largo de un proceso iterativo de estimación de estados.

Durante este proceso iterativo, se utilizan los datos de posición y velocidad obtenidos en los experimentos de la sección 5.3. Se emplea el RMSE entre las predicciones realizadas por el filtro y los datos de velocidad medidos del simulador para cuantificar la precisión del filtro de Kalman en cada iteración. Este ajuste iterativo se repite hasta alcanzar una precisión que se considere aceptable.

Las matrices P_{kalman}, Q_{kalman} y R_{kalman} sintonizadas se muestran en las ecuaciones 5.7,5.8 y 5.9.

$$P_{\text{kalman}} = \text{diag}\,(0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1) \tag{5.7}$$

$$Q_{\text{kalman}} = \text{diag}\,(0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1, 0{,}1) \tag{5.8}$$

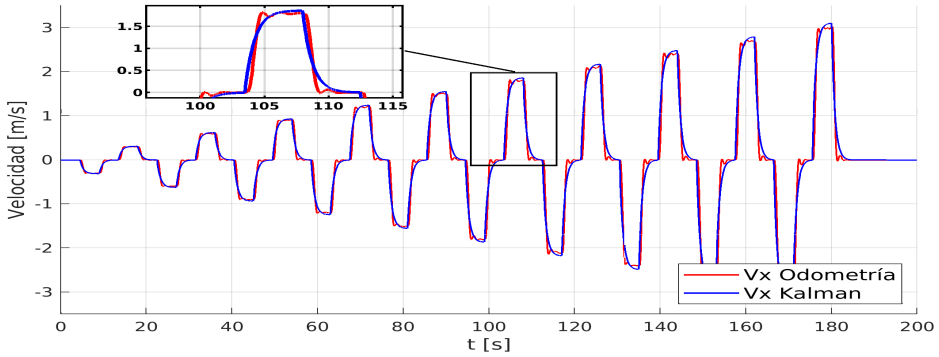
$$R_{\text{kalman}} = \text{diag}\,(0{,}1, 0{,}1, 0{,}1) \tag{5.9}$$

La tabla 5.5 muestra los resultados obtenidos del filtro de Kalman implementado utilizando las matrices anteriormente mostradas. Estos resultados reflejan la Raíz del Error Cuadrático Medio resultado de comparar las mediciones obtenidas por el filtro de Kalman contra las mediciones obtenidas del simulador realizadas en la sección 5.3, indicándonos en promedio cuánto difieren nuestras predicciones de los datos experimentales.

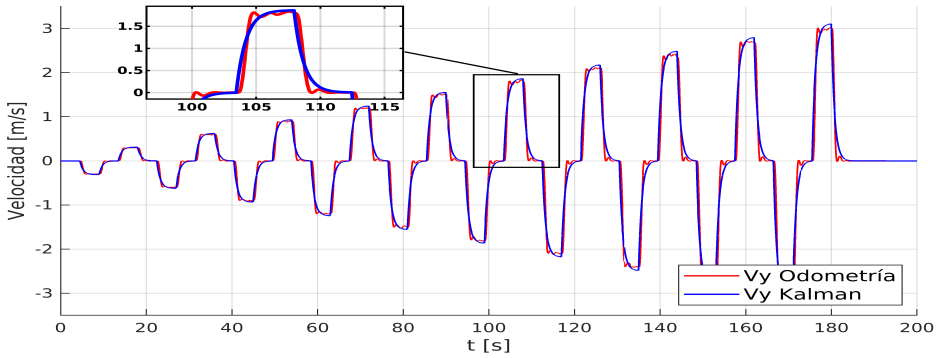
RMSE	$\mathbf{v_x}$	$\mathbf{v_y}$	$\mathbf{v_z}$
m/s	0.1981	0.1977	0.1146

Tabla 5.5: RMSE obtenido entre las mediciones del simulador y las predicciones de velocidad utilizando las matrices sintonizadas Q_{kalman} y R_{kalman}.

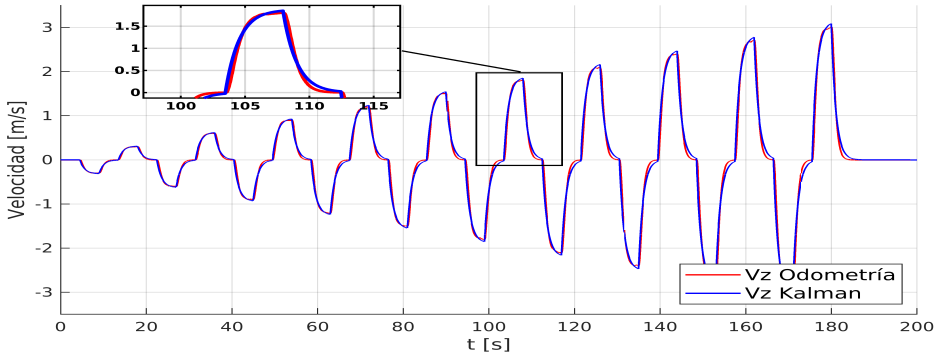
La figura 5.5 proporciona una visualización de los valores de velocidad obtenidos del simulador comparados con las estimaciones generadas por el filtro de Kalman. Este análisis proporciona una perspectiva visual y cuantitativa de la eficacia del filtro en la estimación de la velocidad en cada eje, ofreciendo una evaluación integral de su rendimiento en función de los datos recopilados.



a)



b)



c)

Figura 5.5: Estimación de velocidad obtenida con el filtro de Kalman contra mediciones del simulador para los datos obtenidos en la sección 5.3 para a) v_x, b) v_y y c) v_z.

5.5. Diseño de controles

Una vez definida la altura de seguimiento, conocido el error máximo de estimación en cada zona de observación, identificado el modelo dinámico lineal de la planta y aplicado el filtro de Kalman para estimar los estados de velocidad, se establecen las características de diseño para los controladores. El objetivo principal es lograr un seguimiento preciso del "Objetivo", manteniéndolo lo más cercano posible al UAV. Para lograr esto, se llevan a cabo una serie de experimentos destinados a ajustar los controles, teniendo en cuenta las siguientes consideraciones:

- Dada la velocidad máxima del "Objetivo", fijada en 1.5 m/s, se elige un periodo de estimación de posición de 0.4 s. Esto significa que, después de transcurrir este intervalo, el "Objetivo" habrá avanzado, como máximo, una distancia de 60 cm con respecto a su posición anterior. En consecuencia, se sintonizan los controles con el objetivo de trasladar el UAV de un punto a otro, separados por una distancia de 60 cm, en el menor tiempo posible. La meta ideal es lograr que el control alcance este punto en un lapso de 0.4 s.
- Considerando la capacidad del objetivo para detenerse en cualquier momento, se busca asegurar que los controles no generen un sobre impulso y que se establezcan con un error en estado estacionario menor al 4 %. El objetivo de esto es evitar que, en caso de que el objetivo se detenga, el UAV rebase al "Objetivo" asegurando así un seguimiento preciso y controlado del "Objetivo" en todo momento.

5.5.1. Control PID

La sintonización del control PID se llevó a cabo utilizando el método de Ziegler-Nichols como aproximación inicial, posteriormente se fueron ajustando las ganancias de manera heurística en un proceso iterativo como se detalla a continuación:

Experimento de sintonización PID: El proceso de sintonización comienza colocando el UAV en vuelo estacionario sobre el origen del sistema de coordenadas global, manteniendo una altitud de 1 m (0,0,1), con el UAV orientado hacia el eje $X+$, es decir, ($\psi = 0°$). Se establece un punto de referencia con coordenadas (0.6, 0.6, 1.6) y una orientación de $\psi = 45°$. A partir de este punto, se inicia la sintonización heurística utilizando el método de Ziegler-Nichols. El objetivo principal es ajustar las ganancias del control PID de manera precisa, con el fin de controlar los estados de posición y orientación para garantizar que se alineen adecuadamente con los requisitos de diseño establecidos previamente.

Los valores de las ganancias sintonizadas se presentan en la tabla 5.6, la figura 5.6 ilustra la respuesta obtenida con el control PID sintonizado para alcanzar la referencia establecida en el experimento anterior.

Estado	$\mathbf{k}_p$	$\mathbf{k}_d$	$\mathbf{k}_i$
x	0.23	0.15	0.000075
y	0.18	0.15	0.000075
z	0.8	0.15	0.0003
ψ	1.4	0.15	0.00001

Tabla 5.6: Ganancias sintonizadas para el control de seguimiento PID para el UAV bebop 2.0.

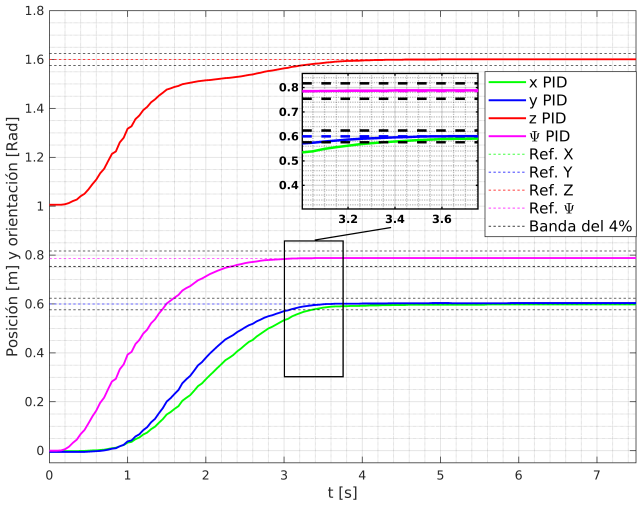


Figura 5.6: Respuesta del control PID sintonizado.

Como se muestra en la figura 5.6, se alcanza el estado estacionario casi simultáneamente en x, y y z en 3.4 s, sin sobre impulso en cada una de ellas, mientras que el estado estacionario para el angulo de guiñada (ψ) se alcanza en 2.35 s, después de 4.65 s se obtuvo un error inferior al 3.33 % en x, e inferiores al 1 % para y, z y ψ cumpliendo con las características de diseño.

5.5.2. MPC

Con la idea de realizar la sintonización de las matrices Q_{MPC} y R_{MPC} para el MPC se realiza el siguiente experimento, teniendo en cuenta que el MPC diseñado únicamente controla el movimiento en X, Y y Z.

Experimento de sintonización MPC: Bajo el mismo escenario del ambiente de simulación descrito en el experimento de sintonización PID, manteniendo ahora el ángulo de Guiñada (ψ= 0 °), se procedió a sintonizar el control MPC resolviendo el problema de optimización planteado en la sección 4.6.2 y ajustando las matrices Q_{MPC} y R_{MPC} hasta obtener el comportamiento deseado. El objetivo principal fue controlar

los estados de posición y garantizar que se alinearan adecuadamente con los requisitos de diseño establecidos previamente.

Las matrices Q_{MPC} y R_{MPC} sintonizadas se muestran a continuación (ecuación 5.10 y 5.11).

$$Q_{\text{MPC}} = \text{diag}\,(60, 60, 60, 30, 30, 35) \tag{5.10}$$

$$R_{\text{MPC}} = \text{diag}\,(65, 65, 80) \tag{5.11}$$

La figura 5.7 ilustra la respuesta obtenida con el MPC sintonizado para alcanzar la referencia establecida en el experimento.

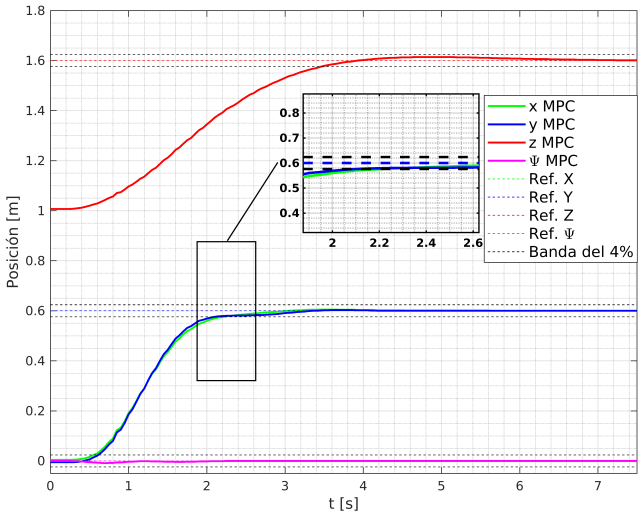


Figura 5.7: Respuesta del control MPC sintonizado.

Como se muestra en la figura 5.7, se alcanza el estado estacionario en x y y casi simultáneamente en un tiempo de 2.25 s, mientras que el estado estacionario en z se alcanza en 3.5 s, sin sobreimpulso en ninguna de ellas. Después de 3.35 s para x y y, se logra un error inferior al 1 %, y después de 5.65 s se alcanzó a la referencia. En cuanto al movimiento en z, se alcanza un error inferior al 1 % después de 3.8 s y se llega a la referencia después de 7.7 s, cumpliendo con las características de diseño establecidas.

5.5.3. Diseño de trayectorias de validación de controles PID y MPC

Se diseñaron las trayectorias “A”, “B”, “C”, “D”, “E” y “F” conforme al requisito 5 de la sección 4.6. Para obtener estas trayectorias, se llevaron a cabo experimentos siguiendo al robot terrestre con el propósito de identificar los puntos estimados que aproximan la trayectoria original seguida por el “Objetivo” en pasos de tiempo constantes. Durante este experimento, se recopiló información crucial que permite replicar la trayectoria estimada conforme el UAV va avanzando mientras se realiza el seguimiento con los

controladores PID y MPC. Este enfoque asegura que los resultados del rendimiento del seguimiento no se vean afectados por las variaciones en la estimación de posición con el fin de comparar los controladores bajo las mismas circunstancias, garantizando una evaluación precisa de su desempeño.

Experimento de obtención de trayectoria:

El experimento se inicia posicionando el UAV en vuelo estacionario sobre el origen del sistema de coordenadas global a una altura de 4 metros, es decir, en las coordenadas (0, 0, 4), con el UAV orientado hacia el eje $X+$ ($\psi = 0°$). Simultáneamente, el "Objetivo" se ubica 1 metro delante (1, 0, 0) con la misma orientación.

Después de 50 iteraciones (2.5 segundos), el UAV entra en modo autónomo y comienza a estimar la posición del objetivo en intervalos de 0.4 segundos mientras se dirige hacia el punto estimado. En cada estimación, se verifica si el punto calculado se ha desplazado más de 0.35 metros con respecto a la estimación anterior. En caso afirmativo, el UAV actualiza el punto meta y se dirige hacia esta nueva ubicación para mantener el "Objetivo" lo más cerca posible (idealmente en la zona de observación "A"), y evitar que salga del campo de visión. En caso contrario, si el punto calculado no se ha desplazado más de 0.35 metros durante este intervalo de tiempo con respecto a la estimación anterior, el UAV omite la estimación obtenida y el punto meta permanece sin cambios.

Durante el experimento, el robot terrestre es conducido manualmente a una velocidad constante de 1.5 m/s a lo largo y ancho de un área de aproximadamente 50 m x 50 m. Durante este recorrido, se realizan transiciones entre curvas y tramos rectos, incluyendo curvas de 90 grados, cerradas semicirculares, en forma de "S" y en forma de "U", con pausas eventuales.

El seguimiento del robot terrestre se lleva a cabo utilizando el controlador PID, durante un total de 4500 iteraciones (225 segundos). Durante este proceso, se almacena el conjunto de datos de la tabla 5.7 en un archivo ".txt" siempre que el punto meta se haya actualizado. El experimento concluye una vez que se han completado las 4500 iteraciones.

Al concluir el experimento, se genera un archivo que contiene tanto las posiciones estimadas y las medidas del simulador, asociadas a la trayectoria original descrita por el robot terrestre. Este archivo incluye información detallada, como la iteración en la que se llevó a cabo la estimación y el porcentaje de detección del "Objetivo" proporcionado por YOLOv8.

El experimento se realiza seis ocasiones para obtener las trayectorias A, B, C, D, E y F. Con los datos recopilados, se determina la distancia total recorrida por el "Objetivo" a lo largo de la trayectoria y se calcula el RMSE entre las posiciones estimadas y las medidas del simulador para la trayectoria realizada. Los resultados se presentan en la tabla 5.8.

Variable	Significado	Unidades
$x_{est.}$	Posición estimada del objetivo en X.	m
$y_{est.}$	Posición estimada del objetivo en Y.	m
$x_{obj.}$	Posición del objetivo en X medida del simulador.	m
$y_{obj.}$	Posición del objetivo en Y medida del simulador.	m
Iteración	Número de iteración.	adim.
Detección	% de confianza de la detección en YOLOv8	adim.

Tabla 5.7: Datos numéricos almacenados en los archivos ".txt", generados durante el seguimiento del "Objetivo" para la estimación de trayectorias de validación de los controladores PID y MPC.

Trayectoria	A	B	C	D	E	F
RMSE [m]	0.38095	0.38544	0.38217	0.40249	0.39589	0.38062
Longitud [m]	295.790	311.946	343.150	271.394	273.053	317.532

Tabla 5.8: Longitud de las trayectorias estimadas y RMSE obtenido.

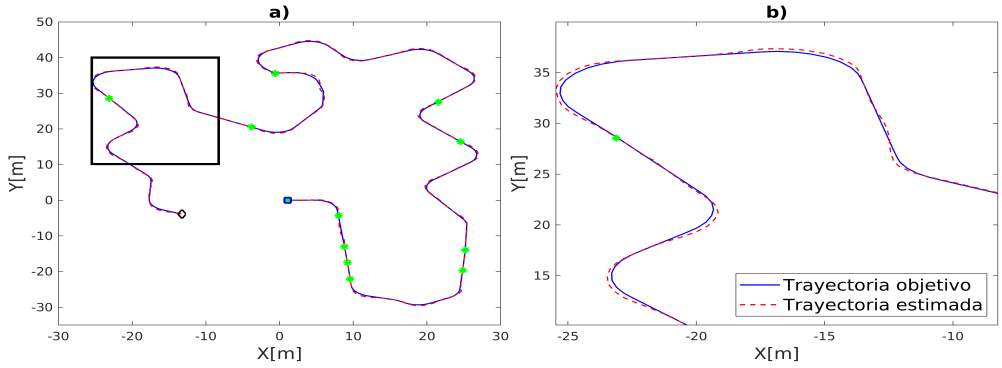
Debido al número y similitud entre las trayectorias realizadas y con el propósito de no extender demasiado este trabajo solo se muestran gráficamente las trayectoria D, E y F (vea figura 5.8) en la que se muestra la trayectoria estimada por el UAV a partir de la trayectoria original descrita por el "Objetivo" durante el experimento.

5.5.4. Validación de los controles PID y MPC

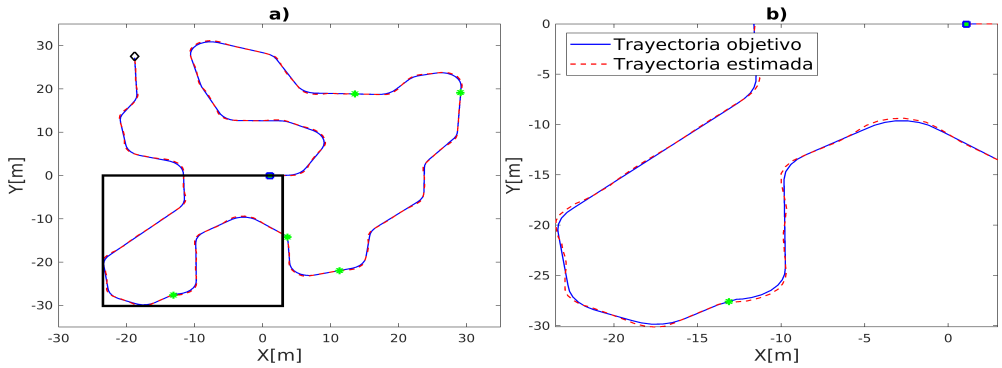
La validación de los controladores implica llevar a cabo el seguimiento de las trayectorias "A", "B", "C", "D", "E" y "F" utilizando los controladores PID y MPC. Durante este proceso, se recopila información esencial para realizar cálculos que permitan medir el desempeño de cada uno de los controladores. A continuación, se describe el experimento realizado para este propósito.

Experimento de validación de controles:

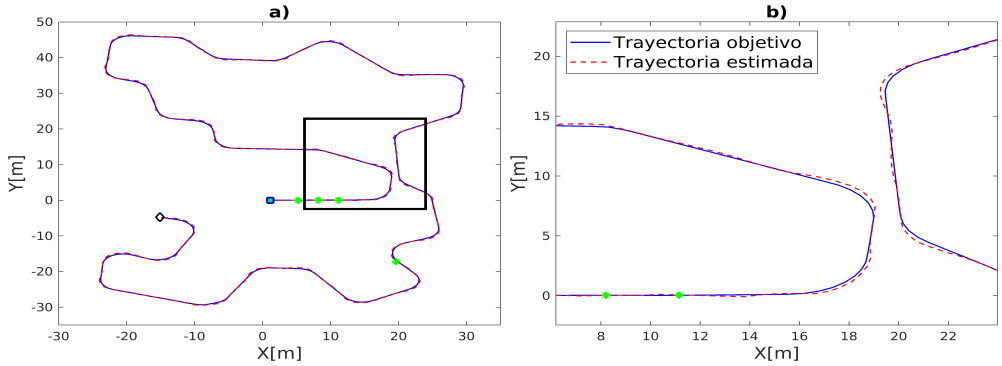
El experimento inicia con el UAV en vuelo estacionario sobre el origen del sistema de coordenadas global, a una altura de 4 metros, es decir, en las coordenadas (0,0,4), con el UAV orientado hacia el eje $X+$ ($\psi = 0°$). Después de un lapso de 50 iteraciones (2.5 segundos), el UAV cambia a modo autónomo y comienza el seguimiento de la trayectoria de referencia. Este proceso implica obtener el punto de destino del archivo ".txt" en la iteración indicada por el mismo, simulando así el seguimiento del "Objetivo" bajo las mismas condiciones experimentales en las que se generaron los puntos guía de la trayectoria estimada. El experimento concluye al completar las 4500 iteraciones, y



Trayectoria “D”



Trayectoria “E”



Trayectoria “F”

Figura 5.8: Trayectoria original y estimada, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: □ inicio , ⋄ fin y ∗ pausas a lo largo la trayectoria “D”, “E” y “F”. Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom.

durante cada iteración, se almacena el conjunto de datos detallado en la tabla 4.2 en un archivo ".txt".

El experimento se llevó a cabo para las trayectorias "A", "B", "C", "D", "E" y "F". Es importante destacar que estos datos se utilizaron para el entrenamiento de las versiones neuronales. La figura 5.9 muestra el seguimiento de las trayectorias estimadas para tres de los experimentos realizados.

Con la información recopilada en cada experimento, se calcularon las métricas establecidas en la sección 4.6.4. Los resultados se presentan en la tabla 5.9 y 5.10.

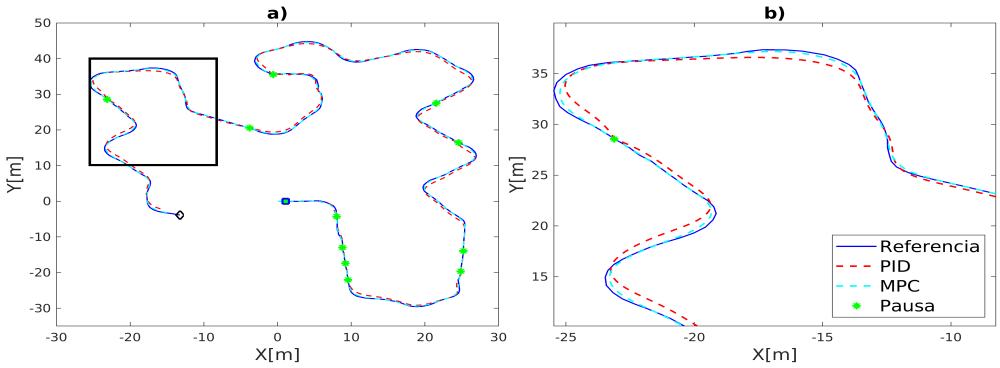
Control	**Trayectoria**	**RMSEX**	**RMSEY**	**RMSEZ**	**SSE prom.**
PID	A	1.3126	1.8653	0.0007	0.0280
	B	1.2812	1.9105	0.0006	0.0216
	C	1.4144	1.7842	0.0007	0.0100
	D	1.2417	1.9446	0.0008	0.0477
	E	1.2893	1.9708	0.0008	0.0246
	F	1.4785	1.7478	0.0007	0.0336
Promedio		$\mathbf{1{,}3362}$	$\mathbf{1{,}8705}$	$\mathbf{7{,}1666 \times 10^{-4}}$	$\mathbf{2{,}75 \times 10^{-2}}$
MPC	A	1.1224	1.2408	0.0101	0.0027
	B	1.1037	1.2690	0.0095	0.0023
	C	1.2225	1.1889	0.0098	0.0
	D	1.0614	1.3281	0.0074	0.0102
	E	1.0913	1.3084	0.0107	0.0014
	F	1.2647	1.1797	0.0093	0.0057
Promedio		$\mathbf{1{,}1443}$	$\mathbf{1{,}2524}$	$\mathbf{9{,}4666 \times 10^{-3}}$	$\mathbf{3{,}7166 \times 10^{-3}}$
Unidades		m	m	m	m

Tabla 5.9: RMSE en X, Y y Z para los controladores PID y MPC durante el seguimiento de las trayectorias A,B,C,D,E Y F.

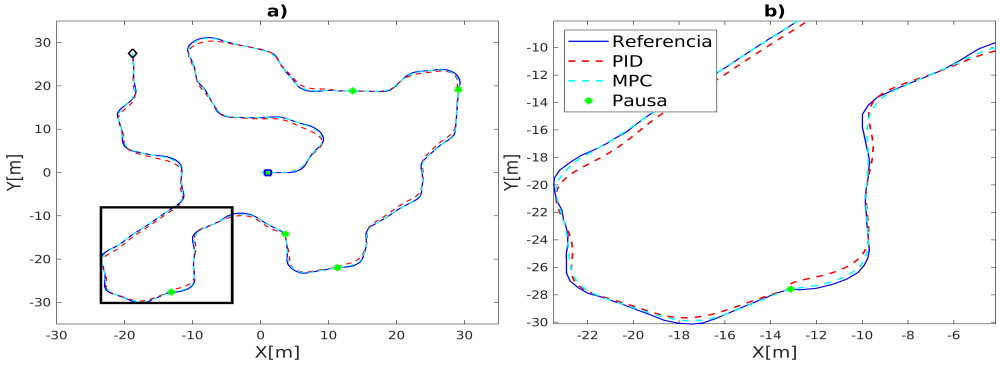
De acuerdo con los resultados obtenidos, se destaca la superioridad del MPC sobre el PID en términos de la precisión del seguimiento de la trayectoria, como evidencian los valores del RMSE promedio para las coordenadas X y Y, presentados detalladamente en la tabla 5.9. En todo momento, el MPC se posiciona más cercano al "Objetivo", llegando incluso a situarse justo encima de la posición de referencia, con una precisión promedio de $3{,}7166 \times 10^{-3}$ m cuando se encuentra en estado estacionario.

Respecto al seguimiento de la altura de referencia, se observa que el PID presenta, en promedio, un desempeño superior al MPC. No obstante, es importante señalar que los resultados obtenidos para el MPC siguen siendo favorables, manteniendo un RMSE promedio de $9{,}4666 \times 10^{-3}$.

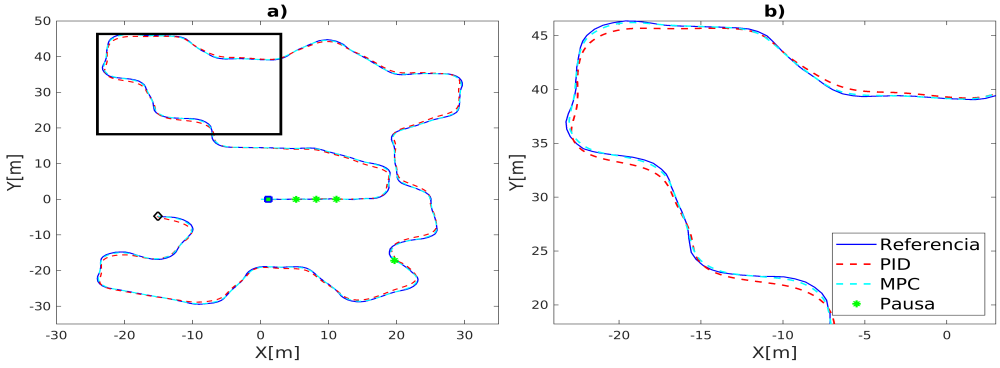
Del análisis de la tabla 5.10, se desprende que la velocidad promedio del UAV a lo largo del seguimiento de las trayectorias es de 1.4931 m/s, un valor muy cercano a



Trayectoria “D”



Trayectoria “E”



Trayectoria “F”

Figura 5.9: Seguimiento de la trayectoria “D”, “E” y “F” utilizando los controladores PID y MPC a) Trayectoria completa b) Zona de Zoom.

Control	Trayectoria	TSR	$V_{\text{prom.}}$	Frec.prom.	Frec.min.
PID	A	12.7931	1.4258	3.6014	43.3788
	B	13.6272	1.4586	3.7187	79.1139
	C	14.0480	1.5026	4.1412	180.8903
	D	18.0288	1.3527	5.0214	490.3330
	E	13.2701	1.4423	4.6863	68.1136
	F	11.5853	1.4852	4.7941	568.7958
Promedio		**13,89**	**1,4445**	**4,3271**	
MPC	A	99.3603	1.4791	0.1641	7.5280
	B	99.7995	1.4933	0.1651	9.1991
	C	100.0	1.5370	0.1743	9.2878
	D	97.8365	1.4319	0.1062	5.02688
	E	97.1563	1.4932	0.1656	7.4366
	F	98.5772	1.5243	0.1452	7.4546
Promedio		**98,7883**	**1,4931**	**0,1534**	
Unidades		%	m/s	kHz	Hz

Tabla 5.10: TSR, velocidad y frecuencia de operación de los controladores PID y MPC durante el seguimiento de las trayectorias A,B,C,D,E Y F.

la velocidad a la que se desplaza el "Objetivo" de 1.5 m/s. El TSR promedio indica que el UAV mantiene una distancia inferior a 2.5 m durante el 98.7883 % del tiempo.

Sin embargo, es importante señalar que la frecuencia promedio de cálculo de las señales de control por parte del controlador MPC es de 0.1534 kHz. Durante el seguimiento de las trayectorias, el cálculo de las señales se redujo hasta alcanzar una frecuencia mínima de 5.02 Hz (lo que equivale al 25 % de la frecuencia mínima de operación requerida), haciendo que en ocasiones el seguimiento utilizando el MPC no se pudiera realizar en "tiempo real". Sin embargo, se optó por realizar pausas en la simulación cada vez que esto ocurría y reanudar el proceso una vez que las señales de control habían sido calculadas lo que permitió obtener los resultados mostrados con anterioridad. Este aspecto contrasta con el desempeño del PID, que no experimentó esta limitación. En el siguiente enlace:Vídeos demostrativos PID y MPC pueden encontrarse algunos vídeos de ejemplo de seguimiento del "Objetivo" utilizando los controles PID y MPC.

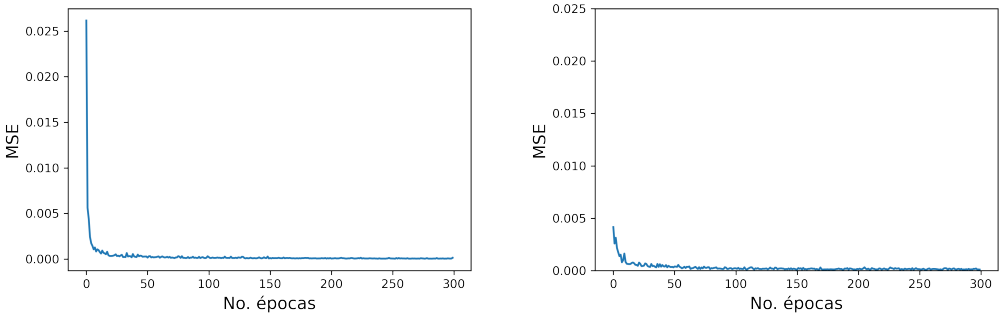
5.5.5. Control Neuronal

Utilizando los datos recopilados en los experimentos de la sección 5.5.4, se entrenan y validan los modelos N-PID y N-MPC diseñados cada uno con su respectivo conjunto de datos. El 80 % de los datos se asigna para entrenamiento, mientras que el 20 % restante

se reserva para validación los cuales serán nuestro “ground thruth”.

Entrenamiento N-PID y N-MPC

El entrenamiento se llevó a cabo según la configuración detallada en la sección 4.6.5. El Error Cuadrático Medio resultante después de 300 épocas es de 0.00013 para el N-PID y 0.00010 para el N-MPC. La figura 5.10 muestra la curva de aprendizaje de ambos modelos.

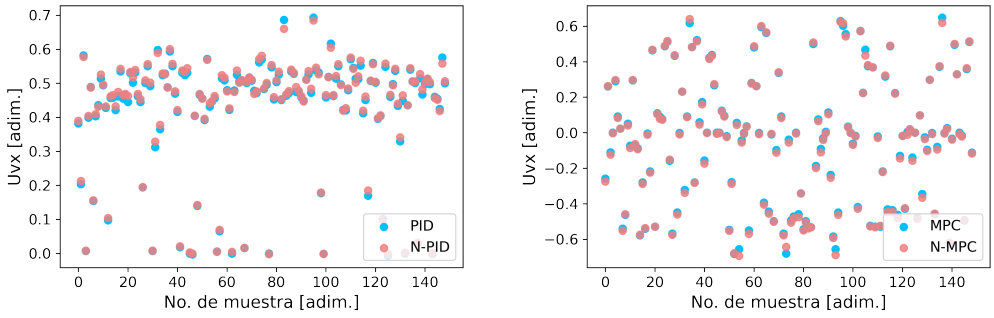


(a) (b)

Figura 5.10: Curva de aprendizaje durante 300 épocas de los modelos a) N-PID y b) N-MPC.

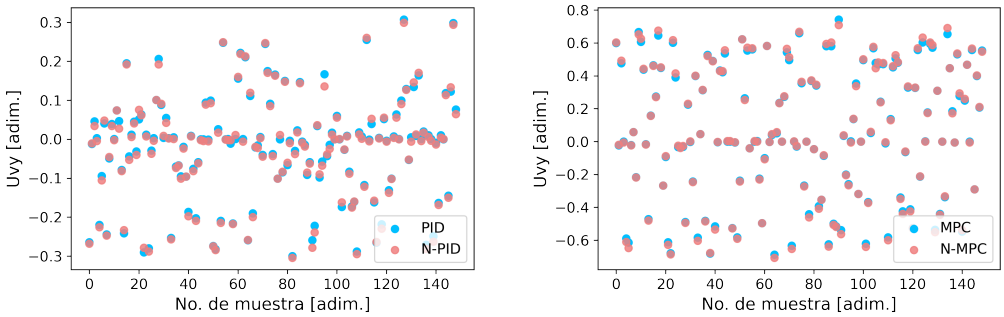
Validación N-PID y N-MPC

Una vez que ambos modelos han sido entrenados, se utiliza el conjunto de datos de validación para predecir las señales de control de este conjunto y obtener el RMSE entre las predicciones realizadas por la red y el “ground thruth”. Las figuras 5.11, 5.12 y 5.13 muestran una comparación de las predicciones realizadas con los modelos N-PID y N-MPC contra su respectivo “ground thruth” para una muestra de 150 señales de control del conjunto de validación.



(a) (b)

Figura 5.11: Comparación entre las señales de control u_{vx} predichas y el “ground truth” para una muestra de 150 datos del conjunto de validación para los modelos a) N-PID y b) N-MPC.



(a) (b)

Figura 5.12: Comparación entre las señales de control u_{vy} predichas y el "ground truth" para una muestra de 150 datos del conjunto de validación para los modelos a) N-PID y b) N-MPC.

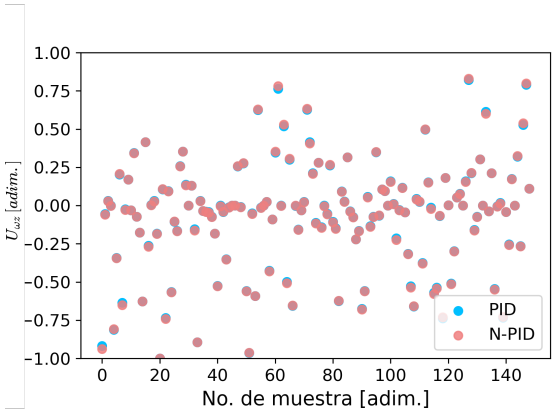


Figura 5.13: Comparación entre las señales de control $u_{\omega z}$ predichas y el "ground truth" para una muestra de 150 datos del conjunto de validación para N-PID.

Luego de calcular el RMSE entre las predicciones de la señal de control y el "ground thruth" de cada controlador, se obtuvieron los resultados que se presentan en la tabla 5.11. Los resultados son considerados satisfactorios para llevar a cabo el seguimiento.

Métrica	**Control**	$\mathbf{u_{vx}}$	$\mathbf{u_{vy}}$	$\mathbf{u_{\omega_z}}$
RMSE	N-PID	0.0068	0.0073	0.0053
	N-MPC	0.0129	0.0123	N/A
Unidades		adim.	adim.	adim.

Tabla 5.11: RMSE obtenido entre las predicciones y los el "ground thruth" del conjunto de validación para los modelos N-PID y N-MPC.

5.5.6. Diseño de trayectorias de validación de controles neuronales

Análogamente al proceso de validación de los controladores PID y MPC, se replican los experimentos realizados en la sección 5.5.3 para obtener 3 trayectorias de validación. En esta ocasión, se utiliza el controlador PID para obtener las trayectorias de referencia G y H, mientras que, la trayectoria I se obtiene utilizando el controlador N-MPC. Aunque se reportan los resultados numéricos obtenidos para cada una de estas trayectorias estimadas, por cuestiones de no extender el presente trabajo, solo se muestra la trayectoria "I" (figura 5.14), la cual muestra la trayectoria original realizada por el "Objetivo" y la trayectoria estimada por el UAV durante el experimento.

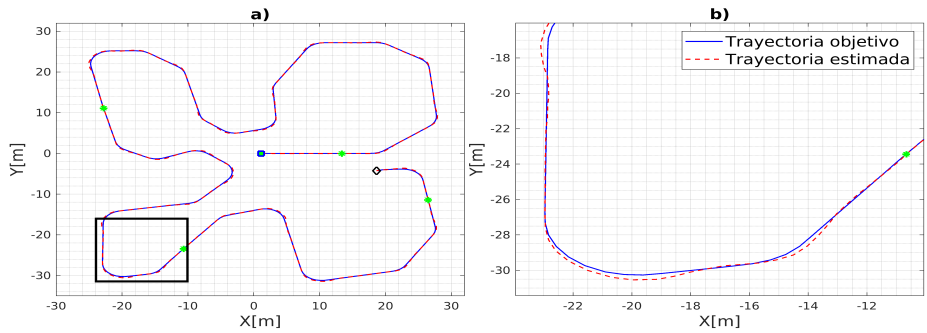


Figura 5.14: Trayectoria original y estimada, con diversas transiciones entre curvas y lineas donde se utiliza la siguiente simbología para denotar: □ inicio , ◇ fin y ∗ pausas a lo largo la trayectoria "I". Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom.

Con los datos recopilados, se determina la distancia total recorrida por el "Objetivo" a lo largo de la trayectoria y se calcula el RMSE entre las posiciones estimadas y las medidas del simulador para la trayectoria realizada. Los resultados se presentan en la tabla 5.15.

Trayectoria	**G**	**H**	**I**
RMSE [m]	0.4088	0.3918	0.2982
Longitud [m]	305.6650	281.2111	306.9196

Tabla 5.12: Longitud de las trayectorias estimadas y RMSE obtenido.

Como se observa el RMSE obtenido para la estimación de la trayectoria I a través del N-MPC es aproximadamente un 25 % menor a las obtenidas estimadas con un control PID, esto se debe a que el N-MPC no realiza movimientos en la Guiñada, y también a la naturaleza del controlador al estar basado en un control óptimo realiza movimientos con mayor precisión en comparación con el PID y el N-PID, manteniendo la cámara del UAV apuntando hacia el nadir sin que esta sufra movimientos muy bruscos.

5.6. Validación de los controles N-PID y N-MPC

La validación de los controladores implica llevar a cabo el seguimiento de las trayectorias "G", "H" e "I"utilizando los controladores N-PID y N-MPC. Sin embargo, también se realiza el seguimiento de estas trayectorias utilizando el controlador PID y MPC con el propósito de contrastar los resultados obtenidos entre cada uno de ellos. Este proceso validación se realiza replicando los experimentos descritos en la sección 5.5.4.

Con la información recopilada en cada experimento, se calcularon las métricas establecidas en la sección 4.6.4. Los resultados se presentan en la tabla 5.13 y 5.14. La figura 5.15 muestra el seguimiento de las trayectorias de referencia G, H e I en cada experimento. Por otro lado, en el siguiente enlace:Vídeos demostrativos N-PID y N-MPC pueden encontrarse algunos vídeos de ejemplo de seguimiento del "Objetivo" utilizando los controles N-PID y N-MPC.

Control	Trayectoria	RMSEX	RMSEY	SSE prom.
PID	G	1.3953	1.8682	0.0252
	H	1.3551	1.8245	0.0436
	I	1.4057	1.8551	0.0311
Promedio		**1,3853**	**1,8492**	**0,0333**
MPC	G	1.1815	1.2307	0.0011
	H	1.1607	1.2307	0.0045
	I	1.1929	1.2530	0.0002
Promedio		**1,1783**	**1,2381**	**0,0019**
N-PID	G	1.3897	1.8740	0.2093
	H	1.3605	1.8272	0.2483
	I	1.4085	1.8404	0.3405
Promedio		**1,3862**	**1,8472**	**0,2660**
N-MPC	G	1.1486	1.2016	0.0237
	H	1.1365	1.1906	0.0299
	I	1.1591	1.2162	0.0341
Promedio		**1,1480**	**1,2028**	**0,0292**
Unidades		m	m	m

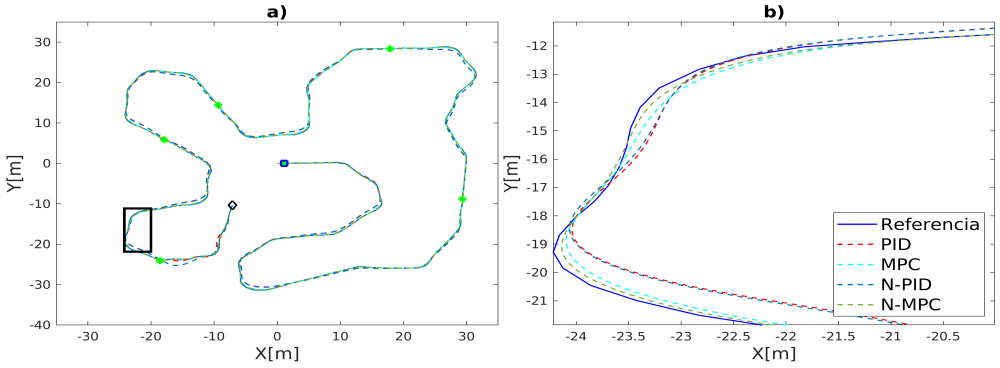
Tabla 5.13: Resultados obtenidos de error de posición y error de posición en estado estacionario para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I.

Control	Trayectoria	TSR	$V_{\text{prom.}}$	Frec.prom.	Frec.min.
PID	G	10.0456	1.4791	4.9392	461.5718
	H	17.6870	1.4351	3.2266	107.3096
	I	10.5263	1.4829	4.0155	50.3004
Promedio		**12,7529**	**1,4657**	**4,0604**	
MPC	G	93.3789	1.5249	0.1623	7.2356
	H	98.1859	1.4799	0.1583	7.4253
	I	98.3157	1.5295	0.1126	6.9320
Promedio		**96,6268**	**1,5114**	**0,1444**	
N-PID	G	11.1872	1.4796	2.1721	145.5243
	H	14.9659	1.4328	2.1721	125.3789
	I	9.6842	1.4821	1.9548	153.5249
Promedio		**11,9457**	**1,4648**	**2,0996**	
N-MPC	G	89.7260	1.5279	1.8762	46.6822
	H	94.1043	1.4894	2.1890	161.5866
	I	93.8947	1.5369	2.1794	59.7249
Promedio		**92,575**	**1,5180**	**2,0815**	
Unidades		%	m/s	kHz	Hz

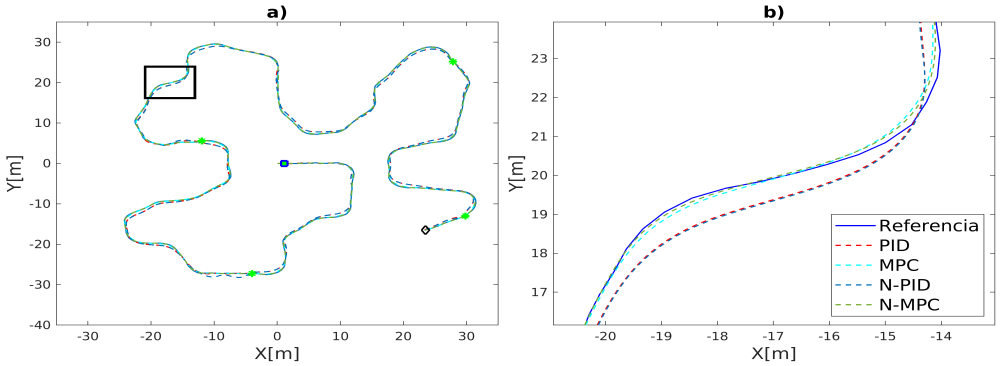
Tabla 5.14: Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I.

5.6.1. Análisis y Discusión de Resultados

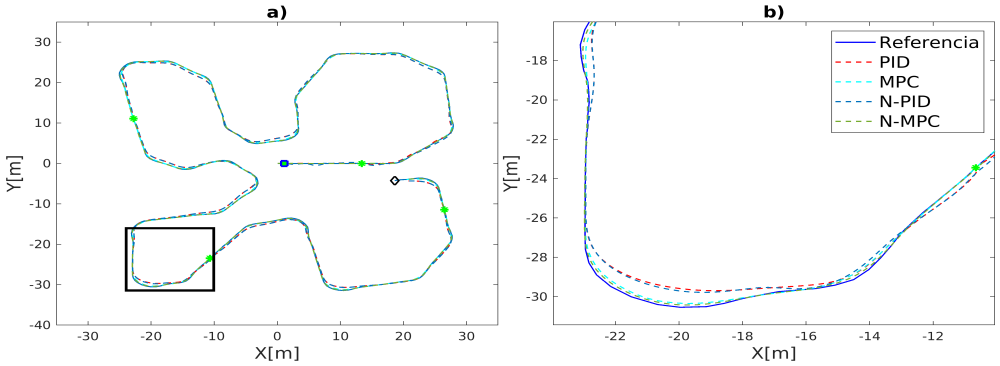
Una mejor visualización de los resultados obtenidos se muestra en los gráficos siguientes: La figura 5.16 presenta los valores de RMSE obtenidos para las coordenadas X y Y durante el seguimiento de las trayectorias de validación G, H e I. Se destaca que el controlador con el RMSE en X y Y más bajo es el N-MPC superando incluso al MPC siempre que el objetivo se encuentre en movimiento, indicando su capacidad para realizar un seguimiento preciso de la trayectoria de referencia y reproducir de manera cercana la ruta deseada. Además, se observa una tendencia donde el RMSE en la coordenada Y es inferior al de la coordenada X. Este comportamiento se explica por el hecho de que el UAV se mantiene detrás del objetivo durante el seguimiento, alineándose con la trayectoria de referencia y resultando en un RMSEY menor que el RMSEX.



Trayectoria “G”

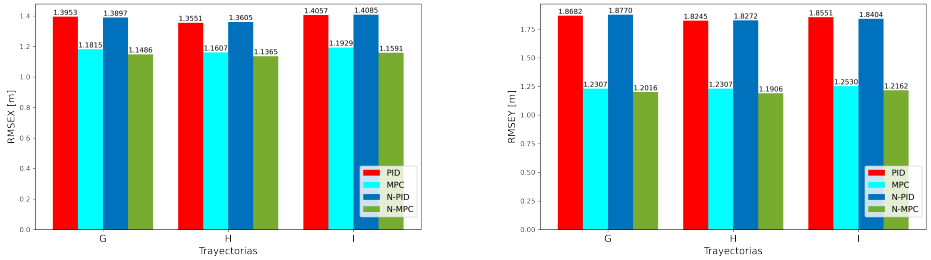


Trayectoria “H”



Trayectoria “I”

Figura 5.15: Seguimiento de la trayectoria “G”, “H” e “I” utilizando los controladores PID, MPC, N-PID, N-MPC. Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom.



(a) (b)

Figura 5.16: RMSE obtenido entre las posiciones de referencia y las alcanzadas con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I. Las visualizaciones muestran (a) RMSEX y (b) RMSEY.

La figura 5.17 representa el Error en Estado Estacionario entre la posición del UAV y el punto de referencia, cuando este último no ha experimentado cambios durante al menos 2 segundos (“pausa”). Se destaca la superioridad de los resultados obtenidos con el controlador MPC, seguido por el N-MPC. Esto sugiere que al utilizar el controlador MPC, el UAV logra posicionarse con una precisión promedio de 0.19 cm sobre el objetivo. Por otro lado, la versión neuronal del controlador MPC alcanza una precisión promedio de 2.92 cm sobre el objetivo, superando los resultados obtenidos por un controlador PID y su versión neuronal.

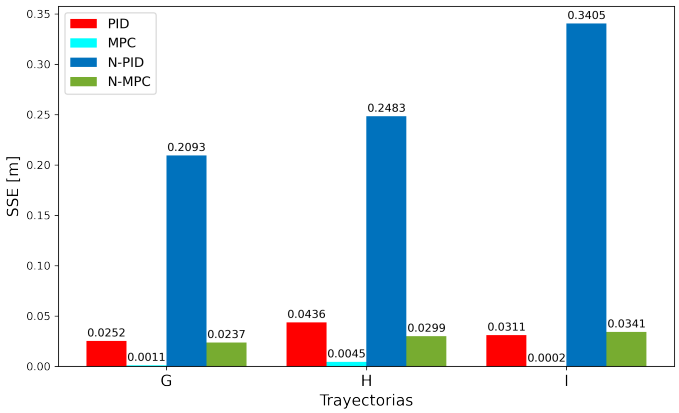


Figura 5.17: Error en Estado Estacionario promedio obtenido con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I.

Los resultados de RMSE obtenidos para el movimiento en los ejes X y Y, junto con el SSE, nos brindan información sobre las distancias entre el UAV y el objetivo a lo largo del seguimiento.

La Tasa de Seguimiento Exitoso (TSR) nos indica el porcentaje del tiempo en el

que el UAV se mantuvo dentro de un radio de proximidad al “Objetivo” menor a 2.5 metros durante el seguimiento (figura 5.18).

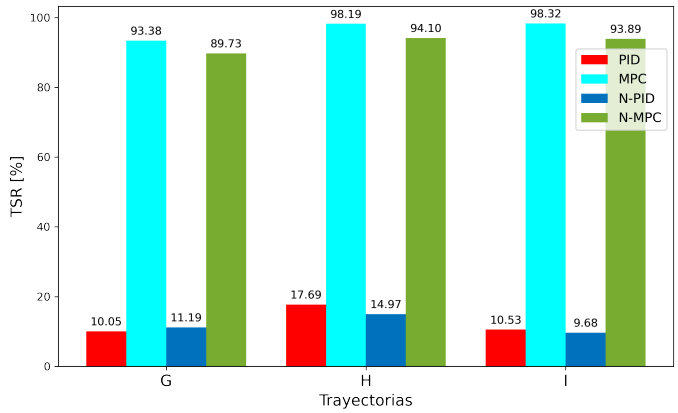


Figura 5.18: TSR obtenido para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I para un umbral de 2.5 m.

Este comportamiento se ilustra con mayor claridad al analizar el siguiente fragmento del seguimiento de la trayectoria, donde se destaca que el N-MPC logra un mejor seguimiento de la trayectoria descrita por el objetivo. A pesar de esto, su TSR promedio es del 92.575 %, inferior al del MPC. Estos resultados se explican porque el N-MPC, al realizar un seguimiento preciso, acercándose a la trayectoria, tiende a incrementar la distancia entre el UAV y el objetivo, especialmente en curvas. No obstante, ambos resultados son considerados positivos para el seguimiento de la trayectoria, ya que el MPC mantiene un TSR más alto que el N-MPC, a pesar de tener valores de RMSEX y RMSEY menores en comparación con los obtenidos con el N-MPC.

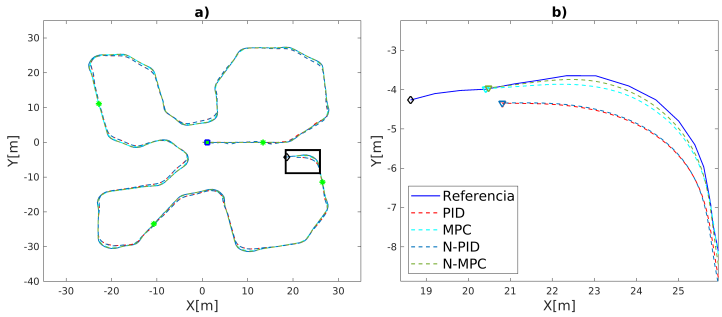


Figura 5.19: Comparación de seguimiento con los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de la trayectoria G. Las visualizaciones muestran a) El seguimiento de la trayectoria y b) Una zona con Zoom donde △ denota la posición del UAV en el mismo instante para cada controlador.

La elección de que controlador consideremos como el mejor, dependerá de la métrica que consideremos más importante: mantener el UAV y el objetivo más cercanos durante el seguimiento o ajustarse más a la trayectoria descrita por el objetivo. Sin embargo, el N-MPC logra un equilibrio entre ambas métricas, como se verá más adelante.

Para comprender mejor el TSR, consideremos el caso de la trayectoria G, que presenta el TSR más bajo para todos los controladores. En la figura 5.20, se presenta un histograma que muestra la distribución de las distancias entre el UAV y el "Objetivo" durante el seguimiento de esta trayectoria. Este histograma proporciona una representación visual de cómo varían estas distancias a lo largo del tiempo de seguimiento.

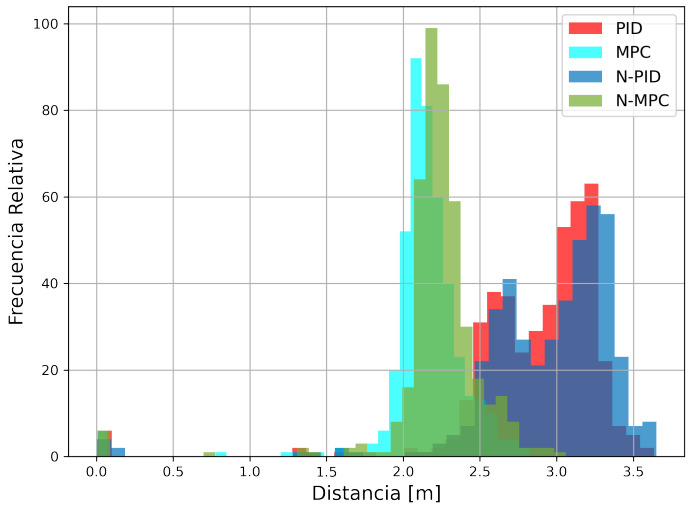


Figura 5.20: Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria G.

Como se observa, los únicos controladores que pueden garantizar mantener al objetivo a una distancia menor a 2.5 metros durante la mayor parte del recorrido de las trayectorias son el MPC y su versión neuronal.

Uno de los resultados más significativos obtenidos a lo largo de este trabajo corresponden al controlador neuronal, que proporciona resultados de desempeño muy similares al de un controlador óptimo, mejorando simultáneamente su eficiencia en términos del tiempo de cálculo de las señales de control. Como se muestra en la figura 5.21, la frecuencia de operación promedio para el PID es de 4.0604 kHz, mientras que, para el MPC es de 0.1444 kHz. Por otro lado, sus versiones neuronales mantienen una frecuencia de operación promedio de 2.0996 kHz y 2.0815 kHz, respectivamente.

Aunque podría pensarse que, en promedio, la frecuencia obtenida para el cálculo de señales de un controlador MPC es superior a los 20 Hz, existen casos en los que las señales se calculan a frecuencias inferiores a los 20 Hz, como se muestra en la

tabla 5.14. Aunque estos casos no son muy frecuentes, cuando se presentan, afectan el seguimiento, permitiendo que el objetivo escape del campo de visión de la cámara y provocando un fallo en el seguimiento por lo cual el MPC implementado no pudo realizar el seguimiento del "Objetivo" sin tener que realizar pausas durante la resolución del problema de optimización.

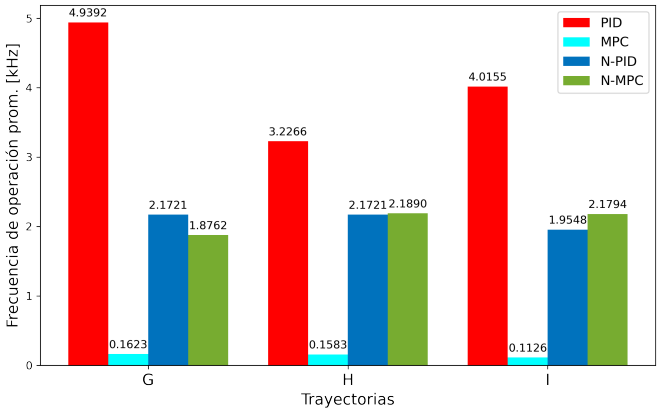


Figura 5.21: Frecuencia de operación promedio de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias G, H e I.

Estos resultados indican que al utilizar un controlador MPC neuronal se pueden obtener resultados muy buenos en términos de desempeño para una tarea de seguimiento. Además, presentan la ventaja de solucionar el problema inherente a los controladores óptimos como el MPC, que están diseñados principalmente para procesos lentos. El controlador MPC neuronal desarrollado mejora la frecuencia de operación en aproximadamente un 1343.98 % en comparación con la frecuencia de operación promedio obtenida para el controlador MPC implementado.

5.6.2. Desempeño del N-MPC para otros escenarios

Una vez validados los controladores N-PID y N-MPC, se llevaron a cabo dos pruebas de seguimiento del objetivo utilizando el N-MPC para determinar la velocidad máxima a la cual el controlador podría realizar el seguimiento desde una altura de 4 metros. En la primera prueba, se obtuvo la trayectoria "J", con el objetivo moviéndose a una velocidad de hasta 2.4 m/s. En la segunda prueba, a una altura de 5 metros, se obtuvo la trayectoria "K", con el objetivo moviéndose a una velocidad de hasta 2.75 m/s, dichas trayectorias se muestran mas adelante durante la etapa de seguimiento.

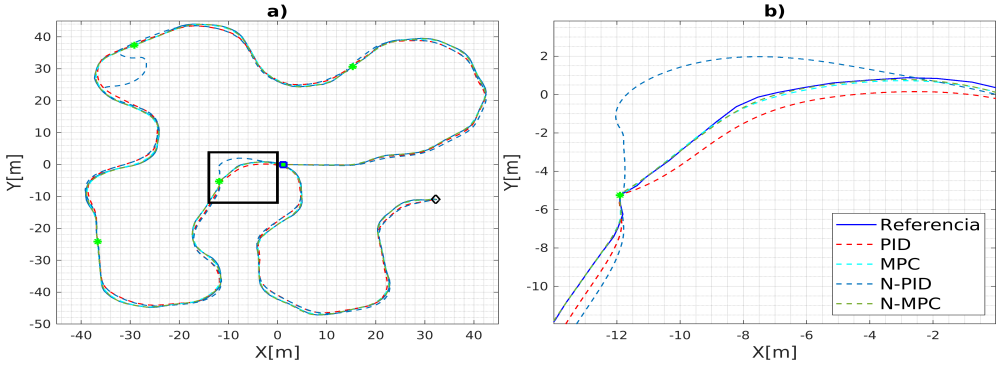
Los resultados para la distancia total recorrida por el "Objetivo" a lo largo de la trayectoria y el RMSE entre las posiciones estimadas y las medidas del simulador para la trayectoria realizada se presentan en la tabla 5.15.

Trayectoria	J	K
RMSE [m]	0.6010	0.6402
Longitud [m]	438.0240	420.3837

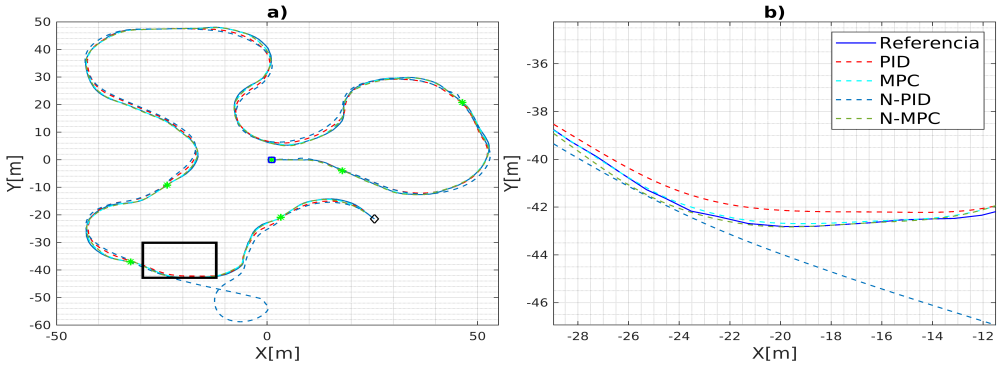
Tabla 5.15: Longitud de las trayectorias estimadas y RMSE obtenido.

En este escenario, al aumentar la velocidad del objetivo, la distancia entre el UAV y objetivo también se incrementa, resultando en un crecimiento del error de estimación. No obstante, se puede afirmar en base a los experimentos realizados que este error es menor que el que se obtendría al realizar la estimación mediante el control PID para el seguimiento a estas velocidades.

Se realizo el seguimiento de estas trayectorias con cada uno de los controladores desarrollados en este trabajo. La figura 5.22 muestra el seguimiento de las trayectorias de referencia J y K en cada experimento.



Trayectoria J.



Trayectoria K.

Figura 5.22: Seguimiento de las trayectorias J y K utilizando los controladores PID, MPC, N-PID, N-MPC. Las visualizaciones incluyen a) la trayectoria completa y b) una zona ampliada con zoom.

Con la información recopilada en cada experimento, se calcularon las métricas establecidas en la sección 4.6.4. Los resultados se presentan en la tabla 5.16 y 5.17.

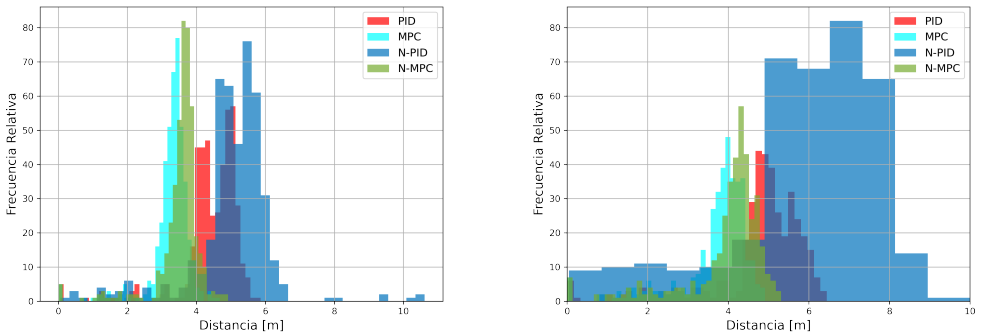
Control	Trayectoria	RMSEX	RMSEY	SSE prom.
PID	J	2.1101	3.0144	0.0241
	K	2.5631	3.0174	0.0571
MPC	J	1.8129	2.02952	0.0001
	K	2.2062	2.1118	0.0041
N-PID	J	2.7367	3.4295	1.1898
	K	5.5425	5.3460	5.2289
N-MPC	J	1.8335	2.0183	0.0459
	K	2.2882	2.1321	0.0364
Unidades		m	m	m

Tabla 5.16: Resultados obtenidos de error de posición y error de posición en estado estacionario para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias J y K

Control	Trayectoria	TSR	$V_{\text{prom.}}$	Frec.prom.	Frec.min.
PID	J	4.7727	2.3141	3.9868	51.4499
	K	8.0103	2.4590	5.1027	50.4535
MPC	J	5.4545	2.3744	0.0940	9.0962
	K	11.8863	2.5393	0.1618	36.6397
N-PID	J	5.0	2.3615	2.2189	10.7114
	K	8.0103	2.4897	2.2938	68.2566
N-MPC	J	5.6818	2.3859	2.1508	129.5018
	K	10.0775	2.5359	2.2780	37.2787
Unidades		%	m/s	kHz	Hz

Tabla 5.17: Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias J y K.

La figura 5.23 muestra los histogramas de distancia entre el UAV y el Objetivo a lo largo del seguimiento de estas trayectorias. Por otro lado, en el siguiente enlace:Vídeos demostrativos N-PID y N-MPC a 2.4 m/s y 2.75 m/s pueden encontrarse algunos vídeos de ejemplo de seguimiento del “Objetivo” utilizando los controles N-PID y N-MPC.



(a) (b)

Figura 5.23: Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria (a) J y (b) K.

De acuerdo con los resultados obtenidos, se observa que el SSE para los controladores PID, MPC y N-MPC se mantiene en un valor aceptable, manteniendo el mismo comportamiento que los obtenidos a una velocidad de 1.5 m/s. Sin embargo esto no ocurre para el N-PID. Como lo evidencia el seguimiento de la trayectoria J, cuando el objetivo se mueve a una velocidad de 2.4 m/s, tanto los controladores MPC como N-MPC permanecen la mayor parte del tiempo a una distancia inferior a 4 metros (aun manteniendo al objetivo dentro del campo de visión). En el caso de la trayectoria K, cuando el objetivo se desplaza a una velocidad de 2.75 m/s, la distancia entre el UAV y el objetivo aumenta hasta aproximadamente 4.5 metros para estos controladores. Esto explica el comportamiento del RMSE en X y Y aumenta y por lo tanto el TSR obtenido disminuye considerando un umbral de 2.5 metros.

No obstante, los resultados obtenidos para el MPC y el N-MPC se consideran exitosos, ya que logran mantener al objetivo dentro del campo de visión (FoV) durante todo el seguimiento. En contraste, para los controladores PID, donde esta distancia se mantiene más allá de este rango saliendo en ocasiones del FoV de la cámara, y para el caso del N-PID, la distancia aumenta aún más, llegando incluso a fallar en el seguimiento de la trayectoria.

5.7. Implementación Física

Se llevó a cabo la implementación física con el propósito de validar la metodología propuesta para el desarrollo de un sistema de control neuronal destinado al seguimiento de objetos móviles. En esta etapa, se implementaron los algoritmos PID[2] y MPC para obtener sus versiones neuronales N-PID y N-MPC respectivamente. A continuación,

[2]Las ganancias utilizadas en la plataforma física fueron; para x: $k_p = 0{,}55$, $k_d = 15$, $k_i = 0{,}00015$. Para y: $k_p = 0{,}55$, $k_d = 15$, $k_i = 0{,}00015$. Para z: $k_p = 1{,}0$, $k_d = 5{,}32$,$k_i = 0{,}00015$. Para ψ: $k_p = 1{,}4$, $k_d = 0{,}15$, $k_i = 0{,}0001$.

se detallan las condiciones experimentales que incluyen hardware, software y entorno utilizado en este proceso.

Consideraciones de Hardware: Se empleó el UAV Bebop 2.0 de Parrot y un vehículo terrestre en configuración diferencial con una velocidad máxima de 0.5 m/s. Controlado mediante comunicación Bluetooth. Se utilizó un Sistema Vicon compuesto por 5 cámaras infrarrojas para registrar el movimiento traslacional (X, Y, Z) del UAV, su orientación (ψ), así como la posición tridimensional del vehículo terrestre, todo ello a una frecuencia máxima de 100 Hz. Como estación terrena, se empleó una laptop equipada con 500 GB SSD, 16 GB de RAM, con una tarjeta de vídeo NVIDIA GTX 1650 Ti. Esta laptop ejecutó los algoritmos previamente desarrollados. La comunicación entre la estación terrena y el UAV se realizó vía ROS a través de conexión WiFi. A su vez, esta estación se enlazó con el servidor del sistema Vicon a través de Ethernet para obtener la posición del UAV y el vehículo terrestre en el área de experimentación.

Consideraciones de Software: La figura 5.24 ilustra la integración del sistema mencionado anteriormente con ROS, a través del cual se obtiene la posición y orientación del UAV, así como la posición del objetivo. La estación terrena opera bajo Ubuntu 20.04 con ROS Noetic y Gazebo 11, junto con los paquetes $bebop\_autonomy$ y $vicon\_bridge$.

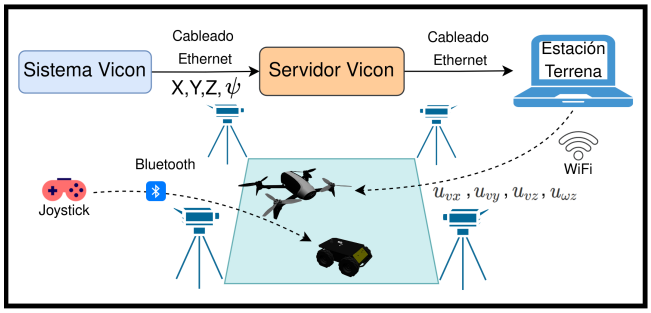


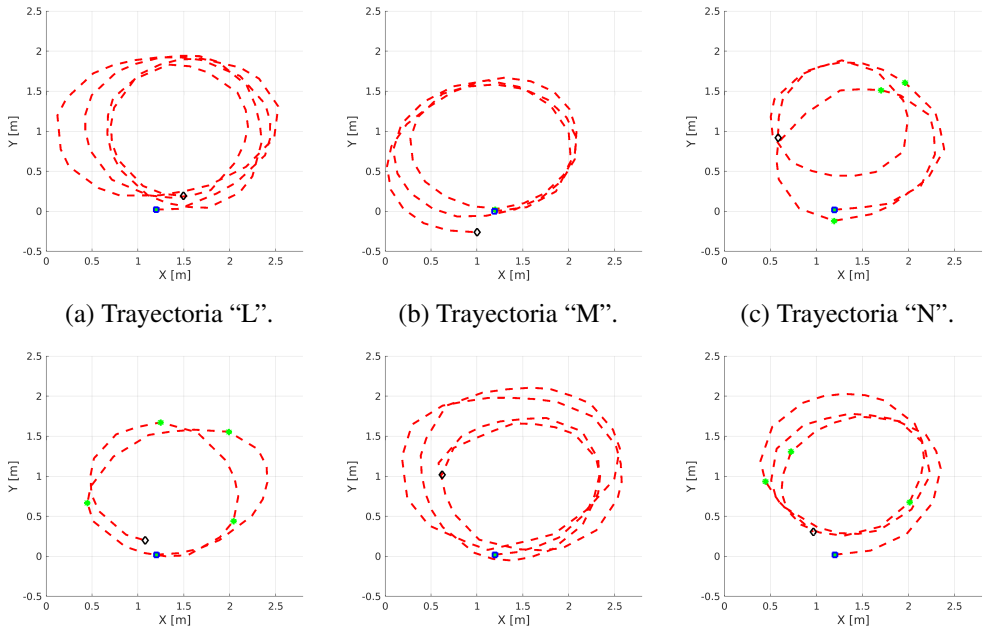
Figura 5.24: Configuración del sistema para la implementación física de los controladores.

Condiciones Experimentales: Los experimentos se llevaron a cabo en un área de 2.5 x 2.5 metros, con el UAV volando a una altura de 1.0 metro sobre la superficie.

Limitaciones: Debido a la falta de un control preciso del movimiento del vehículo terrestre, y las dificultades encontradas para trazar trayectorias suaves (sin cambios abruptos de orientación en puntos contiguos de la trayectoria) al utilizar este vehículo. Se tomó la decisión de eliminar el grado de libertad de orientación (ψ) del UAV durante el seguimiento con los controles PID y N-PID. Para mitigar este problema, se simplificaron las trayectorias eliminando la capacidad de que el UAV cambiara de orientación durante el seguimiento. Además, considerando la altura de vuelo y otros factores relevantes, se optó por prescindir del sistema de estimación de posición del sistema de percepción, en favor de obtener directamente la posición del sistema Vicon. Esta elección se fundamentó en la reconocida precisión y confiabilidad del sistema Vicon en

entornos controlados, esencial dada la necesidad de un control preciso del UAV en un espacio reducido.

Bajo estas premisas, se condujo el robot terrestre manualmente en el área experimental, trazando un total de 6 trayectorias. Después de realizar el seguimiento a través de los controladores PID y MPC de las trayectorias L, M, N y O (vea figura 5.25). Se recopiló la información necesaria para entrenar los modelos N-PID y N-MPC. Por otro lado, las trayectorias P y Q se reservaron específicamente para validar los modelos entrenados. La información relacionada a las trayectorias obtenidas se encuentra en la tabla 5.18.



(a) Trayectoria “L”. (b) Trayectoria “M”. (c) Trayectoria “N”.

(d) Trayectoria “O”. (e) Trayectoria “P”. (f) Trayectoria “Q”.

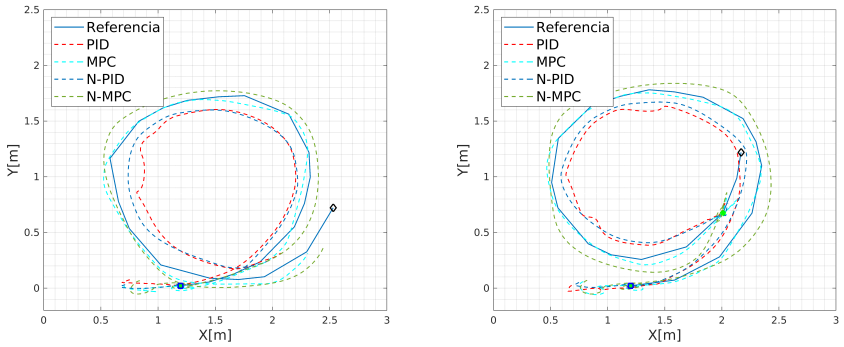
Figura 5.25: Trayectorias L, M , N y O utilizadas para el entrenamiento de los modelos N-PID y N-MPC. T rayectorias P y Q utilizadas para la validación del seguimiento. Se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella.

Trayectoria	**L**	**M**	**N**	**O**	**P**	**Q**
Longitud [m]	23.4167	16.8797	14.4579	10.4143	23.0619	15.6192
Aplicación	•	•	•	•	✓	✓

Tabla 5.18: Longitud de las trayectorias empleadas en la implementación física. Se utiliza el símbolo • para indicar que la trayectoria fue utilizada en el entrenamiento del N-PID y N-MPC y el símbolo ✓ para indicar que la trayectoria fue utiliza en la validación del seguimiento utilizando estos modelos.

Los resultados del seguimiento de estas trayectorias con los modelos entrenados

se presentan en la figura 5.26. Además, se llevaron a cabo múltiples seguimientos utilizando el robot objetivo para verificar su repetibilidad. En el siguiente enlace:Vídeos demostrativos PID, MPC y N-MPC pueden encontrarse algunos vídeos de ejemplo de seguimiento del "Objetivo" utilizando los controles PID, MPC y N-MPC implementados en la plataforma física.



(a) Trayectoria "L". (b) Trayectoria "M".

Figura 5.26: Seguimiento de las trayectorias de validación utilizando los controladores PID, MPC, N-PID y N-MPC. Se utiliza la siguiente simbología para denotar: - - la trayectoria, □ su inicio , ◇ fin y ∗ pausas a lo largo de ella.

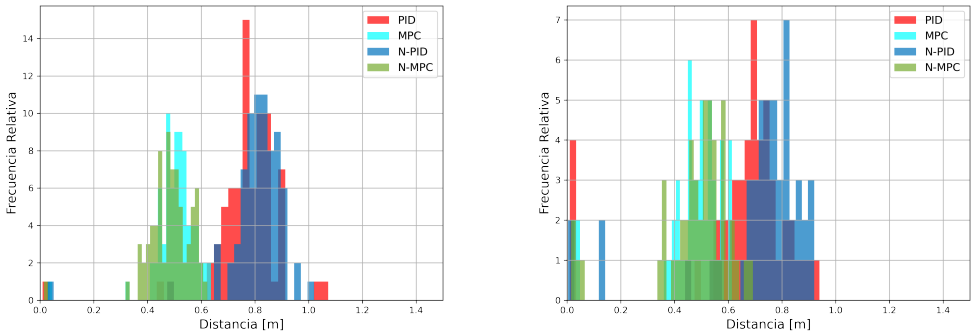
Por otro lado, las tablas 5.19 y 5.20 presentan los resultados obtenidos durante el seguimiento de las trayectorias de validación. Mientras que, la figura 5.27 muestra sus respectivos histogramas del TSR para un umbral de 0.75 m. Este umbral es menor al utilizado en simulación debido principalmente a la velocidad con la que se mueve el "Objetivo" en la plataforma física.

Control	**Trayectoria**	**RMSEX**	**RMSEY**	**SSE prom.**
PID	P	0.3999	0.3904	0.0100
	Q	0.3206	0.3364	0.0237
MPC	P	0.2299	0.2128	0.0294
	Q	0.2068	0.1910	0.0299
N-PID	P	0.4194	0.4049	0.0262
	Q	0.3683	0.3565	0.0763
N-MPC	P	0.1858	0.1645	0.0145
	Q	0.1996	0.1654	0.0345
Unidades		m	m	m

Tabla 5.19: Resultados obtenidos de RMSEX, RMSEY y SSE para los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias P y Q.

Control	Trayectoria	TSR	$V_{\text{prom.}}$	Frec.prom.	Frec.min.
PID	P	30.6666	0.4650	8.7293	1161.8585
	Q	75.5102	0.3808	8.2665	775.5725
MPC	P	100.0	0.5488	0.1518	5.7443
	Q	100.0	0.5019	0.1736	5.7585
N-PID	P	14.6666	0.4735	4.5643	1172.9104
	Q	44.8979	0.3716	4.4684	1581.5527
N-MPC	P	100.0	0.5794	4.3643	734.4246
	Q	100.0	0.5124	4.4547	1967.3034
Unidades		%	m/s	kHz	Hz

Tabla 5.20: Resultados de TSR, velocidad promedio del UAV, frecuencia de operación del controlador y frecuencia mínima de los controladores PID, MPC, N-PID y N-MPC durante el seguimiento de las trayectorias P y Q.



(a) (b)

Figura 5.27: Histograma de distancia entre el UAV y el "Objetivo" durante el seguimiento de la trayectoria (a) P y (b) Q.

Como se observa, el UAV se mantiene en todo momento a una distancia inferior a 0.75 m para los controladores MPC y N-MPC. Los resultados anteriores respaldan la efectividad de la metodología propuesta para desarrollar controladores neuronales y su aplicación para el seguimiento de objetivos móviles. Estos resultados muestran que es posible lograr un desempeño similar al de un controlador óptimo en el seguimiento de trayectorias, y además, mejorando la velocidad de cálculo de las señales de control. Esta mejora demuestra la aplicabilidad de la metodología propuesta para tareas que implican el control en procesos con dinámicas rápidas, en los que la implementación de un controlador óptimo podría resultar complicado o incluso, inviable debido a los requisitos de tiempo de respuesta exigidos.

Capítulo 6

Conclusiones y Trabajo Futuro

Se ha diseñado un sistema autónomo para el seguimiento de objetivos móviles, compuesto por un módulo de percepción, estimación de la posición del objetivo, así como otro dedicado al control de seguimiento de trayectoria. Los resultados obtenidos proporcionan una visión integral del comportamiento del sistema. Sin embargo, durante el proceso de diseño y desarrollo se evaluó individualmente cada uno de ellos.

En simulación, las arquitecturas de control desarrolladas son capaces de predecir señales de control para el movimiento horizontal, calculando las señales de control u_{vx} y u_{vy} en el caso del N-MPC, y también $u_{\omega z}$ para el caso del N-PID . Sin embargo, en la plataforma física se implementó el mismo modelo de red para ambos controladores debido a las condiciones experimentales. Estas predicciones se calculan tomando en cuenta algunos de los estados actuales del UAV y el historial de los últimos 5 puntos en la trayectoria descrita por el objetivo.

En simulación, cada punto en esta trayectoria se obtiene a través de un módulo encargado de estimar la posición tridimensional del objetivo a partir del *bounding box* proporcionado por el módulo de percepción, cuya función es detectar el objetivo. A diferencia de la simulación, en la plataforma experimental la posición del objetivo se obtiene directamente del sistema Vicon. Debido principalmente a la altura de vuelo del UAV. En ambos casos, así se genera el historial de puntos conforme el objetivo avanza, permitiendo a las arquitecturas de control realizar el ajuste en el movimiento horizontal y de guiñada del UAV.

Destaca el desempeño del modelo N-MPC para el seguimiento de la trayectoria del objeto móvil en el plano XY. Los resultados indican que este controlador presenta un rendimiento comparable al de un MPC diseñado para el control de movimiento horizontal, tal como lo demuestran el RMSE en los ejes X y Y, el SSE y el TSR.

A diferencia del MPC implementado en simulación, que enfrentó dificultades para realizar el seguimiento del objetivo en "tiempo real", requiriendo pausas en la simulación cada vez que se resolvía el problema de optimización, el N-MPC presenta una mejora significativa en su velocidad de operación. Esta mejora posibilita el seguimiento en "tiempo real" sin interrupciones. Además, esta mayor eficiencia en la velocidad de operación permite que un controlador de este tipo sea aplicable en plataformas con re-

cursos computacionales limitados y baja capacidad de procesamiento. Mostrando una alternativa para el control de dinámicas donde un MPC convencional no sería factible, debido a su alto costo computacional y a los requisitos de velocidad de respuesta exigentes.

Debido al escenario planteado para el seguimiento del objetivo, en el cual el UAV se mantiene a una altura constante durante el seguimiento. Se optó por controlar la altura a través de un control PID, debido a su eficiencia y rapidez en este tipo de situaciones. Aunque los resultados reportados se obtuvieron para un seguimiento a una altura de 4 metros, es importante destacar que esta altura puede ajustarse durante el seguimiento según las necesidades y que las condiciones experimentales lo permitan.

Los controladores fueron originalmente diseñados para un escenario en el que el objetivo se desplaza a una velocidad máxima de 1.5 m/s. Sin embargo, con el fin de evaluar la capacidad de adaptación y eficacia de los controladores, se realizaron pruebas de seguimiento en simulación bajo condiciones más exigentes, principalmente con el N-MPC. Durante estas pruebas, tanto los controladores sintonizados como los neuronales fueron sometidos a seguimientos cuando el objetivo se movía a velocidades superiores a las previamente establecidas. Los resultados demostraron que el sistema pudo seguir satisfactoriamente al objetivo, incluso cuando este alcanzaba velocidades de hasta 2.4 m/s con el UAV volando a una altura de 4 m, y de hasta 2.75 m/s cuando el UAV volaba a 5 m. Es importante destacar que en todas las pruebas realizadas, el objetivo permaneció dentro del campo de visión en todo momento para el control N-MPC, confirmando así la efectividad del seguimiento incluso en situaciones más desafiantes.

Aunque los resultados reportados se basan en un conjunto de trayectorias de validación, se llevaron a cabo múltiples seguimientos del objetivo y de trayectorias utilizando las velocidades mencionadas anteriormente para verificar la repetibilidad de los resultados obtenidos principalmente con el N-MPC.

A diferencia de algunos enfoques presentados en el estado del arte, hemos abordado un escenario específico en el cual el UAV es capaz de detenerse manteniéndose en vuelo estacionario una vez que haya alcanzado al objetivo, reanudando el seguimiento una vez que el objetivo se desplaza de nuevo.

Limitaciones

Debido a la falta de implementación de un control de seguimiento de trayectoria para el robot HUSKY, este se conduce siempre de forma manual. En consecuencia, al realizar la estimación de la trayectoria mediante la cámara del UAV en simulación, esta depende directamente del controlador de movimiento del UAV. Por ende, no es posible replicar esta trayectoria con el vehículo terrestre y comparar cómo sería la trayectoria estimada al utilizar otro controlador de seguimiento para el UAV.

Como se menciono anteriormente, las trayectorias de entrenamiento en simulación fueron obtenidas a través de conducir de forma manual el objetivo. Se buscó que las trayectorias contaran con curvas en “U”, giros de 90 grados, tramos rectos y movimientos en forma de “S”, incluso incorporando pausas. Sin embargo, en la plata-

forma física la variedad y suavidad de estas trayectorias se vieron limitadas debido al vehículo de control remoto utilizado. Es importante destacar que la fidelidad de estas características depende en gran medida de la experiencia del conductor para realizar estos movimientos y a la precisión de los giros del vehículo seleccionado. En consecuencia, el controlador funcionará de manera óptima para trayectorias muy similares a las utilizadas en el entrenamiento pero no se garantiza que lo haga para movimientos no contemplados en estas trayectorias.

En los escenarios de simulación que involucran el seguimiento a velocidades superiores a 1.5 m/s, y/o con movimientos más complejos, el N-MPC demostró una notable capacidad de adaptación. Logrando un seguimiento exitoso incluso en situaciones no previamente contempladas durante el entrenamiento, como situaciones en las que el objetivo se mueve incluso en reversa para el caso de simulación. Este éxito se atribuye principalmente a la estabilidad mantenida por el UAV al mantener su orientación siempre constante, a diferencia del N-PID, que intenta corregir su orientación mientras se desplaza simultáneamente en el plano XY. En contraste, en algunos de estos casos el N-PID mostró limitaciones llegando incluso experimentar fallos durante el seguimiento. Específicamente, el N-PID puede mostrar debilidades en comparación con el N-MPC cuando se enfrenta a movimientos que no se ajustan a las condiciones específicas del entrenamiento. Cabe destacar que, para movimientos extremadamente complejos, existe la posibilidad de que el rendimiento del N-MPC pueda disminuir en eficiencia e incluso llegar a experimentar errores durante el seguimiento.

La elección de la altura de seguimiento se fundamenta principalmente en el porcentaje de confianza en la detección, el error de estimación de posición, y el área percibida por la cámara. Se llevaron a cabo pruebas en simulación para evaluar la estimación de posición y percepción a alturas de 3, 4 y 5 metros, lo que respalda la viabilidad de realizar seguimientos a estas alturas. Sin embargo, es esencial tener en cuenta que alturas inferiores a 3 metros pueden impactar la visibilidad de la cámara, aumentando el error de estimación y requiriendo velocidades de seguimiento por debajo de 1.5 m/s para evitar que el objetivo salga del campo de visión.

En contraste, alturas superiores también presentan limitaciones, ya que existe un umbral crítico tanto para la velocidad de seguimiento como para el error de estimación, en el cual el módulo de estimación de posición y percepción ya no puedan calcular la posición con precisión. Estos límites son esenciales para garantizar un seguimiento efectivo y preciso en diversas alturas, estableciendo así las condiciones óptimas para el funcionamiento del sistema.

Trabajo futuro

En términos de futuras líneas de investigación, se realizaron pruebas en un entorno simulado utilizando un modelo de curvas de nivel de la Luna, el cual presenta una superficie de exploración no completamente plana, lo cual podría afectar el desempeño del sistema de estimación de posición. Aunque los resultados del seguimiento fueron en su mayoría exitosos, estas pruebas no se incluyeron en los resultados de esta tesis.

Esto señala una oportunidad valiosa para futuras investigaciones, en las que incluso se podrían utilizar los controladores desarrollados para explorar escenarios no convencionales y desafiantes, que empleen curvas de nivel de la superficie marciana para validar y ajustar el sistema propuesto. La metodología presentada en este trabajo puede servir como una guía valiosa para realizar modificaciones efectivas en este contexto.

Se exploró la integración de los comandos de altitud y guiñada en un único modelo de red neuronal. Sin embargo, se identificó que esta inclusión generaba ruido y afectaba la precisión del cálculo de las señales de control. Como futuro desarrollo, se sugiere considerar la implementación de subredes independientes que operen en paralelo, permitiendo así obtener con mayor precisión los comandos de control restantes con el fin de complementar de manera efectiva al N-MPC existente, fortaleciendo el rendimiento general del sistema.

La implementación de un sistema de control de seguimiento para el robot terrestre no solo contribuiría a una definición más precisa del conjunto de entrenamiento, permitiendo su expansión para abordar movimientos no contemplados, sino también mejoraría la robustez del N-MPC en entornos más complejos. El implementar este controlador, posibilitaría además el seguimiento del objetivo a través del UAV utilizando cada uno de los controladores desarrollados, evaluando así el comportamiento de la estimación de posición y trayectoria obtenida durante el seguimiento del objetivo para cada controlador de vuelo.

A diferencia de los resultados obtenidos en simulación, en la que el área de experimentación se desarrollo en un espacio de 50 x 50 m y el conjunto de datos de entrenamiento fue de 27000. En la plataforma física, el entrenamiento se realizó con 4000 datos. Aunque se obtuvieron resultados favorables para este tipo de trayectorias, en un futuro el sistema podría entrenarse para trayectorias mas complejas aumentando el volumen de datos de entrenamiento.

Finalmente, el MPC implementado para la plataforma física utiliza el modelo dinámico del Bebop 2.0 obtenido en la simulación. Sin embargo, podrían obtenerse mejores resultados identificando a la planta física utilizando la metodología utilizada en este trabajo y sintonizar el MPC con el modelo dinámico real de la planta.

Bibliografía

Almozel, A. (2020). *Competitive Drone Racing Using Game Theory* [Tesis doctoral].

Balaram, J., Aung, M., & Golombek, M. P. (2021). The ingenuity helicopter on the perseverance rover. *Space Science Reviews*, *217*(4), 56.

Cai, P., Sun, Y., Chen, Y., & Liu, M. (2019). Vision-based trajectory planning via imitation learning for autonomous vehicles. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2736-2742.

Camacho, E. F., & Bordons, C. (2007). Model Predictive Controllers. En *Model Predictive control* (pp. 376-377). Springer London. https://doi.org/10.1007/978-0-85729-398-5_2

Camacho, E. F., Bordons, C., Camacho, E. F., & Bordons, C. (2007). *Model predictive controllers*. Springer.

Gama, K. P., Hu, C., & Hao, W. (2022). Visual Tracking Based Deep Learning and Control Design Onboard Small-Sized Quadrotor UAV. *2022 41st Chinese Control Conference (CCC)*, 5865-5870.

Gardner, M. W., & Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, *32*(14-15), 2627-2636.

Giernacki, W., Kozierski, P., Michalski, J., Retinger, M., Madonski, R., & Campoy, P. (2020). Bebop 2 quadrotor as a platform for research and education in robotics and control engineering. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1733-1741.

Green, J. L. (2021). Perseverance Rover and Its Search for Life On Mars. *Communications of the Byurakan Astrophysical Observatory*, *68*(2), 464-469.

Grip, H. F., Conway, D., Lam, J., Williams, N., Golombek, M. P., Brockers, R., Mischna, M., & Cacan, M. R. (2022). Flying a helicopter on Mars: How ingenuity's flights were planned, executed, and analyzed. *2022 IEEE Aerospace Conference (AERO)*, 1-17.

Huang, Z.-Y., & Lai, Y.-C. (2020). Image-based sense and avoid of small scale UAV using deep learning approach. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*, 545-550.

Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, *31*(3), 685-695.

Kamel, M., Stastny, T., Alexis, K., & Siegwart, R. (2017). Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. *Robot Operating System (ROS) The Complete Reference (Volume 2)*, 3-39.

Ketkar, N., & Moolayil, J. (2021). *Deep learning with Python: learn best practices of deep learning models with PyTorch*. Springer.

Kim, Y., & Bang, H. (2018). Introduction to Kalman filter and its applications. *Introduction and Implementations of the Kalman Filter*, *1*, 1-16.

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, *3*, 2149-2154.

Li, M., Cai, Z., Zhao, J., Wang, Y., Wang, Y., & Lu, K. (2021). MNNMs Integrated Control for UAV Autonomous Tracking Randomly Moving Target Based on Learning Method. *Sensors*, *21*(21), 7307.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740-755.

Martinez-Carranza, J., & Rojas-Perez, L. O. (2022). Warehouse inspection with an autonomous micro air vehicle. *Unmanned Systems*, *10*(04), 329-342.

Matus-Vargas, A., Rodriguez-Gomez, G., Martinez-Carranza, J., & Munoz-Silva, A. (2017). Numerical optimization techniques for nonlinear quadrotor control. *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1309-1315.

Mercado-Ravell, D. A., Castillo, P., & Lozano, R. (2019). Visual detection and tracking with UAVs, following a mobile object. *Advanced Robotics*, *33*(7-8), 388-402.

Muller, M., Li, G., Casser, V., Smith, N., Michels, D. L., & Ghanem, B. (2019). Learning a controller fusion network by online trajectory filtering for vision-based uav racing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 0–0.

Naskath, J., Sivakamasundari, G., & Begum, A. A. S. (2023). A study on different deep learning algorithms used in deep neural nets: MLP SOM and DBN. *Wireless Personal Communications*, *128*(4), 2913-2936.

Nguyen, H. T., Prasad, N. R., Walker, C. L., & Walker, E. A. (2002). *A first course in fuzzy and neural control*. CRC press.

Okasha, M., Kralev, J., & Islam, M. (2022). Design and Experimental Comparison of PID, LQR and MPC Stabilizing Controllers for Parrot Mambo Mini-Drone. *Aerospace*, *9*(6), 298.

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. *2017 15th international conference on ICT and knowledge engineering (ICT&KE)*, 1-6.

Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach*. .ºReilly Media, Inc.".

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A. Y., et al. (2009). ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, *3*(3.2), 5.

Sa, I., Kamel, M., Khanna, R., Popovic, M., Nieto, J., & Siegwart, R. (2017). Dynamic System Identification, and Control for a cost effective open-source VTOL MAV. *arXiv preprint arXiv:1701.08623*.

Salinas, L. R., Slawiñski, E., & Mut, V. A. (2014). Kinematic Nonlinear Controller for a Miniature Helicopter via L yapunov Techniques. *Asian Journal of Control*, *16*(3), 856-870.

Shinde, P. P., & Shah, S. (2018). A review of machine learning and deep learning applications. *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, 1-6.

Simon, D. (2001). Kalman filtering. *Embedded systems programming*, *14*(6), 72-79.

Szczerbicki, E. (2009). *Intelligent systems for knowledge management* (Vol. 252). Springer.

Taud, H., & Mas, J. (2018). Multilayer Perceptron (MLP). En M. T. Camacho Olmedo, M. Paegelow, J.-F. Mas & F. Escobar (Eds.), *Geomatic Approaches for Modeling Land Change Scenarios* (pp. 451-455). Springer International Publishing. https://doi.org/10.1007/978-3-319-60801-3_27

Tian, X., Jia, Y., Luo, X., & Yin, J. (2022). Small Target Recognition and Tracking Based on UAV Platform. *Sensors*, *22*(17), 6579.

Tzanetos, T., Aung, M., Balaram, J., Grip, H. F., Karras, J. T., Canham, T. K., Kubiak, G., Anderson, J., Merewether, G., Starch, M., et al. (2022). Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout. *2022 IEEE Aerospace Conference (AERO)*, 01-19.

von Ehrenfried, M. '. (2022). Ingenuity. En *Perseverance and the Mars 2020 Mission: Follow the Science to Jezero Crater* (pp. 111-125). Springer International Publishing. https://doi.org/10.1007/978-3-030-92118-7_6

Welch, G., Bishop, G., et al. (1995). An introduction to the Kalman filter.

Wu, D., Zhu, H., & Lan, Y. (2022). A Method for Designated Target Anti-Interference Tracking Combining YOLOv5 and SiamRPN for UAV Tracking and Landing Control. *Remote Sensing*, *14*(12), 2825.

Xie, J., Peng, X., Wang, H., Niu, W., & Zheng, X. (2020). Uav autonomous tracking and landing based on deep reinforcement learning strategy. *Sensors*, *20*(19), 5630.

Zhang, J., Chen, T., & Shi, Z. (2020). A Real-Time Visual Tracking for Unmanned Aerial Vehicles with Dynamic Window. *2020 China Semiconductor Technology International Conference (CSTIC)*, 1-3.

Zhilenkov, A. A., & Epifantsev, I. R. (2018). The use of convolution artificial neural networks for drones autonomous trajectory planning. *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EICon-Rus)*, 1044-1047.