UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

CENTRO DE INVESTIGACIÓN EN TECNOLOGÍAS DE INFORMACIÓN Y SISTEMAS

MAESTRÍA EN CIENCIAS EN AUTOMATIZACIÓN Y CONTROL

IMPLEMENTACIÓN DE UN ALGORITMO DE VISIÓN ARTIFICIAL BASADO EN ORB-SLAM PARA UN HELICÓPTERO MINIATURA DE CUATRO ROTORES.

T E S I S

QUE PARA OBTENER EL GRADO DE MAESTRO EN CIENCIAS EN AUTOMATIZACIÓN Y CONTROL

PRESENTA:

IDAEL OLIVA LEAL

DIRECTORES DE TESIS:

DR. JORGE SAID CERVANTES ROJAS

DR. IVÁN GONZÁLEZ HERNÁNDEZ

MINERAL DE LA REFORMA HGO., MÉXICO 1 DE DICIEMBRE DE 2020

Idael Oliva Leal

Implementación de un algoritmo de visión artificial basado en ORB-SLAM para un helicóptero miniatura de cuatro rotores.

A mis padres, hermano, y en especial a mi abuelo,
quien hizo posible se hiciera realidad este sueño.
A todos ellos por el apoyo incondicional aún desde la distancia.
No solo por este trabajo, sino por estar siempre en mi vida. GRACIAS.

Gracias a

CONACYT, por la beca de Maestría otorgada durante el periodo Agosto 2018 - Julio 2020.

CITIS-UAEH, por la formación académica brindada durante el posgrado y el apoyo para la realización de la presente tesis.

Agradecimientos

A mi abuelo Alberto, por brindarme el apoyo económico necesario para poder dar este paso tan importante en mi vida y realizar este posgrado en el extranjero.

A mis padres Yuderkis e Idael, por confiar en mi y apoyarme en este decisión de seguir adelante con mi superación y futuro aunque fuese lejos de ellos.

A mi hermano Laidel, por siempre estar ahí para sacarme una sonrisa y ser mi mejor amigo.

A toda mi familia, por apoyarme de una manera u otra a lo largo de mi vida y ser mi ejemplo y motivación.

A mis asesores de tesis el Dr. Jorge Said Cervantes Rojas y el Dr. Iván González Hernández, por su paciencia y dedicación durante todo el proceso de esta investigación, por brindarme sus conocimientos y materiales necesarios.

A mis sinodales, Dr. Hugo Romero Trejo, Dr. Carlos Cuvas Castillo, Dra. Liliam Rodríguez Guerrero, por sus comentarios y recomendaciones que hicieron posible la realización de la tesis.

A mis compañeros y amigos de la MCAC-UAEH, Daniel, Juan Pablo, Jhonatan, Emmanuel, Oziel y Alejandro, por compartir muchas experiencias personales y profesionales a lo largo de estos años.

A los coordinadores de la MCAC-UAEH Dr. Jesús Patricio Ordaz Oliver y Dra. Liliam Rodríguez Guerrero, además a mi asesor Dr. Jorge Said Cervantes Rojas, por apoyarme en todo momento con todo el proceso y trámites necesarios para poder venir a este hermosos país y realizar este posgrado.

Al CONACyT, por el apoyo económico.

A México y su gente, por darme la bienvenida con su hermosa cultura y costumbres.

A todos, mis más sinceros agradecimientos, simplemente, GRACIAS.

UNIVERSIDAD AUTÓNOMA DEL ESTADO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences
Área Académica de Computación y Electrónica
Department of Electronics and Computer Science

Mineral de la Reforma, Hgo., a 19 de noviembre de 2020

Número de control: ICBI-AACyE/1213/2020
Asunto: Autorización de impresión de tesis.

M. EN C. JULIO CÉSAR LEINES MEDÉCIGO
DIRECTOR DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

Por este conducto le comunico que el comité revisor asignado al Ing. Idael Oliva Leal, alumno de la Maestría en Ciencias en Automatización y Control, autoriza la impresión del proyecto de tesis titulado "Implementación de un algoritmo de visión artificial basado en ORB-SLAM para un helicóptero miniatura de cuatro rotores", bajo la dirección del Dr. Jorge Said Cervantes Rojas y codirección del Dr. Iván González Hernández, en virtud de que se han efectuado las revisiones y correcciones pertinentes.

A continuación, se integran las firmas de conformidad de los integrantes del jurado.

Presidente:	Dr. Hugo Romero Trejo	UAEH
Secretario:	Dr. Carlos Cuvas Castillo	UAEH
Vocal 1:	Dr. Jorge Said Cervantes Rojas	UAEH
Suplente 1:	Dra. Liliam Rodríguez Guerrero	UAEH
Suplente 2:	Dr. Iván González Hernández	CINVESTAV

Atentamente
"Amor. Orden y Progreso"

Dra. Liliam Rodríguez Guerrero
Coordinadora de la Maestría en Ciencias
en Automatización y Control

LRG/APL

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo km 4.5 Colonia
Carboneras, Mineral de la Reforma, Hidalgo,
México. C.P. 42184
Teléfono: +52 (771) 71 720 00 ext. 2250, 2251
Fax 2109
aacye_icbi@uaeh.edu.mx

www.uaeh.edu.mx

Índice general

Índice general I
Índice de figuras V
Índice de tablas IX
Acrónimos XI
Glosario XIV
Resumen XV
Abstract XVII

1. Introducción **1**
1.1. Planteamiento del problema 3
1.2. Justificación 3
1.3. Hipótesis 3
1.4. Objetivo general 3
1.5. Objetivos particulares 4
1.6. Metodología 4
1.7. Conclusiones del capítulo 5

2. Marco Teórico **7**
2.1. Conceptos de vehículos aéreos. 7
2.1.1. Aplicaciones de los VANTs 7
2.1.2. Clasificación de los VANTs 8
2.2. Sistema Operativo Robótico ROS 10
2.2.1. Estructura de ROS 10
2.2.1.1. Nodos 11
2.2.1.2. Tópicos 11
2.2.1.3. Mensajes 11
2.3. Localización y mapeo simultáneo (SLAM) 12
2.3.1. Estructura y formulación del problema de SLAM 13
2.3.2. Puntos de interés 16
2.3.3. Sensores utilizados en el algoritmo SLAM. 17
2.3.4. Principales algoritmos SLAM existentes 17

2.4. Conclusiones del capítulo 18

3. Algoritmo ORB-SLAM2 **19**
3.1. Módulo de seguimiento 21
3.2. Módulo de mapeo local 24
3.3. Módulo de trayectorias cerradas y BA completo 26
3.4. Gráfico de Covisibilidad y Gráfico Esencial 27
3.5. Reconocimiento del lugar 27
3.6. Conclusiones del capítulo 28

4. Plataforma Experimental **29**
4.1. Sistema de propulsión 30
4.2. Sistema de alimentación 31
4.3. Variadores de velocidad 32
4.4. Controlador de vuelo 32
4.5. Sistema GPS 33
4.6. Radio-receptor 33
4.7. Radio-control 33
4.8. Computadora embebida 34
4.9. Sistema de percepción visual 34
4.10. Conclusiones del capítulo 35

5. Resultados **37**
5.1. Simulación de un controlador PD para la estabilización 37
5.2. Simulación de un controlador PD para seguimiento de trayectorias . . . 41
5.3. Simulación de un controlador no lineal por saturaciones anidadas para la estabilización 43
5.4. Simulación de un controlador no lineal por saturaciones anidadas para seguimiento de trayectorias 46
5.5. Funcionamiento del algoritmo ORB-SLAM2 en una computadora portátil 49
5.6. Comunicación serial entre el Pixhawk y la computadora Jetson Nano 55
5.7. Funcionamiento del algoritmo de ORB-SLAM2 en la computadora embebida Jetson Nano 58
5.8. Control para la estabilización del vehículo 60
5.9. Vuelo autónomo estacionario del vehículo mediante un controlador PD 62
5.10. Vuelo autónomo de seguimiento de trayectorias del vehículo mediante un controlador PD 66
5.11. Conclusiones del capítulo 68

6. Conclusiones y trabajo futuro **69**
6.1. Trabajo futuro 69

Bibliografía **71**

A. Modelado matemático **77**
A.1. Características del vehículo 77
A.2. Modelado dinámico 79

Índice de figuras

2.1. Cuadricóptero en configuración X. 10
2.2. Diagrama de funcionamiento de ROS. 11
2.3. Problema principal del SLAM. Se requiere una estimación simultánea de las ubicaciones del robot y de los puntos de interés. Las ubicaciones reales nunca se conocen ni se miden directamente. Las observaciones se realizan entre el robot y los puntos de interés. 13
2.4. Esquema funcional del algoritmo SLAM. 16

3.1. Diagrama general del algoritmo ORB-SLAM2. 21
3.2. Preprocesamiento de la entrada. 22

4.1. Plataforma experimental. Vistas superior (izquierda) y frontal (derecha). 29
4.2. Modelo CAD de la plataforma experimental en Solid Works. Vistas superior (arriba) y frontal (abajo). 30
4.3. a) Motor Turnigy Multistar 4108-600. b) Batería Graphene Lipo. c) Afro 30A Muti-Rotor ESC. d) Pixhawk versión 2.4.8. e) GPS 3DR uBlox. f) Spektrum AR-8000. g) Jetson Nano. h) Cámara estereoscópica ZED. i) Spektrum DX8. 31

5.1. Posiciones x, y y z respectivamente. Estabilización con controladores PD. 40
5.2. Ángulos de Euler θ, ϕ y ψ respectivamente. Estabilización con controladores PD. 40
5.3. Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Estabilización con controladores PD. 41
5.4. Posiciones x, y y z respectivamente. Seguimiento de trayectorias con controladores PD. 42
5.5. Ángulos de Euler θ, ϕ y ψ respectivamente. Seguimiento de trayectorias con controladores PD. 42
5.6. Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Seguimiento de trayectorias con controladores PD. 43

5.7. Posiciones x, y y z respectivamente. Estabilización con controladores por saturaciones anidadas. 45
5.8. Ángulos de Euler θ, ϕ y ψ respectivamente. Estabilización con controladores por saturaciones anidadas. 45
5.9. Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Estabilización con controladores por saturaciones anidadas. 46
5.10. Diagrama de bloques realizado para la simulación. 47
5.11. Posiciones x, y y z respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas. 48
5.12. Ángulos de Euler θ, ϕ y ψ respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas. 48
5.13. Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas. 49
5.14. Diagrama de comunicación de los nodos en ROS. 51
5.15. Croquis del CITIS y recorrido realizado. 53
5.16. Diagrama del experimento realizado. 53
5.17. Puntos del mapa del fotograma actual. 54
5.18. Vistas del mapa creado del entorno. 55
5.19. Pose obtenida en un punto del mapa. 56
5.22. Trama de datos. 56
5.20. Diagrama de conexión. 57
5.21. Conexión física de la plataforma. 58
5.23. Diagrama de comunicación nivel software. 58
5.24. Posiciones x, y y plano $x - y$, respectivamente. 59
5.25. Diagrama que representa el experimento. 61
5.26. Ángulos de Euler θ, ϕ y ψ, respectivamente. 61
5.27. Señales de control τ_θ, τ_ϕ y τ_ψ, respectivamente. 62
5.28. Ángulos de Euler para un vuelo estacionario. 63
5.29. Señales de control τ_ϕ, τ_θ y τ_ψ respectivamente para un vuelo estacionario. 63
5.30. Posiciones x, y y z respectivamente para un vuelo estacionario. 64
5.31. Señales de control u_x, u_y y u_z respectivamente para un vuelo estacionario. 64
5.32. a) características ORB detectadas en un fotograma, b) mapa creado y pose (vista desde el vehículo), c) vista en tres dimensiones, d) vista superior. 65
5.33. Ángulos de Euler para seguimiento de trayectoria. 66
5.34. Señales de control τ_ϕ, τ_θ y τ_ψ respectivamente para seguimiento de trayectoria. 67
5.36. Señales de control u_x, u_y y u_z respectivamente para seguimiento de trayectoria. 67
5.35. Posiciones x, y y z respectivamente para seguimiento de trayectoria. . 68

A.1. Representación del vector de coordenadas generalizadas, los ángulos de Euler, el esquema de fuerzas y pares para el vehículo aéreo. 79

Índice de tablas

2.1. Clasificación de los VANTs [1]. 9

5.1. Ganancias sintonizadas de los controladores. 60

Acrónimos

VANT: Vehículo Aéreo no Tripulado.

UAV: Vehículo Aéreo no Tripulado (Unmanned Aerial Vehicle por sus siglas en inglés).

ROS: Sistema Operativo Robótico (Robot Operating System por sus siglas en inglés).

SDK: Kit de desarrollo de software (Software Development Kits por sus siglas en inglés).

GPS: Sistema de Posicionamiento Global (Global Positioning System por sus siglas en inglés).

FPS: Fotogramas por segundo.

SLAM: Localización y Mapeo Simultáneo (Simultaneous Localization and Mapping por sus siglas en inglés).

AI: Inteligencia Artificial (Artificial Intelligence por sus siglas en inglés).

RGB-D camera: Cámara con sensor de profundida (Red Green Blue - Depth camera por sus siglas en inglés).

IMU: Unidad de Mediciones Inerciales (Inertial Measurement Unit por sus siglas en inglés).

CCD: Dispositivo de Carga Acoplada (Charge Coupled Device por sus siglas en inglés).

CUDA: Arquitectura Unificada de Dispositivos de Cómputo (Compute Unified Device Architecture por sus siglas en inglés).

GPU: Unidad de Procesamiento Gráfico (Graphics Processing Unit por sus siglas en inglés).

EKF: Filtro de Kalman Extendido (Extended Kalman Filter por sus siglas en inglés)

UART: Transmisor-Receptor Asíncrono Universal (Universal Asynchronous Receiver-Transmitter por sus siglas en inglés)

Glosario

- **Visión Artificial.** Es una disciplina científica que incluye métodos para adquirir, procesar y analizar imágenes del mundo real con el fin de producir información que pueda ser tratada por una máquina.
- **Sensor.** Un sensor es un dispositivo que está capacitado para detectar acciones o estímulos externos y responder en consecuencia.
- **Firmware.** Conjunto de instrucciones de un programa informático que se encuentra registrado en una memoria ROM.
- **Sensor Ultrasónico.** Dispositivo que mide la altura del suelo en un rango de 5 metros por medio de un pulso a altas frecuencias (Frecuencia 40Khz).
- **Giroscopio.** Dispositivo electrónico que mide los movimientos de un dispositivo con un brazo de accionamiento.
- **Acelerómetro.** Dispositivo electrónico responsable de detectar los cambios de orientación.
- **Brújula Digital.** Dispositivo electrónico que permite señalizar nuestra orientación respecto al campo magnético terrestre.
- **Cabeceo.** El eje lateral o transversal es un eje imaginario que se extiende de punta a punta de las alas del avión.
- **Alabeo.** El eje longitudinal es un eje imaginario que se extiende desde el frente a la cola del avión.
- **Guiñada.** El eje vertical es un eje imaginario que, pasando por el centro de gravedad del avión, es perpendicular a los ejes transversal y longitudinal. Este eje está contenido en un plano que pasa por el centro de gravedad desde arriba hacia abajo.
- **Linux™.** Es un sistema operativo libre tipo Unix, multiplataforma, multiusuario y multitarea.
- **Tipo UNIX.** Es un sistema que se comporta de manera similar a un sistema Unix, aunque no es necesario que sea certificado en ninguna versión.
- **Código Abierto.** El código abierto (Open source por sus siglas en inglés) es un modelo de desarrollo de software basado en la colaboración abierta, se enfoca más en los beneficios prácticos de la colaboración.

- **Rosbuild.** Antiguo compilador de ROS.
- **Variables de entorno.** Cadenas que contienen información acerca del entorno para el sistema y el usuario que ha iniciado sesión en ese momento.
- **Interprete de Código.** Es un programa que ejecuta scripts escritos en un lenguaje interpretado como Python y/o Java.
- **Script.** Es una serie de comandos que se almacenan en un archivo de texto y los cuales se caracterizan por presentar un tamaño muy pequeño.
- **Cámara RGB-D.** Es una cámara que cuenta con un sensor de profundidad.
- **Cámara Stereo.** Es una cámara estereoscópica capaz de capturar imágenes (fotografías) en tres dimensiones.
- **Odometría.** Es el estudio de la estimación de la posición de vehículos durante la navegación. Este término también se usa a veces para referirse a la distancia que ha recorrido uno de estos vehículos (pudiéndose emplear otros sensores para su cálculo, como la odometría visual).
- **CUDA.** Arquitectura Unificada de Dispositivos de Cómputo (Compute Unified Device Architecture, por sus siglas en inglés) hace referencia a una plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por nVidia que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU de nVidia.
- **Wrapper.** Es una subrutina en una librería de software que funciona como un adaptador entre un sistema operativo y un controlador, como un controlador de dispositivo, que no fue diseñado para ese sistema operativo.
- **GPU.** Unidad de Procesamiento Gráfico (Graphics Processing Unit por sus siglas en inglés) es un coprocesador dedicado al procesamiento de gráficos u operaciones de coma flotante, para aligerar la carga de trabajo del procesador central en aplicaciones como los videojuegos o aplicaciones 3D interactivas.
- **rviz.** Es una herramienta de visualización en 3D para aplicaciones de ROS. Proporciona una vista del modelo de robot, captura la información de los sensores del robot y reproduce los datos capturados.

Resumen

El presente tema de tesis está dirigido a resolver el problema de la implementación de un algoritmo de visión artificial basado en ORB-SLAM2 para la construcción de mapas de un entorno desconocido y localización de un vehículo aéreo dentro del mismo de manera simultánea, percibido mediante una cámara estereoscópica. Además, se resolverá el control de movimiento del vehículo usando la localización proporcionada por el algoritmo de Localización y Mapeo Simultáneo (SLAM, del inglés, Simultaneous Localization and Mapping). Las técnicas de control que se usarán serán controladores convencionales como lo son Proporcionales-Derivativos PD.

Abstract

This thesis topic is aimed at solving the problem of the implementation of an artificial vision algorithm based on ORB-SLAM2 for the construction of maps of an unknown environment and the location of an air vehicle within it simultaneously, perceived through a stereoscopic camera. In addition, the vehicle movement control will be solved using the location provided by the Simultaneous Localization and Mapping (SLAM) algorithm. The control techniques that will be used will be conventional controllers such as Proportional-Derivative PD.

Capítulo 1

Introducción

En el presente capítulo se define la justificación de la tesis, se plantea el problema a abordar, así como la hipótesis de la investigación. Además, se plasma el objetivo general y los particulares.

Dotar de mayor autonomía a los vehículos aéreos no tripulados (VANTs) ha sido motivo de investigación en trabajos recientes, dado el gran impacto que tienen en la actual sociedad. Para lograr la correcta autonomía del vehículo se deben tener en cuenta aspectos muy importantes como son: la estimación de la orientación y posición (seis grados de libertad).

Mediante el uso de sistemas de posicionamiento global (GPS, del inglés, Global Positioning System) y unidades de medición inercial (IMU, del inglés, Inertial Measurement Unit), se ha resuelto de manera significativa dicho problema en ambientes exteriores. Pero en el caso de ambientes desconocidos, desordenados, y sin señal de GPS, aún siguen representando un desafío considerable. Aunque el problema de estimar la orientación de los VANTs ha sido abordado y resuelto satisfactoriamente [2], la estimación de la posición basada en GPS todavía presenta algunas desventajas. Es decir, se considera que la tecnología GPS no constituye un servicio totalmente confiable, y su disponibilidad puede estar limitada en los denominados cañones urbanos (ciudades con edificaciones altas que pueden interferir en la recepción de la señal del GPS) y es completamente inaccesible en ambientes interiores. Además, aún cuando la señal del GPS está disponible, el problema de la estimación de la posición no puede llevarse a cabo en escenarios, en donde se plantea, por ejemplo, la inspección aérea en plantas industriales en donde se requiere desarrollar maniobras precisas en un ambiente complejo. En este caso, y debido a diversas fuentes de error, la posición obtenida con un GPS puede variar con un error de varios metros en pocos segundos cuando se lleva a cabo una localización estática [3].

Además, es importante destacar, que el uso de las lecturas del GPS, como la señal de realimentación de un sistema de control, puede ser poco fiable porque el sistema de

control no puede distinguir entre ruido del sensor y pequeños movimientos reales del vehículo. Por lo tanto, información sensorial adicional (e.g., información visual) debe ser integrada en el sistema para mejorar la exactitud.

Los problemas mencionados anteriormente se han atacado mediante el uso de cámaras para realizar navegación basada en visión en situaciones cuando la señal del sensor de posición no está disponible, parcialmente disponible, o cuando la navegación local requiere de una alta precisión.

Dada las ventajas del uso de las cámaras, tales como: ser de bajo costo, de peso ligero y de bajo consumo de energía, han podido ser adaptadas en sistemas embebidos. En este sentido, una combinación de visión y mediciones inerciales es implementada frecuentemente para la estimación de la orientación y la posición del vehículo. Esta combinación puede abordarse usando diferentes enfoques, como en [4], en donde la medición de la visión es proporcionada por un sistema externo que mide la trayectoria, obteniéndose directamente la posición y orientación del robot. En [5], una cámara externa como dispositivo de carga acoplada (CCD, del inglés, Charge Coupled Device) es la encargada de proporcionar las mediciones.

Otras técnicas implementadas a bordo del vehículo fueron propuestas en [6] y [7], en donde una cámara embebida usa diferentes marcadores para proporcionar una buena estimación de la posición y orientación. Esta estimación fue obtenida usando la geometría específica de diferentes marcadores y asumiendo que la posición de los marcadores era conocida. La misma idea fue explotada en [8], en donde se implementó un sensor visual remoto Nintendo Wii de bajo costo.

Finalmente, el sensado visual es proporcionado algunas veces por sensores de flujo óptico para estimar la orientación, la posición, y la velocidad, como en [9]. En todos estos enfoques, la estimación de la posición se obtiene usando visión por computadora y la orientación se obtiene también usando visión, ver [4] y [7], o por sensores IMU. Igualmente, se pueden utilizar otros enfoques similares (por ejemplo [10], [11]) que depende de métodos visuales de localización y mapeo simultáneo (SLAM, del inglés, Simultaneous Localization and Mapping). En este caso, el robot móvil opera en un ambiente desconocido a priori usando solo sensores abordo para construir un mapa de sus alrededores y simultáneamente localizarse dentro de este mapa. Los sensores del robot tienen un gran impacto sobre el algoritmo usado en el SLAM. Los primeros trabajos que usan SLAM se enfocan en el uso de sensores de rango, tal como sonares y láseres, ver [12]. Sin embargo, algunas desventajas aparecen cuando se usan sensores de rango en SLAM; esto es, la correspondencia y asociación de datos se vuelve una tarea difícil, los sensores son caros, llegando a costar hasta 5000 pesos mexicanos y tienen un rango de trabajo limitado, algunos solamente de 35 a 250 mm, y algunos de ellos se limitan a aplicaciones de mapeos en dos dimensiones (2D). Para vehículos aéreos no tripulados, existen varias limitaciones que consideran el diseño de la plataforma, movilidad, y capacidad de carga que impone restricciones considerables. Una vez

más, las cámaras representan una buena opción para usarse en los sistemas de SLAM aplicados en VANTs.

1.1. Planteamiento del problema

La localización mediante GPS de vehículos no tripulados, en especial helicópteros de varios rotores, cuando realizan vuelos en exteriores, puede verse deteriorada debido a la presencia de obstáculos como edificios, árboles, vehículos, etc. o también a condiciones climáticas como nubosidad. Pero cuando se trata de interiores es imposible contar con la misma. Por lo tanto, realizar tareas de vuelo autónomas como el seguimiento de una trayectoria de referencia, resultan difícil de resolver utilizando solo los sensores más comunes a bordo del vehículo.

1.2. Justificación

Las tareas de seguimiento autónomas representan una de las problemáticas con gran vigencia hoy en día, sobre todo en ambientes no estructurados y con señales de posición GPS inaccesibles o deficientes. Por lo anterior, es necesario la implementación de sistemas de localización empotrados en el helicóptero de varios rotores, uno de los métodos que han mostrado ser efectivos para resolver este problema son los sistemas de mapeo y localización simultánea, específicamente el algoritmo de ORB-SLAM2 (ORB [13], del inglés Oriented FAST and Rotated BRIEF), que se basa en procesos que requieren un menor costo computacional. Una vez conocida la localización del vehículo utilizando el algoritmo de ORB-SLAM2, es posible implementar leyes de control para realizar tareas de seguimiento autónomas. Además, este trabajo de tesis pretende ser la base para plantear problemas más desafiantes como tareas de navegación autónoma.

1.3. Hipótesis

Aplicando el algoritmo de ORB-SLAM2 podrá realizarse un mapeo de un área desconocida y localización simultánea en el mismo de un helicóptero de cuatro rotores para realizar vuelo estacionario y de seguimiento de trayectorias.

1.4. Objetivo general

Implementar un sistema de localización y mapeo simultáneo a abordo de un helicóptero miniatura de cuatro rotores, utilizando el algoritmo de ORB-SLAM2, para realizar tareas de seguimiento autónomas.

1.5. Objetivos particulares

- Revisar el estado del arte relacionado con los vehículos aéreos, algoritmos de visión, y técnicas de SLAM de manera sistemática para crear las bases teóricas de la presente investigación.
- Estudiar el modelo matemático del helicóptero de cuatro rotores para poder entender su comportamiento dinámico a través de simulaciones numéricas.
- Entender el funcionamiento de una plataforma experimental compuesta por un helicóptero miniatura de cuatro rotores estudiando los sistemas que la integran para adquirir la capacidad de operarla remotamente.
- Implementar leyes de control de tipo proporcional derivativo para estabilizar la orientación, la altura y la posición traslacional del vehículo mediante la modificación de Firmware del autopiloto embebido.
- Entender el funcionamiento de las principales herramientas del entorno de programación del Sistema Operativo Robótico (ROS, del inglés, Robot Operating System) consultando la literatura especializada para poder implementar el algoritmo de ORB-SLAM2.
- Integrar una cámara estereoscópica y una computadora embebida en la plataforma experimental para procesar la información visual del entorno mediante el algoritmo de ORB-SLAM2.
- Realizar tareas de seguimiento mediante la ubicación del helicóptero de cuatro rotores en el mapa generado por el algoritmo de ORB-SLAM2 utilizando el sistema de percepción visual embebido.

1.6. Metodología

En la presente sección se describe la metodología a seguir para cumplir con los objetivos particulares y finalmente llegar al objetivo general.

- Se revisará el estado del arte actualizado constantemente y se irá agregando al marco teórico las ideas más destacadas e importantes para desarrollar toda la investigación.
- Se estudiará el modelo matemático del vehículo aéreo de cuatro rotores para a partir del entendimiento de su comportamiento dinámico sea posible desarrollar, simular e implementar leyes de control.
- A su vez, se estudiará el funcionamiento de la plataforma experimental y de todos sus componentes para poder manipularla y tener siempre el control manual en caso de perder la autonomía por cualquier razón.

- Con el conocimiento necesario del Firmware del autopiloto embebido será posible implementar las leyes de control.
- De manera paralela, se estudiarán y comprenderán las herramientas necesarias de la plataforma ROS que permitan poner en marcha el algoritmo ORB-SLAM2.
- Una vez comprendido en su mayoría el sistema ROS y el algoritmo ORB-SLAM2, este último será implementado en una computadora portátil con una cámara estereocópica para generar un mapa de un espacio no estructurado y de manera simultánea obtener la pose de la cámara en el mismo.
- Ya implementado y probado el algoritmo en la computadora portátil, se procederá a implementar el mismo en una computadora embebida y una cámara estereocópica a bordo del vehículo.
- Finalmente, con el mapa creado y conocida la localización del vehículo en el mismo, se realizarán tareas de seguimiento autónomas, a partir de rutas trazadas en dicho mapa.

1.7. Conclusiones del capítulo

En este capítulo se inició partiendo del planteamiento del problema que se tiene, así como la justificación del mismo, se plantea la hipótesis de la investigación, definiendo los objetivos trazados para llevar a cabo durante la presente tesis. En el siguiente capítulo se estudiará el marco teórico con los principales conceptos necesarios para desarrollar la investigación.

Capítulo 2

Marco Teórico

En el presente capítulo se define el marco teórico necesario para la realización de toda la investigación.

2.1. Conceptos de vehículos aéreos.

En los últimos siglos, la tecnología se ha desarrollado a un ritmo trepidante. La invención de las primeras aeronaves tripuladas, las redes de comunicación o la radiocomunicación, entre otros, han traído consigo múltiples avances, donde los Vehículos Aéreos No Tripulados VANTs son un claro ejemplo de ello.

Los VANTs, comúnmente conocidos como drones, tienen la peculiaridad de no disponer de un piloto en el interior de la aeronave, por lo que esta es dirigida por una persona o sistema electrónico externo, que decide en cada momento el siguiente paso a seguir [14].

Con el paso de los años y fundamentalmente con la mejora e invención de nuevas tecnologías como el GPS, han ido apareciendo varios modos de utilización hasta llegar al actual modo autónomo, que permite a un VANT despegar, realizar cualquier intervención de forma periódica, y aterrizar sin la intervención ni presencia humana [15].

2.1.1. Aplicaciones de los VANTs

Los VANTs usualmente se ubican en dos grandes categorías de aplicación: la militar y la civil. En el caso de aplicaciones militares la cualidad más importante de los VANTs es el vuelo no tripulado. Las aplicaciones de los VANTs militares pueden dividirse en tres categorías principales: patrullaje y reconocimiento, apoyo al combate, y combate. Por su parte, las aplicaciones civiles en comparación con los VANTs militares, los VANTs civiles no tienen el mismo desarrollo que han tenido sus similares militares. No

obstante, poseen un buen potencial, debido a su versatilidad y flexibilidad de operación. Existe un amplio rango de aplicaciones potenciales para los vehículos no tripulados civiles, tal es el caso de investigación científica, en el estudio de la atmósfera, la tierra y el océano, cartografía, apoyo en desastres, detección de fuego en incendios, volcanes y tornados, búsqueda y rescate, vigilancia civil, supervisión marítima, vigilancia urbana, entre otras [16].

2.1.2. Clasificación de los VANTs

Existen varias maneras de clasificar los VANTs, en primer lugar, se tendrá en cuenta la clasificación según el método de control [17], estas pueden ser:

1. Autónomo: La aeronave no necesita de un piloto humano que lo controle desde tierra.

2. Monitorizado: En este tipo de control para VANTs se necesita la presencia de un técnico humano.

3. Supervisado: Un operador pilota directamente la aeronave, aunque esta puede realizar algunas tareas automáticamente.

4. Pre-programado: El VANT sigue un plan de vuelo diseñado previamente y no tiene medios para cambiarlo para adaptarse a posibles cambios.

5. Controlado remotamente: Son los más implantados dentro de los VANTs civiles, son conocidos como VANTs de radio control.

Otra de las posibles clasificaciones de los VANTs es en función de sus alas, distinguiendo así entre fijas y móviles o rotatorias. Las alas fijas consisten en alas adosadas en los laterales de la aeronave. Las móviles o rotatorias consisten en hélices giratorias, generalmente suelen ser cuadricópteros (cuatro motores con hélice), situados en un eje vertical al suelo [18]. Estos pueden ser de 6, 8 o más hélices, teniendo la consideración de que cuente con un número par de rotores, para garantizar la estabilidad de la aeronave [19].

Los VANTs también se pueden clasificar tomando en cuenta su peso, altura de operación, duración de vuelo, etc., de igual forma no existe una clasificación única, a continuación se muestra una tabla con alguna de esas clasificaciones:

Categoría	Peso	Altitud de operación	Radio de la misión	Autonomía de vuelo	Altitud
Micro	<2 kg	60 m	5 km	Pocas horas	Muy poca altitud
Mini	2-20 kg	900 m	25 km	Hasta 2 días	Poca altitud
Pequeño	20-50 kg	1.5 km	50 km	Hasta 2 días	Poca altitud
Táctico	150-600 kg	3 km	200 km	Hasta 2 días	Poca altitud
MALE (Mediana Altitud, gran resistencia)	>600 kg	13.5 km	200 km	Días/semanas	Altitud media
HALE(Gran altitud, gran resistencia)	>600 kg	20 km	Ilimitada	Días/semanas	Gran altitud
Combate	>600 kg	20 km	Ilimitada	Días/semanas	Gran altitud

Tabla 2.1: Clasificación de los VANTs [1].

Otra manera de clasificar los VANTs tipo helicóptero es según la cantidad de motores y la disposición de los mismos. Estas categorías pueden ser:

- Tricópteros [20].
- Cuadricópteros [21].
- Hexacópteros [22].
- Octocópteros [23].

Los cuadricópteros presentan la configuración más común dentro de los helicópteros de varios rotores. Es un helicóptero de 4 hélices, dichas hélices se encuentran en el mismo plano entre sí, son impulsadas mediante 4 motores eléctricos de corriente continua sin escobillas que se encuentran en los extremos de los brazos. El ascenso y descenso del VANT se consigue mediante el aumento y reducción de las revoluciones de los motores. Dos de los cuatro motores giran en el sentido contrario a las agujas del reloj, esto es necesario para poder neutralizar la fuerza generada y alcanzar un equilibrio. Para el movimiento de giro del VANT respecto a su eje vertical, es necesario la aparición de diferentes fuerzas, es necesario que las fuerzas de los motores izquierdo y derecho se encuentren desequilibradas. Para el giro sobre el eje longitudinal del VANT (rotación sobre su propio eje), el propulsor contrario al giro deseado deberá rotar a revoluciones más elevadas y por consiguiente su contrario disminuirá las revoluciones inversamente proporcional, consiguiendo así, el movimiento de rotación. Es necesario que la suma de fuerzas iniciales, sean exactamente iguales que las fuerzas resultantes durante el movimiento del VANT. Los sistemas de giroscopios son los encargados del control y estabilización inteligente del helicóptero de varios rotores [21].

Este tipo de helicóptero de varios rotores dispone de estructuras diversas a pesar de que las hélices deben encontrarse en el mismo plano. La primera configuración en forma de + permite un manejo sencillo, puesto que a la hora de un cambio de dirección sobre el eje vertical u horizontal se produce con el control de un solo motor. La estructura es diferente en el caso de los cuadricópteros con forma de X, ver Figura 2.1, o H. Los motores en este tipo de helicóptero de varios rotores se encuentran separados 45° en dirección a uno de los brazos y orientados en la dirección de vuelo, para ello, será necesario que los cuatro motores sean dirigidos al mismo tiempo. Este

tipo de estructura se considera una de las más apropiadas para el soporte de cámaras, puesto que ofrece un campo de visión mayor al no existir motores que interfieran en la visión [24].

Figura 2.1: Cuadricóptero en configuración X.

2.2. Sistema Operativo Robótico ROS

El Sistema Operativo Robótico (ROS, del inglés, Robot Operating System) [25] es un framework que es ampliamente utilizado en robótica. La filosofía es convertirse en una pieza de software que pueda trabajar en robots haciendo pequeños cambios en el código fuente del software. Se puede concluir que su finalidad es poder usar y compartir software con robots de manera mucho más eficiente y fácil [26].
ROS fue desarrollado en 2007 por el Laboratorio de Inteligencia Artificial de Stanford (SAIL, del inglés, Stanford Artificial Intelligence Laboratory), con el soporte de la *Stanford AI Robot project*. ROS provee las facilidades de un sistema operativo estándar, como la abstracción de hardware de bajo nivel, implementa funciones comúnmente usadas. Está basado en arquitectura gráfica y con topología centralizada, donde los procesos toman lugar en nodos que pueden enviar o recibir, como un sensor, un control, un estado, un actuador y mucho más. La librería está enfocada a sistemas Unix (Ubuntu encabeza la lista con soporte) mientras que otras como Fedora y Mac OS X son consideradas como versiones experimentales [27].

2.2.1. Estructura de ROS

ROS usa un lenguaje de mensajes descriptivos simples para describir los valores de datos que publica por medio de nodos. Con esta descripción, ROS puede generar el código fuente para ese tipo de mensajes a través de diferentes lenguajes de programación. ROS tiene una gran variedad de mensajes predefinidos, pero es posible desarrollar un nuevo tipo de mensaje, definido mediante un paquete personalizado. Un mensaje debe contener dos partes principales, el campo y la constante, el campo se refiere al tipo de

datos por ejemplo *int32*, *float32* y *string*, mientras que las constantes se refieren al valor que se desea enviar.

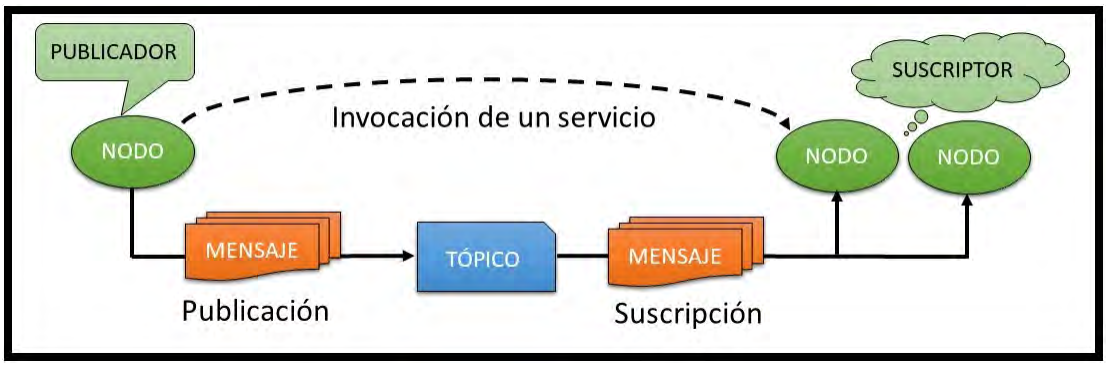

Figura 2.2: Diagrama de funcionamiento de ROS.

En la Figura 2.2 se puede observar el diagrama de funcionamiento de ROS. En primer lugar, un nodo publicador "publica" un mensaje vía un tópico, dicho mensaje puede ser recibido por todos los nodos suscriptores que se encuentren "suscritos" a dicho tópico. Igualmente, se puede hacer uso de servicios para comunicarse entre nodos.

2.2.1.1. Nodos

Los nodos son ejecutables que pueden comunicarse con otros procesos usando tópicos, servicios, o los parámetros, el uso de nodos proporciona el desarrollo de un código reutilizable para diferentes proyectos, cada nodo debe contener un nombre único dentro del sistema, este nombre lo utiliza el sistema operativo para realizar una comunicación entre otros nodos dentro del sistema, evitando así las ambigüedades.

2.2.1.2. Tópicos

Los tópicos son buses de datos usados por los nodos para el envío de información, estos pueden transmitir información de manera indirecta entre los nodos. Cada tópico tiene un tipo de mensaje definido que es usado para publicar información a través de él, cabe destacar que si otro nodo se desea subscribir a él es necesario que contenga el mismo tipo de mensaje, cada nodo es capaz de soportar a varios subscriptores conectados simultáneamente [28].

2.2.1.3. Mensajes

Un nodo envía información a otro nodo por medio del uso de mensajes que son publicados en los tópicos. Los mensajes tienen una estructura simple que pueden ser

definidos por el usuario, los nombres de los mensajes son definidos por la siguiente convención: el nombre del paquete, seguido por una barra , el nombre, finalizado por la extensión *.msg*, ejemplo: *std_msgs/msg/String.msg*.

2.3. Localización y mapeo simultáneo (SLAM)

La Localización y Mapeo Simultáneo (SLAM, por sus siglas en inglés) es una técnica cuya tarea consiste en ubicar un robot móvil en un ambiente desconocido, el robot irá construyendo un mapa consistente del ambiente, mientras determina su localización simultáneamente en dicho mapa. La solución al problema del SLAM ha sido objeto de estudio de la comunidad mundial, puesto que propicia los medios para que un robot móvil sea verdaderamente autónomo.

La idea del SLAM surge en el año 1986 durante un congreso de robótica y automatización de la IEEE. En ese momento, se había comenzado a introducir los métodos probabilísticos en la inteligencia artificial (AI por sus siglas en inglés) y en la robótica. Varios artículos fueron publicados en los siguientes años, por ejemplo los autores Smith, Cheesman y Durrant-Whyte [29], los cuales establecieron las bases estadísticas para describir relaciones entre los objetos de referencia o puntos de interés y manipular la incertidumbre geométrica.

Al mismo tiempo, Ayache y Faugeras [30] estaban consiguiendo los primeros avances en el campo de la navegación basada en visión y Crowley [31], Chatila y Laumond [32] incursionaban en el uso de sensores tipo sonar para robots móviles usando algoritmos basados en filtros de Kalman.

Los trabajos anteriores no tuvieron en cuenta las propiedades de convergencia del mapa o su comportamiento en estado estacionario. También, por estas razones, el trabajo teórico sobre el problema combinado de localización y mapeo se detuvo temporalmente, centrándose únicamente en el mapeo o la localización como problemas separados.

Finalmente se logró un avance sustancial, los investigadores descubrieron que el problema combinado de mapeo y localización, una vez formulado como un solo problema de estimación, en realidad era convergente. Además, aún más importante, se reconoció que las correlaciones entre puntos de interés mientras más crecían mejor era la solución al problema [33]. Por todo lo anterior, la estructura del problema de SLAM, el resultado de la convergencia y el uso oficial del acrónimo SLAM en si, se presentaron por primera vez en una investigación de robótica móvil presentado en el

Simposio Internacional de Investigación en Robótica en el año 1995.

2.3.1. Estructura y formulación del problema de SLAM

La presente sección fue basada en su mayoría en los artículos [33] y [34], los cuales representan en conjunto los inicios, principales resultados y aplicaciones del problema de SLAM.

Como se mencionó anteriormente, el SLAM es un proceso mediante el cual un robot móvil puede construir un mapa de un entorno y simultáneamente utilizarlo para estimar su ubicación. Tanto la trayectoria del robot como la ubicación de todos los puntos de interés son estimados en línea (online), por lo cual, no es necesario tener ningún previo conocimiento de la ubicación.

A partir de la Figura 2.3, considere un robot móvil que se mueve a través de un entorno desconocido, además, a partir de un sensor a bordo del robot, este toma observaciones relativas de varios puntos de interés desconocidos.

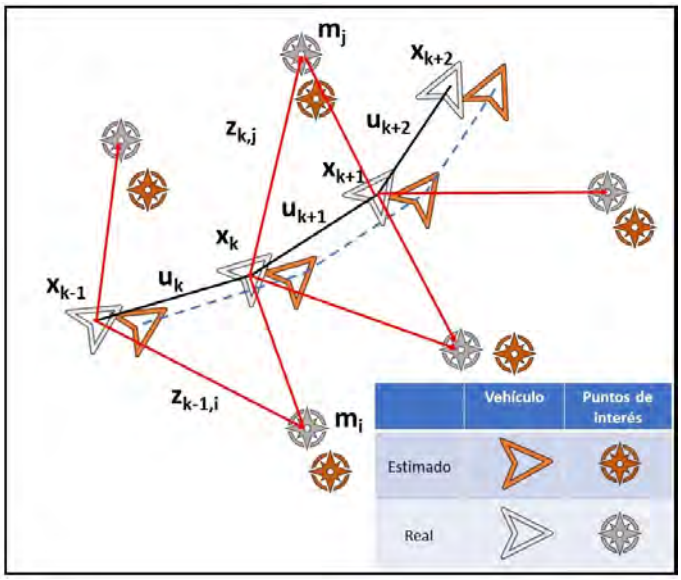

Figura 2.3: Problema principal del SLAM. Se requiere una estimación simultánea de las ubicaciones del robot y de los puntos de interés. Las ubicaciones reales nunca se conocen ni se miden directamente. Las observaciones se realizan entre el robot y los puntos de interés.

En un instante k, se definen las siguientes variables:

- x_k: vector de estado que describe la posición y orientación del vehículo.

- u_k: vector de control, aplicado en el tiempo $k-1$ para mover el vehículo al estado x_k en el tiempo k.
- m_i: vector que describe la localización del i-ésimo punto de interés cuya localización real es asumida invariante en el tiempo.
- z_{ik}: observación tomada desde el vehículo de la ubicación del i-ésimo punto de interés en el tiempo k. Cuando hay múltiples observaciones de puntos de interés en un tiempo dado o cuando el punto de interés específico no es relevante, la observación se tomará simplemente como z_k.

Adicionalmente, se definen las variables siguientes:

- $X_{0:k} = x_0, x_1, ..., x_k = X_{0:k-1}, x_k$: histórico de las localizaciones del vehículo.
- $U_{0:k} = u_1, u_2, ..., u_k = U_{0:k-1}, u_k$: histórico de las señales de control.
- $m = m_1, m_2, ..., m_n$: conjunto de todos los puntos de interés.
- $Z_{0:k} = z_1, z_2, ..., z_k = Z_{0:k-1}, z_k$: conjunto de todas las observaciones de los puntos de interés.

En forma probabilística, el problema de localización y mapeo simultáneo (SLAM), requiere que la distribución de probabilidad

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0), \tag{2.1}$$

debe ser calculada para todos los tiempos k. Esta distribución de probabilidad describe la distribución condicional de los puntos de interés y el vector de estado del vehículo (en el tiempo k), dadas las observaciones registradas y las entradas de control hasta el tiempo k, junto con el estado inicial del vehículo. De manera general, es deseable una solución recursiva al problema de SLAM. Comenzando con una estimación de la distribución $P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1})$ en el tiempo $k-1$, la siguiente iteración, siguiendo un control u_k y observación z_k, se calcula utilizando el teorema de Bayes [33]. Este cálculo requiere que se defina un modelo de transición de estado y un modelo de observación que describan el efecto de la entrada de control y la observación, respectivamente.

El modelo de observación describe la probabilidad de hacer una observación z_k cuando la localización del vehículo y el punto de interés es conocida, generalmente se describe como

$$P(z_k|x_k, m). \tag{2.2}$$

Es razonable suponer que una vez definida la ubicación del vehículo y el mapa, las observaciones son condicionalmente independientes dado el mapa y el estado actual del vehículo.

El modelo para describir el movimiento del vehículo se puede obtener en términos de una distribución de probabilidad en las transiciones de estado de la forma

$$P(x_k|x_{k-1}, u_k). \tag{2.3}$$

Es decir, se supone que la transición de estado es un proceso de Markov [35], en el que el siguiente estado x_k depende solo del estado inmediatamente anterior x_{k-1} y del control aplicado u_k, siendo independiente de las observaciones y del mapa.

El algoritmo SLAM se implementa en dos pasos recursivos (secuenciales), predicción (actualización del tiempo) y corrección (actualización de la medición):

1. Actualización del tiempo

$$P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0) = \int \{P(x_k|x_{k-1}, u_k) \times P(x_{k-1}, m|Z_{0:k-1}, U_{0:k-1}, x_0)\} dx_{k-1}, \tag{2.4}$$

2. Actualización de la medición

$$P(x_k, m|Z_{0:k}, U_{0:k}, x_0) = \frac{P(z_k|x_k, m)P(x_k, m|Z_{0:k-1}, U_{0:k}, x_0)}{P(z_k|Z_{0:k-1}, U_{0:k})}. \tag{2.5}$$

Las ecuaciones (2.4) y (2.5) proporcionan un procedimiento recursivo para calcular la siguiente iteración $P(x_k, m|Z_{0:k}, U_{0:k}, x_0)$ para el estado del vehículo x_k y el mapa m en el tiempo k, basado en todas las observaciones $Z_{0:k}$ y las entradas de control $U_{0:k}$ hasta el tiempo k. La recursión está basada en una función del modelo de vehículo $P(x_k|x_{k-1}, u_k)$ y el modelo de observación $P(z_k|x_k, m)$.

Vale la pena señalar que el problema de construcción del mapa puede formularse como el cálculo de la densidad condicional $P(m|X_{0:k}, Z_{0:k}, U_{0:k})$. Esto supone que la ubicación del vehículo x_k es conocida (o al menos determinista) en todo momento, sujeto al conocimiento de la ubicación inicial. Luego, se construye un mapa m fusionando observaciones desde diferentes ubicaciones. Por su parte, el problema de localización puede formularse como el cálculo de la distribución de probabilidad $P(x_k|Z_{0:k}, U_{0:k}, m)$. Esto supone que las ubicaciones de los puntos de interés se conocen con certeza, y el objetivo es calcular una estimación de la ubicación del vehículo con respecto a estos puntos de interés [33].

Como ya se comentó anteriormente, el objetivo del SLAM es usar el entorno para conseguir estimar la posición del robot. Debido a que la odometría no es del todo fiable, no podemos confiar todo el posicionamiento en ella. Por este motivo se emplean sensores de distancias capaces de corregir los errores cometidos por la odometría. Para ello, se extraen una serie de puntos de interés del entorno que se emplearán como

referencia. Estas características deben ser fácilmente distinguibles y re-observables por el vehículo. El encargado de actualizar la estimación de la posición suele ser un Filtro de Kalman Extendido [36] y es uno de los elementos más importantes del algoritmo de SLAM. En la Figura 2.4 se observa el esquema general de funcionamiento del algoritmo SLAM. Cuando el vehículo se mueve, su odometría cambia y se actualiza empleando el EKF (Extended Kalman Filter por sus siglas en inglés). Por otra parte, los datos obtenidos por el sensor se procesan y se extraen una serie de puntos de interés, posteriormente se determina si se trata de algún punto característico del entorno observado por primera vez o uno que ya se haya captado antes. En el caso de que sea un punto ya conocido, se introduce en el filtro de Kalman para actualizar la posición. Por el contrario, si se trata de un nuevo punto se añade al EKF como nueva observación y se empleará más adelante para actualizar la estimación de la posición.

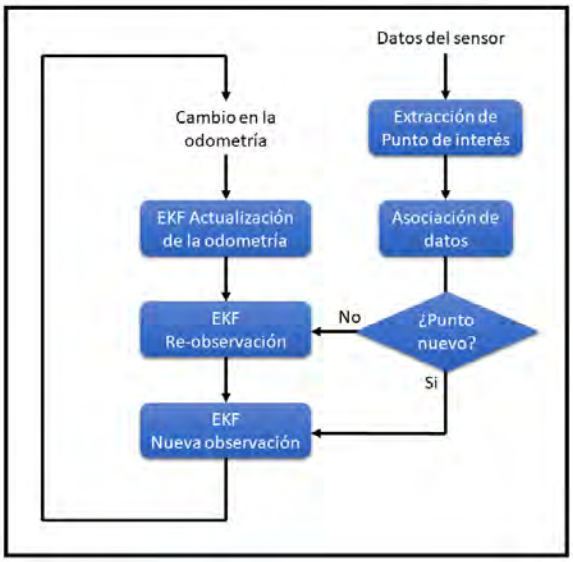

Figura 2.4: Esquema funcional del algoritmo SLAM.

A la hora de generar los mapas de los entornos por los que se va navegando, se pueden diferenciar dos tipos de mapeado: denso y disperso. Los métodos de reconstrucción densos, son aquellos donde se evalúa la correspondencia de todos los píxeles en la imagen, y se calcula la disparidad de cada uno. En los dispersos se utilizan los puntos de interés de la imagen y se halla su correspondiente par para determinar la profundidad o distancia al sistema de cámaras.

2.3.2. Puntos de interés

Los puntos de interés son aquellos puntos característicos del entorno que pueden ser fácilmente re-observables y distinguidos por el vehículo. Estos puntos se emplean

para que el vehículo pueda localizarse dentro del entorno en el cual está navegando. Es importante que los puntos de interés sean observables desde diferentes posiciones y ángulos. Además, deben ser únicos, para que puedan identificarse fácilmente y el algoritmo pueda determinar de forma rápida si se trata de una nueva observación o de un punto que ya se había observado con anterioridad. Para evitar errores, es necesario que estos puntos sean estacionarios y de un tamaño considerable. Un aspecto importante en el algoritmo de SLAM es la asociación de datos, es decir, hacer corresponder las distintas observaciones de un mismo punto de interés en distintas imágenes [33].

2.3.3. Sensores utilizados en el algoritmo SLAM.

Uno de los aspectos más importante a tener en cuenta a la hora de implementar SLAM es el sensor que se va a emplear para extraer la información del entorno por el que se desplaza el vehículo. Uno de los más usados es un escáner láser debido a la precisión que presentan (algunos tienen un error inferior a 10 milímetros) y a que son muy rápidos. Otra de las ventajas que presentan, es el poco procesamiento que necesitan los datos a la salida, ya que pueden ser interpretados directamente. No obstante, son sensores muy caros y no siempre dan buenos resultados [33]. En algunas superficies, como el vidrio, la calidad de las medidas no es adecuada. Tampoco son adecuados para realizar SLAM bajo el agua, ya que en este medio los sensores láser tienen muy poco alcance. Otro de los sensores que más se usan como dispositivo de medida en estas técnicas son los de ultrasonido (sonar). Son más mucho más baratos que los láseres, aunque sus medidas menos precisas. Sin embargo, son adecuados para usarlos bajo agua. Sin embargo, los sensores vistos anteriormente son incapaces de extraer información de las superficies observadas e identificar objetos. Por este motivo, una de las formas más comunes de realizar SLAM es mediante visión. El mayor inconveniente es la gran cantidad de información que se debe procesar tras extraer los datos de las imágenes. No obstante, con el gran desarrollo que ha experimentado este campo, el procesamiento de los datos no supone un gran problema [34].

2.3.4. Principales algoritmos SLAM existentes

Existe gran variedad de algoritmos SLAM que han sido creados e implementado, a continuación, se enumeran algunos de estos algoritmos y se refleja una pequeña explicación de los mismos.

1. Hector SLAM [37]: Es una implementación open source de la técnica de SLAM en 2D que suele ser usada como base en muchos algoritmos de SLAM 3D. El sensor utilizado por este método es un sensor láser, aunque también se puede

usar un dispositivo como Kinect, ya que a través de un tratamiento de la nube de puntos que genera dicho dispositivo se pueden obtener datos similares a los obtenidos por sensor láser.

2. LSD-SLAM [38]: (Large- Scale Direct Monocular SLAM) permite construir mapas de gran escala. En lugar de utilizar características, opera directamente sobre los contrastes de las imágenes tanto para la localización como para el mapeo. La geometría de los mapas se estima usando filtros sobre las imágenes adquiridas en escala de grises.

3. RGB-D SLAM [39]: Es uno de los primeros algoritmos que surgieron para realizar SLAM empleando las imágenes en color y de profundidad de las cámaras RGB-D; comparado con otros métodos es uno de los más fiables y robustos.

4. DP-SLAM [40]: Se basa en sensores láser de rango.

5. ORB-SLAM: Se basa en ORB para detectar las características.

2.4. Conclusiones del capítulo

En el presente capítulo se trataron los principales conceptos teóricos necesarios para llevar a cabo el trabajo de tesis. Principalmente, los conceptos relacionados con vehículos aéreos, el principio de funcionamiento de ROS, el estado del arte referente a los sistemas SLAM, así como los principales algoritmos desarrollados. Destacando el algoritmo presentado ORB-SLAM2, que será el que se utilizará para ser implementado en la computadora embebida en el vehículo aéreo, por lo cual el siguiente capítulo tratará en detalles el funcionamiento del mismo.

Capítulo 3

Algoritmo ORB-SLAM2

En el presente capítulo se describe de manera detallada el funcionamiento del algoritmo ORB-SLAM2 el cuál será implementado en la computadora embebida en el vehículo aéreo.

Existe una gran variedad en cuanto a los sensores utilizados a la hora de implementar los algoritmos de visión artificial, entre los más utilizados se encuentran las cámaras dado su bajo costo y además por proporcionar una información enriquecida del entorno. Por lo tanto, las soluciones en los algoritmos SLAM, en las que el sensor principal es una cámara, son de gran interés en la actualidad. El reconocimiento adecuado del entorno que se desea navegar es un módulo clave de un sistema SLAM para detectar lazos cerrados (es decir, detectar cuándo el sensor regresa a un área previamente mapeada) y además, para reubicar la cámara después de una falla de seguimiento, debido a una oclusión o movimiento agresivo, o en la reinicialización del sistema.

Para la implementación de algoritmos SLAM se puede utilizar como sensor una cámara monocular, la cuál es la configuración de sensor más barata y más pequeña. Sin embargo, como la profundidad no es observable desde una cámara monocular, se desconoce la escala del mapa y la trayectoria estimada. Además, el sistema de inicialización requiere múltiples vistas o técnicas de filtrado para producir un mapa inicial, ya que no se puede triangular desde el primer fotograma. Por último, pero no menos importante, el SLAM monocular sufre desviación de escala y puede fallar si se realizan rotaciones en la exploración. Al utilizar una cámara estereoscópica o RGB-D, todos estos problemas se resuelven y permiten obtener soluciones más confiables.

Existen dos algoritmos principales de ORB-SLAM, el primero es únicamente para cámaras monoculares, conocido como ORB-SLAM [41], desarrollado por Raúl Mur-Artal, J. M. M. Montiel y Juan D. Tardós en el año 2015. El segundo, que permite el uso tanto de cámaras monoculares como estereocópicas y RGB-D fue una modificación del primer algoritmo realizado por los mismo autores en el año 2017, también conocido

como ORB-SLAM2 [42]. Este algoritmo se basa en ORB, el cual es un descriptor binario muy rápido y de menor costo computacional basado en BRIEF [43] que es invariante a la rotación y resistente al ruido, además, permite la detección de características coincidentes en una imagen, permitiendo el reconocimiento de objetos a pesar de presentar rotaciones la imagen.

La presente sección se basa principalmente en los dos artículos de ORB-SLAM mencionados en el párrafo anterior. El método de ORB-SLAM, se trata de un algoritmo de SLAM monocular basado en el reconocimiento de características, capaz de operar en tiempo real, en entornos tanto de pequeño como de gran tamaño, y que se puede usar tanto en exteriores como en interiores. La idea principal de la generación de mapas en esta técnica es emplear las mismas características para todas las tareas de SLAM: seguimiento, creación del mapa, localización y detección de trayectorias cerradas. Se seleccionarán estratégicamente puntos y fotogramas clave para generar los mapas, estos solo variarán si el contenido de la escena cambia. Los puntos más destacados de esta técnica de SLAM son:

- Usar las mismas características para todas las tareas de SLAM. Esto provoca que el sistema sea más eficiente, simple y fiable.
- Capaz de realizar operaciones en tiempo real en entornos de gran tamaño. Es el resultado de usar un grafo de co-visibilidad. Tanto el seguimiento como el mapeo se focalizan en el área co-visible, independientemente del tamaño del mapa completo, consiguiendo así explorar entornos amplios sin aumentar el tiempo de cómputo.
- La estrategia para detectar los bucles cerrados de visión en tiempo real está basada en la optimización de un grafo llamado Gráfico Esencial, se trata de una de las ideas novedosas de este método.
- Localización en tiempo real en localizaciones que se han visitado con anterioridad independientemente de que existan significantes cambios en la iluminación o en el ángulo de visión.

Dada las ventajas que ofrece el uso de cámaras estereoscópicas o RGB-D, el algoritmo que será implementado será el ORB-SLAM2. Dicho algoritmo incorpora tres módulos que se ejecutan en paralelo: seguimiento, mapeo local y trayectorias cerradas. En la Figura 3.1 se puede observar el diagrama general del algoritmo.

A continuación, se describen los procesos que se ejecutan en cada módulo del algoritmo.

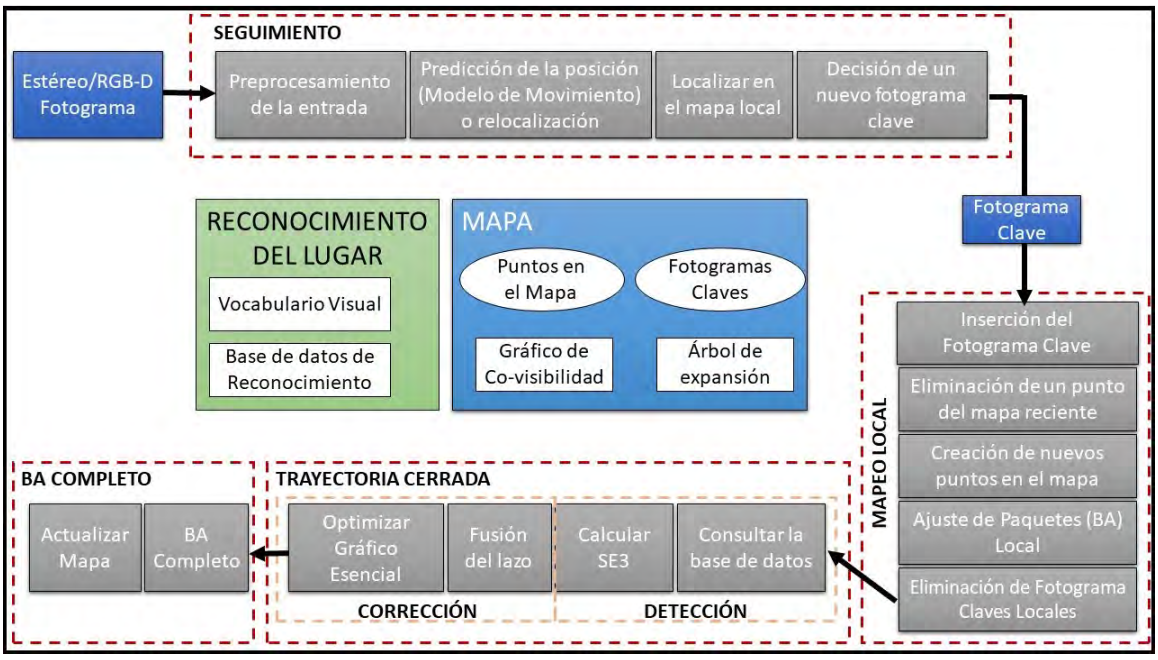

Figura 3.1: Diagrama general del algoritmo ORB-SLAM2.

3.1. Módulo de seguimiento

En esta sección, se describen los pasos del módulo de seguimiento que se realizan con cada fotograma de la cámara. Las optimizaciones de la posición de la cámara, mencionadas en varios pasos, consisten en Ajuste de Paquetes (BA, del inglés, Bundle Adjustment) [44] de solo movimiento.

Preprocesamiento de la entrada

ORB-SLAM2 como es un algoritmo basado en características, procesa previamente la entrada para extraer características de puntos clave con ubicaciones destacadas, como se muestra en la Figura 3.2. Las imágenes de entrada se descartan y todas las operaciones del sistema se basan en estas características para que el sistema sea independiente del sensor estéreo o RGB-D utilizado. El algoritmo maneja puntos claves monoculares y estéreo, que además, se clasifican como cercanos o lejanos.

Los puntos claves estéreos están definidos por tres coordenadas $x_s = (u_L, v_L, u_R)$, siendo (u_L, v_L) las coordenadas en la imagen izquierda y u_R la coordenada horizontal en la imagen derecha. Para cámaras estereoscópicas, se extraen las características ORB [13] en ambas imágenes y para cada característica ORB izquierda buscamos una coincidencia en la imagen derecha. Luego, se genera el punto clave estéreo con las coordenadas de la característica ORB izquierda y la coordenada horizontal de la

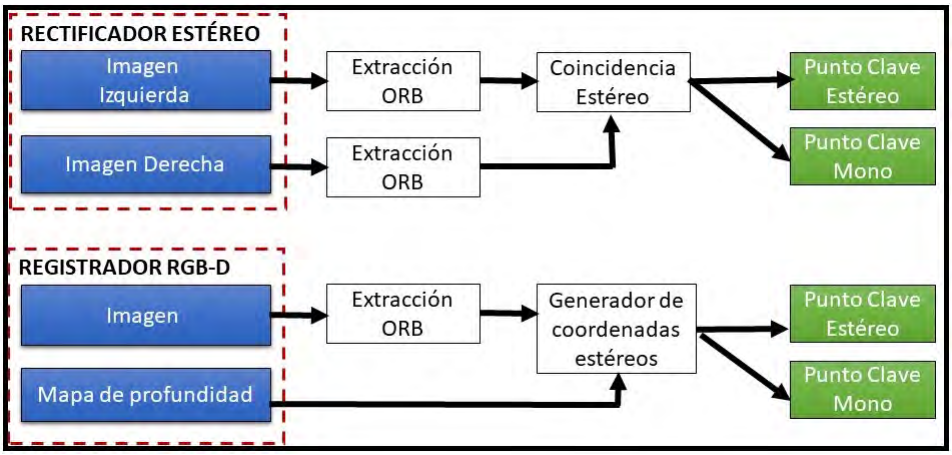

Figura 3.2: Preprocesamiento de la entrada.

coincidencia con la imagen derecha. Para el caso de las cámaras RGB-D, se extraen las características ORB en la imagen RGB y, según lo propuesto por Strasdat et al. [45], para cada entidad con coordenadas (u_L, v_L), transformamos su valor de profundidad d en una coordenada derecha virtual.

$$u_R = u_L - \frac{f_x b}{d}, \tag{3.1}$$

donde f_x es la distancia focal horizontal y b es la línea de base entre el proyector de luz estructurado y la cámara infrarroja, que se puede aproximar a 8 cm para Kinect y Asus Xtion. De esta manera, las características de entrada estéreo y RGB-D son manejadas igualmente por el resto de los módulos.

Un punto clave estéreo se clasifica como cercano si su profundidad asociada es inferior a 40 veces la línea base estéreo/RGB-D, como se sugiere en [46], de lo contrario se clasifica como lejano. Los puntos clave cercanos se pueden triangular de forma segura a partir de un cuadro, ya que la profundidad se estima con precisión y proporciona información de escala, traslación y rotación. Por otro lado, los puntos lejanos proporcionan información de rotación precisa, pero información débil relacionada con la escala y traslación.

Los puntos claves monoculares se definen mediante dos coordenadas $x_m = (u_L, v_L)$ en la imagen izquierda y corresponden a todas aquellas características ORB para las que no se pudo encontrar una coincidencia estéreo o que tienen un valor de profundidad no válido en el caso de RGB-D. Estos puntos solo se triangulan desde múltiples vistas y no proporcionan información de escala, sino que contribuyen a la estimación de rotación y traslación.

Inicialización del sistema

Una de las principales ventajas de utilizar cámaras estereoscópicas o RGB-D es que, al tener información de profundidad de un solo fotograma, no se necesita una estructura específica a partir de la inicialización del movimiento como en el caso monocular. Al iniciar el sistema, se crea un fotograma clave con el primer fotograma, se establece su posición en el origen y se crea un mapa inicial desde todos los puntos claves estéreos.

Un modo de localización es incorporado, que puede ser útil para la localización ligera a largo plazo en áreas bien mapeadas, siempre que no haya cambios significativos en el entorno. En este modo, los módulos de mapeo local y trayectorias cerradas son desactivados y la cámara se localiza continuamente mediante el seguimiento utilizando la reubicación si es necesario. En este modo, el seguimiento aprovecha las coincidencias de odometría visual y las coincidencias con los puntos del mapa. Las coincidencias de odometría visual son coincidencias entre características ORB en el fotograma actual y puntos 3D creados en el fotograma anterior a partir de la información estéreo o de profundidad.

Localizar en el mapa local

Una vez obtenida una estimación de la posición de la cámara y un conjunto inicial de características coincidentes, se puede proyectar el mapa en el fotograma y buscar más correspondencias de puntos en el mapa. Para limitar la complejidad en mapas grandes, solo se realizan dichas proyecciones en un mapa local. Este mapa local contiene el conjunto de fotogramas clave K_1, que comparten puntos del mapa con el fotograma actual, y un conjunto K_2 contiene puntos vecinos a los fotogramas clave en K_1 en el gráfico de covisibilidad. El mapa local también tiene un fotograma clave de referencia $K_{ref} \in K_1$, que comparte la mayoría de los puntos del mapa con el fotograma actual. Ahora, cada punto del mapa observado en K_1 y K_2 se busca en el fotograma actual de la siguiente manera.

1. Realizar la proyección del punto del mapa x en el fotograma actual. Descartar si queda fuera de los límites de la imagen.

2. Calcular el ángulo entre el rayo de visualización actual v y el punto del mapa en la dirección de visualización n. Descartar si $v.n < \cos(60°)$.

3. Calcular la distancia d desde el punto del mapa hasta el centro de la cámara. Descartar si está fuera de la región de escala de invarianza del punto $d \notin [d_{min}, d_{max}]$.

4. Calcular la escala en el fotograma por la relación $\frac{d}{d_{min}}$.

5. Comparar el descriptor representativo D del punto del mapa con las características ORB aún inigualables en el fotograma, en la escala pronosticada y cerca de x, y asociar el punto del mapa con la mejor coincidencia.

La posición final de la cámara es optimizada con todos los puntos del mapa encontrados en el fotograma.

Decisión de un nuevo fotograma clave

El último paso en este primer módulo es decidir si el fotograma actual será considerado como un nuevo fotograma clave. Como hay un mecanismo en el módulo de mapeo local encargado de eliminar fotogramas clave redundantes, se intentará insertar fotogramas clave lo más rápido posible, porque esto hace que la tarea de seguimiento sea más robusta ante movimientos bruscos de la cámara, generalmente rotaciones. Para insertar un nuevo fotograma clave, se deben cumplir todas las condiciones siguientes.

1. Más de 20 fotogramas deben haber pasado desde la última relocalización global.
2. El mapeo local está inactivo, o han pasado más de 20 fotogramas desde la última inserción de un fotograma clave.
3. El fotograma actual contiene al menos 50 puntos.
4. El fotograma actual contiene al menos el 90 % de los puntos del conjunto K_{ref}.

En lugar de utilizar un criterio de distancia a otros fotogramas clave como lo realizan en el algoritmo de Localización y Mapeo Paralelo (PTAM, por sus siglas en inglés)[47], se impone un cambio visual mínimo (condición 4). La condición 1 asegura una buena reubicación y la condición 3 un buen seguimiento. Si se inserta un fotograma clave cuando el mapeo local está ocupado (condición 2), se envía una señal para detener al BA local para que pueda procesar lo antes posible el nuevo fotograma clave.

3.2. Módulo de mapeo local

A continuación, se describen los pasos realizados por el módulo de mapeo local para cada nuevo fotograma clave K_i.

Inserción del fotograma clave

Primeramente, se actualiza el gráfico de covisibilidad, agregando un nuevo nodo para K_i. Luego, se actualiza el árbol de expansión que une K_i con el fotograma clave que tenga la mayoría de puntos en común. Finalmente, se calcula la representación

de las palabras binarias que representan el fotograma clave, lo que ayudará en la asociación de datos para triangular nuevos puntos.

En el caso de ORB-SLAM2, la distinción entre puntos estéreos cercanos y lejanos permite introducir una nueva condición para la inserción de fotogramas clave, que puede ser crítica en entornos complicados donde una gran parte de la escena está lejos del sensor estéreo. En dichos entornos, es necesario tener una cantidad suficiente de puntos cercanos para estimar con precisión la traslación, por lo tanto, si el número de puntos cercanos detectados cae por debajo de τ_t y el fotograma podría crear al menos τ_c nuevos puntos estéreos cercanos, el sistema insertará un nuevo fotograma clave. Se recomienda que $\tau_t = 100$ y $\tau_c = 70$, dichos valores fueron obtenidos de resultados experimentales [42].

Eliminación de un punto del mapa reciente

Los puntos del mapa, para ser retenidos en él, deben pasar una prueba restrictiva durante los primeros tres fotogramas clave después de la creación, lo que garantiza que sean rastreables y no se triangulen erróneamente. Un punto debe cumplir estas dos condiciones.

1. La tarea de seguimiento debe encontrar el punto en más del 25 % de los fotogramas en los que se predice que sea visible.

2. Si más de un fotograma clave ha pasado desde la creación del punto del mapa, este debe observarse desde al menos tres fotogramas clave.

Una vez que un punto del mapa ha pasado esta prueba, solo se puede eliminar si en cualquier momento se observa desde menos de tres fotogramas clave. Esto puede suceder cuando se eliminan los fotogramas clave y cuando la BA local descarta las observaciones atípicas.

Creación de nuevos puntos

Se crean nuevos puntos del mapa triangulando ORB a partir de fotogramas clave conectados K_c en el gráfico de covisibilidad. Para cada característica ORB no coincidente en K_i, se busca una coincidencia con otro punto no coincidente en otro fotograma clave. Los pares de características ORB se triangulan y, para aceptar los nuevos puntos, se verifica la profundidad positiva en ambas cámaras, el error de proyección y la consistencia de la escala. Inicialmente, se observa un punto del mapa a partir de dos fotogramas clave, pero podría coincidir en otros; por lo tanto, se proyecta en el resto de fotogramas clave conectados.

Ajuste de Paquetes (BA) con restricciones monoculares y estéreos

El algoritmo ORB-SLAM2 realiza BA para optimizar la posición de la cámara en la tarea de seguimiento (BA de solo movimiento), para optimizar los fotogramas clave y puntos del mapa en la tarea de mapeo local (BA local), y finalmente en la tarea de detección de trayectorias cerradas para optimizar todos los fotogramas clave y puntos del mapa (BA completo). Para ello, se utiliza el método Levenberg-Marquardt implementado en g2o optimización general de gráficos, herramienta para la optimización de mínimos cuadrados no lineales [48].

El BA de solo movimiento optimiza la orientación y la posición de la cámara, minimizando el error de proyección entre los puntos 3D coincidentes en coordenadas reales y fotogramas clave.

Por su parte, el BA local optimiza un conjunto de fotogramas clave K_L covisibles y todos los puntos P_L vistos en esos fotogramas clave. Todos los demás fotogramas clave K_F, que no están en K_L, los puntos de observación en P_L contribuyen a la función de costo pero permanecen fijos en la optimización.

Finalmente, el BA completo es un caso específico del BA local, donde se optimizan todos los fotogramas clave y puntos en el mapa.

Eliminación de fotogramas clave locales

Para mantener una reconstrucción compacta, el mapeo local intenta detectar fotogramas clave redundantes y eliminarlos. Esto es beneficioso ya que la complejidad de BA aumenta con el número de fotogramas clave, pero también, permite la operación de por vida en el mismo entorno ya que el número de fotogramas clave no crecerá sin límites, a menos que cambie el contenido visual en la escena. Son descartados todos los fotogramas clave en K_c en los cuales el 90 % de los puntos del mapa se han visto en al menos otros tres fotogramas clave en la misma o más fina escala. La condición de escala garantiza que los puntos del mapa mantengan los fotogramas clave desde los que se miden con la mayor precisión. Esta metodología fue inspirada en la propuesta presentada en el trabajo de Tan et al. [49], donde los fotogramas clave se descartaron después de un proceso de detección de cambios.

3.3. Módulo de trayectorias cerradas y BA completo

La tarea de trayectorias cerradas se realiza en dos pasos, primero, la trayectoria cerrada debe ser detectada y validada, y segundo, esta se corrige optimizando un gráfico de posición. A diferencia del ORB-SLAM monocular, donde puede producirse una deriva de escala [50], la información estéreo o de profundidad hace que la escala

sea observable y la validación geométrica y la optimización del gráfico de posición ya no requieren tratar con la deriva de escala y se basan en transformaciones de cuerpo rígido en lugar de similitudes.

En ORB-SLAM2, es incorporado una optimización BA completa después del gráfico de posición para lograr la solución óptima. Esta optimización puede ser muy costosa computacionalmente y, por lo tanto, es realizada como una tarea independiente ejecutada igualmente en paralelo con las demás, lo que permite que el algoritmo continúe creando mapas y detectando trayectorias cerradas. Si se detecta una nueva trayectoria cerrada mientras se ejecuta el BA completo, esta es abortada, lo que la iniciará nuevamente.

3.4. Gráfico de Covisibilidad y Gráfico Esencial

La información de covisibilidad entre fotogramas clave es muy útil en varias tareas de que se desarrollan en el algoritmo y se representa como un gráfico ponderado como en [45]. Cada nodo es un fotograma clave, y existe un borde entre dos fotogramas clave si comparten observaciones de los mismos puntos del mapa (al menos 15), siendo el peso θ del borde el número de puntos comunes del mapa. Para no incluir todos los bordes proporcionados por el gráfico de covisibilidad, que puede ser muy denso, se propone construir un gráfico esencial que conserve todos los nodos (fotogramas clave), pero menos bordes, conservando una red sólida que produzca resultados precisos. El algoritmo construye de forma incremental un árbol de expansión desde el fotograma clave inicial, que proporciona un subgrafo conectado del gráfico de covisibilidad con un número mínimo de bordes. Cuando se inserta un nuevo fotograma clave, se incluye en el árbol vinculado al fotograma clave que comparte la mayoría de las observaciones de puntos del mapa, y cuando la tarea de eliminación borra un fotograma clave, el sistema actualiza los enlaces afectados por ese fotograma clave. El gráfico esencial contiene el árbol de expansión, el subconjunto de bordes del gráfico de covisibilidad con alta covisibilidad ($\theta_{min} = 100$) y los bordes de trayectorias cerradas.

3.5. Reconocimiento del lugar

El algoritmo tiene incorporado un módulo de reconocimiento de lugar basado en un vocabulario de palabras, basado en DBoW22 [51], para realizar la detección de trayectorias cerradas y la reubicación. Las palabras visuales son solo una discretización del espacio del descriptor, que se conoce como vocabulario visual. El vocabulario se crea sin conexión con los descriptores ORB extraídos de un gran conjunto de imágenes. Si las imágenes son lo suficientemente generales, se puede utilizar el mismo vocabulario para diferentes entornos obteniendo un buen rendimiento. El algoritmo construye de

forma incremental una base de datos que contiene un índice invertido, que almacena para cada palabra visual en el vocabulario, en cuales fotogramas clave se ha visto, de modo que la consulta de la base de datos se puede realizar de manera muy eficiente. La base de datos también se actualiza cuando el procedimiento de eliminación elimina un fotograma clave.

3.6. Conclusiones del capítulo

En el capítulo se describió el principio de funcionamiento del algoritmo de visión artificial ORB-SLAM2, mediante diagramas, se mostraron las principales tareas que se realizan de manera simultánea en los diferentes módulos que componen el algoritmo. Una vez conocidas las nociones principales de su funcionamiento, será posible utilizar los paquetes de ROS implementados del algoritmo para poder ser ejecutado en la computadora embebida en el vehículo. En el siguiente capítulo se describen los componentes que conforman la plataforma experimental.

Capítulo 4

Plataforma Experimental

En el presente capítulo se describe la plataforma experimental que se utilizará, exponiendo los principales sistemas que la conforman, todos los parámetros e imágenes fueron obtenidos de sus respectivas hojas de datos. La plataforma consiste en un helicóptero miniatura de cuatro rotores en configuración X.

En las figuras 4.1 y 4.2 se muestran imágenes de diferentes vistas de la plataforma experimental y de su modelo realizado en el software Solid Works.

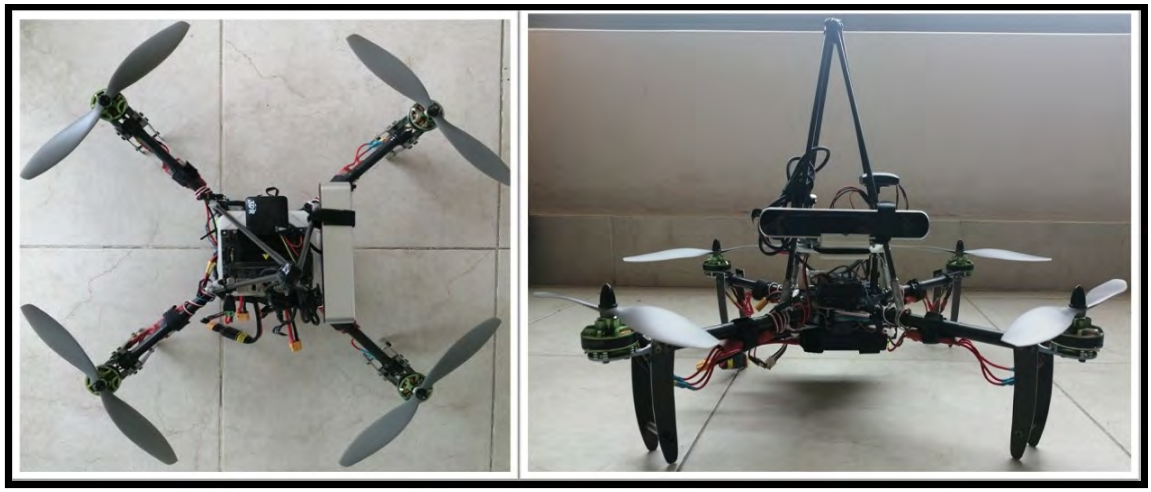

Figura 4.1: Plataforma experimental. Vistas superior (izquierda) y frontal (derecha).

Figura 4.2: Modelo CAD de la plataforma experimental en Solid Works. Vistas superior (arriba) y frontal (abajo).

En la Figura 4.3 se pueden observar todos los componentes que conforman la plataforma. Todas las imágenes fueron cortesía de los respectivos sitios web de cada fabricante.

4.1. Sistema de propulsión

Para el sistema de propulsión se utilizan los motores sin escobillas Turnigy Multistar 4108-600 [52]. La descripción de sus principales características se muestra a continuación.

- KV (RPM/V): 600.
- Células Lipo: 4-6s.
- Potencia máxima: 400w.
- Corriente máxima: 26A.
- Sin corriente de carga: 0.9A / 10V.

Figura 4.3: a) Motor Turnigy Multistar 4108-600. b) Batería Graphene Lipo. c) Afro 30A Muti-Rotor ESC. d) Pixhawk versión 2.4.8. e) GPS 3DR uBlox. f) Spektrum AR-8000. g) Jetson Nano. h) Cámara estereoscópica ZED. i) Spektrum DX8.

- Resistencia interna: 0.098ohm.
- Dimensiones (Diámetro x Longitud): 47 x 26 mm.

4.2. Sistema de alimentación

Se utiliza para el sistema de alimentación la batería Graphene Lipo con una sola capa de carbono [53]. Las principales especificaciones se destacan a continuación.

- Capacidad: 4000 mAh.
- Voltaje: 14.8 V, 4 celdas.
- Descarga: 45C.
- Dimensiones: 144 x 51 x 34 mm.
- Conector: xt-90.

4.3. Variadores de velocidad

Los variadores de velocidad o ESC utilizados son los Afro 30A Muti-Rotor ESC (SimonK Firmware) [54], sus principales características se definen como sigue.

- Corriente: 30A.
- Rango de voltaje: 2-4s Lipoly.
- Frecuencia de entrada: 1KHz.
- Firmware: afro_nfet.hex.
- Descarga cable/conector: 15AWG/Macho 3.5 mm.
- Motor cable/conector: 16AWG/Hembra 3.5 mm.
- Dimensiones: 50 x 25 x 11 mm.

4.4. Controlador de vuelo

Como controlador de vuelo se utiliza un Pixhawk versión 2.4.8 [55], cuyas especificaciones principales se exponen a continuación.

- Procesador: 32 bit 2M memoria flash STM32F427 Cortex M4.
- Frecuencia principal: 256K, 168MHZ RAM.
- Sensores:
  - Giroscopio: L3GD20 Digital 3 ejes 16 bit.
  - Acelerómetro/Magnetómetro: LSM303D 3 ejes 14 bit.
  - Acelerómetro/Magnetómetro: MPU6000 6 ejes.
  - Barómetro: MS5611 de alta precisión.
- Interfaces:
  - 5 UART, 1 compatible de alto voltaje, 2 control de flujo del hardware.
  - Entrada compatible para receptor Spektrum DSM/DSM2/DSM-X.
  - Entrada compatible para receptor Futaba SBUS.
  - Entrada para señal PPM.
  - Entrada RSSI (PWM o voltage).
  - Protocolo I2C.
  - Protocolo SPI.
  - Interfaz externa Micro USB.

4.5. Sistema GPS

Para el sistema de ubicación se utiliza el GPS 3DR uBlox con compás incluido [56]. Sus especificaciones son las siguientes.

- Actualización: 5Hz.
- Antena cerámica de: 25 x 25 x 4 mm.
- Filtros: LNA y SAW.
- Batería recargable: 3V litio.
- Pines: RX, TX, 5V y GND.
- Regulador de bajo ruido de 3.3V.
- Dimensiones: 38 x 38 x 8.5 mm.

4.6. Radio-receptor

El radio-receptor utilizado es el Spektrum AR-8000 de 8 canales [57]. Sus especificaciones son las siguientes.

- Voltaje: 3.5–9.6V.
- Banda: 2.4GHz.
- Modulación: DSM2/DSMX.
- Canales: 8.
- Dimensiones: 34.3 x 32.3 mm.

4.7. Radio-control

El radio-control utilizado es el Spektrum DX8 [57], se resumen sus características a continuación.

- Programación governor.
- Ajuste para governor activo.
- Ajuste para giroscopio activo.
- Gráfico de 5 puntos para curvas throttle y pitch.
- Sincronización de plato cíclico.
- 6 tipos de platos: 1 Servo Normal, 2 Servos 180°, 3 Servos 90°, 3 Servos 120°, 3 Servos 135°, 3 Servos 140°.
- Anillo electrónico.

4.8. Computadora embebida

Para la ejecución e implementación del algoritmo se utilizó la computadora embebida Jetson Nano [58], se resumen sus características a continuación.

- GPU: 128-core Maxwell
- CPU: Quad-core ARM A57 @ 1.43 GHz
- Memoria: 4 GB 64-bit LPDDR4 25.6 GB/s
- Almacenamiento: microSD 64 GB
- Conectividad: Gigabit Ethernet, M.2 Key E, HDMI y display port, 4x USB 3.0, USB 2.0 Micro-B, GPIO, I^2C, I^2S, SPI, UART

4.9. Sistema de percepción visual

Como sistema de percepción visual se utilizó la cámara estereoscópica ZED [59], cuyas principales características se presentan a continuación.

- Captura de vídeo 3D de alta resolución y velocidad de cuadros.
- Percepción de profundidad en interiores y exteriores de hasta 20 m.
- Seguimiento posicional de 6 grados de libertad.
- Mapeo espacial.
- Vídeo
    - Modo: 2.2K, FPS: 15, Resolución: 4416x1242.
    - Modo: 1080p, FPS: 30, Resolución: 3840x1080.
    - Modo: 720p, FPS: 60, Resolución: 2560x720.
    - Modo: WVGA, FPS: 100, Resolución: 1344x376.
- Rango de profundidad: 0.5 - 20 m (1.64 to 65 ft).
- Formato de profundidad: 32-bits.
- Linea base estéreo: 120 mm (4.7”).
- Precio: 8700 pesos mexicanos aproximadamente.

4.10. Conclusiones del capítulo

En el capítulo en cuestión se describieron los principales componentes que conforman la plataforma experimental con la cual se realizarán las pruebas experimentales. Además, fue necesario agregar una batería adicional con el objetivo de alimentar la computadora embebida de manera independiente y así evitar ruidos en el voltaje generados por los motores. También, se disminuyó el voltaje de entrada a la Jetson Nano a 5V mediante un UBEC. Igualmente, se diseñó en SolidWorks e imprimió mediante una impresora 3D los soportes necesarios para sujetar la cámara y la computadora. En el siguiente capítulo se muestran los principales resultados numéricos y experimentales obtenidos.

Capítulo 5

Resultados

En este capítulo se presentan los principales resultados numéricos obtenidos durante la investigación, a partir de las simulaciones desarrolladas, además, se desarrollan los resultados experimentales a partir de pruebas realizadas sobre la plataforma.

5.1. Simulación de un controlador PD para la estabilización

El controlador PD tiene la siguiente forma [60],

$$u = K_p e + K_d \dot{e}.$$

Partiendo del sistema (A.8), aplicando una aproximación de ángulos pequeños ($\theta \approx 0 \Rightarrow \sin\theta \approx \theta, \phi \approx 0 \Rightarrow \sin\phi \approx \phi$) y considerando que el ángulo de guiñada se fija en la posición deseada cero ($\psi_d = 0 \Rightarrow \sin\psi_d \approx 0, \cos\psi_d \approx 1$). Además, cuando el vehículo está suspendido en el aire compensa la gravedad, por tanto se reduce el sistema a las siguientes ecuaciones,

$$\begin{aligned}
\ddot{x} &= g\theta, \\
\ddot{y} &= -g\phi, \\
\ddot{z} &= \frac{1}{m}(u - mg), \\
\ddot{\phi} &= \tilde{\tau_\phi}, \\
\ddot{\theta} &= \tilde{\tau_\theta}, \\
\ddot{\psi} &= \tilde{\tau_\psi}.
\end{aligned} \tag{5.1}$$

En el caso del control en altura z, para hacer que $z \to z_d \Rightarrow e_z = z_d - z \to 0$, esto se logra con $u = mg + k_{pz}e_z + k_{dz}\dot{e_z}$. Por lo tanto, el sistema en lazo cerrado sería,

$$\ddot{z} = \frac{1}{m}(k_{pz}e_z + k_{dz}\dot{e_z}),$$

además, considerando aceleraciones pequeñas ($\ddot{z_d} \approx 0$) quedaría de la siguiente forma,

$$\ddot{e_z} = -\frac{k_{pz}}{m}e_z - \frac{k_{dz}}{m}\dot{e_z}.$$

Definiendo como variables de estado $e_z = e_{z1}, \dot{e_z} = e_{z2}$,

$$\begin{bmatrix} \dot{e_{z1}} \\ \dot{e_{z2}} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k_{pz}}{m} & -\frac{k_{dz}}{m} \end{bmatrix}}_{A_z} \begin{bmatrix} e_{z1} \\ e_{z2} \end{bmatrix}.$$

Calculando los valores propios de la matriz A_z,

$$\mid (sI - A_z) \mid = s^2 + \frac{k_{dz}}{m}s + \frac{k_{pz}}{m}.$$

Aplicando el criterio de Routh - Hurwitz [61],

$$\begin{array}{ccc} s^2 & 1 & \frac{k_{pz}}{m} \\ s^1 & \frac{k_{dz}}{m} & 0 \\ s^0 & \frac{k_{pz}k_{dz}}{m^2} & \end{array}$$

$$\frac{k_{dz}}{m} > 0, \frac{k_{pz}k_{dz}}{m^2} > 0.$$

Puesto que k_{pz}, k_{dz}, m son constantes positivas entonces el sistema es exponencialmente estable y $\lim_{t\to\infty} e_z \to 0 \Rightarrow z \to z_d$.

Para el control en traslación x e y, suponiendo que las dinámicas de orientación son suficientemente rápidas, entonces $\theta \to \theta_d$ y $\phi \to \phi_d$ rápidamente. Por tanto, $\theta_d = u_x, \phi_d = u_y$, pueden utilizarse como entradas de control virtuales. Para que $x \to x_d \Rightarrow e_x = x_d - x \to 0$, de igual forma, $y \to y_d \Rightarrow e_y = y - y_d \to 0$, esto se logra con,

$$u_x = k_{px}e_x + k_{dx}\dot{e_x}$$

$$u_y = k_{py}e_y + k_{dy}\dot{e_y}$$

Siguiendo la misma metodología y tomando las mismas consideraciones que en el caso del control en altura, se obtienen los siguientes sistemas en espacio de estado para los ejes x e y,

$$\begin{bmatrix} \dot{e_{x1}} \\ \dot{e_{x2}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -gk_{px} & -gk_{dx} \end{bmatrix} \begin{bmatrix} e_{x1} \\ e_{x2} \end{bmatrix},$$

$$\begin{bmatrix} \dot{e_{y1}} \\ \dot{e_{y2}} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -gk_{py} & -gk_{dy} \end{bmatrix} \begin{bmatrix} e_{y1} \\ e_{y2} \end{bmatrix}.$$

Calculando los valores propios y aplicando el criterio de estabilidad de Routh - Hurwitz se obtienen las condiciones $gk_{dx} > 0, g^2k_{dx}k_{px} > 0, gk_{dy} > 0, g^2k_{dy}k_{py} > 0$. Puesto que $k_{px}, k_{dx}, k_{py}, k_{dy}, g$ son constantes positivas entonces el sistema es exponencialmente estable y $\lim_{t\to\infty} e_x \to 0 \Rightarrow x \to x_d, \lim_{t\to\infty} e_y \to 0 \Rightarrow y \to y_d$.

Para la orientación se sigue el mismo procedimiento, definiendo,

$$\tau_\psi = -k_{p\psi}\psi - k_{d\psi}\dot{\psi},$$

$$\tau_\phi = -k_{p\phi}\phi - k_{d\phi}\dot{\phi},$$

$$\tau_\theta = -k_{p\theta}\theta - k_{d\theta}\dot{\theta}.$$

Por lo tanto, la representación en espacio de estado se define como,

$$\begin{bmatrix} \dot{\psi_1} \\ \dot{\psi_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\psi} & -k_{d\psi} \end{bmatrix} \begin{bmatrix} \psi_1 \\ \psi_2 \end{bmatrix},$$

$$\begin{bmatrix} \dot{\phi_1} \\ \dot{\phi_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\phi} & -k_{d\phi} \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix},$$

$$\begin{bmatrix} \dot{\theta_1} \\ \dot{\theta_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\theta} & -k_{d\theta} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}.$$

Calculando los valores propios y aplicando el criterio de estabilidad de Routh - Hurwitz se obtienen las condiciones $k_{d\psi} > 0, k_{p\psi}k_{d\psi} > 0, k_{d\phi} > 0, k_{p\phi}k_{d\phi} > 0, k_{d\theta} > 0, k_{p\theta}k_{d\theta} > 0$. Puesto que $k_{p\psi}, k_{d\psi}, k_{p\phi}, k_{d\phi}, k_{p\theta}, k_{d\theta}$ son constantes positivas, de igual forma los sistemas en orientación son exponencialmente estables.

Con las leyes de control Proporcionales-Derivativas PD, se procede a implementarlas en una simulación del modelo matemático del vehículo en el software Simulink. El diagrama en bloques realizado para simular los controladores se puede observar en la Figura 5.10, pero únicamente sustituyendo la referencia sigmoidal por cero y el control por saturaciones anidadas por un PD.

Se fijó como altura deseada 5 m y como posiciones y orientaciones iniciales $x_0 = 1\,m$, $y_0 = 2\,m$, $z_0 = 2\,m$, $\theta_0 = 0\,rad$, $\phi_0 = 0\,rad$ y $\psi_0 = 0.5\,rad$. Luego de ejecutada la simulación con tiempo de muestreo de 0.001 segundos y método numérico Runge Kutta de cuarto orden (ode4), se obtuvieron los siguientes resultados tanto para la posición como la orientación del vehículo, además de las señales de control.

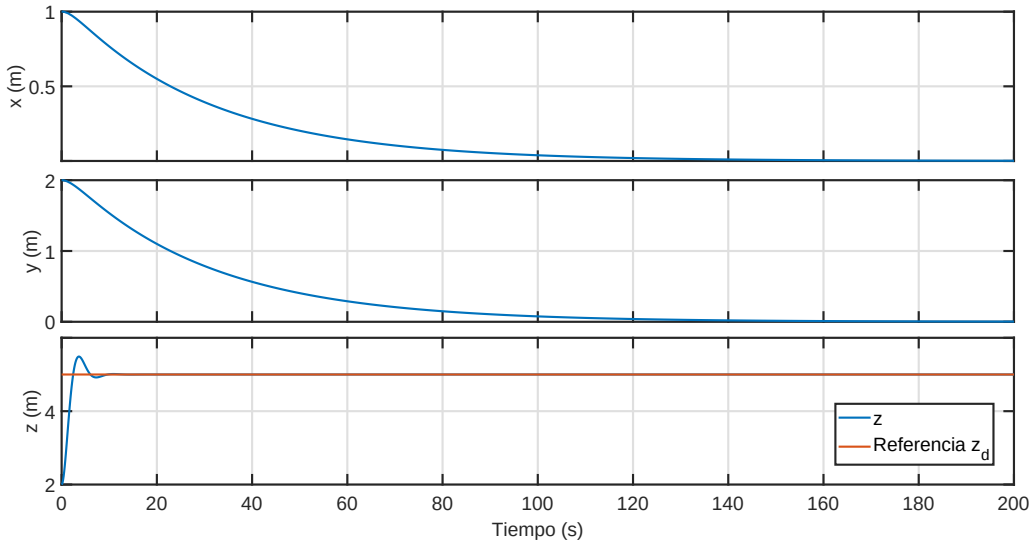

Figura 5.1: Posiciones x, y y z respectivamente. Estabilización con controladores PD.

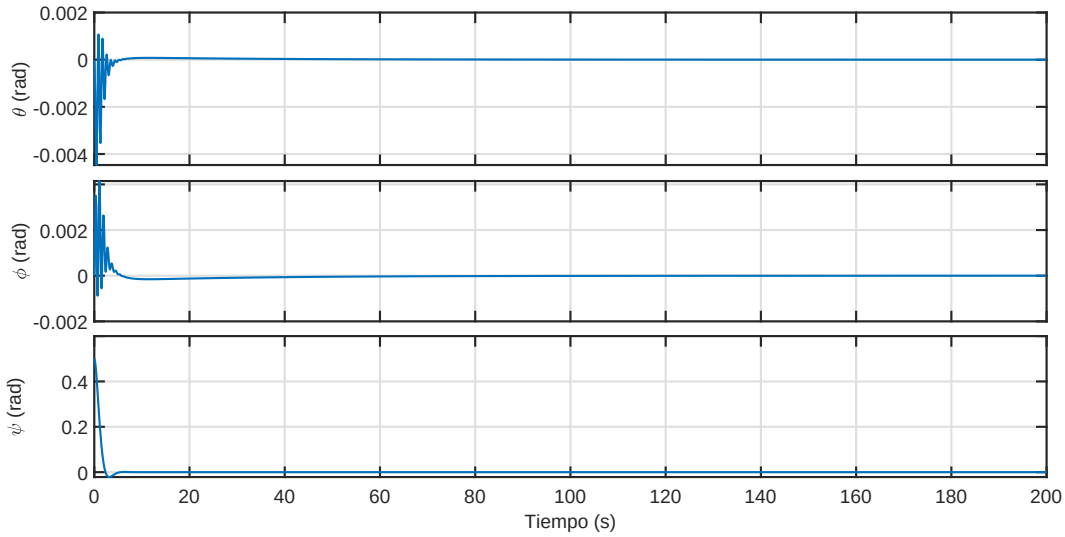

Figura 5.2: Ángulos de Euler θ, ϕ y ψ respectivamente. Estabilización con controladores PD.

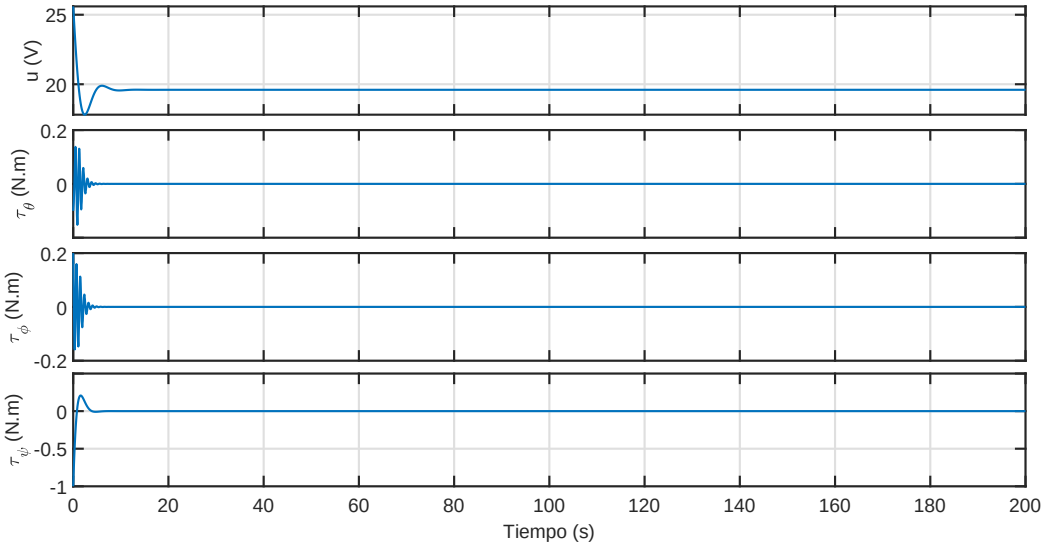

Figura 5.3: Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Estabilización con controladores PD.

Como se puede observar en la Figura 5.1 tanto las posiciones x e y se estabilizan en el origen $(x, y = 0)$, además, la posición z converge a la altura deseada, destacando que presentan un mayor tiempo de establecimiento en comparación con el controlador por saturaciones anidadas. También, en la Figura 5.2, se garantiza la estabilización de la orientación del vehículo para cada uno de los ángulos de Euler. Por último, las respectivas señales de control son ilustradas en la Figura 5.2.

5.2. Simulación de un controlador PD para seguimiento de trayectorias

A partir de leyes de control Proporcionales-Derivativas PD para la orientación, traslación y altura, se realizan las siguientes simulaciones numéricas del modelo matemático del vehículo en el software Simulink.

Se fijó como altura deseada $5\,m$ y como posiciones y orientaciones iniciales $x_0 = 10\,m$, $y_0 = 10\,m$, $z_0 = 0\,m$, $\theta_0 = 0\,rad$, $\phi_0 = 0\,rad$ y $\psi_0 = 0.5\,rad$. Además, las trayectorias a seguir en los ejes x e y fueron sigmoides de amplitud $20\,m$. Luego de ejecutada la simulación con los mismos parámetros descritos en la sección anterior, se obtuvieron los siguientes resultados tanto para la posición como la orientación del vehículo, además de las señales de control.

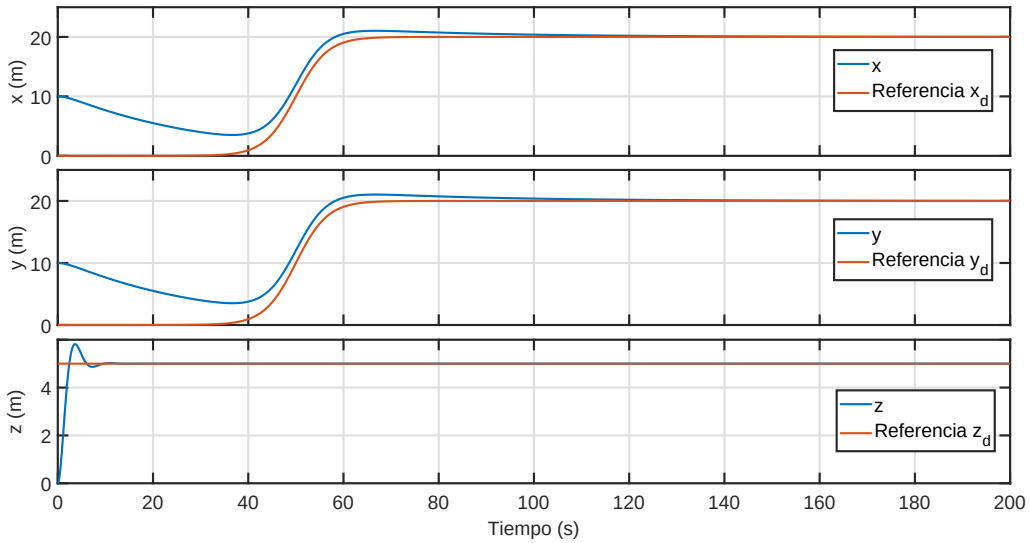

Figura 5.4: Posiciones x, y y z respectivamente. Seguimiento de trayectorias con controladores PD.

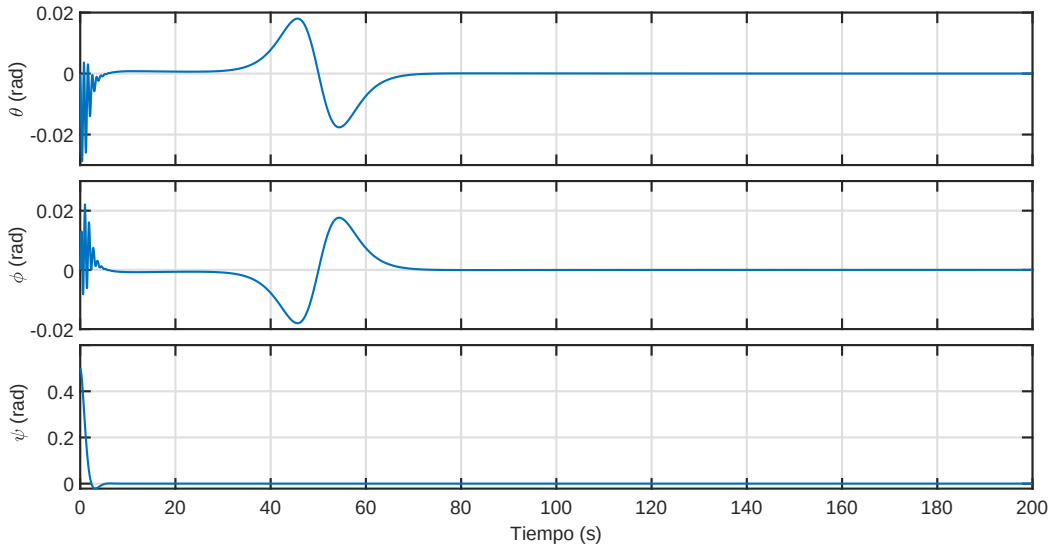

Figura 5.5: Ángulos de Euler θ, ϕ y ψ respectivamente. Seguimiento de trayectorias con controladores PD.

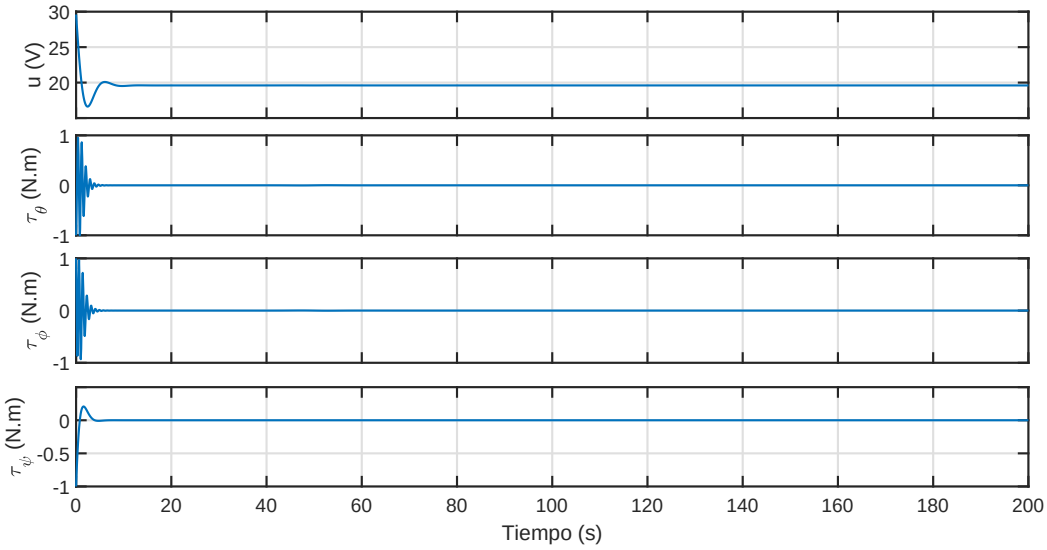

Figura 5.6: Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Seguimiento de trayectorias con controladores PD.

Como se puede observar en la Figura 5.4 tanto las posiciones x e y parten de sus respectivas posiciones iniciales y al trascurrir el tiempo convergen a la referencias deseadas, siguiendo las sigmoides, pero se puede apreciar un error de seguimiento que los controladores PD no logran corregir de manera exacta. También, en la Figura 5.5, se garantiza la estabilización de la orientación del vehículo para cada uno de los ángulos de Euler, observándose cómo responden los controles en traslación de manera directa sobre la orientación.

5.3. Simulación de un controlador no lineal por saturaciones anidadas para la estabilización

A partir del modelo matemático expresado por las ecuaciones (A.8), se aplicó una ley de control por saturaciones anidadas [62] para el control de la orientación y posición del vehículo, además de la altura. Los controles obtenidos a partir de [62] se muestran a continuación.

Para controlar la altura se propone el control

$$u = \frac{r_1 + mg}{\cos\phi\cos\theta}$$

definiendo

$$r_1 = -k_{d1}\dot{z} - k_{p1}(z - z_d)$$

donde k_{d1} y k_{p1} son constantes del controlador y z_d la altura deseada.

Para mantener el ángulo de guiñada ψ en cero se propone el control

$$\tilde{\tau}_\psi = -k_{d2}\dot{\psi} - k_{p2}\psi$$

donde k_{d2} y k_{p2} son constantes del controlador.

En el caso del control de la posición en el eje x es necesario el control del ángulo de cabeceo θ, para mantener dicha posición en cero se propone la siguiente ley de control

$$\tilde{\tau}_\theta = -\sigma_{\theta 1}(\dot{\theta} + \sigma_{\theta 2}(\theta + \dot{\theta} + \sigma_{\theta 3}(2\theta + \dot{\theta} - \frac{\dot{x}}{g} + \sigma_{\theta 4}(\dot{\theta} + 3\theta - 3\frac{\dot{x}}{g} - \frac{x}{g}))))$$

donde $\sigma_{\theta 1}, \sigma_{\theta 2}, \sigma_{\theta 3}, \sigma_{\theta 4}$ son las constantes de las funciones de saturación.

Para controlar el par $(\phi - y)$, es decir, controlar la posición en el eje y a partir del ángulo de alabeo ϕ, se propone la siguiente ley de control para llevar la posición a cero

$$\tilde{\tau}_\phi = -\sigma_{\phi 1}(\dot{\phi} + \sigma_{\phi 2}(\phi + \dot{\phi} + \sigma_{\phi 3}(2\phi + \dot{\phi} + \frac{\dot{y}}{g} + \sigma_{\phi 4}(\dot{\phi} + 3\phi + 3\frac{\dot{y}}{g} + \frac{y}{g}))))$$

donde $\sigma_{\phi 1}, \sigma_{\phi 2}, \sigma_{\phi 3}, \sigma_{\phi 4}$ son las constantes de las funciones de saturación.

Con las leyes de control obtenidas, se procede a implementarlas en una simulación del modelo matemático del vehículo en el software Simulink. El diagrama en bloques realizado para simular los controladores se puede observar en la Figura 5.10, pero únicamente sustituyendo la referencia sigmoidal por cero.

Se fijaron las referencias de posiciones y condiciones iniciales con los mismos valores del experimento de estabilización con controladores PD. Igualmente, se utilizaron los parámetros de simulación del experimento anterior, se obtuvieron los siguientes resultados tanto para la posición como la orientación del vehículo, además de las señales de control.

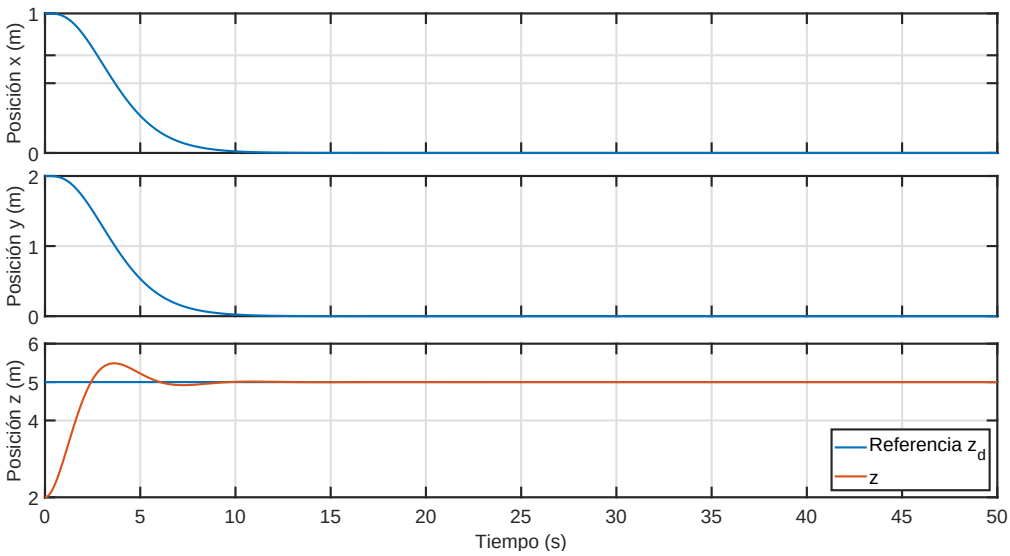

Figura 5.7: Posiciones x, y y z respectivamente. Estabilización con controladores por saturaciones anidadas.

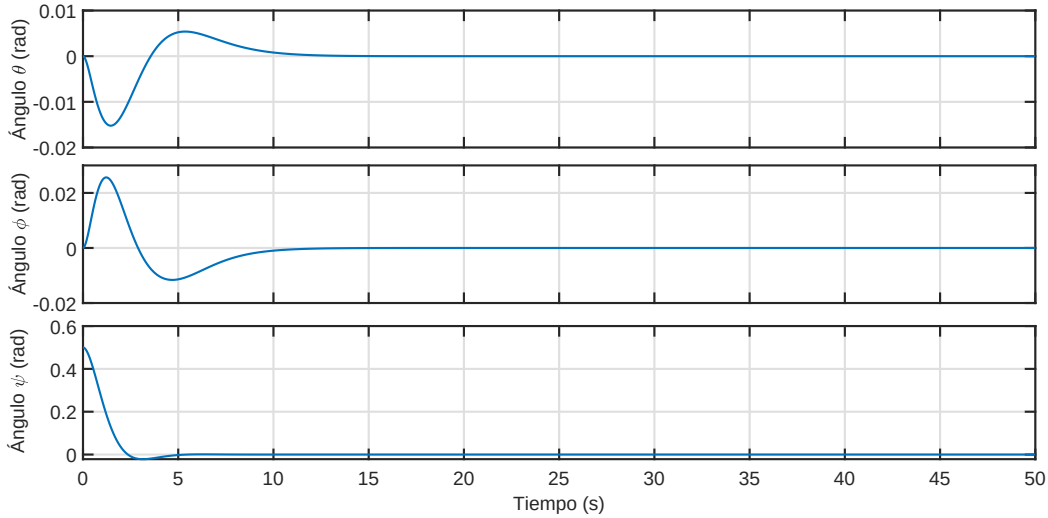

Figura 5.8: Ángulos de Euler θ, ϕ y ψ respectivamente. Estabilización con controladores por saturaciones anidadas.

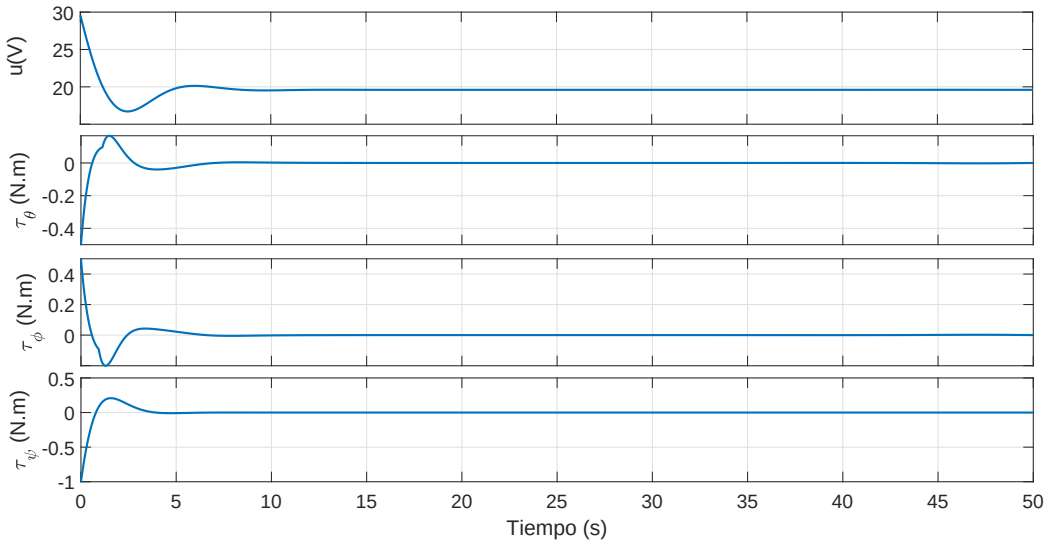

Figura 5.9: Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Estabilización con controladores por saturaciones anidadas.

Como se puede observar en la Figura 5.7 tanto las posiciones x e y se estabilizan en el origen $(x, y = 0)$, además, la posición z converge a la altura deseada. También, en la Figura 5.8, se garantiza la estabilización de la orientación del vehículo para cada uno de los ángulos de Euler. Por último, las respectivas señales de control son ilustradas en la Figura 5.9, cuyos valores se encuentran saturados debido al controlador no lineal aplicado de saturaciones anidadas, permitiendo a cambios grandes de las señales de error acciones de control menos bruscas.

5.4. Simulación de un controlador no lineal por saturaciones anidadas para seguimiento de trayectorias

Siguiendo la misma metodología desarrollada en [62], definiendo las señales de error como sigue:

$$e_x = \frac{1}{g}(x - x_d)$$

$$e_y = \frac{1}{g}(y - y_d)$$

y sus respectivas derivadas. Se obtienen las siguientes leyes de control para la traslación, los controles en altura y guiñada se mantienen como en el caso de estabilización.

$$\tilde{\tau}_\phi = -\sigma_{\phi 1}(\dddot{e}_y + \sigma_{\phi 2}(\ddot{e}_y + \dddot{e}_y + \sigma_{\phi 3}(\dddot{e}_y + 2\,\ddot{e}_y + \dot{e}_y + \sigma_{\phi 4}(\dddot{e}_y + 3\,\ddot{e}_y + 3\,\dot{e}_y + e_y))))$$

$$\tilde{\tau}_\theta = -\sigma_{\theta 1}(\dddot{e}_x + \sigma_{\theta 2}(\ddot{e}_x + \dddot{e}_x + \sigma_{\theta 3}(\dddot{e}_x + 2\,\ddot{e}_x - \dot{e}_x + \sigma_{\theta 4}(\dddot{e}_x + 3\,\ddot{e}_x - 3\,\dot{e}_x - e_x))))$$

.

Con las leyes de control obtenidas, se procede a implementarlas en una simulación del modelo matemático del vehículo en el software Simulink. El diagrama en bloques realizado para simular los controladores se puede observar en la Figura 5.10.

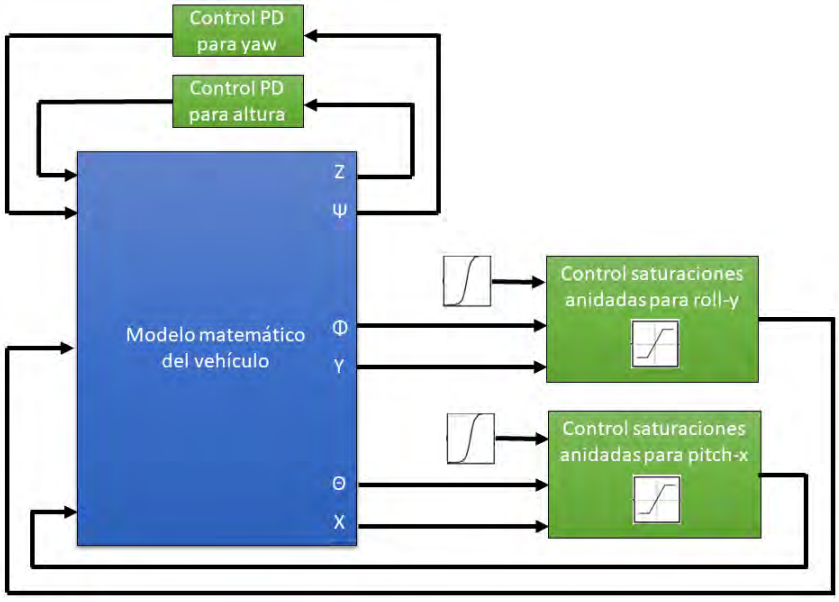

Figura 5.10: Diagrama de bloques realizado para la simulación.

Se fijaron las referencias de posiciones como sigmoides y condiciones iniciales con los mismos valores del experimento de seguimiento de trayectorias con controladores PD. Igualmente, se utilizaron los parámetros de simulación del experimento anterior, se obtuvieron los siguientes resultados tanto para la posición como la orientación del vehículo, además de las señales de control.

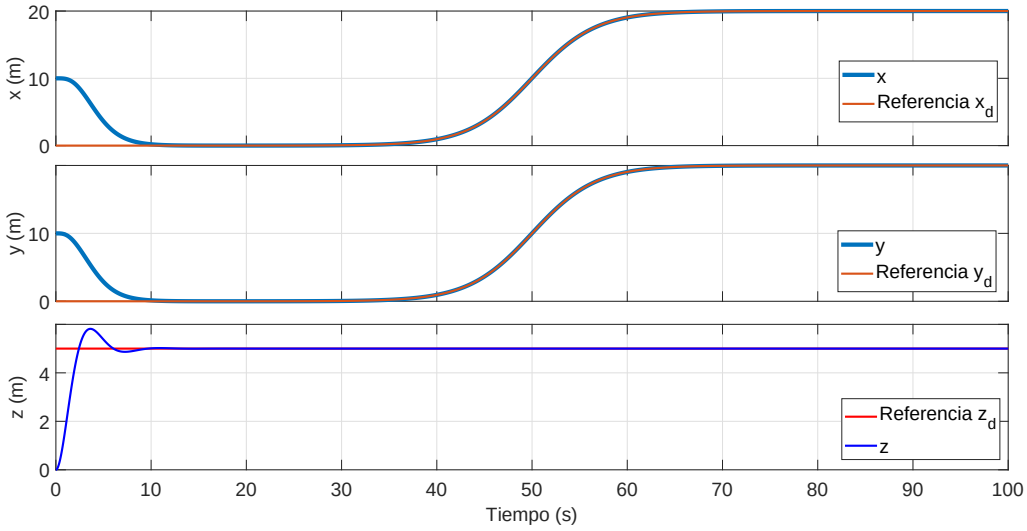

Figura 5.11: Posiciones x, y y z respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas.

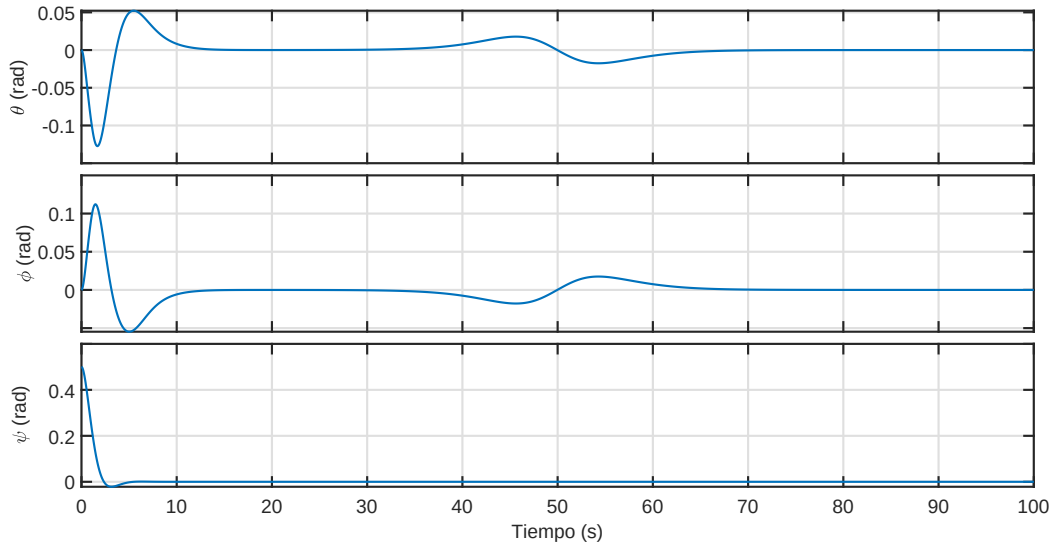

Figura 5.12: Ángulos de Euler θ, ϕ y ψ respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas.

Como se puede observar en la Figura 5.11 tanto las posiciones x e y parten de sus respectivas posiciones iniciales y al trascurrir el tiempo convergen a la referencias deseadas, siguiendo las sigmoides. También, en la Figura 5.12, se garantiza la estabilización de la orientación del vehículo para cada uno de los ángulos de Euler.

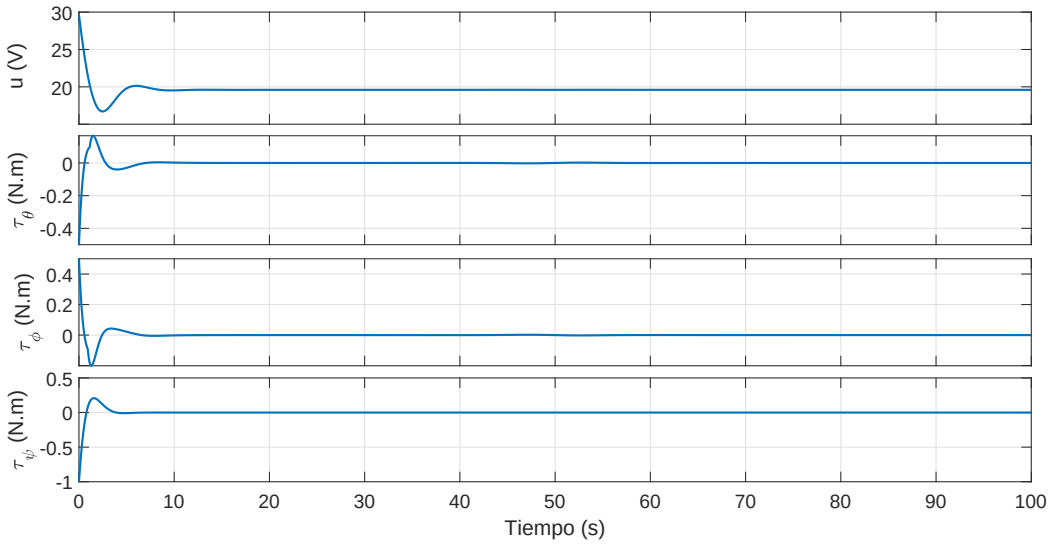

Figura 5.13: Señales de control u, τ_θ, τ_ϕ y τ_ψ respectivamente. Seguimiento de trayectorias con controladores por saturaciones anidadas.

Con las simulaciones numéricas realizadas se pudo observar un mejor desempeño del controlador no lineal por saturaciones anidadas sobre los controladores convencionales PD, principalmente en el caso del seguimiento de trayectorias.

5.5. Funcionamiento del algoritmo ORB-SLAM2 en una computadora portátil

También se realizaron pruebas del algoritmo de visión ORB-SLAM2 sobre una computadora portátil y utilizando una cámara estereoscópica ZED de StereoLabs. A continuación, se muestran las principales características de la computadora.

- CPU: Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz.
- Memoria RAM: 8.00 GB.
- GPU: nVIDIA GTX 1050 Ti.
- Sistema Operativo: Ubuntu 16.04.4 LTS.
- ROS: Kinetic.

Para poder utilizar dicha cámara sobre la plataforma Linux es necesario tener instalados los siguientes programas en las versiones recomendadas:

- NVIDIA Vídeo Drivers versión 384 (para Ubuntu 16.04.4 LTS). Controladores de la tarjeta de vídeo NVIDIA en la versión recomendada.
- CUDA 9.0 (para Ubuntu 16.04.4 LTS). Plataforma de computación en paralelo incluyendo un compilador y un conjunto de herramientas de desarrollo creadas por nVIDIA que permiten a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPU.
- ZED SDK 2.8.3 (para Ubuntu 16.04.4 LTS). Kit de desarrollo de software de la cámara ZED.
- ZED ROS wrapper (para Ubuntu 16.04.4 LTS y ROS Kinetic). Librería que funciona como un adaptador entre el sistema operativo y la cámara ZED.

Una vez configurados todos los programas necesarios se procede a crear un archivo launch para ejecutar el algoritmo. Para ello, se creó un nuevo nodo para el algoritmo ORB-SLAM2 estéreo, consecuentemente se remapearon los tópicos mediante los cuales los nodos publicadores de la cámara publican las imágenes del lente derecho e izquierdo rectificadas con los tópicos correspondientes del algoritmo. Además, se conectó el tópico de la cámara que publica los parámetros de calibración opencv tales como distancia focal y distorsión. Por último, se fijaron los parámetros ORB, el número de características por imagen, el factor de escala, entre otros. Existen dos grupos de parámetros, dinámicos, es decir, que pueden ser modificados en línea, y parámetros estáticos, que solo se modifican una vez creado el nodo. A continuación, se explica en que consiste cada uno.

- Parámetros estáticos
    - load_map: Tipo 'Bool'. Si es declarado como verdadero el nodo intentará cargar al inicio el mapa cuyo nombre se especificó en el parámetro map_file.
    - map_file: Tipo 'String'. El nombre del archivo en el cual se guarda el mapa creado.
    - settings_file: Tipo 'String'. La localización del archivo de configuraciones de la cámara.
    - voc_file: Tipo 'String'. La ubicación del archivo de vocabulario de configuración.
    - publish_pose: Tipo 'Bool'. Si se debe publicar un mensaje PoseStamped, que contiene la posición estimada de la cámara.
    - publish_pointcloud: Tipo 'Bool'. Si la nube de puntos que contiene todos los puntos claves (el mapa) debería publicarse.
    - pointcloud_frame_id: Tipo 'String'. El identificador del fotograma de la nube de puntos / mapa.

- camera_frame_id: Tipo 'String'. El identificador del fotograma de la posición de la cámara.

- Parámetros dinámicos

  - localize_only: Tipo 'Bool'. Activar el modo de solo localización. El SLAM ya no agregará nuevos puntos al mapa.
  - reset_map: Tipo 'Bool'. Activar para borrar el mapa y comenzar de nuevo. Después de restablecer, el parámetro se actualizará automáticamente a falso.
  - min_num_kf_in_map: Tipo 'Int'. Número de fotogramas claves que debe tener un mapa para que no se restablezca después de perder el seguimiento.

En la Figura 5.14 se puede observar un diagrama de como se comunican los nodos publicadores y suscriptores a través de los respectivos tópicos, para lograr el funcionamiento correcto del algoritmo con la cámara ZED.

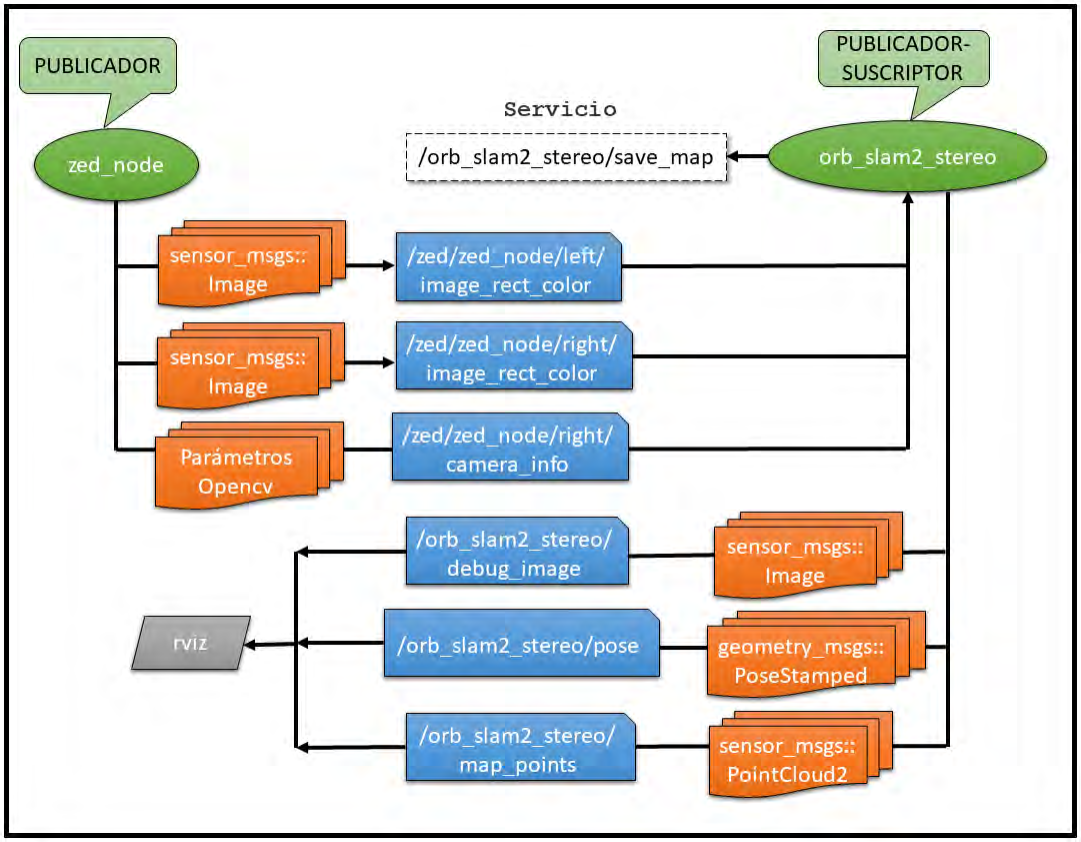

Figura 5.14: Diagrama de comunicación de los nodos en ROS.

Como se puede observar, el nodo *zed_node* publica los mensajes del tipo imagen para cada uno de los lentes de la cámara vía sus respectivos tópicos */zed/zed_node/left/image_rect_color, /zed/zed_node/right/image_rect_color*, dichos mensajes son recibidos por el nodo *orb_slam2_stereo* que se encuentra suscrito a ambos tópicos. Una vez disponibles las imágenes rectificadas de cada lente en el algoritmo, se procesan todas las tareas programadas en el algoritmo ORB-SLAM2 y finalmente los resultados son publicados en cada tópico. El algoritmo publica tres mensajes diferentes mediante tres tópicos respectivamente. Mediante el tópico */orb_slam2_stereo/debug_image* se publica un mensaje tipo imagen donde se muestran las características y puntos claves detectados por el algoritmo para cada fotograma. El tópico */orb_slam2_stereo/pose* publica mensajes del tipo Pose, en los cuales se encuentra información relacionada con la posición (X, Y, Z) y orientación (ψ, ϕ, θ) de la cámara en el espacio. Por último, un mensaje del tipo PointCloud2 que contiene una nube de puntos con los puntos del mapa creados, es publicado mediante el tópico */orb_slam2_stereo/map_points*.

En todo momento el nodo de ORB-SLAM2 dispone de un servicio el cual permite guardar el mapa creado hasta el momento en un archivo .bin, por defecto el mapa es guardado en la dirección de la variable de entorno de ROS *ROS_HOME*. Posteriormente, el mapa guardado puede ser cargado nuevamente para realizar tareas de localización.

Para la visualización de todo el proceso de creación del mapa y localización en el mismo, se utiliza la herramienta de ROS *rviz*. Mediante la herramienta se vizualizará la información publicada en los tópicos */orb_slam2_stereo/debug_image*, */orb_slam2_stereo/pose* y */orb_slam2_stereo/map_points*, con lo anterior se tiene información de los puntos del mapa creados en el fotograma actual, la posición y orientación de la cámara y la nube de puntos procesada por el algoritmo ORB-SLAM2, respectivamente.

Una vez compilado el wrapper de la cámara ZED y el algoritmo ORB-SLAM2 en el espacio de trabajo catkin_ws de ROS en la computadora, se procede a ejecutar los archivos launch para poner en marcha todos los nodos necesarios de cada una de las herramientas. Adicionalmente, se ejecuta la herramienta rviz para visualizar los datos procesados por todos los nodos.

Para el presente experimento fue creado un mapa en el área del CITIS, en la Figura 5.15 se observa un croquis del lugar y se representa el recorrido realizado, de tal manera que se mapeara tanto en interiores como exteriores para comprobar el funcionamiento correcto del algoritmo. En la Figura 5.16 se describe gráficamente el experimento.

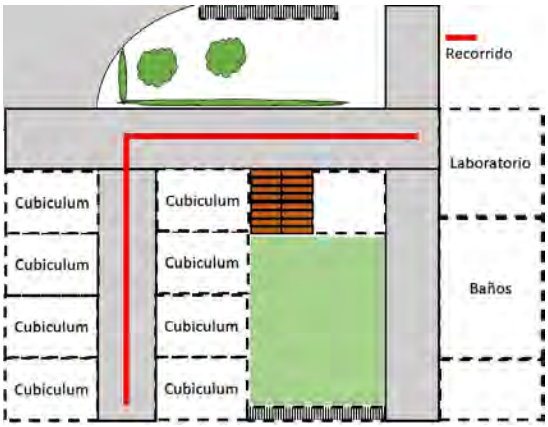

Figura 5.15: Croquis del CITIS y recorrido realizado.

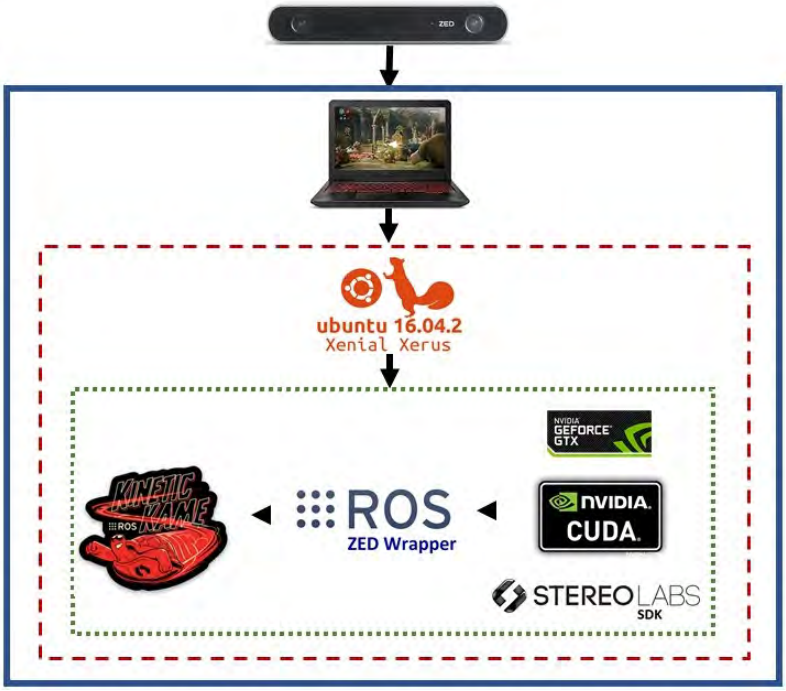

Figura 5.16: Diagrama del experimento realizado.

La Figura 5.17 muestra un fotograma del entorno mapeado, en ella se pueden observar los características detectadas por el algoritmo ORB-SLAM2 para crear los puntos del mapa, la cantidad máxima de características detectadas por el algoritmo pueden ser modificadas mediante el parámetro *ORBextractor.nFeatures*.

Una vez realizado todo el recorrido y construido el mapa, se puede visualizar en la Figura 5.18 diferentes vistas del mismo.

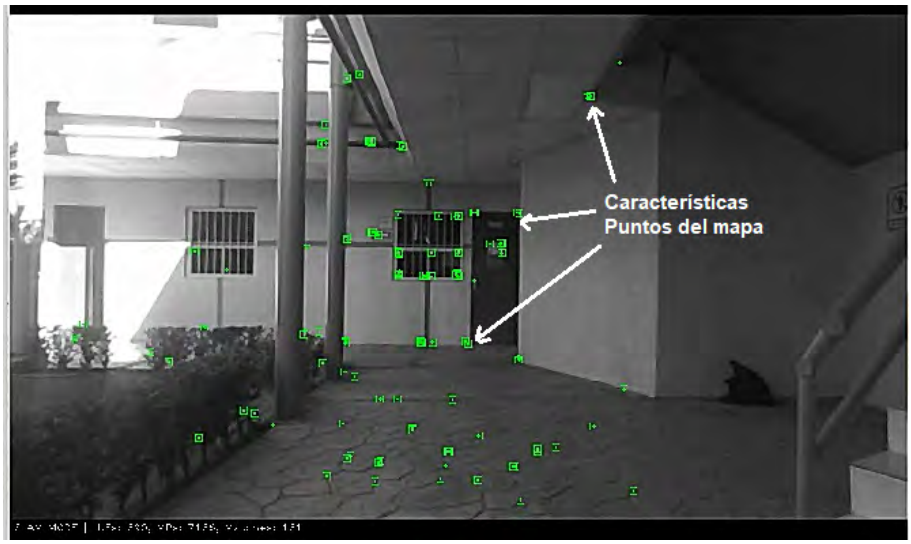

Figura 5.17: Puntos del mapa del fotograma actual.

Con la información del mapa guardado se ejecuta nuevamente el algoritmo ORB-SLAM2 pero en modo de solo localización, en dicho modo no es actualizado y solamente se buscan coincidencias para así poder determinar la pose de la cámara en el entorno, en ocasiones puede que se pierda la misma por movimientos bruscos, pero el algoritmo tiene la capacidad de re-localización una vez que detecta puntos coincidentes en el mapa [42]. La pose se muestra en rviz mediante un vector (flecha roja), que define la posición y orientación de la cámara en el mapa. En la Figura 5.19 se puede observar la pose obtenida en un punto del mapa.

Con las pruebas realizadas se pudo obtener un mapa en el cual posteriormente se pudo obtener la posición y orientación de la cámara. Todos los algoritmos fueron implementados en una computadora portátil, para una vez funcionando ser implementados sobre una computadora embebida Jatson Nano acoplada en conjunto con la cámara en el vehículo aéreo.

Con las pruebas experimentales realizadas hasta el momento se comprobó que la velocidad de los fotogramas por segundo (FPS) de la cámara influye en las tareas realizadas por el algoritmo ORB-SLAM2, a menos FPS existe mayor posibilidad de que el algoritmo pierda la localización. También, la resolución de la imagen de la cámara igualmente influye en la respuesta del algoritmo, a una menor resolución le resulta más complicado al algoritmo reconocer características ORB de cada fotograma. Igualmente, se pudo comprobar la robustez del algoritmo en cuanto a movimientos bruscos, y su capacidad para la re-localización.

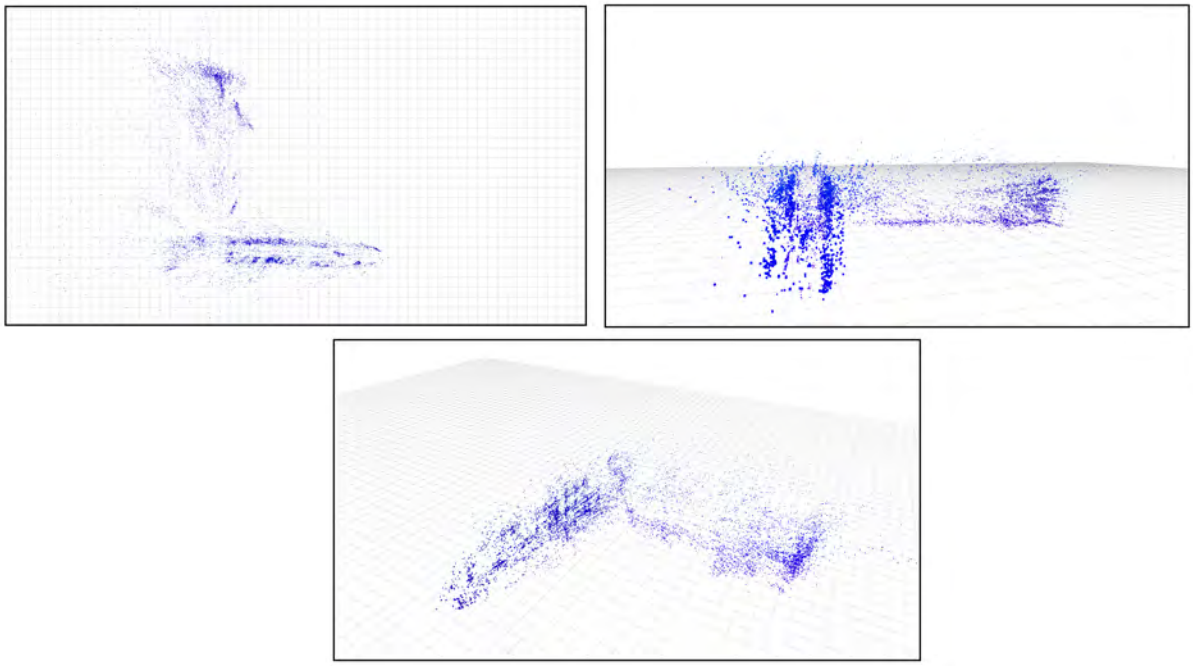

Figura 5.18: Vistas del mapa creado del entorno.

5.6. Comunicación serial entre el Pixhawk y la computadora Jetson Nano

El algoritmo ORB-SLAM2 se ejecutará en la computadora embebida Jetson Nano, con el objetivo de estimar la pose (posición y orientación) de la cámara, esta a su vez será la pose del vehículo dado que la misma estará empotrada sobre él. Dichos valores serán enviados vía serial al controlador de vuelo Pixhawk, para ser utilizados en el cálculo de la señal de error y de control. De los 5 puertos serial UART disponibles en el Pixhawk se hará uso de TELEM2, en el caso de la Jetson Nano se utilizará el puerto UART en la GPIO J41. Para la comunicación se fijó un baudrate de 115200, 8 bits de datos, sin paridad y un bit de parada. En la Figura 5.20 se muestra el diagrama de la conexión basado en el pinout de cada puerto de los dispositivos.

De manera adicional, fue necesario construir un cable para la conexión física de los dispositivos, en la Figura 5.21 se muestra la interconexión real de la plataforma.

Los valores de pose generados por el algoritmo consisten en coordenadas (x, y, z) para la posición y la orientación un cuaternión (x, y, z, w). Para controlar el vehículo solo se hará uso de la posición (x, y, z) para sustituir la lectura del GPS y barómetro. Además, para la orientación serán tomados directamente de las mediciones de la IMU.

Para enviar los valores mediante el puerto UART de x, y y z, es necesario construir

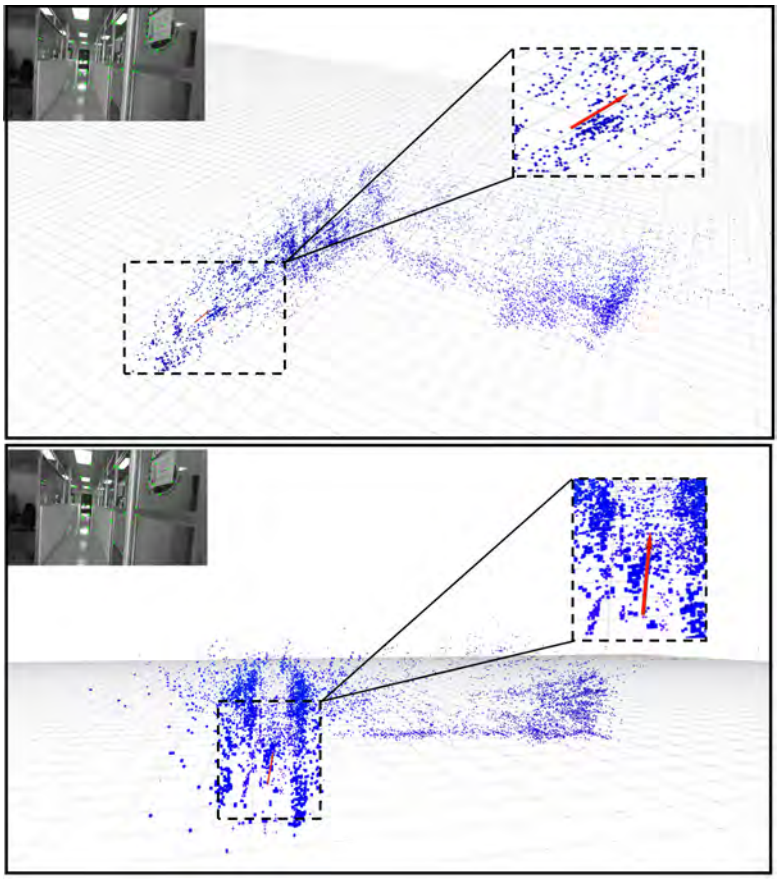

Figura 5.19: Pose obtenida en un punto del mapa.

una trama de datos organizada para poder ser interpretada de la misma manera una vez recibida en el Pixhawk. En la Figura 5.22 se muestra como está constituido dicho paquete de datos.

Cada uno de los caracteres será enviado mediante su código ASCII, por lo cual al ser recibido será necesario convertirlo nuevamente. En el caso de los valores de x, y y z serán enviados en centímetros y cada dígito que lo conforma será enviado igualmente en código ASCCI, por lo cual de manera similar tendrá que ser reconstruido el valor real.

X	Valor de x	Y	Valor de y	Z	Valor de z	F

Figura 5.22: Trama de datos.

La trama está conformada en primer lugar por el caracter X, indicando que los

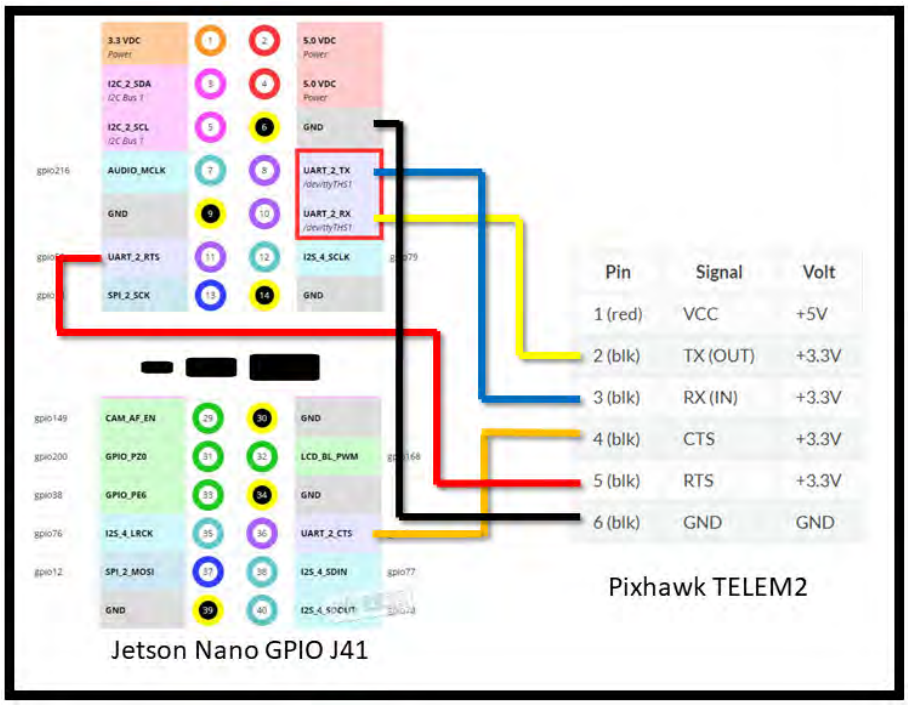

Figura 5.20: Diagrama de conexión.

restantes caracteres serán todos los dígitos que conforman el valor de x hasta recibir el siguiente carácter Y que nuevamente indicará que seguidamente se envían los dígitos del valor de y. Por último, se envía el valor de z antecedido por Z que es el carácter que lo indica, y finalmente F para advertir el fin de la trama.

A nivel software, se creó un nuevo paquete de ROS, el cual contiene un nodo programado en python encargado de suscribirse al tópico del algoritmo ORB-SLAM2 donde se publican los valores de la pose. En primer lugar, dicho nodo adquiere los valores requeridos de x, y y z en metros y los convierte a centímetros. Luego, se construye la trama de datos y la misma es enviada vía serial por el puerto UART de la Jetson Nano. Por su parte, una función es programada en el firmware modificado del Pixhawk encargada de recibir esta trama de datos y conformar los respectivos valores de x, y y z para posteriormente ser utilizados en las diferentes señales de control. Las velocidades traslacionales son calculadas numéricamente con el método de Euler y un paso de 0.1. En la Figura 5.23 se puede observar un diagrama que representa lo explicado con anterioridad.

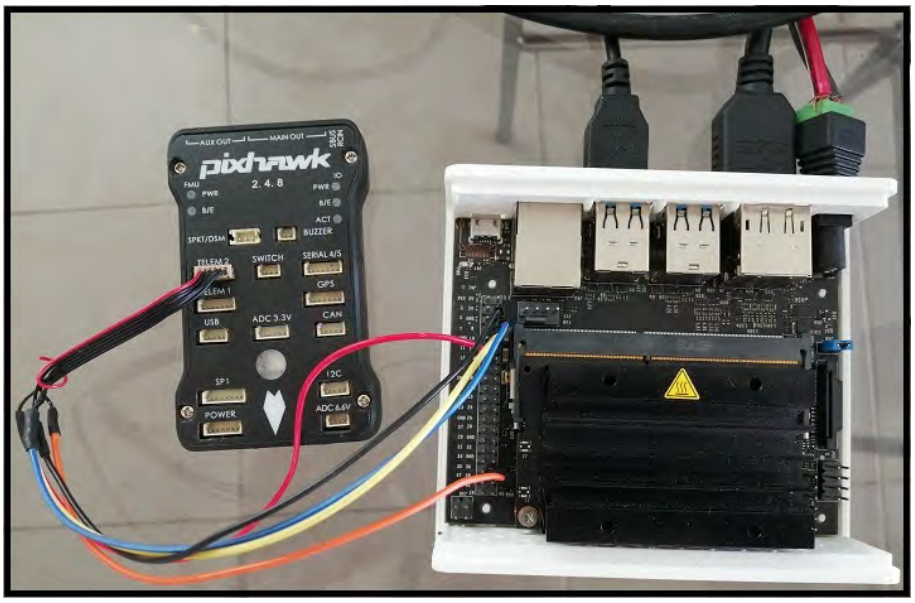

Figura 5.21: Conexión física de la plataforma.

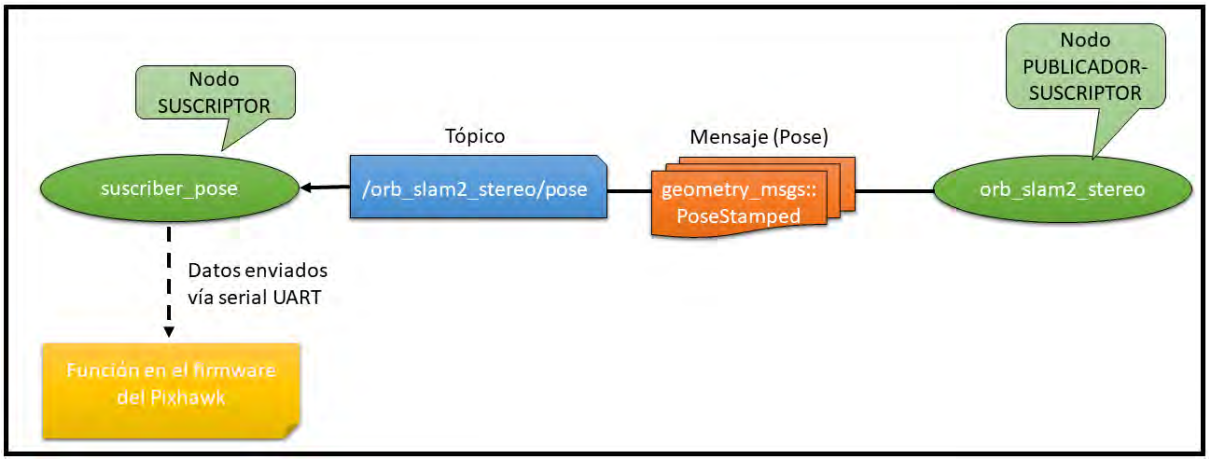

Figura 5.23: Diagrama de comunicación nivel software.

5.7. Funcionamiento del algoritmo de ORB-SLAM2 en la computadora embebida Jetson Nano

Se realizará un experimento donde se ejecutará el algoritmo ORB-SLAM2 en la Jetson Nano utilizando la cámara estereoscópica ZED. El objetivo de dicho experimento es comprobar que la computadora embebida es capaz de soportar el procesamiento del algoritmo. La siguientes especificaciones fueron las utilizadas durante el experimento.

- Computadora Jetson Nano Alimentada por puerto DC 5V-4A.
- Cámara ZED en modo WVGA, resolución 1344x376 a 60 FPS.
- Algoritmo ORB-SLAM2 configurado con 1200 características por fotograma.

Se realizó un recorrido para conformar un cuadrado de 1.2 metros de largo por 1.2 metros de ancho. Las gráficas de los resultados obtenidos para los valores de x, y y el plano $x - y$ se muestran en la Figura 5.24.

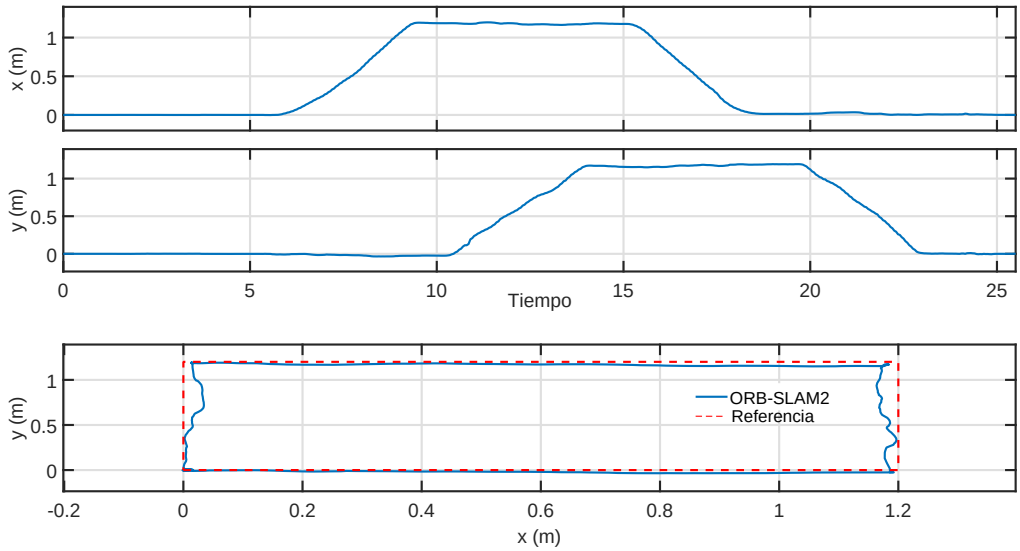

Figura 5.24: Posiciones x, y y plano $x - y$, respectivamente.

Como se pudo observar en las figura anterior, la posición de la cámara en el plano $x - y$ estimada por el algoritmo, muestra una exactitud relativamente buena para posteriormente ser utilizada como las señales necesarias para el cálculo de las leyes de control.

Para lograr dichos resultados fue necesario variar el parámetro ThDepth que representa el umbral de profundidad, esto se realizó a prueba y error, dicho parámetro es el que influye directamente en la correcta estimación de la pose por parte del algoritmo. Finalmente se comprobó que el valor de ThDepth = 22 arrojó las mejores estimaciones y resultados.

Otro detalle que se pudo detectar es la frecuencia de publicación de la pose por parte del algoritmo, esta es de aproximadamente 10Hz y dicho valor no ha podido ser modificado, sería deseable una velocidad más rápida para obtener mejores resultados, pero no obstante, dicha frecuencia puede ser utilizada sin ningún problema para el cálculo de la señal de control.

Con dicho experimento se pudo lograr los resultados esperados, por lo cual, la computadora embebida es capaz de procesar el algoritmo ORB-SLAM2 y a su vez

enviar vía serial los datos necesarios al controlador de vuelo. Hasta el momento ya se cuenta con la posición x, y y z, estimados por el algoritmo en sustitución de los datos generados por el GPS, por lo tanto, ya se puede continuar con la implementación de leyes de control en el controlador de vuelo tanto de posición como orientación.

Con los experimentos realizados hasta el momento, se dispone de una correcta comunicación entre la computadora embebida Jetson Nano y el controlador de vuelo Pixhawk. El procesamiento del algoritmo ORB-SLAM2 en la computadora embebida permite la estimación de la pose de la cámara (y el vehículo a su vez por estar empotrados), de la cuál se envía vía serial UART la posición x, y y z al controlador de vuelo y las respectivas velocidades calculadas numéricamente. Con los datos disponibles, en primer lugar se implementarán leyes de control PD para estabilizar el vehículo. Una vez logrado lo anterior, se prosigue a implementar las leyes de control descritas, pero en el caso de seguir una trayectoria sigmoidal suave en el eje x e y desde un punto A a un punto B.

5.8. Control para la estabilización del vehículo

En primer lugar, se realizaron pruebas de campo sobre la plataforma para comprobar el funcionamiento del control de orientación. Para su programación se modificó el Firmware del Pixhawk 2.4.8, además, se utilizaron las señales de la unidad de mediciones inerciales, principalmente los giroscopios. Además, se utiliza el radiocontrol DX8 y el radioreceptor AR8000 para poder manipular el vehículo de manera manual remotamente, lo anterior se puede observar en el diagrama de la Figura 5.25. Las ganancias de los controladores sintonizadas de forma heurística se pueden observar a continuación:

Ganancia	Valor	Ganancia	Valor
Kp_θ	120	Kd_θ	40
Kp_ϕ	120	Kd_ϕ	40
Kp_ψ	90	Kd_ψ	70

Tabla 5.1: Ganancias sintonizadas de los controladores.

En las figuras 5.26 y 5.27 se muestran los resultados experimentales obtenidos con los controles de orientación.

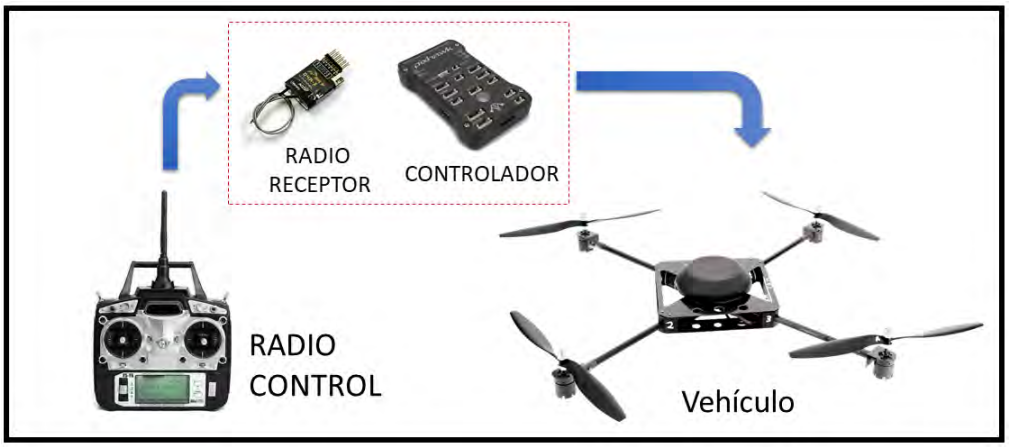

Figura 5.25: Diagrama que representa el experimento.

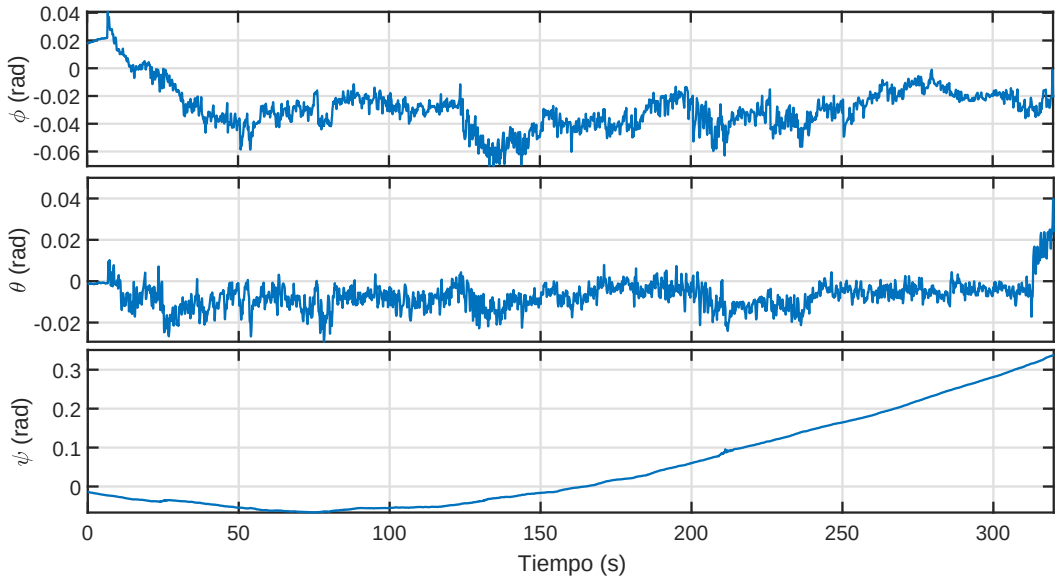

Figura 5.26: Ángulos de Euler θ, ϕ y ψ, respectivamente.

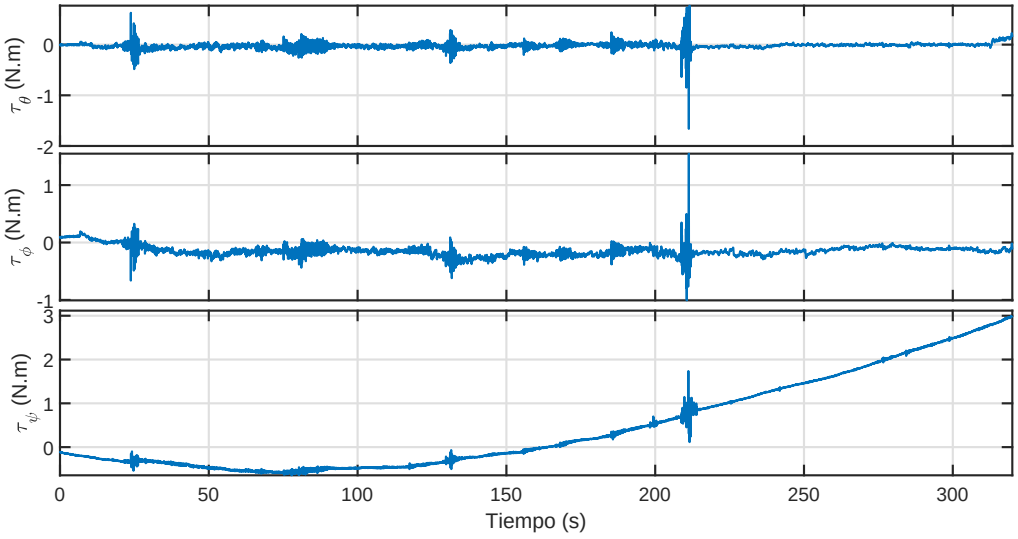

Figura 5.27: Señales de control τ_θ, τ_ϕ y τ_ψ, respectivamente.

Como se puede observar, todos los ángulos de Euler convergen a cero y se mantienen aproximadamente sobre ese valor durante todo el experimento, garantizando la estabilidad del vehículo, la cuál es de suma importancia para posteriormente poder aplicar los controles de traslación y altura. En las gráficas de las señales de control se manifiestan algunos momentos en los que estas crecen o tienen algunos picos, lo anterior sucede puesto que se manipulan los diferentes ángulos desde el radiocontrol para forzar que actúe el control y los regrese a cero. Con dicho experimento se pudo observar un correcto comportamiento en la orientación del vehículo.

5.9. Vuelo autónomo estacionario del vehículo mediante un controlador PD

El presente experimento consiste en realizar un vuelo estacionario o hover con la plataforma experimental funcionando con todos los sistemas a bordo. Durante el experimento, las mediciones de la posición son calculadas por el algoritmo mediante la cámara en la computadora embebida y enviadas vía serial al controlador de vuelo. En el Pixhawk serán procesadas e introducidas a los controles PD de orientación, traslación y altura. Una vez el vehículo en el aire, son accionados dos switches en el radiocontrol encargados de activar los controles de traslación y altura, en ese mismo instante la posición actual del vehículo es tomada como referencia para el cálculo de los errores. En las siguientes figuras son mostrados los resultados gráficos obtenidos del experimento.

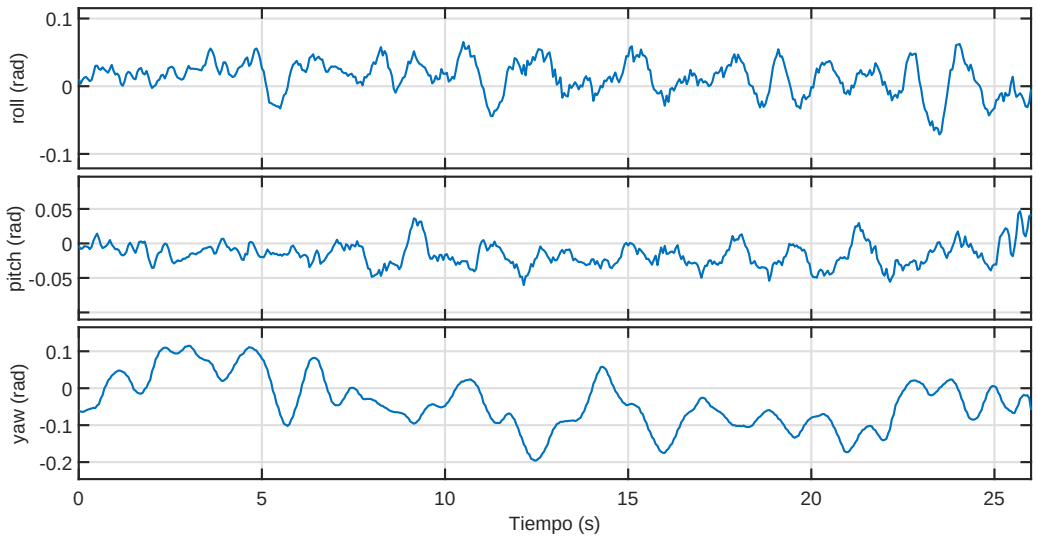

Figura 5.28: Ángulos de Euler para un vuelo estacionario.

En la Figura 5.28 se evidencia como los respectivos ángulos de Euler se mantienen oscilando alrededor de cero, las señales de control que garantizan lo anterior se muestran en la Figura 5.29. Igualmente, la altura converge hacia la referencia fijada en 1.2 metros aproximadamente.

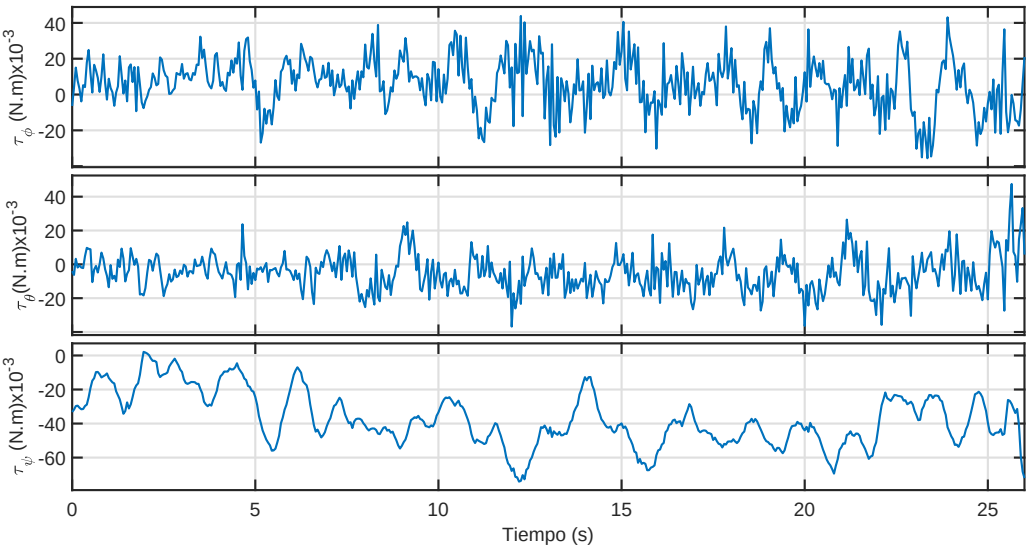

Figura 5.29: Señales de control τ_ϕ, τ_θ y τ_ψ respectivamente para un vuelo estacionario.

Como se observa en la Figura 5.30 las posiciones x e y igualmente se mantienen sobre el origen $(0, 0)$ el cual se fijó como punto de referencia. Las respectivas señales

de control se pueden ver en la Figura 5.31.

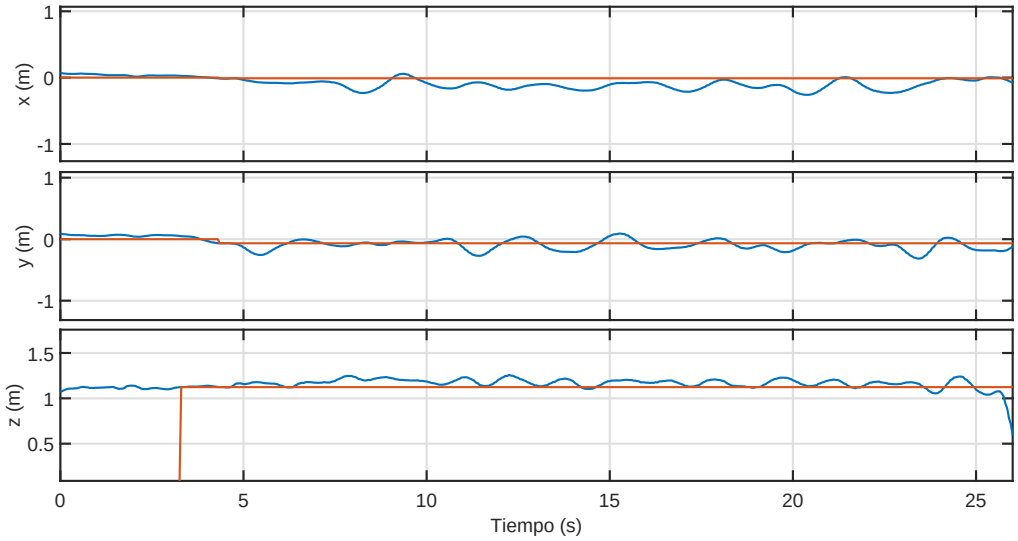

Figura 5.30: Posiciones x, y y z respectivamente para un vuelo estacionario.

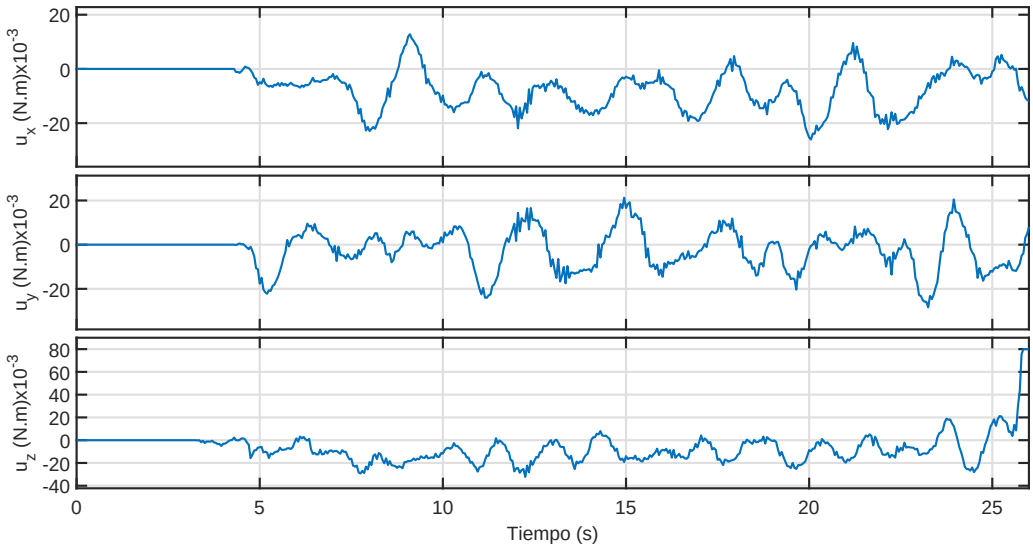

Figura 5.31: Señales de control u_x, u_y y u_z respectivamente para un vuelo estacionario.

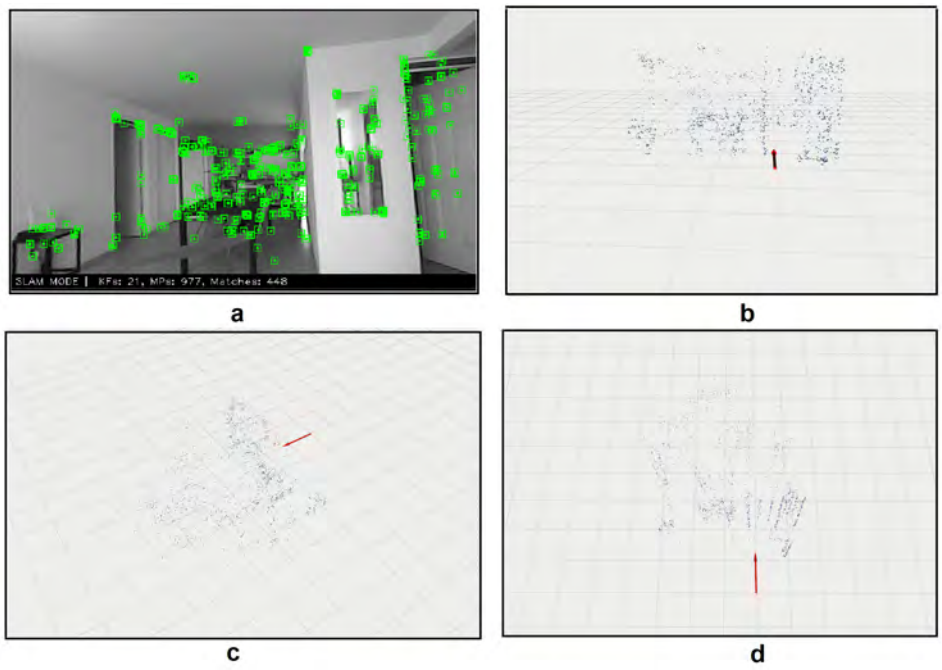

Figura 5.32: a) características ORB detectadas en un fotograma, b) mapa creado y pose (vista desde el vehículo), c) vista en tres dimensiones, d) vista superior.

En la Figura 5.32 se muestra una instantánea con las características ORB detectadas en un fotograma, además, diferentes vistas del mapa que se generó y la obtención de la pose del vehículo en dicho mapa de manera simultánea. Lo anterior se obtuvo mediante la herramienta de visualización de ROS rviz.

Con la realización de este experimento se pudo comprobar el correcto funcionamiento de la plataforma en interiores prescindiendo de la señal del GPS y el barómetro para obtener su posición, y en el caso de la orientación utilizando las mediciones de la IMU. Además, es importante destacar que las pruebas fueron realizadas en un espacio de pequeña dimensión por lo tanto las propias turbulencias creadas por las hélices del vehículo actúan como una perturbación externa sobre el sistema, ante las cuales los controles respondieron de manera eficiente permitiendo de manera satisfactoria el vuelo estacionario. Logrando lo anterior, ya se pueden realizar pruebas de seguimiento de trayectorias en los ejes x e y.

5.10. Vuelo autónomo de seguimiento de trayectorias del vehículo mediante un controlador PD

En el siguiente experimento se pretende seguir una trayectoria en los ejes x e y aplicando un control PD en traslación. Como trayectoria a seguir en ambos ejes se seleccionó una sigmoide de amplitud 1 metro, ya que dicha señal permite un movimiento suave en el desplazamiento de un punto a otro, esto garantiza que existan pequeñas señales de error y por consiguiente señales de control pequeñas que eviten la desestabilización del vehículo. Las respectivas gráficas que representan el experimento se plasman a continuación.

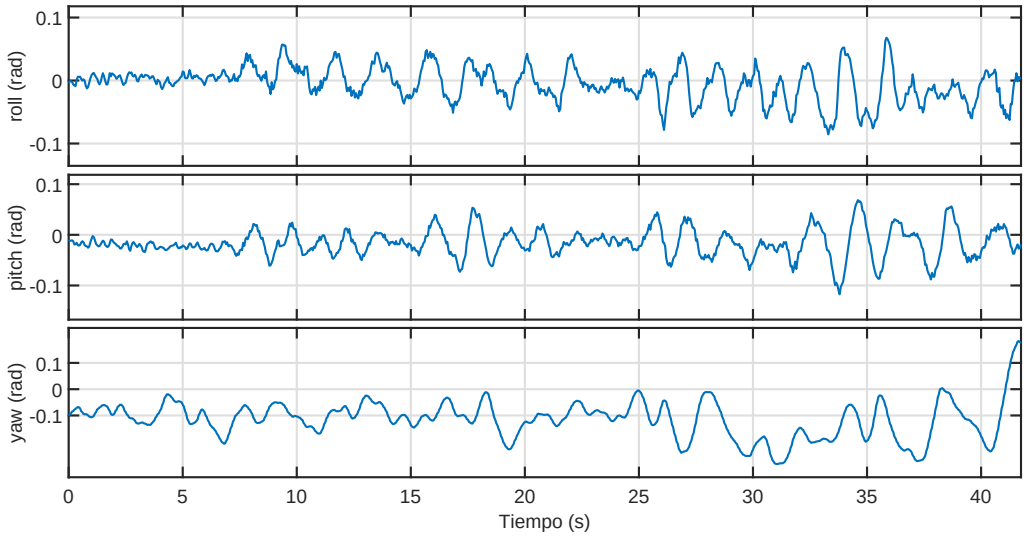

Figura 5.33: Ángulos de Euler para seguimiento de trayectoria.

En las figuras 5.33 y 5.34 se presencian los ángulos de Euler y la altura, así como sus respectivas señales de control que garantizan el comportamiento de convergencia a sus respectivas referencias. En este caso se utilizaron las mismas ganancias de los controladores PD para realizar el vuelo estacionario.

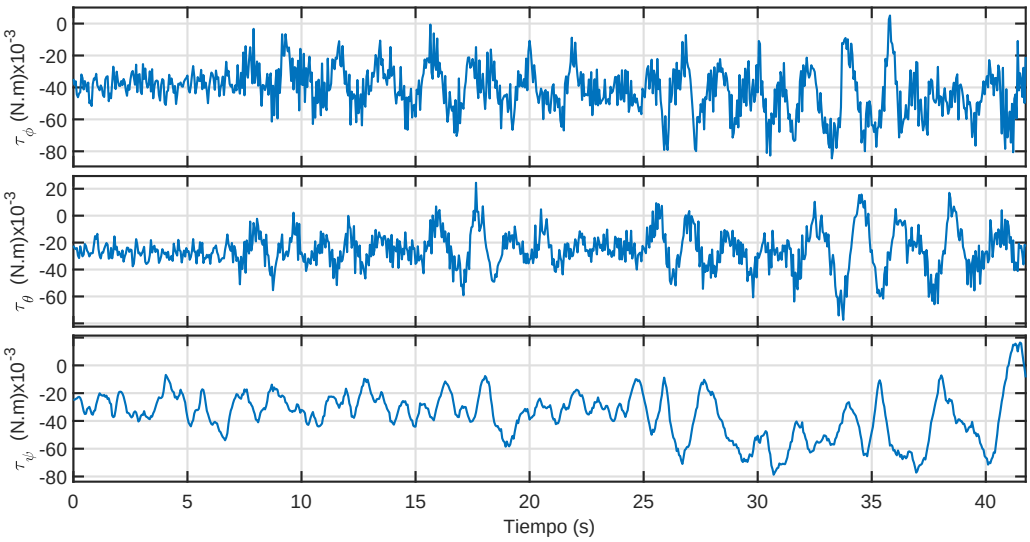

Figura 5.34: Señales de control τ_ϕ, τ_θ y τ_ψ respectivamente para seguimiento de trayectoria.

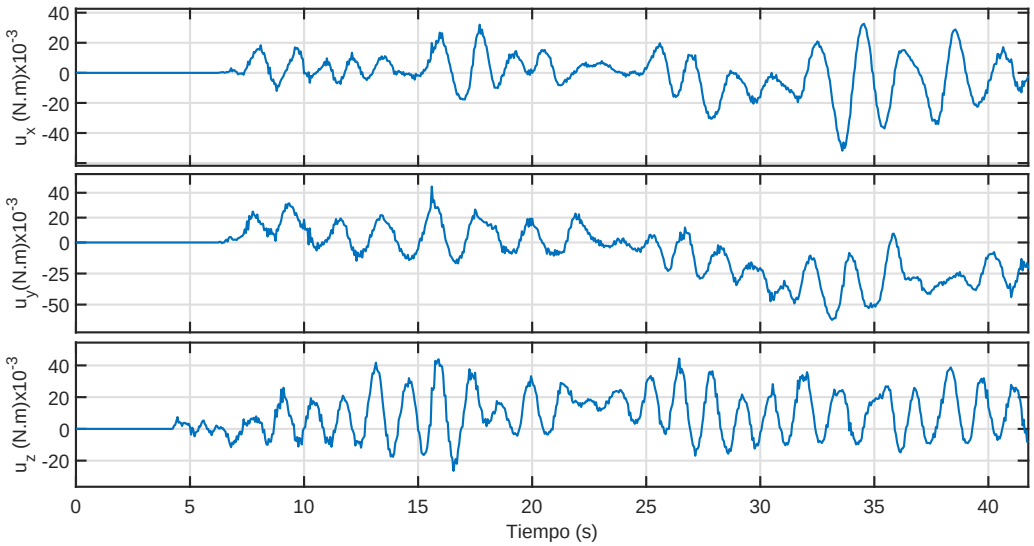

Figura 5.36: Señales de control u_x, u_y y u_z respectivamente para seguimiento de trayectoria.

En el caso de los controles PD de traslación se tuvieron que ajustar ligeramente las ganancias para lograr seguir las trayectorias sigmoidales de referencia. Lo anterior se puede observar en las figuras 5.35 y 5.36. Se pudo comprobar que es posible realizar el seguimiento de trayectorias sencillas del vehículo en espacios interiores igualmente

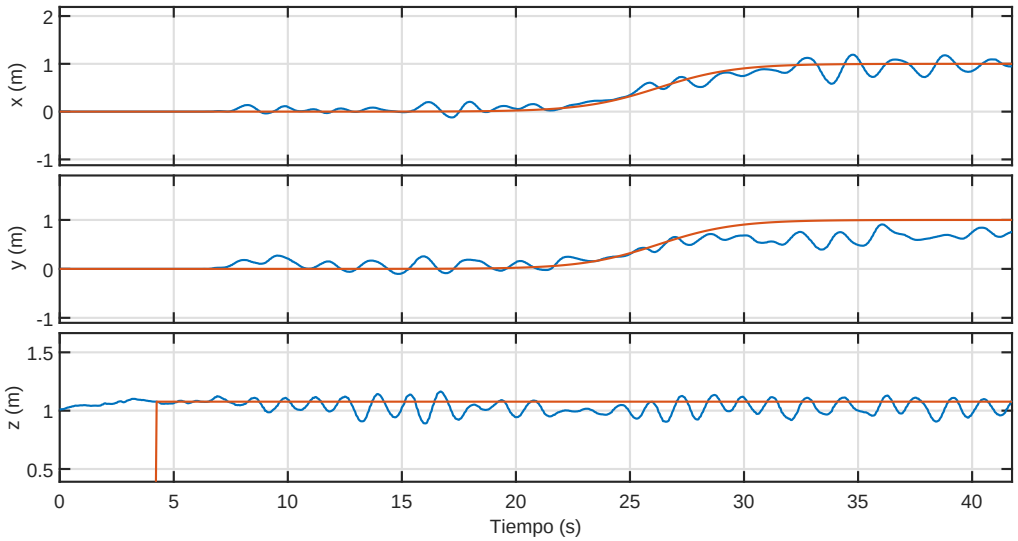

Figura 5.35: Posiciones x, y y z respectivamente para seguimiento de trayectoria.

sustituyendo las mediciones del GPS y barómetro por las generadas por el algoritmo, con lo anterior será posible implementar en trabajos futuros trayectorias más complejas u otros controladores y así realizar tareas de navegación autónomas más sofisticadas.

5.11. Conclusiones del capítulo

En el presente capítulo se desarrollaron simulaciones numéricas de vuelos estacionarios y seguimientos de trayectorias tanto para controladores PD como de saturaciones anidadas, con el objetivo de comparar el comportamiento del modelo del sistema a partir de las gráficas obtenidas. Además, en un primer momento se comprobó el funcionamiento del algoritmo ORB-SLAM2 en una computadora portátil para posteriormente ser implementado en la computadora embebida empotrada al vehículo. Igualmente, fue necesario establecer la comunicación física y lógica entre el Pixhawk y la Jetson Nano, para así permitir el envío de las posiciones generadas por el algoritmo hacia el controlador de vuelo. Por último, se realizaron pruebas experimentales sobre la plataforma, que consistieron en vuelos estacionarios y de seguimiento de trayectorias mediante controladores PD. En el próximo y último capítulo se plasman las conclusiones obtenidas a lo largo de la tesis, así como el trabajo futuro que se podría desarrollar.

Capítulo 6

Conclusiones y trabajo futuro

Siguiendo la metodología planteada para dar cumplimiento a los objetivos específicos hasta llegar al general, se llevaron a cabo una serie de experimentos los cuales arrojaron los resultados esperados. En primer lugar, la investigación del estado del arte relacionado con los algoritmos de visión artificial permitió seleccionar el ORB-SLAM2 principalmente por su menor costo computacional. Igualmente esto propició que fuese implementado en una computadora embebida Jetson Nano de pequeña dimensión. Además, el uso de la cámara estereoscópica ZED y de su wrapper de ROS, garantizó la integración de ambos componentes en dicho sistema operativo robótico. También, se logró realizar vuelos en hover y tareas de seguimiento de trayectorias en interiores prescindiendo del GPS y el barómetro, únicamente las mediciones estimadas por el algoritmo. Por lo anterior, se dispone de un vehículo aéreo equipado con un sistema de percepción visual encargado de estimar la posición del mismo en ambientes no estructurados o en los que la señal del GPS es débil o no disponible, pudiendo ser utilizado para realizar en trabajos futuros tareas de navegación autónomas más complejas, como por ejemplo la evasión de obstáculos.

6.1. Trabajo futuro

El trabajo futuro propuesto derivado de la presente tesis es el siguiente:

- Diseñar e implementar leyes de control más complejas y no lineales sobre la plataforma experimental.
- Proveer la plataforma experimental con un algoritmo y otros sensores capaces de evadir obstáculos.

- Realizar tareas de seguimiento de trayectorias más complejas como el caso de circunferencias.
- Realizar tareas de navegación autónoma más complejas.
- Realizar experimentos tanto en interiores como exteriores, donde se completen trayectorias cerradas para explotar esta funcionalidad del algoritmo de ORB-SLAM2.
- De ser posible utilizar otros tipos de cámaras como las RGB-D para comparar el funcionamiento del algoritmo con la cámara estereoscópica.

Bibliografía

[1] L. S. Padilla, N. K. Valverde, G. V. Herrera, F. R. Quiroz, and L. O. Toledo, "Desarrollo de un sistema difuso aplicado al despegue de micro-uav."

[2] R. Munguía and A. Grau, "A practical method for implementing an attitude and heading reference system," *International Journal of Advanced Robotic Systems*, vol. 11, no. 4, p. 62, 2014.

[3] B. W. Parkinson, P. Enge, P. Axelrad, and J. J. Spilker Jr, *Global positioning system: Theory and applications, Volume II.* American Institute of Aeronautics and Astronautics, 1996.

[4] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus, "Energy-efficient autonomous four-rotor flying robot controlled at 1 khz," in *Proceedings 2007 IEEE International Conference on Robotics and Automation.* IEEE, 2007, pp. 361–366.

[5] J. Kim, M.-S. Kang, and S. Park, "Accurate modeling and robust hovering control for a quad-rotor vtol aircraft," in *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009.* Springer, 2009, pp. 9–26.

[6] R. Mori, K. Hirata, T. Tamaki, and N. Yonezawa, "Vision-based guidance control of a small-scale unmanned helicopter," *Journal of the Robotics Society of Japan*, vol. 26, no. 8, pp. 905–912, 2008.

[7] T. Zhang, Y. Kang, M. Achtelik, K. Kuhnlenz, and M. Buss, "Autonomous hovering of a vision/imu guided quadrotor," in *2009 International Conference on Mechatronics and Automation.* IEEE, 2009, pp. 2870–2875.

[8] K. E. Wenzel, P. Rosset, and A. Zell, "Low-cost visual tracking of a landing place and hovering flight control with a microcontroller," *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1-4, p. 297, 2010.

[9] E. Rondon, L.-R. Garcia-Carrillo, and I. Fantoni, "Vision-based altitude, position and speed regulation of a quadrotor rotorcraft," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 628–633.

[10] J. Artieda, J. M. Sebastian, P. Campoy, J. F. Correa, I. F. Mondragón, C. Martínez, and M. Olivares, "Visual 3-d slam from uavs," *Journal of Intelligent and Robotic Systems*, vol. 55, no. 4-5, p. 299, 2009.

[11] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[12] H. Zhao, M. Chiba, R. Shibasaki, X. Shao, J. Cui, and H. Zha, "Slam in a dynamic large outdoor environment using a laser scanner," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 1455–1462.

[13] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf." in *ICCV*, vol. 11, no. 1. Citeseer, 2011, p. 2.

[14] S. Moyano Díaz, "Diseño y construcción de un quadcopter," B.S. thesis, Universitat Politècnica de Catalunya, 2014.

[15] A. Luque-Luque, "Diseño y construcción de un robot móvil como un agente autónomo," 2015.

[16] R. ESCAMILLA NÚÑEZ, "Diseño, construcción, instrumentación y control de un vehículo aéreo no tripulado (uav)," *México DF, Noviembre*, 2010.

[17] I. García García *et al.*, "Estudio sobre vehículos aéreos no tripulados y sus aplicaciones," 2017.

[18] A. Barrientos, J. Del Cerro, P. Gutiérrez, R. San Martín, A. Martínez, and C. Rossi, "Vehículos aéreos no tripulados para uso civil. tecnología y aplicaciones," *Universidad politécnica de Madrid, Madrid*, 2007.

[19] C. Cuerno Rejado, L. Garcia Hernandez, A. Sanchez Carmona, A. Carrió Fernández, J. L. Sanchez Lopez, and P. Campoy Cervera, "Evolución histórica de los vehículos aéreos no tripulados hasta la actualidad," *Dyna*, vol. 91, no. 3, pp. 282–288, 2016.

[20] C. Soria, F. Rossomando, A. Veca, and P. Campillo, "Modelado y diseño del control de un tricóptero uav."

[21] J. Cisneros Fernández and C. Andrade Zarcero, "Diseño y construcción de un cuadricóptero," B.S. thesis, Universitat Politècnica de Catalunya, 2013.

[22] O. Chávez, J. Enrique, and D. A. P. Sebastián, "Diseño, construcción y control de un hexacóptero de monitoreo," Master's thesis, Quito, 2015., 2015.

[23] D. B. Hinojosa Espinoza, M. Pilatásig, and C. Paúl, "Diseño y construcción de una aeronave no tripulada tipo octocóptero con modelamiento de objetos en 3d a partir de imágenes 2d para el laboratorio de mecatrónica." B.S. thesis, Universidad de las Fuerzas Armadas ESPE Extensión Latacunga. Carrera de . . . , 2014.

[24] R. Rubio and W. C. Richard, "Diseño y construcción de un prototipo de cuadricóptero para aplicaciones de adquisición de imágenes aéreas," 2019.

[25] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[26] A. Koubâa, *Robot Operating System (ROS).* Springer, 2017.

[27] M. Quigley, B. Gerkey, and W. D. Smart, *Programming Robots with ROS: a practical introduction to the Robot Operating System.* .ºReilly Media, Inc.", 2015.

[28] S. Cousins, "Exponential growth of ros [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 1, no. 18, pp. 19–20, 2011.

[29] H. F. Durrant-Whyte, "Uncertain geometry in robotics," *IEEE Journal on Robotics and Automation*, vol. 4, no. 1, pp. 23–31, 1988.

[30] N. Ayache and O. D. Faugeras, "Building, registrating, and fusing noisy visual maps," *the International journal of Robotics Research*, vol. 7, no. 6, pp. 45–65, 1988.

[31] J. L. Crowley, "World modeling and position estimation for a mobile robot using ultrasonic ranging," in *Proceedings, 1989 International Conference on Robotics and Automation*. IEEE, 1989, pp. 674–680.

[32] R. Chatila and J.-P. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2. IEEE, 1985, pp. 138–145.

[33] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[34] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE robotics & automation magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[35] R. A. Howard, "Dynamic programming and markov processes." 1960.

[36] L. Ljung, "Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems," *IEEE Transactions on Automatic Control*, vol. 24, no. 1, pp. 36–50, 1979.

[37] S. Kohlbrecher, J. Meyer, T. Graber, K. Petersen, U. Klingauf, and O. von Stryk, "Hector open source modules for autonomous mapping and navigation with rescue robots," in *Robot Soccer World Cup*. Springer, 2013, pp. 624–631.

[38] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[39] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[40] A. Eliazar and R. Parr, "Dp-slam: Fast, robust simultaneous localization and mapping without predetermined landmarks," in *IJCAI*, vol. 3. Acapulco, Mexico, 2003, pp. 1135–1142.

[41] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[42] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[43] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*. Springer, 2010, pp. 778–792.

[44] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[45] H. Strasdat, A. J. Davison, J. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2352–2359.

[46] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "Large-scale 6-dof slam with stereo-in-hand," *IEEE transactions on robotics*, vol. 24, no. 5, pp. 946–957, 2008.

[47] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE Computer Society, 2007, pp. 1–10.

[48] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g 2 o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 3607–3613.

[49] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular slam in dynamic environments," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 209–218.

[50] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," *Robotics: Science and Systems VI*, vol. 2, no. 3, p. 7, 2010.

[51] D. Gálvez-López and J. D. Tardos, "Bags of binary words for fast place recognition in image sequences," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1188–1197, 2012.

[52] Turnigy multistar. [Online]. Available: http://www.turnigy.com/esc/multistar

[53] Graphene lipo. [Online]. Available: https://hobbyking.com

[54] Afro muti-rotor esc. [Online]. Available: https://www.gensace.de

[55] Pixhawk. [Online]. Available: https://pixhawk.org

[56] Gps 3dr ublox. [Online]. Available: https://www.getfpv.com

[57] Spektrum. [Online]. Available: https://www.spektrumrc.com

[58] Jetson nano. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[59] Zed camera. [Online]. Available: https://www.stereolabs.com/zed/

[60] Kiam Heong Ang, G. Chong, and Yun Li, "Pid control system analysis, design, and technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559–576, 2005.

[61] E. X. DeJesus and C. Kaufman, "Routh-hurwitz criterion in the examination of eigenvalues of a system of nonlinear ordinary differential equations," *Phys. Rev. A*, vol. 35, pp. 5288–5290, Jun 1987. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.35.5288

[62] A. R. Teel, "Global stabilization and restricted tracking for multiple integrators with bounded controls," *Systems & control letters*, vol. 18, no. 3, pp. 165–171, 1992.

[63] P. Castillo, P. García, R. Lozano, and P. Albertos, "Modelado y estabilización de un helicóptero con cuatro rotores," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 4, no. 1, pp. 41–57, 2007.

[64] H. Romero-Trejo, "Modélisation et asservissement visuel d'un mini hélicoptère," Ph.D. dissertation, Compiègne, 2008.

[65] H. Goldstein, C. Poole, and J. Safko, "Classical mechanics addison-wesley," *Reading, MA*, p. 426, 1980.

[66] G. G. Slabaugh, "Computing euler angles from a rotation matrix," *Retrieved on August*, vol. 6, no. 2000, pp. 39–63, 1999.

Apéndice A

Modelado matemático

En este apéndice se describe de manera detallada el modelado matemático del helicóptero miniatura de cuatro rotores basados en el modelado dinámico descrito por las ecuaciones de Euler-Lagrange.

A.1. Características del vehículo

El helicóptero miniatura de cuatro rotores es un prototipo muy utilizado ya que permite experimentar casi todos los fenómenos aerodinámicos que ocurren en los vehículos aéreos que realizan vuelo estacionario.

Este vehículo, a diferencia del helicóptero clásico no cuenta con un plato cíclico, por lo tanto, se controla variando la velocidad angular de cada uno de sus cuatro rotores. Esto implica, que la fuerza f_i con $i = 1, 2, 3, 4$ producida por cada cada motor es proporcional al cuadrado de su respectiva velocidad angular, es decir, $f_i = k\omega_i^2$. Además, cada motor solo puede girar en una dirección fija, lo cual implica que la fuerza producida siempre será positiva [63], lo anterior se puede observar en la Figura A.1.

También, en la Figura A.1 se puede observar que tanto el motor delantero M_1 y el trasero M_3 giran en sentido anti-horario, mientras que los motores M_2 y M_4 giran a favor de las manecillas del reloj. Lo anterior implica que los efectos giroscópicos y los momentos aerodinámicos tienden a cancelarse en vuelo estacionario. La entrada de control principal u consiste en la suma de todas las fuerzas producidas por cada

motor de manera independiente, es decir:

$$u = \sum_{i=1}^{4} f_i \tag{A.1}$$

donde f_i es la fuerza producida por el motor M_i para $i = 1, 2, 3, 4$. De forma general, $f_i = k_i\omega_i^2$ donde k_i es una constante y ω_i es la velocidad angular del i-ésimo motor.

Para describir los movimientos producidos por el vehículo, se utilizarán los ángulos de Euler, los cuales son, ψ o ángulo de guiñada (yaw, del inglés), θ o ángulo de cabeceo (pitch, del inglés) y ϕ o ángulo de alabeo (roll, del inglés) [63], ver Figura A.1.

El momento de cabeceo (θ, pitch) es producido por la diferencia de $(f_2+f_4-f_3-f_1)\ell$, el momento de alabeo (ϕ, roll) es producido por $(f_3 + f_2 - f_1 - f_4)\ell$, y el momento de guiñada (ψ, yaw) es la diferencia de $(-f_3 - f_1 + f_2 + f_4)k_{fm}$, donde ℓ es la distancia entre los motores y el centro de gravedad y k_{fm} es el factor de escalamiento de fuerza a momento.

El momento o par del motor es opuesto por una fricción aerodinámica τ_{drag}, tal que:

$$I_{rot}\dot{\omega}_i = \tau_{M_i} - \tau_{drag}, \tag{A.2}$$

donde I_{rot} es el momento de inercia del motor alrededor de su eje.

Por definición se tiene que la fricción aerodinámica es:

$$\tau_{drag} = \frac{1}{2}\rho A v^2, \tag{A.3}$$

donde ρ es la densidad del aire, A es el área frontal de la hélice, y v es su velocidad relativa con respecto al aire.

La velocidad angular ω es igual en magnitud a la velocidad linear v dividida entre el radio de rotación r, es decir: $\omega = v/r$. Reescribiendo la fricción aerodinámica (A.3), se obtiene:

$$\tau_{drag} = k_{drag}\omega^2, \tag{A.4}$$

donde $k_{drag} > 0$ es una constante, que depende entre otros factores, de la densidad del aire, del radio y de la forma de la hélice.

En el caso de maniobras quasi-estacionarias, la velocidad angular ω es una constante, entonces de la ecuación (A.2) se tiene que:

$$\tau_{M_i} = \tau_{drag}. \tag{A.5}$$

Para que el vehículo avance se debe incrementar la velocidad del motor trasero M_3 y reducir la velocidad del motor delantero M_1. De manera similar, el movimiento

lateral del vehículo se obtiene con los motores laterales M_2 y M_4. El movimiento de guiñada (ψ, yaw), se obtiene incrementando el par en el motor delantero y trasero (τ_{M_1}, τ_{M_3}), mientras que se reduce el par en los motores laterales (τ_{M_2}, τ_{M_4}). Estos movimientos se deben realizar manteniendo la fuerza principal, u, constante [63].

A.2. Modelado dinámico

Antes de comenzar el modelado matemático del vehículo se tendrán en cuenta las siguientes consideraciones para simplificar todo el proceso. En primer lugar, se considera el vehículo como un cuerpo rígido que evoluciona en un espacio tridimensional sujeto a una propulsión principal y tres momentos [64].

El vector de coordenadas generalizadas de un vehículo aéreo puede escribirse como sigue:

$$q = (x, y, z, \psi, \theta, \phi)^T \in \mathbb{R}^6,$$

donde $\xi = (x, y, z)^T \in \mathbb{R}^3$ denota la posición del centro de masa del vehículo relativo al marco de referencia inercial (I), y $\eta = (\psi, \theta, \phi)^T \in \mathbb{R}^3$ contiene los tres ángulos de Euler, ψ o ángulo de guiñada (yaw, del inglés), θ o ángulo de cabeceo (pitch, del inglés) y ϕ o ángulo de alabeo (roll, del inglés), los cuales representan la orientación del vehículo. Lo anterior se puede observar en la Figura A.1.

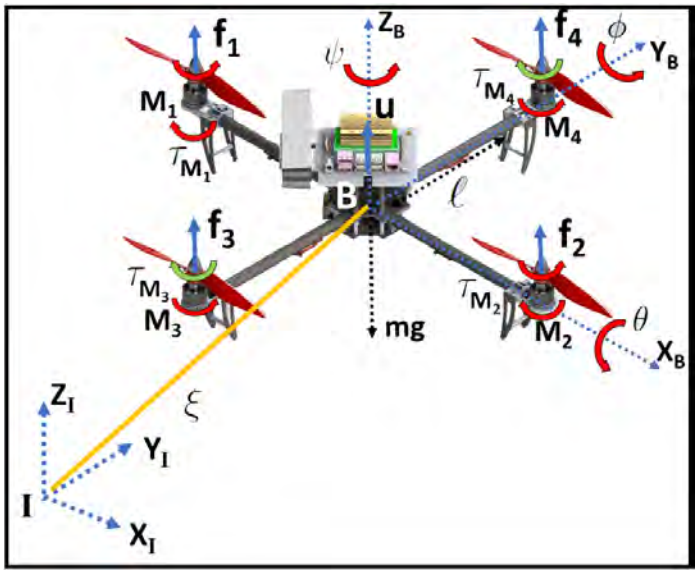

Figura A.1: Representación del vector de coordenadas generalizadas, los ángulos de Euler, el esquema de fuerzas y pares para el vehículo aéreo.

Para aplicar las ecuaciones de Euler-Lagrange primeramente se debe definir el Lagrangiano ($\mathcal{L}$):

$$\mathcal{L} = T - U,$$

$$\mathcal{L}(q, \dot{q}) = T_{tras} + T_{rot} - U,$$

donde $T_{tras} = \frac{1}{2} m \dot{\xi}^T \dot{\xi}$ es la energía cinética traslacional, $T_{rot} = \frac{1}{2} \Omega^T \mathbb{I} \Omega$ es la energía cinética rotacional, donde $\Omega = [p, q, r]^T$ es el vector de velocidades angulares con respecto al marco del cuerpo (B), $U = mgz$ es la energía potencial del vehículo, z es la altura del vehículo, m denota la masa del quadri-rotor, $\mathbb{I}$ es la matriz de inercia, y $g = 9.8 m/s^2$ es la aceleración gravitacional. El vector de velocidades angulares Ω respecto a los ejes de coordenadas del cuerpo (B) se relaciona con las velocidades generalizadas $\dot{\eta}$ (en la región donde los ángulos de Euler son válidos) utilizando una relación estándar cinemática [65],

$$\Omega = W_n \dot{\eta}, \tag{A.6}$$

donde $W_n = \begin{bmatrix} -\sin\theta & 0 & 1 \\ \cos\theta \sin\phi & \cos\phi & 0 \\ \cos\theta \cos\phi & -\sin\phi & 0 \end{bmatrix}$.

Sustituyendo la ecuación (A.6) en $T_{rot} = \frac{1}{2} \Omega^T \mathbb{I} \Omega$, se tiene:

$$T_{rot} = \frac{1}{2} \dot{\eta}^T W_n^T \mathbb{I} W_n \dot{\eta},$$

$$\mathbb{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}.$$

Definiendo $\mathbb{J}(\eta) = W_n^T \mathbb{I} W_n$, entonces

$$T_{rot} = \frac{1}{2} \dot{\eta}^T \mathbb{J} \dot{\eta}.$$

La matriz $\mathbb{J} = \mathbb{J}(\eta)$ actúa como la matriz de inercia para la energía cinética total rotacional del vehículo, expresada directamente en términos de las coordenadas generalizadas η.

Finalmente el Lagrangiano ($\mathcal{L}$) queda expresado de la siguiente forma:

$$\mathcal{L} = \frac{1}{2} m \dot{\xi}^T \dot{\xi} + \frac{1}{2} \dot{\eta}^T \mathbb{J} \dot{\eta} - mgz.$$

Para obtener el modelo dinámico del vehículo aéreo de cuatro rotores se aplican las ecuaciones de Euler-Lagrange con las fuerzas generalizadas externas:

$$\frac{d}{dt}\frac{\partial \mathcal{L}}{\partial \dot{q}} - \frac{\partial \mathcal{L}}{\partial q} = \begin{bmatrix} F_\xi \\ \tau \end{bmatrix},$$

donde $F_\xi = R\hat{F} \in \mathbb{R}^3$ es la fuerza traslacional aplicada al vehículo debido principalmente a la entrada de control principal, $\tau \in \mathbb{R}^3$ representa los momentos de yaw, pitch, y roll, y R denota la matriz rotacional $R(\psi, \theta, \phi)$ representando la orientación del quadri-rotor relacionada al marco de referencia fijo.

La matriz rotacional [66] $R(\psi, \theta, \phi)$ se define mediante rotaciones consecutivas en yaw, pitch, y roll como se muestra a continuación:

$$R = R_\psi R_\theta R_\phi,$$

donde,

$$R_\psi = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, R_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_\phi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},$$

por lo tanto

$$R = \begin{bmatrix} \cos\theta\cos\psi & \cos\psi\sin\theta\sin\phi - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\theta\sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}.$$

A partir de la Figura A.1, se tiene que:

$$\hat{F} = \begin{bmatrix} 0 \\ 0 \\ u \end{bmatrix},$$

donde u, es la entrada de control principal o fuerza principal, en la dirección del eje z, expresada según la ecuación (A.1).

Por lo tanto,

$$F_\xi = \begin{bmatrix} u(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta) \\ u(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi) \\ u\cos\theta\cos\phi \end{bmatrix}.$$

Los momentos o pares generalizados se definen de la siguiente manera:

$$\tau = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} (f_3 + f_2 - f_1 - f_4)\ell \\ (f_2 + f_4 - f_3 - f_1)\ell \\ (-f_3 - f_1 + f_2 + f_4)k_{fm} \end{bmatrix}.$$

donde ℓ es la distancia entre los motores y el centro de gravedad, y k_{fm} es el factor de escalamiento de fuerza a momento.

Puesto que el Lagrangiano no contiene términos combinados entre las energías cinéticas $\dot{\xi}$ y $\dot{\eta}$, las ecuaciones de Euler-Lagrange pueden ser divididas en las dinámicas para las coordenadas de ξ y η.

La ecuación de Euler-Lagrange para el caso del movimiento de traslación es:

$$\frac{d}{dt}\frac{\partial \mathcal{L}_{tras}}{\partial \dot{\xi}} - \frac{\partial \mathcal{L}_{tras}}{\partial \xi} = F_\xi,$$

$$\mathcal{L}_{tras} = \frac{1}{2} m \dot{\xi}^T \dot{\xi} - mgz,$$

realizando las respectivas derivaciones,

$$\frac{\partial \mathcal{L}_{tras}}{\partial \dot{\xi}} = \frac{m}{2} 2\dot{\xi} = m\dot{\xi},$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}_{tras}}{\partial \dot{\xi}} = \frac{d}{dt} m\dot{\xi} = m\ddot{\xi},$$

$$\frac{\partial \mathcal{L}_{tras}}{\partial \xi} = -mgZ_I,$$

por lo tanto,

$$m\ddot{\xi} + mgZ_I = F_\xi.$$

Ahora, en el movimiento de rotación, las ecuaciones serían:

$$\frac{d}{dt}\frac{\partial \mathcal{L}_{rot}}{\partial \dot{\eta}} - \frac{\partial \mathcal{L}_{rot}}{\partial \eta} = \tau,$$

$$\mathcal{L}_{rot} = \frac{1}{2} \dot{\eta}^T \mathbb{J} \dot{\eta},$$

realizando las respectivas derivaciones,

$$\frac{\partial \mathcal{L}_{rot}}{\partial \dot{\eta}} = \frac{1}{2} 2\mathbb{J}\dot{\eta} = \mathbb{J}\dot{\eta},$$

$$\frac{d}{dt}\frac{\partial \mathcal{L}_{rot}}{\partial \dot{\eta}} = \mathbb{J}\ddot{\eta} + \dot{\mathbb{J}}\dot{\eta},$$

$$\frac{\partial \mathcal{L}_{rot}}{\partial \eta} = \frac{1}{2}\dot{\eta}^T \frac{\partial \mathbb{J}}{\partial \eta}\dot{\eta},$$
$$\mathbb{J}\ddot{\eta} + \dot{\mathbb{J}}\dot{\eta} - \frac{1}{2}\dot{\eta}^T \frac{\partial \mathbb{J}}{\partial \eta}\dot{\eta} = \tau,$$
$$\mathbb{J}\ddot{\eta} + \{\dot{\mathbb{J}} - \frac{1}{2}\dot{\eta}^T \frac{\partial \mathbb{J}}{\partial \eta}\}\dot{\eta} = \tau.$$

Definiendo el término $\mathbb{C}(\eta, \dot{\eta})$ que representa los términos de Coriolis, el cual contiene los efectos giroscópicos y centrífugo asociados con η,

$$\mathbb{C}(\eta, \dot{\eta}) = \dot{\mathbb{J}} - \frac{1}{2}\dot{\eta}^T \frac{\partial \mathbb{J}}{\partial \eta}.$$

Reescribiendo el modelo dinámico anterior se obtiene:

$$m\ddot{\xi} + mgZ_I = F_\xi,$$

$$\mathbb{J}\ddot{\eta} + \mathbb{C}(\eta, \dot{\eta})\dot{\eta} = \tau.$$

Haciendo un cambio de coordenadas, se proponen unas nuevas variables de entrada para τ

$$\ddot{\eta} = \mathbb{J}^{-1}\{\tau - \mathbb{C}(\eta, \dot{\eta})\dot{\eta}\} = \tilde{\tau} \tag{A.7}$$
$$\ddot{\eta} = \tilde{\tau}.$$

Finalmente, se obtiene el modelo matemático del vehículo aéreo en miniatura de cuatro rotores [63]

$$\begin{aligned}
m\ddot{x} &= u(\sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta),\\
m\ddot{y} &= u(\cos\phi\sin\theta\sin\psi - \cos\psi\sin\phi),\\
m\ddot{z} &= u\cos\theta\cos\phi - mg,\\
\ddot{\phi} &= \tilde{\tau_\phi},\\
\ddot{\theta} &= \tilde{\tau_\theta},\\
\ddot{\psi} &= \tilde{\tau_\psi},
\end{aligned} \tag{A.8}$$

donde x e y son las coordenadas en el plano horizontal, z es la posición vertical, y $\tilde{\tau_\psi}$, $\tilde{\tau_\theta}$ y $\tilde{\tau_\phi}$ son los momentos de yaw, pitch, y roll, respectivamente, los cuales están relacionados con los momentos generalizados τ_ψ, τ_θ y τ_ϕ por la ecuación (A.7).