CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

LABORATORIO UMI-LAFMIA 3157

"Navegación autónoma de un UAV con percepción activa en tiempo real aplicando cuantificación de daños"

TESIS

Que presenta

M.I.M Jairo Olguin Roque

Para obtener el grado de

Doctor en Ciencias

en la Especialidad de

Sistemas Autónomos de Navegación Aérea y Submarina

Directores de Tesis:

Dr. Ivan Gonzales Hernandez
Dr. Sergio Salazar Cruz

Ciudad de México Noviembre 2024.

Dedicatoria

*Dedico este trabajo en primer lugar a **Dios**, por ser la fuente de mi fortaleza, guía y fe, a mis padres, **Fernando** y **Aurelia**, quienes con su amor incondicional, su esfuerzo incansable y su apoyo constante han sido el pilar fundamental de mi vida y mi educación.*

*A mis hermanos, **Gael**, **Oldair** y **Yoel**; mi familia.*

Agradecimiento

Quisiera comenzar expresando mi más sincero agradecimiento al **Consejo Nacional de Humanidades, Ciencia y Tecnología (CONAHCYT)**, institución que me ha otorgado el respaldo económico necesario para realizar mis estudios de posgrado. Sin su apoyo, este camino académico habría sido considerablemente más difícil de recorrer.

A mis padres, quienes siempre han sido mi refugio y fuente de fortaleza, les agradezco profundamente. Ellos, con su amor incondicional y aliento constante, han sido la base sobre la cual he construido cada logro, acompañándome en cada paso y dándome el valor para enfrentar cualquier obstáculo.

A mis asesores, el **Dr. Iván González Hernández** y el **Dr. Sergio Salazar Cruz**, mi admiración y gratitud son incomparables. A través de su paciencia, guía y dedicación incansable. Su apoyo y sabiduría han dejado una marca indeleble en mi formación académica, y llevo conmigo su ejemplo como modelo a seguir.

A mis asesoras, la **Dra. Dulce Citlalli Martínez Peón** y la **Dra. Belem Saldívar**, les expreso mi más sincero agradecimiento por su constante apoyo y orientación en cada paso de este proceso. Su perspectiva y conocimientos enriquecieron profundamente mi trabajo, y su dedicación fue una fuente constante de inspiración. Agradezco el tiempo y esfuerzo que dedicaron a guiarme, así como la confianza que depositaron en mí para alcanzar este logro.

Mi gratitud se extiende también al **Centro de Investigación y de Estudios Avanzados (CINVESTAV)**, por estar en sus instalaciones y proporcionarme un entorno enriquecedor para el desarrollo de mis capacidades. A todo el equipo del laboratorio **UMI-LAFMIA**, gracias por fomentar un ambiente de colaboración y aprendizaje que ha sido fundamental en mi crecimiento profesional.

Finalmente, quisiera recordar las palabras de Alan Turing: *"A veces las personas que nadie imagina capaces de nada son las que logran lo inimaginable."* Esta frase resuena conmigo, pues me recuerda la importancia de confiar en uno mismo y el poder de aquellos que han creído en mí, incluso cuando las adversidades parecían insuperables. Cada logro alcanzado es un testimonio no solo de mi esfuerzo, sino también del apoyo incondicional y la fe que muchos depositaron en mí.

Resumen

Este trabajo de tesis presenta el desarrollo de una plataforma de UAV (Unmanned Aerial Vehicle) capaz de llevar a cabo la navegación autónoma con percepción activa y cuantificación de daños en tiempo real. Para lograrlo, se ha diseñado e implementado un algoritmo de control robusto basado en la técnica FTSMC (Fixed Time Sliding Mode Control) para asegurar la estabilidad y precisión de la trayectoria del UAV en condiciones adversas y cambiantes especialmente en situaciones donde existen perturbaciones que pueden afectar su rendimiento.

La innovación central de este proyecto reside en la capacidad del UAV para identificar y evaluar desastres naturales en tiempo real mediante el uso de inteligencia artificial. La percepción activa, a través de sensores avanzados y procesamiento de datos en vuelo, permite al UAV detectar y cuantificar el daño en el entorno circundante, lo que resulta fundamental en situaciones de desastre.

La implementación de visión computacional permite que el UAV analice los datos de percepción y los combine con información geoespacial en tiempo real para identificar patrones de desastre, evaluar la magnitud del daño y tomar decisiones informadas sobre las operaciones de respuesta. Esto amplía significativamente las capacidades de respuesta ante desastres naturales, mejorando la eficiencia y la seguridad de las operaciones de búsqueda y rescate, evaluación de daños y toma de decisiones.

En resumen, este proyecto representa un avance significativo en la aplicación de UAV en la detección y respuesta a desastres naturales, combinando tecnologías de control robusto y percepción activa con inteligencia artificial.

Los resultados obtenidos respaldan la eficacia y la robustez del algoritmo propuesto, lo que indica su potencial aplicación en el control de cuadricópteros y su capacidad para perturbaciones externas. Esta investigación contribuye al avance de los sistemas de control en aeronaves no tripuladas y proporciona una base solida para futuros desarrollos y mejoras en esta área.

Abstract

This thesis presents the development of a UAV (Unmanned Aerial Vehicle) platform capable of performing autonomous navigation with active perception and damage quantification in real-time. To achieve this, a robust control algorithm based on the FTSMC (Fixed Time Sliding Mode Control) technique has been designed and implemented to ensure the stability and accuracy of the UAV trajectory in adverse and changing conditions, especially in situations where there are disturbances that may affect its performance.

The core innovation of this project lies in the UAV's ability to identify and assess natural disasters in real time through artificial intelligence. Through advanced sensors and in-flight data processing, active sensing allows the UAV to detect and quantify damage in the surrounding environment, which is critical in disaster situations.

Implementation of computational vision allows the UAV to analyze perception data and combine it with real-time geospatial information to identify disaster patterns, assess the extent of damage, and make informed decisions about response operations. This significantly expands natural disaster response capabilities, improving the efficiency and safety of search and rescue, damage assessment and decision-making operations.

In summary, this project represents a significant advance in the application of UAVs in natural disaster detection and response, combining robust control and active perception technologies with artificial intelligence.

The results support the proposed algorithm's effectiveness and robustness, indicating its potential application in quadcopter control and its capability for external disturbances. This research contributes to the advancement of unmanned aircraft control systems and provides a sound basis for future developments and improvements in this area.

Índice general

Índice de figuras

Lista de Acrónimos

ANN Artificial Neural Network. v

API Application Programming Interface. v

CAD Computer-Aided Design. v

CNN Convolutional Neural Network. v

FTSMC Fixed Time Sliding Mode Control. v

GAN Generative Adversarial Network. v

GPS Global Positioning System. v

IMU Inertial Measurement Unit. v

IoU Intersection over Union. v

LiDAR Light Detection and Ranging. v

mAP Mean Average Precision. v

MAVLink Micro Air Vehicle Link. v

MAVROS MAVLink to ROS. v

NTSMC Nonlinear Time Sliding Mode Control. v

PID Proportional Integral Derivative. v

ROS Robot Operating System. v

SMC Sliding Mode Control. v

SVM Support Vector Machine. v

UAV Unmanned Aerial Vehicle. v

VANT Vehiculo Aereo No Tripulado. v

YOLO You Only Look Once. v

1
Introducción

En esta tesis, se investigarán y analizarán las aeronaves no tripuladas, conocidas como UAVs (Unmanned Aerial Vehicles en inglés), son conocidos también como drones y han estado transformando la forma en que se llevan a cabo diversas operaciones en diferentes sectores industriales como se muestra en la Figura 1.1. Los UAVs tienen la capacidad de volar a alturas elevadas y en áreas de difícil acceso. Gracias a esto, junto con la utilización de tecnologías avanzadas de sensores y cámaras, se pueden realizar tareas de inspección y vigilancia con mayor precisión y seguridad [1, 2]. En el ámbito de la agricultura de precisión, los UAVs pueden emplearse para obtener información detallada sobre las condiciones de los cultivos. Esto permite una gestión más eficiente de los recursos de tierra y agua [3, 4].

Figura 1.1: Vehículo Aéreo No Tripulado (VANT) Matrice 100
[5]

Además, los drones también son utilizados para generar mapas de terrenos y áreas geográficas particulares, lo cual resulta extremadamente útil en actividades como la planificación de

infraestructuras y la gestión de recursos naturales [6]. En situaciones de desastres naturales, los drones pueden emplearse para realizar evaluaciones rápidas y precisas de los daños y las necesidades de las comunidades afectadas [7].

De manera similar, en el campo del transporte aéreo de cargas, los drones proporcionan una opción rentable y segura para la entrega de suministros y materiales en áreas remotas o de difícil acceso. Además, la versatilidad y eficiencia de los drones los convierten en herramientas esenciales en diversos sectores industriales y sociales [8, 9, 10].

El control proporcional-integral-derivativo (PID) es una técnica de control ampliamente empleada en la industria. Su implementación implica ajustar de manera iterativa los parámetros correspondientes. En un estudio reciente [11], se presentó una propuesta de controlador PID no lineal para regular el empuje total y los momentos aplicados con el fin de seguir una trayectoria deseada en un vehículo aéreo no tripulado.

Este controlador resistente es altamente adaptable a distintos tipos de vehículos aéreos no tripulados (UAV) y proporciona un enfoque prometedor para el control de estos vehículos en situaciones desafiantes y dinámicas, donde se necesitan respuestas rápidas, precisas y resistentes.

Además, en el estudio realizado en [12], se comparó la capacidad de seguimiento de tres controladores distintos en el diseño de un controlador de vuelo para un Quadrotor V-Tail. Los controladores evaluados fueron el proporcional-derivativo (PD), el proporcional-integral-derivativo (PID) y el controlador de modo deslizante (SMC). Según los resultados obtenidos en las simulaciones, se observó que el controlador PD presentaba un error en estado estacionario que no podía corregirse, lo cual podía suponer un inconveniente en aplicaciones que requieren una alta precisión. Por otro lado, el controlador SMC permitía que el cuadricóptero convergiera rápidamente hacia el punto deseado.

En este contexto, el SMC presentan un fenómeno intrínseco conocido como “chattering” el cual representa un desafío para los ingenieros, ya que causa desgaste y reduce la vida útil de los actuadores en un proceso determinado.[13] Este efecto se caracteriza por oscilaciones rápidas y de alta frecuencia en la señal de control. Además, el chattering puede ocasionar problemas en el funcionamiento del sistema, como la generación de ruido de alta frecuencia, el desgaste de los componentes mecánicos y la degradación de la precisión en la respuesta del sistema en algunos casos. Por esta razón, se han llevado a cabo investigaciones para abordar el chattering y encontrar soluciones que ayuden a reducir este efecto indeseado.

Dentro de las soluciones propuestas para mitigar el chattering en sistemas de control basados en modos deslizantes, se han explorado diversas estrategias. Entre ellas se incluyen el empleo de técnicas de filtrado, la adaptación de la ganancia del controlador y el diseño de algoritmos de control adaptativo. Estas soluciones han demostrado ser efectivas para reducir el chattering y mejorar el rendimiento del sistema de control, según se ha comprobado en investigaciones previas [14].

Por ende, este estudio se enfoca en uno de los métodos de control que ha sido investigado en la literatura para mejorar la precisión y la robustez del control de vuelo del UAV. En el artículo de referencia [15], se propone un enfoque basado en un controlador de modo deslizante (SMC) junto con un observador de perturbaciones de tiempo fijo para mejorar el control de vuelo en presencia de perturbaciones externas. Del mismo modo, el artículo [16] presenta un enfoque de control combinado de modo backstepping-deslizante utilizando un lazo de retroalimentación para el seguimiento de la trayectoria de los cuadricópteros.

No obstante, el desafío del control de seguimiento en tiempo finito ha sido abordado en numerosos estudios de investigación. En el artículo de referencia [17], los autores presentan un enfoque de control integral de deslizamiento (ISM) para el control de altitud de un Quadrotor con el objetivo de mejorar la robustez y precisión del controlador de altitud. De manera similar, en el artículo [18], se propone un enfoque de control en modo deslizante para mejorar la precisión y robustez del control de actitud y altitud de un Quadrotor. Por otro lado, en las referencias [19, 20], los autores se centran en el control de seguimiento de trayectoria para un Quadrotor utilizando una combinación de un controlador dinámico de superficie (DSC) y un observador de estado extendido (ESO). Asimismo, en el artículo [21], se propone un enfoque de controlador adaptativo en modo deslizante para mejorar la precisión en el seguimiento de trayectoria con capacidad de rechazo de perturbaciones externas. Además, en el artículo [22], los autores presentan un control adaptativo no singular y robusto en modo deslizante para un Quadrotor UAV sujeto a perturbaciones.

Asimismo, en el artículo de referencia [23], se empleó un controlador adaptativo para ajustar los parámetros del controlador de modo deslizante (SMC) en tiempo real, lo cual permitió mejorar aún más la capacidad del sistema para adaptarse a condiciones ambientales adversas. La técnica del SMC se utiliza con el propósito de mejorar la robustez y adaptabilidad del control de vuelo frente a perturbaciones externas, como ráfagas de viento. De manera similar, el artículo

mencionado en la referencia [24] aborda el fenómeno del chattering característico del control en modo deslizante. Las oscilaciones inherentes a este tipo de control pueden afectar la trayectoria, por lo que se propone un controlador de modo deslizante con acción integral para reducir la vibración y mejorar la precisión en el seguimiento de la trayectoria del UAV.

El enfoque propuesto en la referencia [25] utiliza un control de velocidad de modo deslizante mejorado que combina una función de conmutación suavizada y un filtro de paso bajo para atenuar el ruido y mejorar la precisión del control de velocidad del motor, mientras que en la referencia [26], se propone un método de control donde los ángulos estimados convergen al valor deseado para el alcance de la orientación y por lo tanto ejecutan el seguimiento de la posición del vehículo a la referencia dada.

A continuación, en la referencia [27] también se utiliza un observador basado en un controlador de modo deslizante para estabilizar una aeronave Quadrotor en vuelo estacionario en ambiente exterior, mientras que en la referencia [28] se utiliza un observador para estimar perturbaciones y una ley de control de seguimiento en tiempo finito para conseguir un seguimiento robusto de la trayectoria en un vehículo aéreo. Además, en la referencia [29] también se utiliza un control robusto basado en modos deslizantes de tiempo fijo para controlar la actitud de una aeronave Quadrotor en presencia de perturbaciones externas.

La contribución de este proyecto es la implementación de un algoritmo de control basado en un controlador de modo deslizante de tiempo fijo (FTSMC) para mejorar los errores de seguimiento de trayectoria hasta cero, proporcionando resultados robustos en presencia de perturbaciones externas como ráfagas de viento. El FTSMC es una técnica de control utilizada en sistemas dinámicos no lineales, como las aeronaves Quadrotor, que permite alcanzar un estado de control estable en un tiempo fijo. Esta técnica utiliza una superficie de deslizamiento para mantener el sistema en un estado estable, incluso en presencia de perturbaciones o incertidumbres en el sistema [30]. El FTSMC es un algoritmo de control ventajoso para aplicar en aeronaves Quadrotor debido a su capacidad para proporcionar un rendimiento de control robusto y rápido en tiempo real. Este algoritmo utiliza una ley de control basada en una superficie deslizante para reducir el error entre la salida deseada y la salida real del sistema. Además, esta estrategia reduce las perturbaciones externas y tiene en cuenta la dinámica no modelada. Por tanto, FTSMC es una potente técnica de control para aeronaves Quadrotor que permite alcanzar una gran precisión y estabilidad en la navegación y posicionamiento de estos vehículos aéreos

no tripulados.

Al mismo tiempo, los resultados obtenidos han demostrado que el FTSMC implementado proporciona robustez al realizar un seguimiento más preciso de la trayectoria hasta la referencia deseada con una capacidad de mejora para el rechazo de perturbaciones externas que afectan al vehículo aéreo. Este algoritmo propuesto se desarrolla utilizando los errores no lineales de la dinámica de la aeronave Quadrotor. Finalmente, las simulaciones obtenidas del control propuesto se comparan con un NTSMC (non-singular terminal sliding mode controller) para verificar su efectividad y el buen funcionamiento del algoritmo de control robusto propuesto en este trabajo.

La visión computacional ha experimentado un avance significativo en las últimas décadas, convirtiéndose en una tecnología esencial para una variedad de aplicaciones, que van desde la vigilancia hasta el entretenimiento y la educación. Uno de los usos más recientes y prometedores de la visión computacional es la detección de desastres naturales, que puede facilitar en gran medida los esfuerzos de respuesta y recuperación en caso de eventos catastróficos como terremotos [31]. Un estudio reciente ha demostrado la capacidad de la visión computacional para detectar daños estructurales en edificios después de un terremoto, lo que puede ser vital para la toma de decisiones durante los esfuerzos de rescate y recuperación [32].

Como se ha mencionado anteriormente además del control de vuelo, este proyecto también se propone utilizar técnicas de visión por computadora para la detección de desastres naturales. La visión por computadora, alimentada por avances recientes en aprendizaje profundo, ha demostrado ser efectiva en una variedad de tareas, desde la detección de objetos [33] hasta la segmentación semántica [34] y la clasificación de imágenes [35]. En esta tesis, se pretende aplicar estas técnicas para identificar de desastres naturales a partir de las imágenes capturadas por el cuadricóptero.

Además de la navegación autónoma y el control robusto de un UAV, se ha desarrollado un enfoque de vanguardia para la detección de objetos relevantes y la segmentación semántica en tiempo real.

Para lograr esto, se emplea inteligencia artificial, utilizando redes neuronales convolucionales implementadas con TensorFlow [36] y el sistema YOLO (You Only Look Once) [37] para la detección y clasificación de objetos en imágenes capturadas por el UAV. Estos modelos, basados en arquitecturas de redes neuronales convolucionales.

La integración de estos modelos de visión computacional ha permitido que el UAV procese

continuamente los datos de la cámara ZED Mini en tiempo real, identificando y segmentando objetos y áreas relevantes en la escena.

Este enfoque completo, que combina el control robusto y la detección de objetos/segmentación semántica basada en aprendizaje profundo, promete mejorar significativamente la eficacia y la eficiencia de las operaciones de respuesta a desastres naturales. Además, sienta las bases para futuras investigaciones en la aplicación de UAV y tecnologías de inteligencia artificial en situaciones de emergencia.

1.1. Motivación

La motivación detrás de este proyecto de tesis surge de la creciente importancia y aplicación de los cuadricópteros en diversas áreas, como la vigilancia, la inspección de infraestructuras. Los UAVs han demostrado su eficacia y versatilidad en una amplia gama de tareas, lo que ha generado un aumento significativo en su demanda y desarrollo.

Sin embargo, a pesar de los avances en el diseño y la fabricación de cuadricópteros, el control preciso y eficiente de estas aeronaves sigue siendo un desafío importante. La presencia de perturbaciones externas, condiciones ambientales cambiantes y errores de modelado puede afectar su estabilidad y desempeño, lo que limita su capacidad para cumplir con precisión las trayectorias y tareas asignadas.

La necesidad de superar estos desafíos y lograr un control robusto en cuadricópteros ha impulsado la motivación de este proyecto. El desarrollo de un algoritmo basado en un controlador de modo deslizante de tiempo fijo (FTSMC) se presenta como una solución potencial para abordar estos problemas y mejorar significativamente el rendimiento de los cuadricópteros.

Los resultados obtenidos de este proyecto de investigación pueden tener un impacto importante en el desarrollo y la implementación de sistemas de control en aeronaves no tripuladas. La capacidad de controlar con precisión los cuadricópteros en presencia de perturbaciones externas abre nuevas posibilidades en términos de aplicaciones y despliegue de estas aeronaves en entornos desafiantes.

Además una de las principales motivaciones para utilizar UAV en la inspección de estructuras es la mejora en la eficiencia y el ahorro de costos. Los UAV pueden cubrir grandes áreas en un tiempo relativamente corto, capturando imágenes de alta resolución y recopilando datos pre-

cisos sobre la condición de una estructura. Esto reduce significativamente el tiempo necesario para realizar una inspección manual y minimiza los costos asociados con el empleo de personal y equipos especializados.

Muchas estructuras, como puentes o edificios altos, presentan desafíos en términos de acceso físico para los inspectores humanos. En estos casos, los UAV ofrecen una solución invaluable, ya que pueden volar cerca de la estructura y capturar imágenes detalladas desde ángulos difíciles de alcanzar. Esto permite una evaluación exhaustiva y precisa de la condición de la estructura sin poner en peligro la seguridad de los inspectores.

1.2. Planteamiento del problema.

La inspección de estructuras utilizando UAVs ha demostrado ser una solución eficiente y segura para evaluar posibles daños causados por desastres naturales. Sin embargo, uno de los desafíos clave en la implementación exitosa de esta tecnología es garantizar la estabilidad de la trayectoria del UAV durante la inspección. La estabilidad de la trayectoria es crucial para asegurar la captura de imágenes de calidad, evitar colisiones y maximizar la precisión de la detección de daños.

Actualmente, los UAVs se controlan utilizando sistemas de control estándar, que pueden ser susceptibles a perturbaciones y condiciones ambientales variables. Estos factores pueden afectar negativamente la estabilidad de la trayectoria del UAV, lo que resulta en movimientos bruscos, desviaciones no deseadas y dificultades en la captura de imágenes claras y precisas.

Por lo tanto, surge la necesidad de desarrollar una propuesta de control robusto que aborde los desafíos de estabilidad de la trayectoria en la inspección de estructuras utilizando UAV. Este nuevo control robusto debe ser capaz de lidiar eficientemente con perturbaciones y condiciones cambiantes, garantizando una trayectoria suave y precisa para el UAV durante toda la inspección de diversas condiciones.

Además el control robusto también debe permitir una trayectoria precisa y suave del UAV. Esto implica minimizar oscilaciones indeseadas, evitar desviaciones bruscas y asegurar una trayectoria estable y controlada que facilite la captura de imágenes de alta calidad y la detección precisa de posibles daños en la estructura.

En resumen, la estabilidad de la trayectoria es un desafío crucial en la inspección de estruc-

turas utilizando UAV. El desarrollo de un nuevo control robusto que aborde eficientemente este problema es fundamental para garantizar una inspección precisa y segura. Este control robusto debe ser tolerante a perturbaciones, mejorar la precisión de la trayectoria y estar integrado con los algoritmos de detección de daños. Al abordar estos aspectos, se podrá maximizar el potencial de los UAV en la inspección de estructuras y mejorar la seguridad de las infraestructuras en situaciones de desastres naturales.

1.3. Justificación.

La inspección de estructuras utilizando UAV ha demostrado ser una solución altamente efectiva para evaluar posibles daños causados por desastres naturales. Sin embargo, la estabilidad de la trayectoria del UAV durante la inspección es un aspecto crítico que requiere atención especial. La implementación de un control robusto para garantizar la estabilidad de la trayectoria del UAV presenta varias justificaciones importantes.

En primer lugar, la estabilidad de la trayectoria del UAV tiene un impacto directo en la precisión de la detección de daños. Un control robusto asegura una trayectoria suave y precisa, lo que permite capturar imágenes claras y nítidas de la estructura. Esto mejora la precisión de los algoritmos de detección de daños, ya que se basan en información visual precisa para identificar grietas, desplazamientos u otras anomalías en la estructura.

Además, la estabilidad de la trayectoria es crucial para garantizar la seguridad del UAV y de los inspectores humanos. Un control robusto reduce el riesgo de movimientos bruscos, desviaciones no deseadas o colisiones durante la inspección. Esto protege la integridad del UAV y minimiza el riesgo de daños o accidentes que podrían poner en peligro a los inspectores humanos en situaciones peligrosas o de difícil acceso.

Otra justificación importante es la optimización del tiempo y los recursos. Un control robusto garantiza una inspección más eficiente, permitiendo que el UAV cubra áreas más extensas en un tiempo menor. Esto reduce los costos asociados con la mano de obra y los equipos utilizados en la inspección, lo que resulta en un uso más eficiente de los recursos disponibles.

Finalmente, la implementación de un control robusto para la estabilidad de la trayectoria del UAV implica un avance tecnológico y contribuye al desarrollo de la industria de los UAV. Estos avances estimulan la innovación y la mejora continua de las capacidades y aplicaciones de los

UAV en la inspección de estructuras.

1.4. Objetivos.

Objetivo general

Desarrollar un sistema de inspección de estructuras utilizando un UAV y un control robusto para garantizar la estabilidad de la trayectoria durante la inspección implementando un sistema de visión donde identificará la condición de una estructura y así optimizar el tiempo y los recursos empleados en la inspección de estructuras.

Objetivos Específicos

1. **Diseñar el Control Robusto con FTSMC:**
   - Desarrollar el control robusto basado en la técnica FTSMC para garantizar la estabilidad y precisión en la trayectoria del UAV.
   - Realizar simulaciones para evaluar su desempeño y eficacia en términos de estabilidad debido al rechazo de perturbaciones.
2. **Entrenamiento de Modelos de Detección de Objetos:**
   - Recolectar y preparar conjuntos de datos relevantes para la detección de objetos en desastres naturales (edificios en buen estado y derrumbados).
   - Entrenar modelos de detección de objetos basados en redes neuronales utilizando TensorFlow.
   - Evaluar la precisión y el rendimiento de los modelos de detección de objetos.
3. **Adquisición de Datos y Configuración del Hardware:**
   - Configurar y asegurar que la Jetson Nano y la cámara ZED Mini estén funcionando correctamente.
   - Establecer la comunicación entre la cámara y la plataforma para la adquisición de datos de imagen estéreo.
4. **Implementación de Segmentación Semántica:**
   - Diseñar y entrenar modelos de segmentación semántica para identificar áreas específicas en las imágenes relacionadas con derrumbes de edificios.
   - Integrar estos modelos en el flujo de procesamiento de imágenes en tiempo real.

Organización de la Tesis

Para obtener los resultados planteados, esta tesis se estructura en los siguientes capítulos:

- **Capítulo 1: Introducción**

  En este capítulo introductorio, se presentan los objetivos de la investigación que se llevará a cabo en esta tesis. Se destaca la importancia del trabajo en el contexto actual y se proporciona una visión general de la estructura de la tesis para que el lector se familiarice con el contenido.

- **Capítulo 2: Estado del Arte**

  Este capítulo se dedica a explorar investigaciones previas y desarrollos relevantes relacionados con el tema de la tesis. Se busca contextualizar la investigación actual en el contexto de las contribuciones existentes en el campo de estudio.

- **Capítulo 3: Modelo Matemático del UAV**

  Aquí se presenta en detalle el modelo matemático de las ecuaciones dinámicas de un cuadricóptero. Se describen las fundamentos del modelado matemático y se explican las ecuaciones que gobiernan el comportamiento del vehículo aéreo no tripulado.

- **Capítulo 4: Control Robusto**

  En este capítulo, se detalla la aplicación del algoritmo y de la estrategia de control utilizados en el trabajo de investigación. Se establecen los objetivos del control, que incluyen el seguimiento de trayectoria y la estabilidad del cuadricóptero.

- **Capítulo 5: Sistema de Visión Computacional**

  Este capítulo se centra en el desarrollo del sistema de visión computacional integrado en el cuadricóptero. Se explican las tecnologías avanzadas utilizadas y cómo contribuyen a la capacidad del vehículo aéreo no tripulado.

- **Capítulo 6: Simulaciones y Análisis de Resultados**

  Aquí se lleva a cabo la realización de simulaciones numéricas del sistema propuesto. Se detalla el proceso de simulación y se realiza un análisis exhaustivo de los resultados obtenidos en comparación con los objetivos establecidos.

- **Capítulo 7: Conclusiones y Recomendaciones Futuras**

  En este capítulo final, se presentan las principales conclusiones derivadas de la investigación. Además, se ofrecen recomendaciones para futuros trabajos y posibles desarrollos adicionales en el área de estudio.

2
Estado del Arte

El estado del arte en el desarrollo de sistemas cautivos se puede dividir en dos áreas principales. La primera abarca el aspecto científico, donde se presentan artículos que abordan la resolución de problemas desde una perspectiva matemática. La segunda se orienta hacia el ámbito industrial y tecnológico, destacando plataformas comerciales diseñadas para aplicaciones específicas.

2.1. Desarrollo Científico

El uso de UAVs para la inspección de estructuras y la detección de daños ha sido un tema de interés en la comunidad científica en la última década. En particular, la visión por computadora y el aprendizaje automático han mostrado ser herramientas prometedoras para esta aplicación.

Los UAVs han surgido como una herramienta prometedora para la inspección de daños estructurales. Los UAV ofrecen una serie de ventajas sobre los métodos de inspección tradicionales, como su capacidad para volar en zonas peligrosas o inaccesibles, su capacidad para recopilar datos de alta resolución de forma rápida y sencilla, y su capacidad para funcionar de forma autónoma.

Agregando en [38], [39] propusieron un método para detectar daños en estructuras a partir de imágenes utilizando visión computacional. Este trabajo representa un avance significativo en la aplicación de la visión por computadora para identificar y clasificar daños estructurales ver Figura 2.1.

Del mismo modo, en [40] se exploraron el uso de detección de estructuras dañadas después de un colapso utilizando imágenes capturadas por UAVs. Propusieron un método de aprendizaje

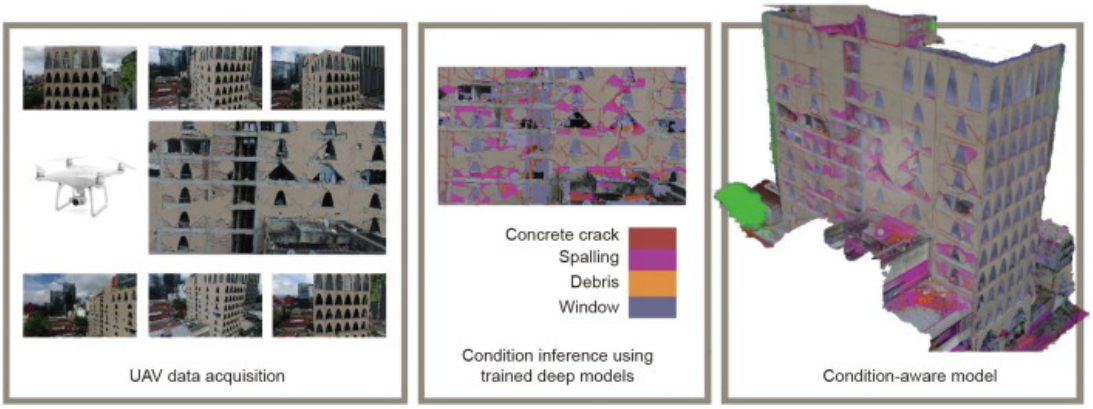

Figura 2.1: Un modelo consciente de la condición de un edificio dañado durante el terremoto del centro de México en septiembre de 2017. [38]

profundo para la detección de daños estructurales, lo que demuestra el potencial del aprendizaje automático en esta área en la Figura 2.2.

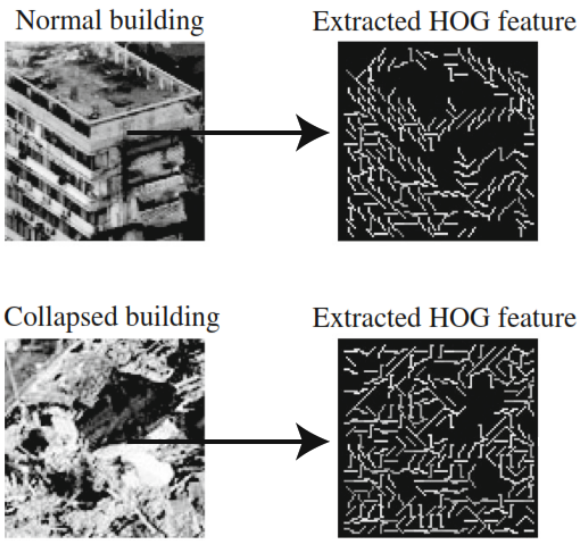

Figura 2.2: Las variaciones de apariencia de edificios normales y derrumbados con la característica HOG (Histogram of Oriented Gradients). [40]

Además del análisis de imágenes, este estudio [41] explora la utilización de sistema de visión modificada para la detección y evaluación de daños en infraestructuras civiles, como las grietas en los edificios. Los autores se centraron en superar las limitaciones de los métodos actuales de inspección de daños, que suelen ser costosos, laboriosos y subjetivos, utilizando UAVs para capturar imágenes de las infraestructuras y aplicar técnicas de inteligencia artificial (IA) y aprendizaje automático (AM).

En su estudio [42], los autores presentan una metodología para automatizar la inspección de edificios utilizando vehículos aéreos no tripulados (UAVs) y visión por computadora, con el objetivo de facilitar la construcción de edificios y la evaluación de daños durante los terremotos. Proponen un enfoque que incluye la recopilación de datos de imágenes basadas en UAV y una biblioteca de software para el procesamiento posterior, lo que ayuda a estimar los parámetros estructurales sísmicos, incluyendo las distancias entre los edificios adyacentes, la forma del plano del edificio, el área del plano del edificio, los objetos en la azotea y la disposición de la azotea en Figura 2.3.

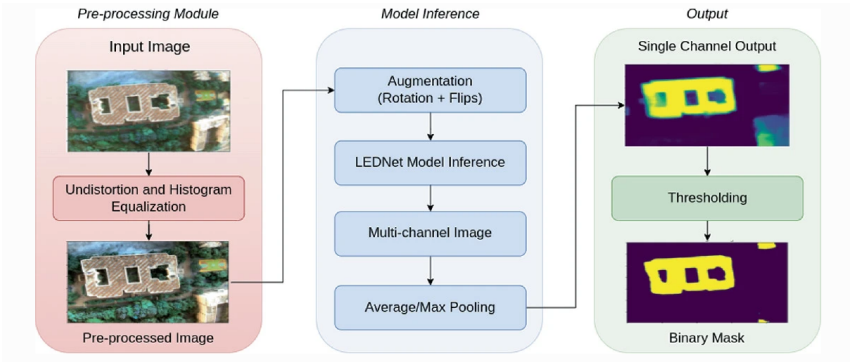

Figura 2.3: Las variaciones de apariencia de edificios normales y derrumbados con la característica HOG.[42]

En la misma línea, [43] propone un nuevo marco de detección de daños en edificios utilizando técnicas de fotogrametría de vehículos aéreos no tripulados y procesamiento de puntos 3D. Este marco se basa en la segmentación refinada por límites y la clasificación de asignación Dirichlet latente (LDA), se mejora el método de segmentación tradicional para segmentar las nubes de puntos.

2.2. Desarrollo Tecnológico

DroneDeploy es una empresa líder en el campo de la tecnología de drones y ofrece soluciones de mapeo y análisis utilizando visión computacional Figura 2.4. En el contexto de los desastres naturales, su plataforma permite la recopilación de datos aéreos utilizando drones equipados con cámaras de alta resolución. Estas imágenes son procesadas utilizando algoritmos de visión por computadora para detectar y evaluar los efectos de los desastres naturales, como inundaciones, incendios forestales o deslizamientos de tierra [44].

La plataforma de DroneDeploy utiliza técnicas de procesamiento de imágenes y aprendizaje automático para analizar los datos capturados por los drones y generar informes detallados sobre la extensión de los daños. Esto incluye la identificación de áreas afectadas, la evaluación de la gravedad de los daños y la generación de mapas de alta resolución que ayudan a los equipos de respuesta a planificar y coordinar las operaciones de rescate y recuperación.

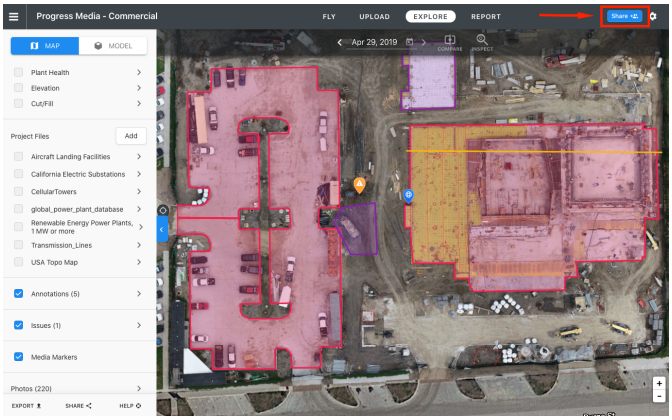

Figura 2.4: Plataforma de DroneDeploy [45].

PrecisionHawk es otra empresa reconocida en el campo de la tecnología de drones y ofrece soluciones de mapeo y análisis utilizando visión computacional para la detección de desastres naturales. Utilizando drones equipados con cámaras y sensores especializados, PrecisionHawk captura datos aéreos de alta calidad en áreas afectadas por desastres [45].

La empresa emplea algoritmos de visión por computadora y análisis geoespacial para procesar los datos recopilados y generar información relevante para la gestión de desastres. Esto puede incluir la identificación de áreas dañadas, el análisis de patrones de inundación, la evaluación de riesgos de deslizamiento de tierra y la detección temprana de incendios forestales. La visión computacional permite a PrecisionHawk generar mapas precisos y detallados que ayudan a los equipos de respuesta a comprender rápidamente la situación y tomar decisiones informadas Figura 2.5.

Kespry es una empresa que ofrece soluciones de drones para la recopilación de datos en diversos sectores, incluyendo la gestión de desastres naturales. Utilizando visión computacional y análisis avanzados, Kespry permite la detección y evaluación precisa de áreas afectadas por desastres [46]. La empresa utiliza drones equipados con cámaras de alta resolución y sensores especializados para capturar imágenes y datos geoespaciales en tiempo real. Estos datos son

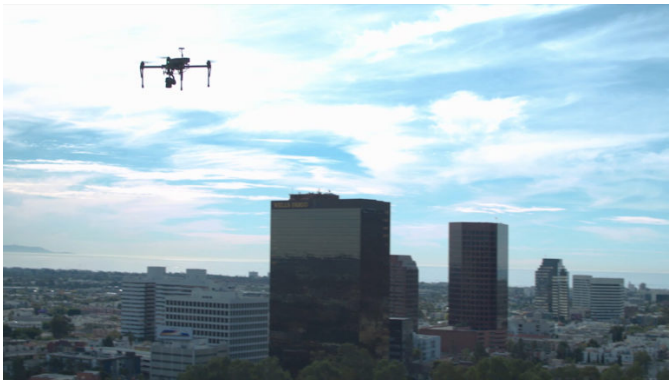

Figura 2.5: Ejecución de Software PrecisionHawk [46].

procesados utilizando algoritmos de visión por computadora para identificar y analizar áreas dañadas, evaluar los riesgos potenciales y generar informes detallados que apoyan las operaciones de respuesta a desastres. La visión computacional utilizada por Kespry permite una rápida interpretación de los datos recopilados y una toma de decisiones más efectiva. Figura 2.6

En resumen, las empresas mencionadas utilizan la visión computacional y la tecnología de drones para capturar y analizar desastres naturales. Sus soluciones permiten la detección temprana de áreas afectadas, la evaluación de los daños y la generación de informes detallados que ayudan a los equipos de respuesta a coordinar las operaciones de recuperación. La visión computacional desempeña un papel clave en el procesamiento de imágenes y datos geoespaciales para identificar patrones, realizar análisis cuantitativos y proporcionar información crítica para la toma de decisiones en situaciones de desastre [47].

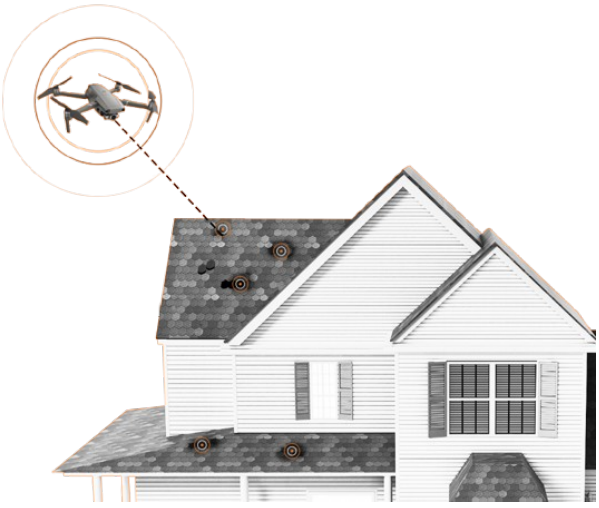

Figura 2.6: Plataforma de análisis en daños [47].

3

Modelado matemático del UAV

Para analizar el impacto de una perturbación aleatoria en la posición y en los ángulos de inclinación de la plataforma, es fundamental comprender el modelo matemático de un cuadricóptero, como se describe en la investigación de [48].

Un cuadricóptero es una aeronave de tipo UAV que se caracteriza por tener cuatro rotores, dispuestos en forma de cruz, que proporcionan el sustento y el control de la aeronave. Cada rotor está impulsado por un motor eléctrico y puede ajustar su velocidad de rotación de manera independiente, lo que permite al cuadricóptero controlar su posición y orientación en el aire.

El cuadricóptero se basa en los principios de vuelo de los helicópteros, pero a diferencia de estós, no cuenta con un rotor principal y un rotor de cola. En su lugar, los cuatro rotores del cuadricóptero trabajan en conjunto para generar las fuerzas necesarias para el vuelo. Al ajustar las velocidades de los rotores de forma diferencial, el cuadricóptero puede cambiar su actitud de inclinación (yaw) , balanceo (roll) y cabeceo (pitch) además de realizar maniobras como ascender, descender, girar y desplazarse en diferentes direcciones.

Con el propósito de comprender los distintos movimientos que efectúa el UAV, es necesario realizar un análisis detallado de su comportamiento físico. Para ello, se emplea un diagrama de fuerzas y sistemas de referencia, como se muestra en la Figura 3.1. Este diagrama ofrece una perspectiva general del cuadricóptero, resaltando los sistemas de coordenadas (x, y, z), las fuerzas f_i, los momentos τ_{Mi} y las velocidades angulares ω_i generadas por cada uno de los cuatro motores $(i = 1, 2, 3, 4)$. La configuración estructural adoptada para este modelo tiene una forma de cruz, donde los motores están ubicados en los extremos de cada brazo, y los ejes

de coordenadas se alinean con los brazos de la estructura.

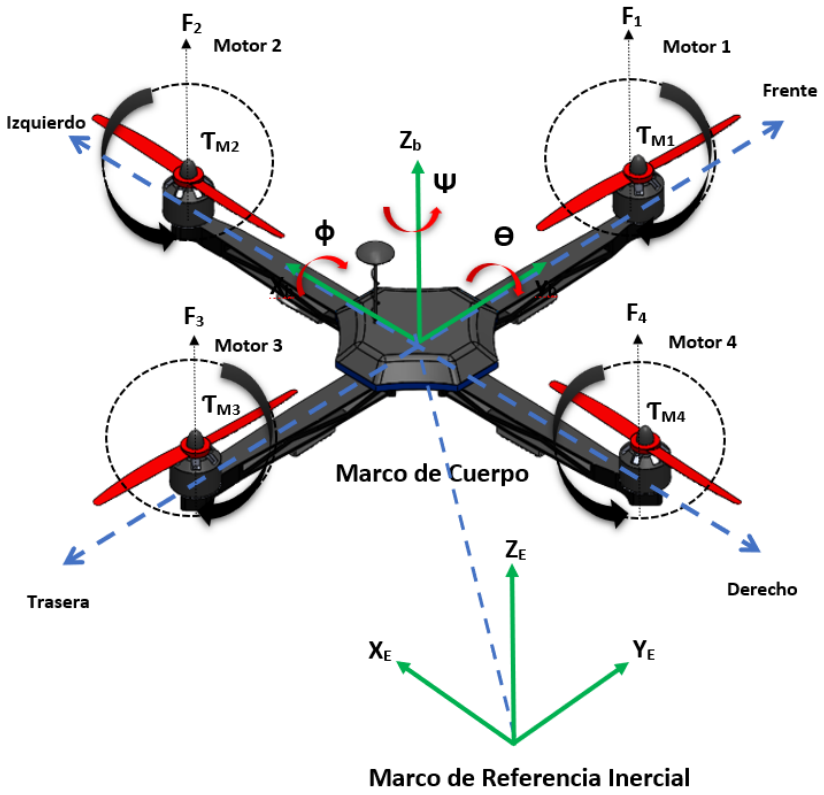

Figura 3.1: Configuracion del UAV, donde se muestran los marcos de referencia, los actuadores y la estructura geométrica del cuerpo del dron.

El cuadricóptero presenta características distintivas, entre las cuales se destaca que el motor delantero y el motor trasero giran en sentido anti-horario, mientras que los otros dos motores giran en sentido de las manecillas del reloj. Esta configuración tiene como objetivo contrarrestar los efectos giroscópicos y los momentos aerodinámicos durante el vuelo estacionario. La fuerza principal o entrada de control se obtiene mediante la suma de las fuerzas generadas por cada uno de los motores.

$$u = \sum_{i=1}^{4} f_i \tag{3.1}$$

El momento de $pitch$ es producido por la diferencia de $f_1 - f_3$, el momento $roll$ es producido por $f_2 - f_4$, y el momento de yaw es la suma de $\tau_{M_1} + \tau_{M_2} + \tau_{M_3} + \tau_{M_4}$, donde τ_{M_i} es el momento de reacción del motor i (Figura 3.1).

Para lograr que el cuadricóptero avance, es necesario aumentar la velocidad del motor tra-

sero M_3 y disminuir la velocidad del motor delantero M_1. Del mismo modo, el movimiento lateral del cuadricóptero se logra utilizando los motores laterales. El giro alrededor de su eje, conocido como movimiento de yaw, se obtiene aumentando el par en los motores delantero y trasero (τ_{M_1}, τ_{M_3}) mientras se minimiza el par en los motores laterales (τ_{M_2}, τ_{M_4}). Es importante destacar que estos movimientos deben realizarse manteniendo constante la fuerza principal, u.

3.1. Modelo Dinámico

Se expone el modelo dinámico del cuadricóptero. El modelo matemático se ha obtenido al considerar al vehículo como un cuerpo sólido que se desplaza en tres dimensiones, sujeto a una fuerza principal y tres momentos o pares. El modelo se deriva utilizando dos enfoques: las ecuaciones de Euler-Lagrange y las leyes de Newton. Mediante estos métodos se obtiene una descripción matemática que permite analizar el comportamiento y la evolución del cuadricóptero en su entorno tridimensional.

$$q = (x, y, z, \psi, \theta, \phi) \in \mathbb{R}^6 \tag{3.2}$$

donde $\xi = (x, y, z) \in \mathbb{R}^3$ indica la posición del centro de masa del cuadricóptero en relación con el sistema de referencia $(\mathcal{I})$, mientras que $\eta = (\psi, \theta, \phi) \in \mathbb{R}^3$ representa los tres ángulos de Euler: ψ es el ángulo de guiñada (yaw), θ el ángulo de alabeo (pitch), y ϕ el ángulo de cabeceo (roll).

La matriz de rotación $R(\psi, \theta, \phi)$ describe la orientación del fuselaje C con respecto a un sistema inercial fijo.

$$R = \begin{pmatrix} c_\theta c_\psi & s_\psi s_\theta & -s_\theta \\ c_\psi s_\theta s_\phi - s_\psi c_\phi & s_\psi s_\theta s_\phi + c_\psi c_\phi & c_\theta s_\phi \\ c_\psi s_\theta c_\phi + s_\psi s_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi & c_\theta c_\phi \end{pmatrix} \tag{3.3}$$

donde S y C representan $\sin$ y $\cos$), respectivamente, de los ángulos θ, ϕ, y ψ.

Estos ángulos describen la orientación del cuadricóptero en relación con el sistema de coordenadas de referencia y son fundamentales para comprender y controlar los movimientos de la aeronave.

La ecuación describe la energía cinética total del sistema, expresada como:

$$T_{\text{trans}} = \frac{m}{2}\dot{\xi}^T\dot{\xi} = \frac{m}{2}\left(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\right) = \frac{1}{2}\dot{\eta}^T W_\eta^T I W_\eta \dot{\eta} = \frac{1}{2}\Omega^T I \Omega \tag{3.4}$$

donde T_{tras} es la energía cinética de traslación del UAV, m es la masa del UAV, y $\dot{\xi} = (\dot{x}, \dot{y}, \dot{z})$ representa el vector de velocidades en el sistema de coordenadas inercial, con $\dot{x}$, $\dot{y}$ y $\dot{z}$ siendo las velocidades en las direcciones x, y y z, respectivamente. El término $\dot{\eta} = (\dot{\psi}, \dot{\theta}, \dot{\phi})$ se refiere al vector de velocidades angulares (yaw, pitch, roll). La matriz W_η es la matriz de transformación entre el sistema de coordenadas del cuerpo y el sistema inercial, I es la matriz de inercia, y Ω es el vector de velocidades angulares en el sistema de referencia del cuerpo. Esta ecuación incorpora tanto la energía cinética de traslación como la energía cinética de rotación del UAV.

Se considera que la energía potencial del cuerpo está formada por la energía potencial gravitatoria.

$$U = -mgz \tag{3.5}$$

Ahora $T_{\text{tras}} = \frac{m}{2}\dot{\xi}^T\dot{\xi}$ representa la energía cinética de traslación del cuadricóptero, $T_{\text{rot}} = \frac{1}{2}\omega^T \mathbf{I}\omega$ relacionada con la energía cinética rotacional, donde T_{rot} es la energía cinética de rotación, ω es el vector de velocidad angular del sistema en el marco de referencia del cuerpo, e I es la matriz de inercia del cuerpo rígido. La ecuación indica que la energía de rotación depende de las velocidades angulares y las propiedades de inercia del sistema. El término $\frac{1}{2}\omega^T I\omega$ representa una forma cuadrática que relaciona las velocidades angulares con la matriz de inercia, lo que permite calcular la energía rotacional del sistema, $U = mgz$ representa la energía potencial del vehículo, donde z es la altura del cuadricóptero. En estas ecuaciones, m denota la masa del cuadricóptero, ω es el vector de velocidad angular, $\mathbf{I}$ representa la matriz de inercia y g corresponde a la aceleración debida a la gravedad. El vector de velocidad angular ω, en relación con los ejes de coordenadas del cuerpo, está asociado con las velocidades generalizadas $\dot{\eta}$.

Se define el Lagrangiano como sigue:

$$\begin{aligned} L(q, \dot{q}) &= T_{\text{tras}} + T_{rot} - U \\ &= \frac{m}{2}\dot{\xi}^T\dot{\xi} + \frac{1}{2}\dot{\eta}^T M(\eta)\dot{\eta} - mgz, \end{aligned} \tag{3.6}$$

Ahora, para obtener el modelo dinámico del UAV, se consideran las ecuaciones de Euler-Lagrange, teniendo en cuenta el vector de fuerzas externas generalizadas que está definido como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{g}}\right) - \frac{\partial L}{\partial g} = F \tag{3.7}$$

donde $F = (F_\xi, \tau)$, F_ξ es el vector de fuerzas que se ejerce sobre el cuadricóptero. Estas fuerzas son los comandos de control que generan movimientos de traslación, y τ es el vector de momentos generalizados, generados por los comandos de control, que inducen movimientos de rotación. Se ignoran todas las demás fuerzas aerodinámicas en el sistema debido a que son significativamente menores en comparación con las fuerzas generadas por los comandos de control principales, bajo la suposición de que las velocidades a las que se mueve el UAV son bajas. Por lo tanto, las fuerzas que actúan sobre el vehículo con respecto a la referencia $\mathbf{R}_b$ son las siguientes.

$$F_b = \begin{pmatrix} 0 \\ 0 \\ u \end{pmatrix} \tag{3.8}$$

y como u queda definida como en (3.1). Además, ω_i es la velocidad angular del motor i; este conjunto de fuerzas expresadas en $\mathbf{R}_w$.La matriz de rotación Rot esta definida por la ecuación (3.3).

$$F_\xi = \mathrm{Rot}\, F_b, \tag{3.9}$$

Ahora en representación de la matriz de rotación es las más utilizada en aplicaciones aéreas el vector de momentos generalizados sobre las variables de η está definido como:

$$\tau \triangleq \begin{pmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{pmatrix} \tag{3.10}$$

como:

$$\begin{aligned}\tau_\phi &= ((f_1 + f_2) - (f_3 + f_4))\, l_c, \\ \tau_\theta &= ((f_2 + f_3) - (f_1 + f_4))\, l_c, \\ \tau_\psi &= \sum_{i=1}^{4} \tau_{M_i},\end{aligned} \tag{3.11}$$

Donde l_c representa la distancia entre los ejes de los motores y el centro de gravedad del vehículo, que se encuentra en el origen de $\mathbf{R}b$, y τM_i es el momento generado por el motor M_i.

Ya que el Lagrangiano no tiene ningún término que combine $\dot{\xi}$ y $\dot{\eta}$ en la ecuación (3.6), se puede dividir la ecuación de Euler-Lagrange (3.7) en dos subsistemas: uno para la dinámica de traslación ξ y el otro para la dinámica de rotación η. La ecuación de Euler-Lagrange para la dinámica de traslación se expresa como sigue.

$$\frac{d}{dt}\left(\frac{\partial L_{tras}}{\partial \dot{\xi}}\right) - \frac{\partial L_{tras}}{\partial \xi} = F_\xi, \tag{3.12}$$

posteriormente se tiene,

$$m\ddot{\xi} + \begin{pmatrix} 0 \\ 0 \\ mg \end{pmatrix} = F_\xi \tag{3.13}$$

Tomando en cuenta la ecuación (3.9), donde F_b y Rot están establecidas en las ecuaciones (3.8) y la matriz de rotación se muestra en la ecuación (3.3). Por lo cual, se obtiene finalmente las ecuaciones que rigen las dinámicas de traslación:

$$\begin{aligned} m\ddot{x}(t) &= u(\sin\phi\sin\psi + \sin\theta\cos\phi\cos\psi), \\ m\ddot{y}(t) &= u(\sin\theta\sin\psi\cos\phi - \sin\phi\cos\psi), \\ m\ddot{z}(t) &= u(\cos\theta\cos\phi) - mg. \end{aligned} \tag{3.14}$$

Posteriormente simplificamos las ecuaciones de traslación, esto se basa en dos técnicas principales. Primero, se asumen valores pequeños para algunos ángulos, lo que permite utilizar aproximaciones como $\sin(\theta) \approx \theta$ y $\cos(\theta) \approx 1$ para simplificar las expresiones trigonométricas. Segundo, se eliminan términos menores o combinaciones complejas de ángulos, como $\sin(\phi)\sin(\psi)$ o $\cos(\phi)\cos(\psi)$, que se consideran insignificantes en configuraciones típicas de

vuelo. Estas aproximaciones permiten reducir la complejidad de las ecuaciones originales, manteniendo únicamente las relaciones esenciales entre el empuje u, los ángulos θ y ϕ, y las aceleraciones en los ejes x, y y z, lo que da como resultado las ecuaciones simplificadas:

$$\begin{aligned} m\ddot{x} &= u\sin\theta, \\ m\ddot{y} &= u\cos\theta\sin\phi, \\ m\ddot{z} &= u\cos\theta\cos\phi - mg, \end{aligned} \tag{3.15}$$

De forma similar, para la dinámica de rotación, se considera ahora la ecuación de Euler-Lagrange correspondiente a este tipo de movimiento:

$$\frac{d}{dt}\left(\frac{\partial L_{rot}}{\partial \dot{\eta}}\right) - \frac{\partial L_{rot}}{\partial \eta} = \tau \tag{3.16}$$

donde;

t : Tiempo, L_{rot} es el Lagrangiano para la rotación, η corresponde a Coordenada generalizada, a menudo un ángulo en contextos de rotación, $\dot{\eta}$ es la Velocidad angular, derivada de η con respecto al tiempo, τ se describe como el Torque o momento de fuerza aplicado al sistema.

Por lo tanto, según [48] se llega al siguiente modelo simplificado para la dinámica de rotación:

$$\begin{aligned} \ddot{x} &= u\sin\theta + \Gamma_x \\ \ddot{y} &= u\cos\theta\sin\phi + \Gamma_y \\ \ddot{z} &= u\cos\theta\cos\phi - mg + \Gamma_z \\ \ddot{\theta} &= \tau_\theta + \Gamma_\theta \\ \ddot{\phi} &= \tau_\phi + \Gamma_\phi \\ \ddot{\psi} &= \tau_\psi + \Gamma_\psi \end{aligned} \tag{3.17}$$

donde u es el empuje total, τ_θ, τ_ϕ y τ_ψ son los pares de los movimientos de rotación, m es la masa, x es el desplazamiento horizontal, y es el desplazamiento vertical, z es el desplazamiento de altitud y g es la aceleración gravitatoria. Γ_x, Γ_y y Γ_z representan las perturbaciones de los movimientos de traslación de los ejes x, y y z; ahora en Γ_θ, Γ_ϕ y Γ_ψ representan las perturbaciones de los movimientos de rotación de los ángulos θ, ϕ y ψ. Todas las perturbaciones son externas. En este modelo, no se contemplan perturbaciones en la dinámica traslacional $(y - x)$

por ser éste el foco principal de análisis, ya que se propone utilizar información visual para obtener referencias (x,y); para ello, tendremos un valor de $\Gamma_z \approx 0$, y posteriormente utilizaremos información GPS para estabilizar el eje z.

4

Control Robusto

El control de UAVs presenta desafíos únicos debido a su naturaleza dinámica y la presencia de perturbaciones externas impredecibles, como ráfagas de viento y turbulencias atmosféricas. Estas perturbaciones pueden afectar el rendimiento del UAV y comprometer su estabilidad y capacidad de seguimiento de trayectorias deseadas.

En este contexto, el presente capítulo se centra en el diseño de un controlador robusto para UAVs, con el objetivo de garantizar la estabilidad y precisión del vehículo en presencia de perturbaciones externas y errores de modelado. El enfoque adoptado se basa en la teoría del control robusto, que busca garantizar el rendimiento del sistema incluso en condiciones adversas y con incertidumbre.

En este Capítulo 4, se abordarán los aspectos fundamentales del diseño del controlador robusto para el UAV. Se comenzará por una revisión exhaustiva de la literatura existente, destacando las investigaciones y avances más relevantes en el campo del control robusto aplicado al UAVs.

Uno de los objetivos principales de aplicar esta técnica en el modelo deslizante es permitir reducir el chattering en el sistema esto mejorará el comportamiento en tiempo real reduciendo las vibraciones que existan en la trayectoria del UAV, es por ello que sea investigado en aplicar un control de modo deslizante, en el articulo [49] habla acerca de la estabilidad y estabilización robusta en tiempo fijo, se proponen nuevos resultados de estabilidad robusta global en tiempo fijo para sistemas escalares utilizando coeficientes de exponente constante y variable, también se aplican a la estabilización global robusta en tiempo fijo de una clase de sistemas no lineales inciertos de segundo orden mediante el uso de control en modo deslizante. El método del control

deslizante ha sido investigado y desarrollado por Utkin [50] en esto el objetivo principal es obligar al sistema dinámico a alcanzar una superficie llamada “superficie de deslizamiento” con un controlador apropiado que garantice el cumplimiento de una restricción sobre la variable de deslizamiento, posteriormente una vez realizado esta restricción las trayectorias del sistema se “deslizan” por la superficie de deslizamiento hacia el equilibrio deseado.

Uno de los objetivos de este estudio es proponer un algoritmo robusto para la estabilización y la trayectoria de un UAV. El algoritmo propuesto se basa en una combinación de la técnica de Control de Modo deslizante en Tiempo Fijo (Fixed-Time Sliding Mode Control) que permiten al UAV adaptarse y responder de manera eficiente a perturbaciones externas y cambios en las condiciones del entorno, a continuación se presenta la metodología para obtener ese algoritmo robusto.

Ahora siguiendo la ecuación de modelo dinámico del UAV de acuerdo con [48] la altitud z puede estabilizarse utilizando la siguiente entrada de control de empuje.

$$u = \frac{m(g + r)}{\cos\theta \cos\phi} \tag{4.1}$$

donde r se define como una señal auxiliar de control utilizada para la estabilización de la altitud del UAV. En particular, se puede observar que:

$$r = -k_{z1}\dot{z} - k_{z2}(z - z_d) \tag{4.2}$$

donde $\dot{z}$ es la velocidad vertical (derivada de la posición z), z_d es la altitud deseada, k_{z1} y k_{z2} son constantes positivas de ganancia del controlador. El primer término, $-k_{z1}\dot{z}$, actúa como un control proporcional a la velocidad vertical del UAV, mientras que el segundo término, $-k_{z2}(z - z_d)$, se encarga de reducir el error de posición entre la altitud actual z y la altitud deseada z_d. Este término r se incorpora en la ecuación de control para garantizar que el UAV siga una trayectoria estable en el eje z.

La posición angular de guiñada puede controlarse aplicando una aproximación de modo deslizante,

$$\tau_\psi = \dot{\psi} - k_\psi \text{sgn}\left(\dot{\psi} + \beta\psi\right) \tag{4.3}$$

donde $k_\psi \geq |\Gamma_\psi|$, β y k_ψ>0; así m tiene un valor constante de 1 como se muestra en la ecuación (4.1) además se sustituye en el modelo 3.17 y esto puede reescribirse como:

$$\ddot{x} = \frac{\tan\theta(g+r)}{\cos\phi} + \Gamma_x \tag{4.4}$$

Para el eje y, es esto:

$$\ddot{y} = \tan\phi(g+r) + \Gamma_y \tag{4.5}$$

Ahora para el eje z, obtenemos:

$$\ddot{z} = r + \Gamma_z \tag{4.6}$$

Posteriormente, para los ángulos de rotación tenemos:

$$\ddot{\theta} = \tau_\theta + \Gamma_\theta \tag{4.7}$$

$$\ddot{\phi} = \tau_\phi + \Gamma_\phi \tag{4.8}$$

$$\ddot{\psi} = \tau_\psi + \Gamma_\psi \tag{4.9}$$

4.1. Modos deseados para la dinámica del eje y del desplazamiento vertical

En este capítulo se estudia el diseño de modos deseados para la dinámica del eje y y el desplazamiento vertical en un UAV. Esta parte del trabajo se enfoca en cómo controlar el movimiento vertical y la posición en el eje y del UAV mediante el ajuste del ángulo de cabeceo. Se propone una metodología que modela la aceleración en el eje y como una función del ángulo de cabeceo y de una entrada de control virtual deseada, representada por $\tan\phi_d$. Este modelo permite que, ajustando el ángulo de cabeceo, se controle efectivamente el desplazamiento vertical del UAV.

La entrada de control deseada, $\tan\phi_d$, es clave en el control de la trayectoria y se diseña para compensar el error no lineal en la dinámica del eje y. La minimización de este error se realiza a través de una ley de control específica, que incorpora una variable de deslizamiento modificada y constantes de control que mejoran la respuesta dinámica del UAV.

La introducción de la dinámica deseada para el eje y y el desplazamiento vertical se logra mediante la definición de una estrategia de control que se enfoca en la estabilización y precisión del seguimiento de trayectoria. El control robusto del desplazamiento vertical implica un análisis detallado de cómo los cambios en el ángulo de cabeceo afectan la elevación y el movimiento del UAV, considerando factores como la gravedad y posibles perturbaciones.

En resumen, este capítulo detalla el enfoque metodológico y técnico adoptado para desarrollar un control preciso del desplazamiento vertical y del movimiento en el eje y del cuadricóptero. La propuesta incluye un análisis matemático y una ley de control que buscan garantizar que el UAV mantenga un comportamiento estable y preciso en su trayectoria, afrontando las incertidumbres y perturbaciones del entorno operativo.

Ahora de la ecuación (4.5), se deduce que:

$$\ddot{y} = \tan\phi(g + r) \pm \tan\phi^d(g + r) + \Gamma_y \tag{4.10}$$

donde tenemos;

$$\ddot{y} = \tan\phi^d(g + r) + e'_y(g + r) + \Gamma_y \tag{4.11}$$

y el error no lineal es:

$$e'_y = \tan\phi - \tan\phi^d \tag{4.12}$$

Además, $\tan\phi^d$ se considera como la entrada de control virtual deseada.

Se define la siguiente variable.

$$\nu_y = \dot{y} + k_y(y - y_d) \tag{4.13}$$

Diferenciando, obtenemos ν_y.

$$\dot{\nu}_y = \ddot{y} + k_y \dot{y} \tag{4.14}$$

con $k_y > 0$ y después de introducir la ecuación (4.11):

$$\dot{\nu}_y = \ddot{y} + k_y \dot{y} = (\tan\phi^d (g + r) + e'(g + r)) + k_y \dot{y} + \Gamma_y \tag{4.15}$$

para obtener la siguiente expresión.

$$\dot{\nu}_y = -\nu_y + e_y \tag{4.16}$$

donde el error e_y es:

$$e_y = e'_y (g + r) \tag{4.17}$$

Se propone la entrada de control virtual deseada $\tan\phi^d$ tal que:

$$(g + r)\tan\phi^d + e'_y (g + r) + k_y \dot{y} = -\nu_y + e'_y (g + r) + \Gamma_y \tag{4.18}$$

obteniendo $\tan\phi^d$.

$$(g + r)\tan\phi^d = -\nu_y - k_y \dot{y} \tag{4.19}$$

De las ecuaciones (4.12), (4.17) y (4.19), se deduce que:

$$e_y = \left(\tan\phi - \tan\phi^d\right)(g + r) = \tan\phi (g + r) + \nu_y + k_y \dot{y} + \Gamma_y \tag{4.20}$$

Derivando lo anterior:

$$\dot{e}_y = \frac{\dot{\phi}}{\cos^2\phi}(g + r) + \tan\phi \dot{r} + \dot{\nu}_y + k_y \ddot{y} + \dot{\Gamma}_y \tag{4.21}$$

Diferenciando de nuevo e introduciendo la ecuación (4.5), se deduce que:

$$\ddot{e}_y = 2\frac{\tan\phi}{\cos^2\phi}\dot{\phi}^2 (g + r) + 2\frac{\dot{\phi}}{\cos^2\phi}\dot{r} + \tan\phi \ddot{r} + \ddot{\nu}_y + k_y y^{(3)} + (g + r)\frac{\tau_\phi + \Gamma_\phi}{\cos^2\phi} + \ddot{\Gamma}_y \tag{4.22}$$

Es importante mencionar que $\ddot{\Gamma}_y$ es la doble derivada de la perturbación en el movimiento traslacional de y y Γ_ϕ es la perturbación en el movimiento rotacional de ϕ. Obsérvese que las derivadas de segundo y tercer orden de r, x y ν_x que aparecen arriba pueden calcularse en función de sus derivadas de primer orden de la siguiente manera.

De las ecuaciones (4.2) y (4.6):

$$\begin{aligned} r &= -k_z\dot{z} - k_z(z - z_d) \\ \dot{r} &= -k_z r - k_z\dot{z} \\ \ddot{r} &= -k_z\dot{r} - k_z\ddot{r} \end{aligned} \tag{4.23}$$

donde $z_d > 0$ es una constante, de la ecuación (4.5):

$$\begin{aligned} \ddot{y} &= \tan\phi(g + r) + \Gamma_y \\ y^{(3)} &= \frac{\dot{\phi}}{\cos^2\phi}(g + r) + \tan\phi\dot{r} + \dot{\Gamma}_y \end{aligned} \tag{4.24}$$

y de la ecuación (4.13) se sustituye.

$$\begin{aligned} \nu_y &= \dot{y} + k_y(y - y_d) \\ \dot{\nu}_y &= \ddot{y} + k_y\dot{y} \\ \ddot{\nu}_y &= y^{(3)} + k_y\ddot{y} \end{aligned} \tag{4.25}$$

Teniendo en cuenta la ecuación (4.22), se propone una superficie de deslizamiento s_{ey}:

$$s_{ey} \triangleq \dot{e_y} + \beta_y e_y \tag{4.26}$$

A continuación, proponemos una función de Lyapunov en la ecuación (4.26):

$$V = \frac{1}{2}s_{ey}^2 \tag{4.27}$$

Diferenciando lo anterior, tenemos

$$\dot{V} = \dot{s}_{ey} = s_{ey}(\ddot{e}_y + \beta_y\dot{e}_y), \;\; \beta_y > 0 \tag{4.28}$$

Ahora se propone el siguiente controlador robusto de modo deslizante.

$$\ddot{e}_y + \beta_y \dot{e}_y = -k_{1y}\,\mathrm{sgn}(s_{ey}) - k_{2y}|s_{ey}|^{\alpha_y}\,\mathrm{sgn}(s_{ey}) - k_{3y}|s_{ey}|^{\gamma_y}\,\mathrm{sgn}(s_{ey}) - k_{4y}y \tag{4.29}$$

Entonces, esto conduce a:

$$\begin{aligned}\dot{V} &= -2k_{1y}|s_{ey}| - 2k_{2y}|s_{ey}|^{\alpha_y+1} - 2k_{3y}|s_{ey}|^{\gamma_y+1} - 2k_{4y}y^2 \\ &\leq -2\left(k_{1y} - \delta_y\right)|s_{ey}| - 2k_{2y}|s_{ey}|^{\alpha_y+1} \\ &\leq -2\left(k_{1y} - \delta_y\right)V(s_{ey})^{\frac{1}{2}} - 2k_{2y}V(s_{ey})^{\frac{\alpha_y+1}{2}}\end{aligned} \tag{4.30}$$

Las ganancias correspondientes de este eje son las mismas que en el apartado siguiente.

Obtengamos ahora τ_ϕ y añadamos un controlador de modo deslizante.

$$\tau_\phi = \frac{1}{g+r}\cos^2\phi[-k_{1y}\,\mathrm{sgn}(s_{ey}) - k_{2y}|s_{ey}|^{\alpha_y}\,\mathrm{sgn}(s_{ey}) - k_{3y}|s_{ey}|^{\gamma_y}\,\mathrm{sgn}(s_{ey}) - k_{4y}y - \beta_y\dot{e}_y \tag{4.31}$$

$$-2\frac{\tan\phi}{\cos^2\phi}\dot{\phi}^2(g+r) - 2\frac{\dot{\phi}}{\cos^2\phi}\dot{r} - \tan\phi\ddot{r} - \ddot{\nu}_y - k_y y^{(3)} - \dot{\Gamma}_\phi - \ddot{\Gamma}_y]$$

donde δ_y es una perturbación externa tal que $|\Gamma_\phi + \ddot{\Gamma}_y| < \delta_y$ para un $\delta_y > 0$ dado; entonces se deduce que $e_y \to 0$ en un tiempo fijo y de la ecuación (4.16):

$$V_y \to 0$$

Después de la ecuación (4.13):

$$y \to y_d$$

y

$$\phi^d \to 0$$

Por lo tanto, a partir de la ecuación (4.19):

$$\phi \to \phi^d$$

Cabe destacar que las trayectorias x y y serán utilizadas por visión artificial y luego analizadas aplicando un filtro de paso bajo que nos permitirá filtrar la perturbación de entrada en la información del eje y.

4.2. Modos deseados para la dinámica del eje x del desplazamiento horizontal

De la ecuación (4.4):

$$\ddot{x} = \frac{\tan\theta(g+r)}{\cos\phi} + \Gamma_x \tag{4.32}$$

Suponiendo que $\cos\phi$ debe tener un valor cercano a 0 porque el eje y se estabiliza en un tiempo fijo, la ecuación (4.32) es:

$$\ddot{x} = \tan\theta(g+r) + \Gamma_x \tag{4.33}$$

Entonces, de la ecuación (4.4) se deduce que:

$$\ddot{x} = \tan\theta^d(g+r) + e'_x(g+r) + \Gamma_x \tag{4.34}$$

donde obtenemos:

$$e'_x = \tan\theta - \tan\theta^d \tag{4.35}$$

y sigue el mismo procedimiento presentado en la subsección anterior. Se define la siguiente variable y se añade que k_x>0.

$$\nu_x = \dot{x} + k_x(x - x_d) \tag{4.36}$$

y también definiendo la variable s_x.

$$\dot{\nu}_x = \ddot{x} + k_4\dot{x} \tag{4.37}$$

Posteriormente, se realiza la misma metodología del eje y para obtener e_x, $\dot{e}_x$ y $\ddot{e}_x$ en el eje x.

$$e_x = \tan\theta(g + r) + \nu_x + k_x\dot{x} + \Gamma_x \tag{4.38}$$

Derivando lo anterior:

$$\dot{e}_x = \frac{\dot{\theta}}{\cos^2\theta}(g + r) + \tan\theta\dot{r} + \dot{\nu}_x + k_x\ddot{x} + \dot{\Gamma}_x \tag{4.39}$$

Derivando de nuevo e introduciendo la ecuación (4.4), se deduce que:

$$\ddot{e}_x = 2\frac{\tan\theta}{\cos^2\theta}\dot{\theta}^2(g + r) + 2\frac{\dot{\theta}}{\cos^2\theta}\dot{r} + \tan\theta\ddot{r} + \ddot{\nu}_x + k_x x^{(3)} + (g + r)\frac{\tau_\theta + \Gamma_\theta}{\cos^2\theta} + \ddot{\Gamma}_x \tag{4.40}$$

donde $\ddot{\Gamma}_x$ es la doble derivada de la perturbación en el movimiento traslacional de x y Γ_θ es la perturbación en el movimiento rotacional de θ. A la vista de la ecuación (4.40), proponemos una superficie de deslizamiento s_{ey}:

$$s_{ex} \triangleq \dot{e_x} + \beta_x e_x \tag{4.41}$$

A continuación, se propone una función de Lyapunov:

$$V = \frac{1}{2}s_{ex}^2 \tag{4.42}$$

Derivando lo anterior:

$$\dot{V} = \dot{s}_{ex} = s_{ex}(\ddot{e}_x + \beta_x\dot{e}_x), \;\; \beta_x > 0 \tag{4.43}$$

Se propone el siguiente controlador de modo deslizante:

$$\ddot{e}_x + \beta_x\dot{e}_x = -k_{1x}\,\mathrm{sgn}(s_{ex}) - k_{2x}|s_{ex}|^{\alpha_x}\,\mathrm{sgn}(s_{ex}) - k_{3x}|s_{ex}|^{\gamma_x}\,\mathrm{sgn}(s_{ex}) - k_{4x}x \tag{4.44}$$

Por fin tenemos:

$$\tau_\theta = \frac{1}{g+r}\cos^2\phi[-k_{1x}\,\mathrm{sgn}(s_{ex}) - k_{2x}|s_{ex}|^{\alpha_x}\,\mathrm{sgn}(s_{ex}) - k_{3x}|s_{ex}|^{\gamma_x}\,\mathrm{sgn}(s_{ex}) - k_{4x}x - \dot{e}_x \quad (4.45)$$

$$-2\frac{\tan\theta}{\cos^2\theta}\dot{\theta}^2(g+r) - 2\frac{\dot{\theta}}{\cos^2\theta}\dot{r} - \tan\theta\ddot{r} - \ddot{\nu}_x - k_x x^{(3)} - \Gamma_\theta - \ddot{\Gamma}_x]$$

donde δ_x es una perturbación externa tal que $|\Gamma_\theta + \ddot{\Gamma}_x| < \delta_x$ para un $\delta_x > 0$, dado.

5

Sistema de Visión Computacional

La visión computacional es una rama de la informática que busca emular la capacidad humana de procesar y comprender la información visual. En el caso de los UAV, la visión computacional es esencial para que estos dispositivos puedan capturar y procesar la información visual de su entorno y tomar decisiones en tiempo real. Para ello, se utilizan técnicas avanzadas de procesamiento de imágenes que permiten a los UAV detectar objetos, clasificarlos, evadir obstáculos y reconocer patrones.

Entre las técnicas de visión más utilizadas en los UAVs se encuentran la detección de objetos, la estimación de la profundidad y la navegación autónoma. La detección de objetos es fundamental para que los UAVs puedan identificar y clasificar objetos en su entorno, lo que les permite, por ejemplo, evitar colisiones o reconocer áreas de interés. La estimación de la profundidad es necesaria para que los UAVs puedan determinar la distancia entre ellos y otros objetos o superficies, lo que es esencial para evitar obstáculos y mantener una trayectoria estable. La navegación autónoma, por su parte, permite a los UAVs tomar decisiones en tiempo real y ajustar su trayectoria de acuerdo a los cambios en su entorno.

Este capítulo presenta el desarrollo e implementación de un sistema de detección y evaluación de daños en edificios por desastres naturales utilizando un UAV y técnicas de visión computacional a través del software OpenCV. El sistema utiliza una cámara ZED Mini y una placa Jetson Nano para la captura y procesamiento de imágenes.

En la Figura 5.1 se presenta un diagrama de flujo que ilustra el proceso completo del desarrollo del sistema, desde la adquisición de datos hasta la evaluación continua del modelo. El

diagrama se organiza en varios bloques principales:

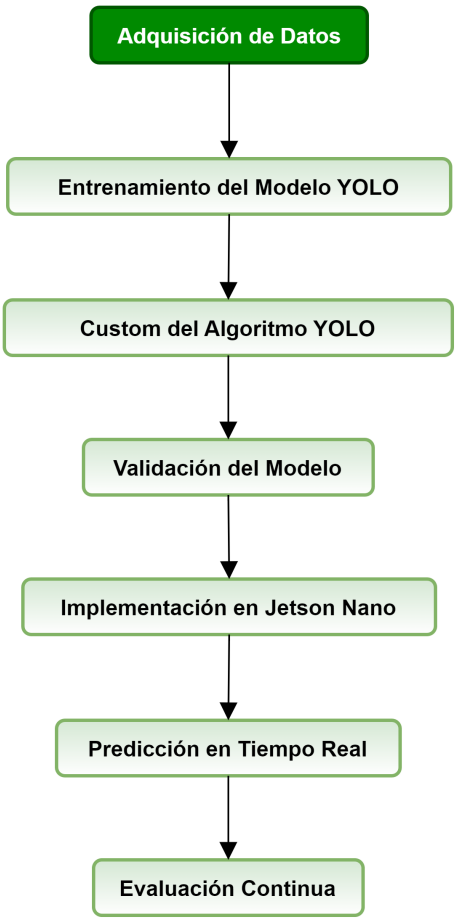

Figura 5.1: Diagrama de Flujo

Este flujo de trabajo garantiza un enfoque estructurado y sistemático para el desarrollo y la evaluación continua de un sistema de detección de daños eficiente y preciso.

5.1. Desarrollo del Algoritmo

En esta sección se centra en el desarrollo de un algoritmo que tiene como objetivo detectar edificios o estructuras y evaluar su condición, es decir, si se encuentran derrumbados o no, en respuesta a un desastre natural. Para ello, se utilizará Python, un lenguaje de programación ampliamente utilizado para la ciencia de datos y el aprendizaje automático, y YOLO(You Only Look Once versión 8) [61] , una biblioteca de funciones de visión por computadora y aprendizaje automático en tiempo real. El objetivo es crear un sistema robusto y confiable que pueda

detectar edificios o estructuras, y evaluar su condición.

En este proyecto [62], se utilizó y se modificó la red neuronal YOLOv8 (You Only Look Once versión 8) específicamente para este algoritmo de detección , es un modelo de vanguardia que se basa en el éxito de versiones anteriores de YOLO e introduce nuevas características y mejoras para aumentar aún más el rendimiento y la flexibilidad. YOLOv está diseñada para ser rápida, precisa y fácil de usar, lo que lo convierte en una excelente opción para una amplia gama de tareas de detección y seguimiento de objetos, segmentación de instancias, clasificación de imágenes y estimación de pose.

En la siguiente sección, se detallará el proceso del desarrollo de este algoritmo, incluyendo la recolección, preparación de datos y entrenamiento del algoritmo.

5.1.1. Adquisición de Datos

La adquisición de datos es una parte esencial de la creación de modelos de entrenamiento con inteligencia artificial que puedan identificar daños estructurales y localizar a las víctimas después de los desastres naturales. Un conjunto de datos bien estructurado y representativo es crucial para entrenar modelos que produzcan resultados precisos y confiables. En esta sección, se analiza el proceso de creación y estructuración del conjunto de datos que se utilizó para entrenar nuestro modelo de detección, que abarca las categorías “Derrumbado”, “Edificio” y “Víctima”.

Para entrenar eficazmente el modelo de detección de objetos, es importante contar con un conjunto de datos robusto y bien representado. A continuación, se visualiza un diagrama de flujo Figura 5.2 donde se describen los pasos para lograrlo.

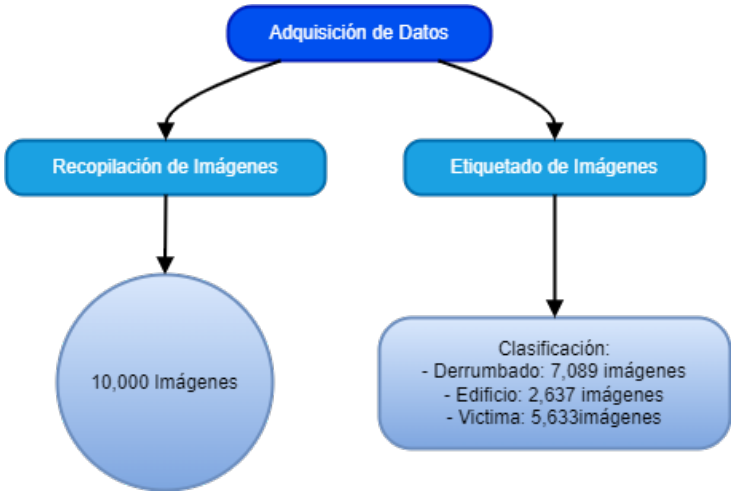

Figura 5.2: Adquisición de Datos

Se han agregado imágenes a distintas clases para mejorar la precisión del algoritmo en la identificación de diferentes tipos de daños y condiciones. Estas clases incluyen:

- **Derrumbado:** Esta clase contiene imágenes de escombros, edificios dañados en sus estructuras, edificios derrumbados como se muestra en la Figura 5.3, etc.

Figura 5.3: Ejemplo de imágenes de la clase Derrumbado.

- **Edificio:** Esta clase incluye imágenes de edificios, casas, rascacielos y otras estructuras intactas Figura 5.4.

Figura 5.4: Ejemplo de imagen de la clase Edificio.

- **Víctima:** Esta clase comprende imágenes de personas tiradas, fallecidas o partes de cuerpos se visualiza en la Figura 5.5.

Figura 5.5: Ejemplo de imagen de la clase Víctima.

Definición de Clases

- **Derrumbado**: Esta categoría incluye imágenes de edificios colapsados o estructuras muy dañadas. Se reunió un total de 7089 imágenes.
- **Edificio**: Esta categoría contiene imágenes de estructuras intactas, como edificios, casas y rascacielos. Se reunió un total de 2637 imágenes.
- **Víctima**: Esta categoría incluye imágenes de personas en situación de víctima, posiblemente fallecidas, o partes del cuerpo. Se reunió un total de 5633 imágenes.

Recopilación de Imágenes

- **Fuentes de Imágenes**: Las imágenes provienen de diversas fuentes para asegurar una representación adecuada. Se utilizan maquetas físicas, simulaciones, y bases de datos públicas y privadas.
- **Condiciones de recopilación**: Se agregaron imágenes en diferentes condiciones de luz y desde varios ángulos para que el modelo pueda adaptarse a diversas situaciones.

Etiquetado de Datos

- **Herramientas Utilizadas**: Se utilizaron herramientas como LabelImg y CVAT para etiquetar manualmente algunas imágenes. Esto implica marcar los objetos de interés con cuadros y asignarles las etiquetas correspondientes.
- **Proceso de Validación**: Se revisaron y validaron las etiquetas para asegurar que sean precisas. Se hicieron revisiones manuales para corregir posibles errores y garantizar la calidad de los datos.

Organización del Conjunto de Datos

- **Estructura del Conjunto de Datos**: Las imágenes se organizaron en carpetas según su categoría y se acompañaron de archivos de anotación adecuados para el modelo de detección utilizado.
- **Formato de Datos**: Los datos se prepararon en un formato compatible con el modelo de detección de objetos (YOLOv8), asegurando que la estructura y las anotaciones fueran correctas.

Preparación para el Entrenamiento

- **Aumento de Datos**: Se aplicaron técnicas de aumento de datos como rotaciones, cambios de tamaño, traslaciones y ajustes de brillo y contraste. Esto ayuda a simular diversas condiciones y aumenta la cantidad de datos efectivos para el entrenamiento.

- **División de Datos**: El conjunto de datos se dividió en tres partes: entrenamiento (70 %), validación (20 %) y prueba (10 %). Esto permite evaluar el rendimiento del modelo de manera efectiva y asegurar que generalice bien a datos nuevos.

Organizar y etiquetar correctamente el conjunto de datos es fundamental para el entrenamiento del algoritmo de detección de objetos. Asegurar la diversidad y calidad de las imágenes ayudará a crear un modelo robusto y preciso para detectar daños y víctimas en situaciones reales de desastre.

5.1.2. Desarrollo del Algoritmo

El proceso de entrenamiento de una red neuronal para identificar las tres clases (Edificio, Dañado y Víctima) se muestra en el Diagrama de Flujo en la Figura 5.6.

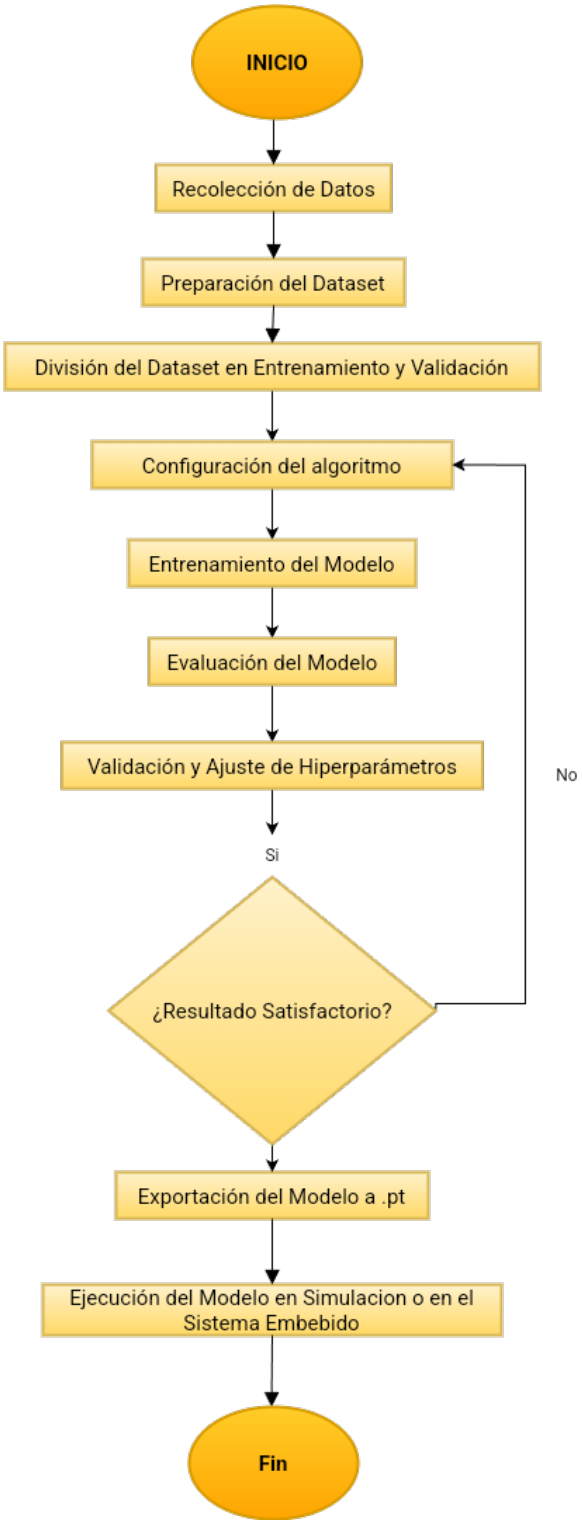

Figura 5.6: Diagrama de flujo del procesamiento de la percepción activa en tiempo real.

El desarrollo del algoritmo comienza con la recolección y anotación de datos, donde se recopilan imágenes relevantes y se etiquetan con las clases correspondientes. Luego, se prepara el dataset y se divide en conjuntos de entrenamiento y validación. Posteriormente, se configura el modelo del algoritmo y se entrena utilizando los datos de entrenamiento. El modelo se evalúa con los datos de validación y, si los resultados no son satisfactorios, se ajustan los hiperparámetros y se repite el entrenamiento. Una vez que se obtienen resultados satisfactorios, el modelo se exporta a un archivo PyTorch (.pt). Finalmente, el modelo exportado se implementa y ejecuta en una simulación o en un sistema embebido, concluyendo así el proceso.

Es de especificar que sea utilizado la arquitectura de YOLOv8 de código abierto capaz de realizar cambios con respecto a nuestros parámetros a entrenar para entrenar un modelo de detección de objetos debe ser capaz de identificar edificios y clasificarlos como dañados o intactos. El entrenamiento implicó utilizar una GPU Tesla T4.

Para realizar el entrenamiento se especifica y personaliza el algoritmo, en la siguiente linea de código se muestra las configuración del entrenamiento de este algoritmo.

```
!yolo task=detect mode=train model=yolov8.pt data={dataset.
   location}/data.yaml epochs=250 imgsz=640
```

Donde:

- task = es especificar el objetivo del algoritmo detectar, segmentar o clasificar.
- train = el modo a especificar entrenamiento, validación, predicción, exportar.
- model = es el modelo preentrenado por Yolov8, detección de objeto.
- data = es la dirección donde se encuentran las imágenes ya organizadas para su entrenamiento.
- epochs: número de épocas para las que queremos entrenar.
- imgsz: el tamaño de la imagen. La resolución predeterminada es 640.

Configuración del Algoritmo

Para desarrollar el algoritmo robusto de detección de daños en estructuras a escala real, se entrenó la siguiente red neuronal. A continuación se describe el proceso de entrenamiento del modelo:

1. **Entorno:** Se configuro el entorno de trabajo en Jupyter y trabajando con Python.
2. **Instalación de Bibliotecas:** Se instalaron las bibliotecas necesarias, incluyendo YOLO.
3. **Descarga del Dataset:** Se cargó un conjunto de datos desde el cual incluyen imágenes clasificadas en las categorías “Derrumbado”, “Edificio” y “Víctima”.
4. **Entrenamiento del Modelo:** Se entrenó el modelo YOLOv utilizando el conjunto de datos descargado. El entrenamiento se llevó a cabo durante 100 épocas con imágenes de alta resolución (640x640). El archivo de configuración del modelo se modificó para detectar las tres clases y ajustar las capas según sea necesario:

```
1
2     # Descarga y modificacion del archivo de configuracion
3     !wget https://github.com/ultralytics/yolov8/releases/download/v8
          .0.196/yolov8x.yaml -O yolov8_custom.yaml
4
5     # Modificacion del archivo para las tres clases
6     # Modificacion del archivo 'yolov8_custom.yaml' como se muestra a
          continuacion
7     nc: 3  # number of classes (Derrumbado, Edificio, V ctima)
8     depth_multiple: 1.0  # model depth multiple
9     width_multiple: 1.0  # layer channel multiple
10
11    # YOLOv8 backbone
12    backbone:
13      [[-1, 1, Conv, [64, 3, 1]],
14       [-1, 1, Conv, [128, 3, 2]],
15       [-1, 2, C2f, [128]],
16       [-1, 1, Conv, [256, 3, 2]],
17       [-1, 6, C2f, [256]],
18       [-1, 1, Conv, [512, 3, 2]],
```

```
      [-1, 9, C2f, [512]],
      [-1, 1, Conv, [1024, 3, 2]],
      [-1, 3, C2f, [1024]],
      [-1, 1, Conv, [1024, 3, 1]],
      [-1, 1, SPPF, [1024, 5]]]  # SPPF layer

  # YOLOv8 head
  head:
    [[-1, 1, Conv, [512, 1, 1]],
     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
     [[-1, 6], 1, Concat, [1]],
     [-1, 3, C2f, [512]],
     [-1, 1, Conv, [256, 1, 1]],
     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
     [[-1, 4], 1, Concat, [1]],
     [-1, 3, C2f, [256]],
     [-1, 1, Conv, [128, 1, 1]],
     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
     [[-1, 2], 1, Concat, [1]],
     [-1, 3, C2f, [128]],
     [[-1, 12], 1, Concat, [1]],
     [-1, 1, C2f, [512, False]],  # Detect(P3, P4, P5)
     [-1, 1, Conv, [256, 3, 2]],
     [[-1, 11], 1, Concat, [1]],
     [-1, 1, C2f, [512, False]],
     [-1, 1, Conv, [128, 3, 2]],
     [[-1, 9], 1, Concat, [1]],
     [-1, 1, C2f, [256, False]],
     [-1, 1, Detect, [nc, anchors]]]  # Detect(P3, P4, P5)

  # Entrenamiento del modelo personalizado
  !yolo task=detect mode=train model=yolov8_custom.yaml data={
      dataset.location}/data.yaml epochs=100 imgsz=640 plots=True
```

5. **Validación del Modelo:** Se validó el modelo entrenado para evaluar su rendimiento en términos de precisión, sensibilidad y especificidad:

```
# Validacion del modelo entrenado
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/
    best.pt data={dataset.location}/data.yaml
```

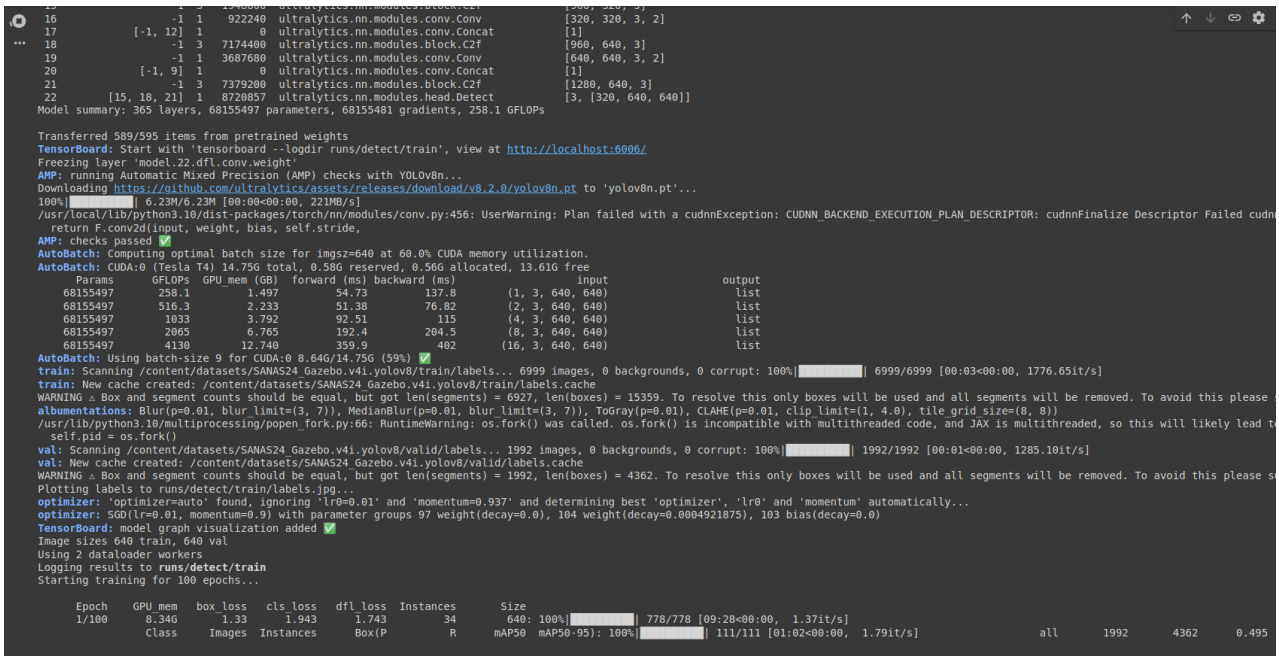

Figura 5.7: Entrenamiento del modelo.

Inicio del Entrenamiento

El entrenamiento comienza con 100 épocas, utilizando dos trabajadores de recopilación de datos. El proceso de entrenamiento registra las pérdidas (`box_loss` y `cls_loss`), así como las métricas de evaluación como `mAP` (media de precisión promedio) y `mAP50`.

Detalles del Entrenamiento por Época La tabla en la parte inferior de la imagen muestra las métricas del entrenamiento por época:

- **Epoch**: Número de época actual.
- **GPU_mem**: Memoria de GPU utilizada.
- **box_loss**: Pérdida asociada a la precisión de los cuadros delimitadores.

- **cls_loss**: Pérdida asociada a la clasificación.
- **dfl_loss**: Pérdida asociada a la función de distancia.
- **Instances**: Número de instancias procesadas.
- **mAP50**: Precisión promedio para una intersección sobre unión (IoU) de 0.50.

La Figura 5.7 proporcionada muestra el proceso de entrenamiento del algoritmo para la detección de objetos, utilizando un conjunto de datos específico. A continuación se describen los detalles clave y las etapas del proceso que se visualizan en la captura de pantalla.

Transferencia de Pesos Preentrenados El proceso de entrenamiento comienza con la transferencia de 589 de 595 elementos de pesos preentrenados. Estos pesos sirven como punto de partida para el algoritmo, permitiéndole aprovechar el conocimiento previo adquirido de otras tareas similares y mejorar su capacidad de generalización.

Configuración y Precisión Mixta Automática (AMP) El algoritmo utiliza Precisión Mixta Automática (AMP) para mejorar la eficiencia del entrenamiento. La AMP permite realizar cálculos en precisión mixta (combinando precisión simple y doble) para acelerar el entrenamiento y reducir el consumo de memoria sin sacrificar la precisión del algoritmo.

Cálculo de Tamaño de Lote Óptimo (AutoBatch) La utilidad AutoBatch calcula el tamaño de lote óptimo para utilizar la memoria de la GPU de manera eficiente. En este caso, se determina que el tamaño de lote óptimo es 64 para una GPU Tesla T4.

Escaneo y Creación de Caché del Conjunto de Datos El proceso de entrenamiento escanea el conjunto de datos ubicado en `/content/datasets/SANASA24_Gazebo.v4i.yolov8/train/labels.cache`,

compuesto por 10000 imágenes. Se crean cachés para acelerar el acceso a los datos durante el entrenamiento.

Configuración del Optimizador y Parámetros del Entrenamiento Se configura el optimizador utilizando SGD (Descenso de Gradiente Estocástico) con un momento de 0.9. Se establecen parámetros adicionales, como el peso de decaimiento y las tasas de aprendizaje.

Visualización de TensorBoard Se habilita la visualización del algoritmo en TensorBoard, permitiendo el monitoreo en tiempo real de las métricas de entrenamiento y validación.

El uso del algoritmo permite una detección rápida y precisa de daños en las imágenes, gracias a su arquitectura optimizada para la detección de objetos. Durante el entrenamiento, se generaron matrices de confusión, gráficos de resultados y ejemplos visuales de las predicciones realizadas por el algoritmo, esto demuestra su efectividad en la identificación de daños estructurales.

Este proceso asegura que el algoritmo pueda ser entrenado, validado y desplegado eficientemente, proporcionando resultados precisos y útiles para la detección de daños en estructuras..

5.1.3. Gráficas de entrenamiento

Matriz de Confusión

Tras 12 horas de entrenamiento, se obtuvieron los resultados, entre los cuales se encuentra la Matriz de Confusión, presentada en la Figura 5.2. Esta matriz es una herramienta fundamental para evaluar el rendimiento de un algoritmo de clasificación. A continuación, se describe la matriz de confusión obtenida de nuestro algoritmo entrenado para las clases "Derrumbado", .[Ed]ificioz "Víctima".

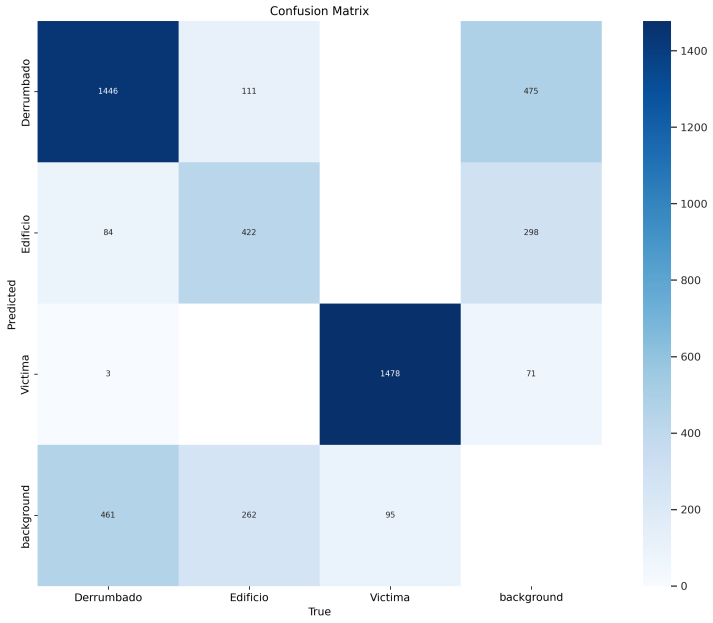

Figura 5.8: Matriz de Confusión.

La matriz de confusión de la Figura 5.8 se compone de cuatro filas y cuatro columnas, donde cada celda indica el número de instancias predichas por el algoritmo en relación con las clases verdaderas. Las filas representan las clases predichas, mientras que las columnas representan las clases reales (verdaderas).

- **Derrumbado (True Positives)**:
  - **Predicciones Correctas (Diagonal Principal)**: La celda en la intersección de la primera fila y la primera columna muestra que el algoritmo predijo correctamente 1446 instancias de “Derrumbado”.
  - **Falsos Positivos**: La primera fila y segunda columna muestra 111 instancias en las que el algoritmo predijo “Derrumbado” cuando la clase real era “Edificio”.
  - **Falsos Negativos**: La primera fila y tercera columna muestra 475 instancias en las que el algoritmo predijo “Derrumbado” cuando la clase real era “Víctima”.
- **Edificio (True Positives)**:
  - **Predicciones Correctas (Diagonal Principal)**: La celda en la intersección de la segunda fila y la segunda columna muestra que el algoritmo predijo correctamente 422 instancias de “Edificio”.

- **Falsos Positivos**: La segunda fila y primera columna muestra 84 instancias en las que el algoritmo predijo “Edificio” cuando la clase real era “Derrumbado”.
- **Falsos Negativos**: La segunda fila y tercera columna muestra 298 instancias en las que el algoritmo predijo “Edificio” cuando la clase real era “Víctima”.

- **Víctima (True Positives)**:
  - **Predicciones Correctas (Diagonal Principal)**: La celda en la intersección de la tercera fila y la tercera columna muestra que el algoritmo predijo correctamente 1478 instancias de “Víctima”.
  - **Falsos Positivos**: La tercera fila y primera columna muestra 3 instancias en las que el algoritmo predijo “Víctima” cuando la clase real era “Derrumbado”.
  - **Falsos Negativos**: La tercera fila y segunda columna muestra 71 instancias en las que el algoritmo predijo “Víctima” cuando la clase real era “Edificio”.
- **Background**:
  - **Predicciones Correctas (Diagonal Principal)**: La celda en la intersección de la cuarta fila y la cuarta columna muestra la cantidad de instancias predichas correctamente como “background”.
  - **Falsos Positivos y Falsos Negativos**: Las demás celdas en la cuarta fila y columna muestran las instancias predichas incorrectamente como “background” o desde “background” a otras clases.

Interpretación de la Matriz

- **Precisión de las Clases**:
  - La clase “Derrumbado” tiene una alta precisión, con 1446 predicciones correctas frente a 111 y 475 errores.
  - La clase “Edificio” tiene una precisión más baja, con 422 predicciones correctas y errores notables en 84 y 298 instancias.
  - La clase “Víctima” muestra una excelente precisión, con 1478 predicciones correctas y errores mínimos en 3 y 71 instancias.
- **Errores Comunes**:

- Las confusiones más comunes ocurren entre las clases “Derrumbado” y “Edificio”, así como entre “Derrumbado” y “Víctima”. Esto podría deberse a características visuales similares en ciertos contextos.
- El algoritmo tiene más dificultades para distinguir correctamente la clase “Edificio” de las otras dos clases, lo que sugiere que se necesita mejorar el entrenamiento o la calidad de las etiquetas en esta clase.

- **Calidad del algoritmo**:

  - La diagonal principal de la matriz muestra un alto número de predicciones correctas, lo que indica que el algoritmo es generalmente preciso.
  - Los valores fuera de la diagonal ayudan a identificar áreas específicas donde el algoritmo podría mejorar.

Esta matriz de confusión proporciona una visión detallada del rendimiento del algoritmo, destacando tanto sus fortalezas como sus áreas de mejora.

Curva de Recall-Confianza

Como siguiente resultado en la Figura 5.9 se muestra la Curva de Recall-Confianza, esta herramienta es utilizada en la evaluación del rendimiento de clasificadores binarios. Esta curva nos indica de manera visual la relación entre la precisión y la sensibilidad del algoritmo, se muestra en la Figura 5.3 que la curva se acerca al valor ideal de la precisión que es 1.

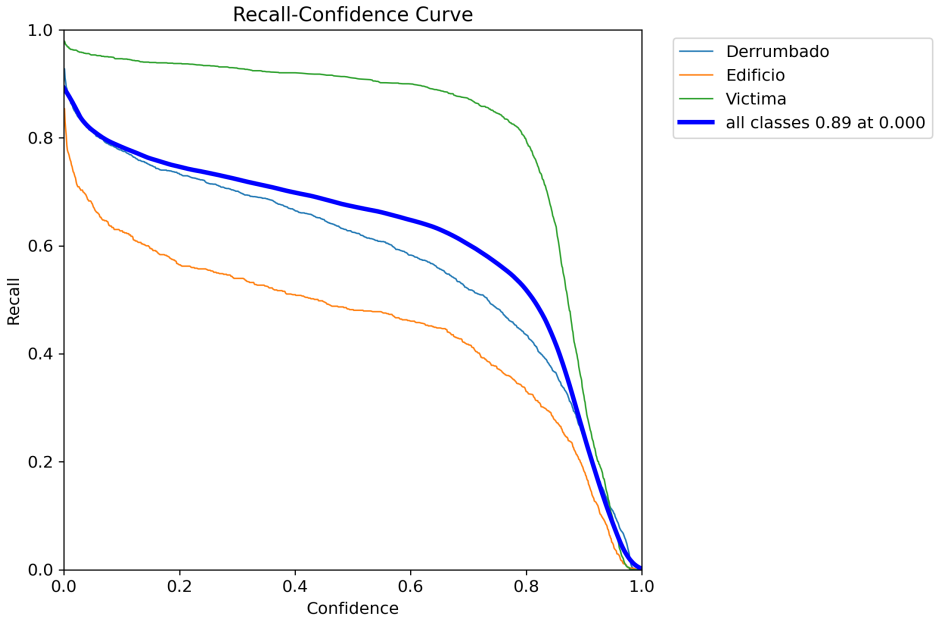

Figura 5.9: Curva de Recall-Confianza.

Descripción de la Curva

- **Ejes**:
  - El eje x representa el nivel de confianza del algoritmo, que varía de 0 a 1. Un nivel de confianza más alto indica que el algoritmo es más seguro de sus predicciones.
  - El eje y representa el recall, que mide la capacidad del algoritmo para encontrar todas las instancias reales de una clase. Un valor de recall más alto indica que el algoritmo está detectando más instancias verdaderas.
- **Clases Representadas**:
  - **Derrumbado**: Representado por la línea azul.
  - **Edificio**: Representado por la línea naranja.
  - **Víctima**: Representado por la línea verde.
  - **Todas las Clases**: Representado por la línea azul gruesa.

Interpretación de la Curva

- **Recall Alto a Bajos Niveles de Confianza**:

- Cuando el nivel de confianza es bajo (cerca de 0), el recall es alto para todas las clases. Esto se debe a que el algoritmo es menos selectivo y clasifica más instancias como verdaderas, aumentando la sensibilidad.

- **Reducción del Recall a Medida que Aumenta la Confianza**:
  - A medida que el nivel de confianza aumenta, el recall disminuye para todas las clases. Esto ocurre porque el algoritmo se vuelve más selectivo y solo clasifica como verdaderas las instancias de las que está más seguro, reduciendo el número de verdaderos positivos detectados.

- **Comparación entre Clases**:
  - **Clase Víctima**: Mantiene un recall alto incluso a niveles de confianza altos, lo que indica que el algoritmo tiene un buen desempeño en la detección de víctimas.
  - **Clase Derrumbado**: Presenta una disminución del recall a medida que aumenta la confianza, pero sigue manteniendo un rendimiento aceptable.
  - **Clase Edificio**: Muestra un recall significativamente más bajo en comparación con las otras clases, indicando que el algoritmo tiene más dificultades para detectar correctamente edificios intactos.

- **Línea de Todas las Clases**:
  - La línea azul gruesa representa el recall promedio para todas las clases. Este valor proporciona una visión general del desempeño global del algoritmo.

Conclusiones

- **Balance entre Recall y Confianza**:
  - Existe un balance inherente entre el recall y la confianza. Un nivel de confianza más alto reduce el recall, mientras que un nivel de confianza más bajo lo aumenta.

- **Desempeño por Clase**:
  - El algoritmo tiene el mejor desempeño en la detección de víctimas, seguido por derrumbados, y tiene más dificultades con los edificios intactos.

- **Ajuste de Umbrales**:
  - Dependiendo de la aplicación específica, se puede ajustar el umbral de confianza para optimizar el recall deseado. Para aplicaciones donde es crucial no perder ninguna instancia verdadera, un umbral de confianza más bajo sería preferible.

Esta curva proporciona una comprensión profunda del comportamiento del algoritmo a diferentes niveles de confianza y es una herramienta valiosa para ajustar y mejorar el rendimiento del algoritmo en tareas de detección.

Curva de F1-Confianza

La curva de F1-confianza 5.10 es una representación gráfica que muestra la relación entre la puntuación F1 del algoritmo y el nivel de confianza en sus predicciones. La puntuación F1 es la media armónica entre precisión y recall, proporcionando una métrica que equilibra ambos aspectos.

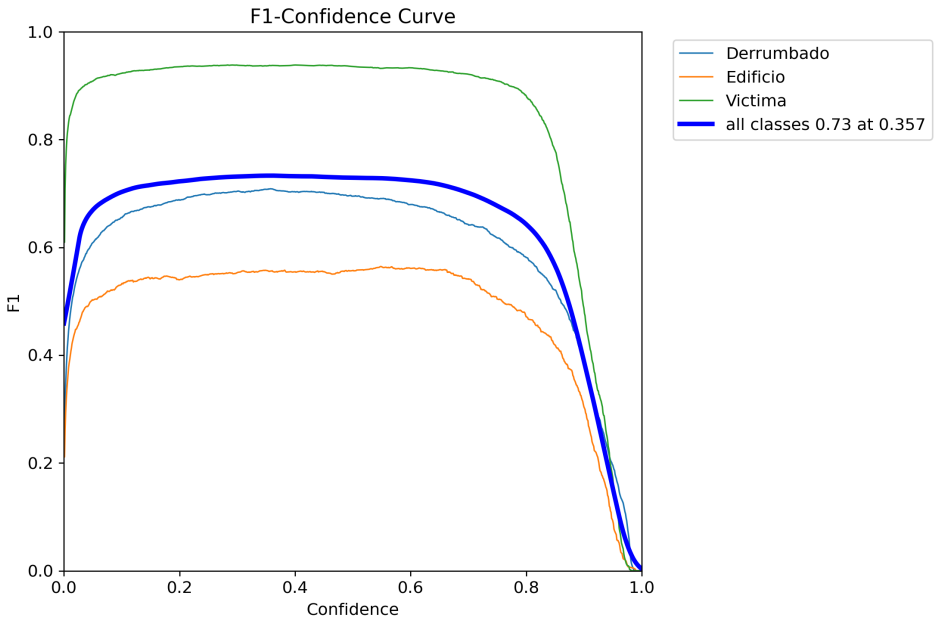

Figura 5.10: Curva de F1-Confianza.

Descripción de la Curva

- **Ejes**:
  - El eje x representa el nivel de confianza del algoritmo, que varía de 0 a 1. Un nivel de confianza más alto indica que el algoritmo es más seguro de sus predicciones.
  - El eje y representa la puntuación F1, que varía de 0 a 1. Una puntuación F1 más alta indica un mejor equilibrio entre precisión y recall.
- **Clases Representadas**:
  - **Derrumbado**: Representado por la línea azul.
  - **Edificio**: Representado por la línea naranja.
  - **Víctima**: Representado por la línea verde.
  - **Todas las Clases**: Representado por la línea azul gruesa.

Interpretación de la Curva

- **Puntuación F1 Alta a Bajos Niveles de Confianza**:
  - Cuando el nivel de confianza es bajo (cerca de 0), la puntuación F1 es alta para todas las clases. Esto se debe a que el algoritmo es menos selectivo y clasifica más instancias como verdaderas, manteniendo un buen equilibrio entre precisión y recall.
- **Reducción de la Puntuación F1 a Medida que Aumenta la Confianza**:
  - A medida que el nivel de confianza aumenta, la puntuación F1 disminuye para todas las clases. Esto ocurre porque el algoritmo se vuelve más selectivo, aumentando la precisión pero disminuyendo el recall, lo que afecta la puntuación F1.
- **Comparación entre Clases**:
  - **Clase Víctima**: Mantiene una puntuación F1 alta incluso a niveles de confianza altos, lo que indica que el algoritmo tiene un buen equilibrio entre precisión y recall en la detección de víctimas.
  - **Clase Derrumbado**: Presenta una disminución moderada de la puntuación F1 a medida que aumenta la confianza, manteniendo un rendimiento aceptable.

- **Clase Edificio**: Muestra una puntuación F1 significativamente más baja en comparación con las otras clases, indicando que el algoritmo tiene más dificultades para mantener un buen equilibrio entre precisión y recall en la detección de edificios intactos.

- **Línea de Todas las Clases**:
  - La línea azul gruesa representa la puntuación F1 promedio para todas las clases. Este valor proporciona una visión general del desempeño global del algoritmo.

Conclusiones

- **Balance entre Puntuación F1 y Confianza**:
  - Existe un balance inherente entre la puntuación F1 y la confianza. Un nivel de confianza más alto reduce la puntuación F1, mientras que un nivel de confianza más bajo la aumenta.

Esta curva proporciona una comprensión profunda del comportamiento del algoritmo a diferentes niveles de confianza y es una herramienta valiosa para ajustar y mejorar el rendimiento del algoritmo.

Curva de Precisión-Confianza

La curva de precisión-confianza 5.11 es una representación gráfica que muestra la relación entre la precisión del algoritmo y el nivel de confianza en sus predicciones. La precisión mide la proporción de predicciones correctas entre todas las predicciones realizadas para una clase.

Descripción de la Curva

- **Ejes**:
  - El eje x representa el nivel de confianza del algoritmo, que varía de 0 a 1. Un nivel de confianza más alto indica que el algoritmo es más seguro de sus predicciones.

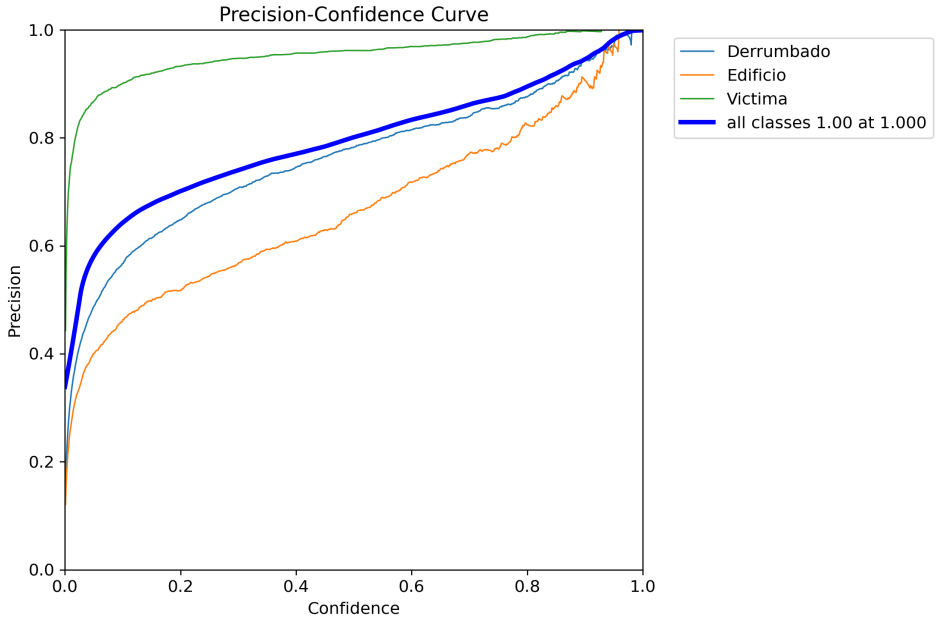

Figura 5.11: Curva de Precisión-Confianza.

- El eje y representa la precisión, que mide la proporción de predicciones correctas entre todas las predicciones realizadas.

- **Clases Representadas**:
  - **Derrumbado**: Representado por la línea azul.
  - **Edificio**: Representado por la línea naranja.
  - **Víctima**: Representado por la línea verde.
  - **Todas las Clases**: Representado por la línea azul gruesa.

Interpretación de la Curva

- **Precisión Bajos a Niveles de Confianza**:
  - Cuando el nivel de confianza es bajo (cerca de 0), la precisión es baja para todas las clases. Esto se debe a que el algoritmo es menos selectivo y clasifica más instancias incorrectamente, lo que disminuye la precisión.
- **Aumento de la Precisión a Medida que Aumenta la Confianza**:
  - A medida que el nivel de confianza aumenta, la precisión también aumenta para todas las clases. Esto ocurre porque el algoritmo se vuelve más selectivo y

clasifica como verdaderas solo las instancias de las que está más seguro, aumentando la proporción de predicciones correctas.

- **Comparación entre Clases**:
  - **Clase Víctima**: Mantiene una precisión alta a través de diferentes niveles de confianza, lo que indica que el algoritmo tiene un buen desempeño en la detección de víctimas.
  - **Clase Derrumbado**: Presenta una precisión moderada a medida que aumenta la confianza, manteniendo un rendimiento aceptable.
  - **Clase Edificio**: Muestra una precisión significativamente más baja en comparación con las otras clases, indicando que el algoritmo tiene más dificultades para detectar correctamente edificios intactos.
- **Línea de Todas las Clases**:
  - La línea azul gruesa representa la precisión promedio para todas las clases. Este valor proporciona una visión general del desempeño global del algoritmo.

Conclusión de gráficas

- **Balance entre Precisión y Confianza**:
  - Existe un balance inherente entre la precisión y la confianza. Un nivel de confianza más alto aumenta la precisión, mientras que un nivel de confianza más bajo la disminuye.
- **Desempeño por Clase**:
  - El algoritmo tiene el mejor desempeño en la detección de víctimas, seguido por derrumbados, y tiene más dificultades con los edificios intactos.
- **Ajuste de Umbrales**:
  - Dependiendo de la aplicación específica, se puede ajustar el umbral de confianza para optimizar la precisión deseada. Para aplicaciones donde es crucial mantener un buen equilibrio entre precisión y recall, se puede seleccionar un nivel de confianza óptimo.

5.1.4. Resumen de Gráficas

A continuación se describen las tres gráficas clave utilizadas para evaluar el rendimiento de un algoritmo de clasificación: la curva de recall-confianza, la curva de F1-confianza, y la curva de precisión-confianza. A continuación, se resumen los objetivos de cada una de estas gráficas, así como sus diferencias.

Curva de Recall-Confianza

Objetivo: La curva de recall-confianza muestra la relación entre el recall (sensibilidad) del algoritmo y el nivel de confianza en sus predicciones. El recall mide la capacidad del algoritmo para detectar todas las instancias verdaderas de una clase. Esta curva es útil para entender cómo varía el recall al ajustar el umbral de confianza del algoritmo.

Diferencia: Esta gráfica se enfoca en la sensibilidad del algoritmo, es decir, su capacidad para identificar correctamente todas las instancias de una clase. A medida que aumenta la confianza, el recall tiende a disminuir porque el algoritmo se vuelve más selectivo.

Curva de F1-Confianza

Objetivo: La curva de F1-confianza muestra la relación entre la puntuación F1 del algoritmo y el nivel de confianza en sus predicciones. La puntuación F1 es la media armónica entre precisión y recall, proporcionando una métrica que equilibra ambos aspectos. Esta curva ayuda a evaluar el rendimiento general del algoritmo teniendo en cuenta tanto la precisión como el recall.

Diferencia: Esta gráfica proporciona una visión más equilibrada del rendimiento del algoritmo, considerando tanto la precisión como el recall. A medida que aumenta la confianza, la puntuación F1 tiende a disminuir debido a la reducción del recall.

Curva de Precisión-Confianza

Objetivo: La curva de precisión-confianza muestra la relación entre la precisión del algoritmo y el nivel de confianza en sus predicciones. La precisión mide la proporción de

predicciones correctas entre todas las predicciones realizadas para una clase. Esta curva es útil para entender cómo varía la precisión al ajustar el umbral de confianza del algoritmo.

Diferencia: Esta gráfica se centra en la precisión del algoritmo, es decir, su capacidad para realizar predicciones correctas sobre el total de predicciones. A medida que aumenta la confianza, la precisión tiende a aumentar porque el algoritmo clasifica como verdaderas solo las instancias de las que está más seguro.

Conclusión

Cada una de estas gráficas proporciona una perspectiva diferente sobre el rendimiento del algoritmo:

- **Curva de Recall-Confianza**: Evalúa la sensibilidad del algoritmo a diferentes niveles de confianza.
- **Curva de F1-Confianza**: Ofrece una visión equilibrada entre precisión y recall.
- **Curva de Precisión-Confianza**: Mide la proporción de predicciones correctas a diferentes niveles de confianza.

La combinación de estas gráficas permite una evaluación completa y detallada del desempeño del algoritmo de clasificación, ayudando a ajustar los umbrales de confianza para optimizar el rendimiento del algoritmo.

5.1.5. Resultados del Entrenamiento

En esta sección se presentan los resultados obtenidos del entrenamiento del algoritmo de detección de objetos utilizando el conjunto de datos compuesto por las clases “Derrumbado”, “Edificio” y “Víctima”. Las figuras adjuntas muestran ejemplos de las predicciones realizadas por el algoritmo sobre un subconjunto de validación.

La Figura 5.12 muestra varias imágenes de validación donde se han identificado y etiquetado las diferentes clases de interés. Cada cuadro delimitador en las imágenes representa una predicción del algoritmo y está etiquetado con la clase correspondiente.

- **Clase Derrumbado**: Etiquetada en color rojo, muestra áreas donde el algoritmo ha identificado estructuras colapsadas o escombros.

- **Clase Edificio**: Etiquetada en color verde, indica las estructuras que el algoritmo ha identificado como edificios intactos.
- **Clase Víctima**: Etiquetada en color naranja, señala las áreas donde el algoritmo ha detectado posibles víctimas.

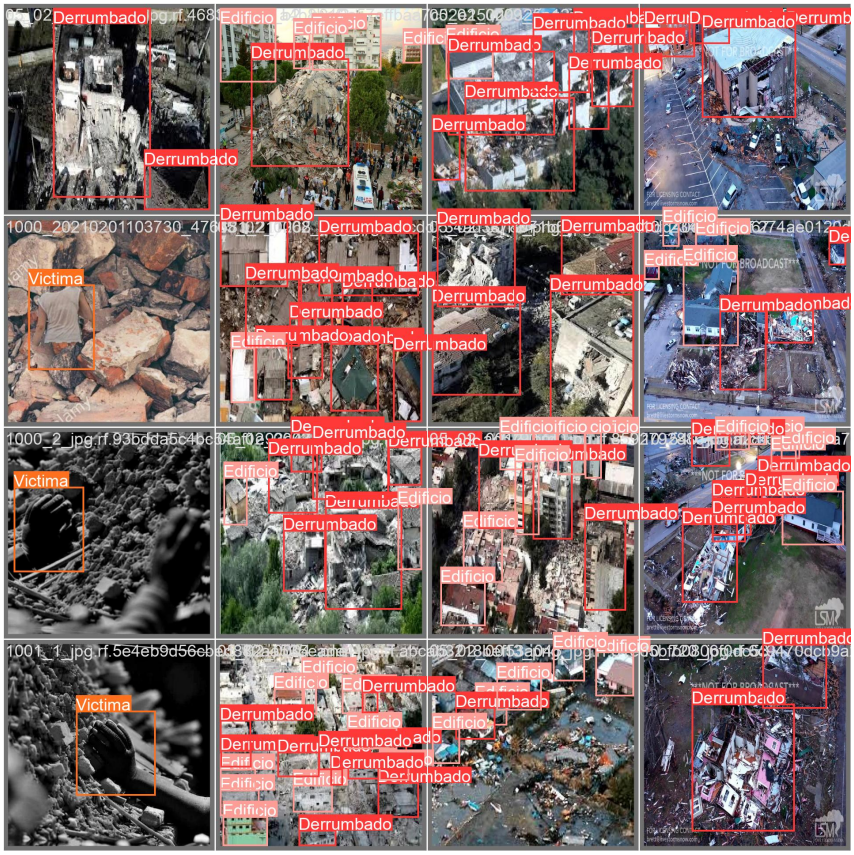

Figura 5.12: Resultado de Predicción en el Conjunto de Validación.

La segunda Figura 5.13, también muestra ejemplos de imágenes de validación con predicciones del algoritmo. Al igual que en la primera figura, los cuadros delimitadores representan las predicciones del algoritmo y están etiquetados con la clase correspondiente.

- **Clase Derrumbado**: Etiquetada en color rojo, muestra áreas donde el algoritmo ha identificado estructuras colapsadas o escombros.
- **Clase Edificio**: Etiquetada en color verde, indica las estructuras que el algoritmo ha identificado como edificios intactos.

- **Clase Víctima**: Etiquetada en color naranja, señala las áreas donde el algoritmo ha detectado posibles víctimas.

El objetivo de estos resultados es demostrar la capacidad del algoritmo para detectar y clasificar correctamente las diferentes clases en un entorno realista. Las predicciones visualizadas permiten evaluar de manera cualitativa el rendimiento del algoritmo, identificando áreas de alta precisión y aquellas donde puede haber confusiones o errores.

Las figuras muestran que el algoritmo puede detectar con precisión las tres clases en diversas condiciones y tipos de escenas. Sin embargo, también se pueden observar algunos casos de traslapamiento entre las clases “Derrumbado” y “Edificio”, lo cual sugiere que el algoritmo podría beneficiarse de un mayor refinamiento en la diferenciación de estas dos clases.

A partir de los resultados mostrados en las figuras se visualiza un claro rendimiento del algoritmo en la tarea de detección de objetos.

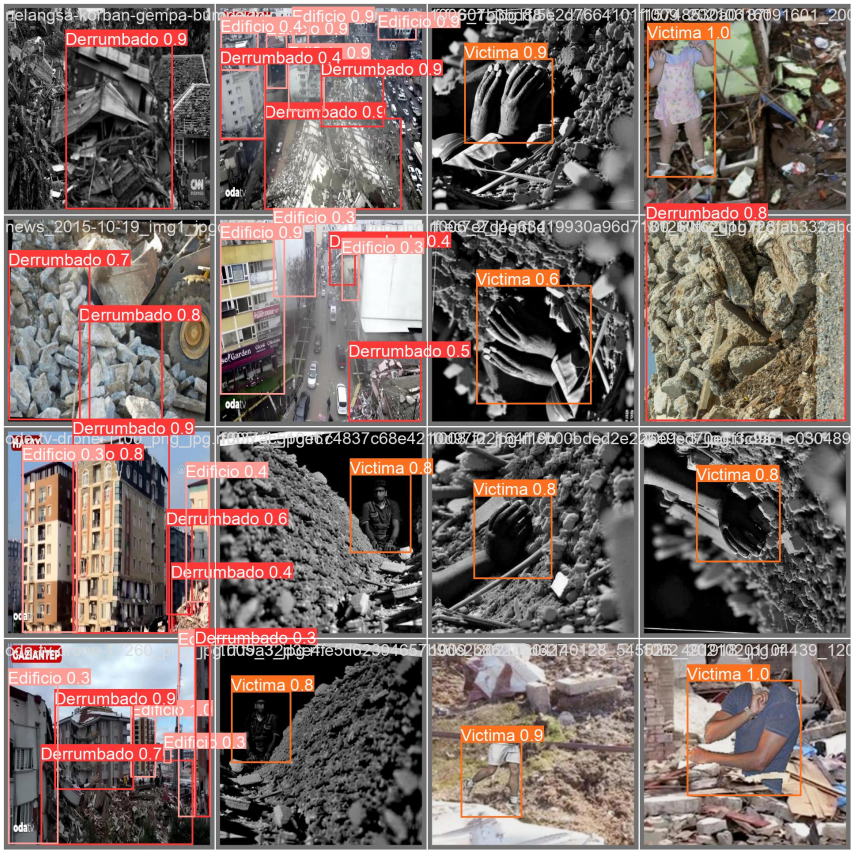

Figura 5.13: Ejemplos de Predicciones en el Conjunto de Validación

6

Resultados en Simulación

En esta sección, los resultados obtenidos se validan mediante simulaciones numéricas y se realiza una comparación con el algoritmo NTSMC propuesto y el controlador PD. Esto permitirá determinar la eficiencia y superioridad del controlador de seguimiento de trayectoria visualizando la rápida velocidad de respuesta, la alta precisión de seguimiento y la minimización de la perturbación; el siguiente controlador se compara respecto a nuestro algoritmo propuesto. Para verificar la robustez del algoritmo de control propuesto, se ha implementado el algoritmo Dryden en las simulaciones con el fin de recrear un entorno real con ráfagas de viento que pudieran afectar a la trayectoria del vehículo aéreo.

El control en modo deslizante terminal no singular (NTSMC) es una técnica de control que destaca por su alta precisión y robustez, y ha sido implementada con éxito en una variedad de aplicaciones. El NTSMC integra las ventajas del control deslizante en modo deslizante (SMC) y el control en modo terminal (TMC), permitiendo un control rápido y preciso de sistemas dinámicos.

En comparación con el SMC convencional, el NTSMC tiene la ventaja de evitar el problema del fenómeno de singularidad, que puede producirse cuando el sistema está cerca de su punto de equilibrio. Además, el NTSMC puede garantizar la convergencia global del sistema a su punto de equilibrio, incluso cuando el algoritmo del sistema es incierto o no lineal.

El NTSMC se publicó en la referencia [64], donde un sistema dinámico no lineal se expresa como sigue:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= f(\boldsymbol{x}) + g(\boldsymbol{x}) + b(\boldsymbol{x})u \end{aligned} \tag{6.1}$$

donde $\boldsymbol{x} = [x_1, x_2]^T$ es el vector de estado del sistema, $f(\boldsymbol{x})$ es la función no lineal suave de $\boldsymbol{x}$ y $g(\boldsymbol{x})$ representa las incertidumbres y perturbaciones que satisfacen $\|g(\boldsymbol{x})\| \leqslant l_g$ donde $l_g > 0$, y u es la entrada de control escalar.

El algoritmo NTSMC propuesto se describe a continuación:

$$s = x_1 + \frac{1}{\beta}\boldsymbol{x}_2^{p/q} \tag{6.2}$$

donde $\beta > 0$ es una constante de diseño, y p y q son enteros impares positivos, que satisfacen la siguiente condición:

$$p > q \tag{6.3}$$

Para el sistema (6.1) con el NTSMC (6.2), la entrada de control obtenida es:

$$u = -b^{-1}(\boldsymbol{x})\left(f(\boldsymbol{x}) + \beta\frac{q}{p}x_2^{2-p/q} + (l_g + \eta)\,\mathrm{sgn}(s)\right) \tag{6.4}$$

donde 1<p / q<2, $\eta > 0$; la prueba de estabilidad se muestra en la referencia [64].

Ahora, para representar el control NTSMC en un cuadricóptero, la variable deslizante s_y se expresa de la siguiente manera:

$$s_y = e_y + \frac{1}{\beta_y} e_y^{p_y/q_y} \tag{6.5}$$

Definiendo:

$$e_y = y^d - y \tag{6.6}$$

Derivando dos veces:

$$\ddot{e}_y = \ddot{y}^d - \ddot{y} \tag{6.7}$$

y de la ecuación (4.4) y considerando que r converge a 0 después de un cierto tiempo.

$$\ddot{e}_y = \ddot{y}^d - g\phi + \Gamma_y \tag{6.8}$$

Entonces, proponiendo ϕ^{d_y} como entrada virtual del sistema (6.8), $\ddot{e}_y$ converge a 0 basando en [64].

$$\phi^{d_y} = \frac{1}{g}(B_y \frac{q_y}{p_y} e_y + (l_{gy} + \eta_y)\,\mathrm{sgn}(s_y) - \ddot{y}^d) \tag{6.9}$$

Para la variable deslizante, s_ϕ se representa como:

$$s_\phi = e_\phi + \frac{1}{\beta_\phi} e_\phi^{p_\phi/q_\phi} \tag{6.10}$$

Ahora definiendo:

$$e_\phi = \phi^{d_y} - \phi \tag{6.11}$$

Derivando dos veces:

$$\ddot{e}_{\phi} = \ddot{\phi}^{d_y} - \ddot{\phi} \tag{6.12}$$

y de la ecuación (4.8):

$$\ddot{e}_{\phi} = \ddot{\phi}^{d_y} - \tau_{\phi} + \Gamma_{\phi} \tag{6.13}$$

entonces, proponiendo ϕ^{d_ϕ} como la entrada virtual del sistema (6.13).

$$\phi^{v_\phi} = \tau_{\phi} \left(B_{\phi} \frac{q_{\phi}}{p_{\phi}} e_{\phi} + (l_{g\phi} + \eta_{\phi}) \operatorname{sgn}(s_{\phi}) - \ddot{\phi}^{d_y} \right) \tag{6.14}$$

Es importante aclarar que $\ddot{e}_{\phi}$ converge a 0 basándonos en la referencia [64]; utilizaremos para los controles del eje y y del cabeceo el mismo procedimiento utilizado en el apartado anterior para el control del alabeo y del eje x.

Para evaluar la efectividad, se han realizado resultados numéricos utilizando herramientas de simulación como Matlab. Además, comparamos el FTSMC propuesto con el NTSMC típico en simulaciones utilizando los mismos valores de parámetros y además se agregó otro controlador PD, esto con la finalidad de comparación y visualización de la efectividad de estos controladores.

El objetivo del control propuesto es conseguir un rendimiento preciso de la cuadricóptero en la posición y actitud en diferentes condiciones de vuelo. Los resultados de la simulación muestran que el FTSMC puede regular con éxito el vehículo aéreo, incluso en presencia de perturbaciones, ruido y dinámicas parásitas. La Tabla 1 muestra los parámetros y las condiciones iniciales.

Es importante destacar que se utilizaron los mismos valores de parámetros y perturbaciones para cada uno de los controladores con el fin de obtener un mejor resultado en la comparación e identificar la superioridad de nuestro algoritmo de control propuesto.

El rendimiento del seguimiento de trayectorias del método de control en modo deslizante de tiempo fijo (FTSMC) propuesto se compara con el del control en modo deslizante terminal no-singular (NTSM).

La Figura 6.4 muestra la convergencia del desplazamiento del eje y a la trayectoria y_d deseada; el controlador FTSMC propuesto tiene una respuesta superior, mientras que el controlador NTSMC se retrasa unos segundos. Es importante mencionar que ambos tienen las mismas ganancias de la perturbación Γ_y. La Figura 6.5 muestra la convergencia del desplazamiento del eje x a la trayectoria x_d deseada. El controlador FTSMC propuesto también se comporta mejor en comparación con el controlador NTSMC, ambos con una perturbación Γ_x, y la Figura 6.6 muestra la convergencia a la altitud deseada de 4 m con una perturbación de Γ_z. La propuesta puede proporcionar una mayor precisión de seguimiento y menos vibración en la entrada de control. Como podemos ver, el controlador FTSMC propuesto puede proporcionar una rápida convergencia a las posiciones deseadas.

La Figura 6.7 muestra que el control FTSMC converge en un periodo de tiempo de 12 s. Sin embargo, el NTSMC tiene un tiempo de convergencia más largo y se produce un pequeño chattering tanto en Γ_ϕ como en θ. La Figura 6.8 también presenta una convergencia del FTSMC en unos 10 s mientras que el NTSMC converge en 18 s, ambos con una perturbación de Γ_θ.

Finalmente, la Figura 6.9 presenta su convergencia a la trayectoria deseada con una perturbación de Γ_ψ además las Figuras 6.10 y 6.11 muestran el comportamiento de la entrada de control τ_ϕ de ambos controles FTSMC y NTSMC, y τ_θ para ambos controles FTSMC y NTSMC donde se presenta el efecto chattering. τ_ψ se muestra en la Figura 6.12. Es importante mencionar que se ha agregado un controlador PD para visualizar su comportamiento con respecto a los otros controladores propuestos y por último, el resultado de

la simulación de la entrada de control u se muestra en la Figura 6.13; los valores de las perturbaciones añadidas se presentan en la Tabla 6.1.

Gain FTSMC	**Value**	**Gain NTSMC**	**Value**	**I. C.**	**Value**	**Perturbation**	**Value**
k_{z1}	1	η_y	1	y_0	$1[\mathrm{m}]$	Γ_y	$\sin 45t[\mathrm{rad/s}]$
k_{z2}	1	q_y	1	$\dot{y}_0$	$0.1[\mathrm{m/s}]$	Γ_x	$\sin 40t[\mathrm{rad/s}]$
k_y	1	p_y	2	x_0	$1[\mathrm{m}]$	Γ_z	$\sin 30t[\mathrm{rad/s}]$
k_x	1	η_x	1	$\dot{x}_0$	$0.1[\mathrm{m/s}]$	Γ_ϕ	$\sin 35t[\mathrm{rad/s}]$
k_ψ	1	q_x	1	z_0	$0.1[\mathrm{m}]$	Γ_θ	$\sin 30t$ $[\mathrm{rad/s}]$
k_{1x}	2.5	p_x	2	$\dot{z}_0$	$0.1[\mathrm{m/s}]$	Γ_ψ	$\sin 45t$ $[\mathrm{rad/s}]$
k_{2x}	1	η_ϕ	1	ϕ_0	$0.1[\mathrm{rad}]$	z_d	4
k_{3x}	1	q_ϕ	1	$\dot{\phi}_0$	$0[\mathrm{rad/s}]$	x_d	6
k_{4x}	1	p_ϕ	2	θ_0	$0.1[\mathrm{rad}]$	y_d	6
k_{1y}	2.5	η_θ	1	$\dot{\theta}_0$	$0[\mathrm{rad/s}]$		
k_{2y}	1	q_θ	1	ψ_0	$0.1[\mathrm{rad}]$		
k_{3y}	1	p_ϕ	2	$\dot{\psi}_0$	$0[\mathrm{rad/s}]$		
k_{4y}	1	l_{gy}	$\sin 45t[\mathrm{rad/s}]$				
α_x	1.5	l_{gx}	$\sin 40t[\mathrm{rad/s}]$				
γ_x	0.5	$l_{g\phi}$	$\sin 35t[\mathrm{rad/s}]$				
α_y	1.5	$l_{g\theta}$	$\sin 30t$ $[\mathrm{rad/s}]$				
γ_y	0.5						
β_y	0.35						
β_x	0.5						

Tabla 6.1: Parámetros de Simulación y Condiciones iniciales

6.1. Interfaz de Simulación

Para desarrollar la interfaz de simulación, se optó por utilizar el modelo de un UAV cuadrirotor. En este capítulo se detalla la estructura de la interfaz de simulación, explicando cómo se integraron los distintos componentes de simulación para observar los efectos de perturbaciones aleatorias en la cobertura proporcionada.

6.1.1. Simulación del Modelo Dinámico

Dentro de la clasificación de vehículos se incluyen aeronaves con características propias de los UAV. Esta configuración es comúnmente utilizada para analizar algunas propiedades específicas de estos vehículos, ya que incorporan todos los componentes necesarios para el control de un vehículo aéreo real. Los UAV poseen seis grados de libertad $(x, y, z, \phi, \theta, \psi)$, que describen su posición y orientación en el espacio.

6.1.2. Integración del Modelo Dinámico

Para llevar a cabo la simulación, se definieron las dinámicas rotacionales y traslacionales del sistema según la ecuación (3.17), ya que estas variarán en función de si se busca desplazar o ajustar la orientación del cuadricóptero en una dirección particular.

6.1.3. Descripción de la plataforma de simulación

El diseño de la plataforma de simulación se organizó como se muestra en el esquema general de la Figura 6.1. Para simular la plataforma UAV, se optó por utilizar Simulink, un entorno de simulación integrado con Matlab que ofrece una amplia gama de herramientas para representar prácticamente cualquier sistema mediante diagramas de bloques. Este entorno resulta especialmente útil, ya que permite introducir las perturbaciones en el modelo de la plataforma UAV.

El vector de salida en el modelo dinámico del UAV está representado por $\mathbf{x}$ e incluye las variables ϕ, θ, ψ, $\dot{\phi}$, $\dot{\theta}$, $\dot{\psi}$, x, y, z, $\dot{x}$, $\dot{y}$ y $\dot{z}$. Las variables empleadas en el modelo son x_7, x_9 y x_{11}, que corresponden al movimiento del UAV en los ejes x, y y z, respectivamente. De manera similar, se utilizan las variables x_1, x_3 y x_5, que representan a ϕ, θ y ψ, respectivamente. Estos valores se extraen para calcular los niveles de potencia recibidos en un programa adicional, mostrado en el bloque de la Figura 6.2 e implementado en Matlab.

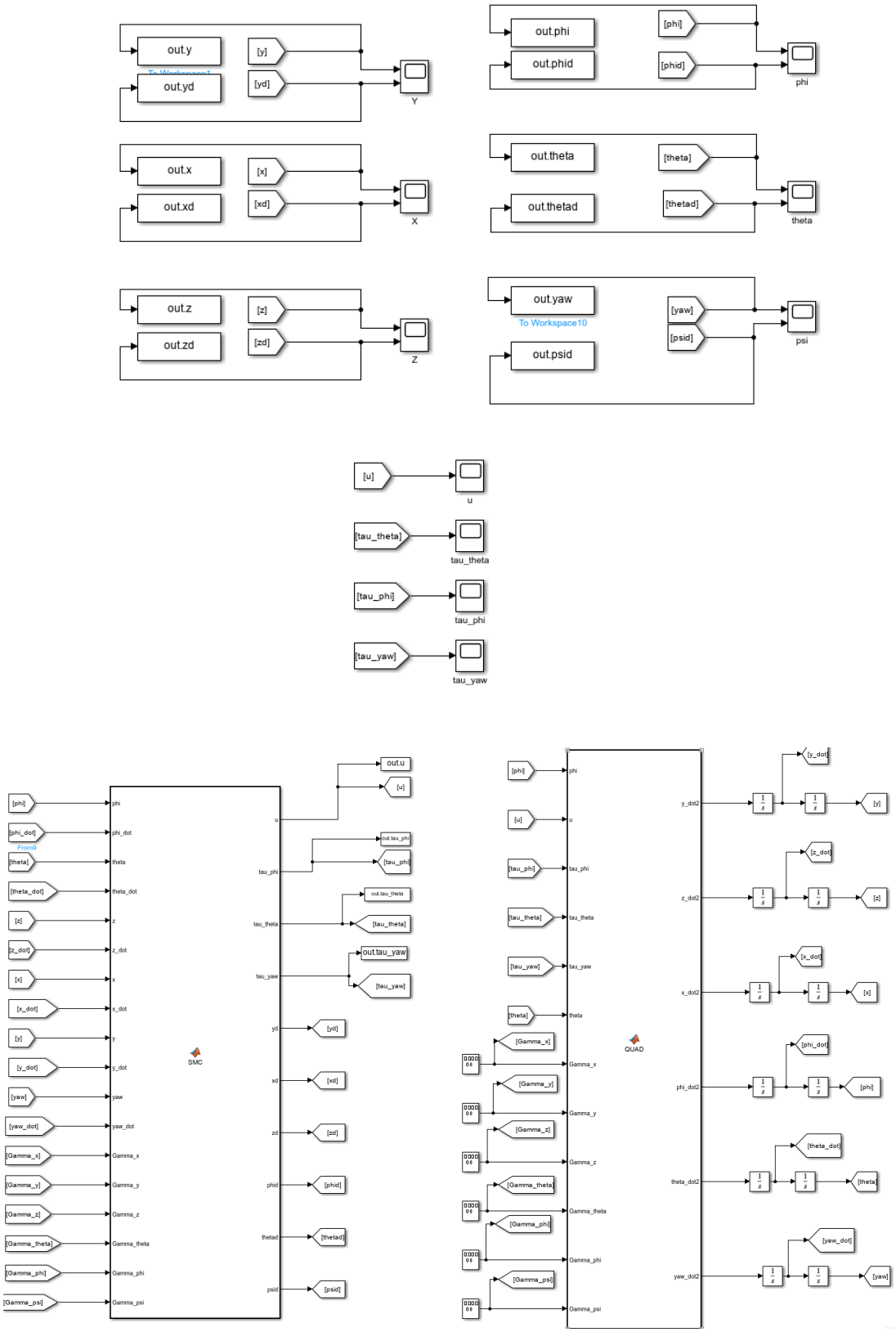

Figura 6.1: Diagrama completo a bloques de la simulación.

Ahora en la Figura 6.1 se muestra el diagrama de bloques para la plataforma de simulación, al igual se visualizan en la parte superior los tres ejes de traslación X,Y, Z creados en la simulación así como en la parte derecha se visualiza los bloque creados para los movimientos de rotación ϕ, θ, ψ y en la parte inferior se crearon las entradas de la señal de u, τ_θ, τ_ϕ y τ_ψ, como se muestra en la Figura 6.2. Al mismo tiempo en la Figura 6.3 se muestran los diagramas a bloques de las variables de entrada para el modelo matemático dentro de ellas se retroalimentan los movimientos de traslación y rotación además de los movimientos deseados de x_d, y_d, z_d y ϕ_d, θ_d, ψ_d.

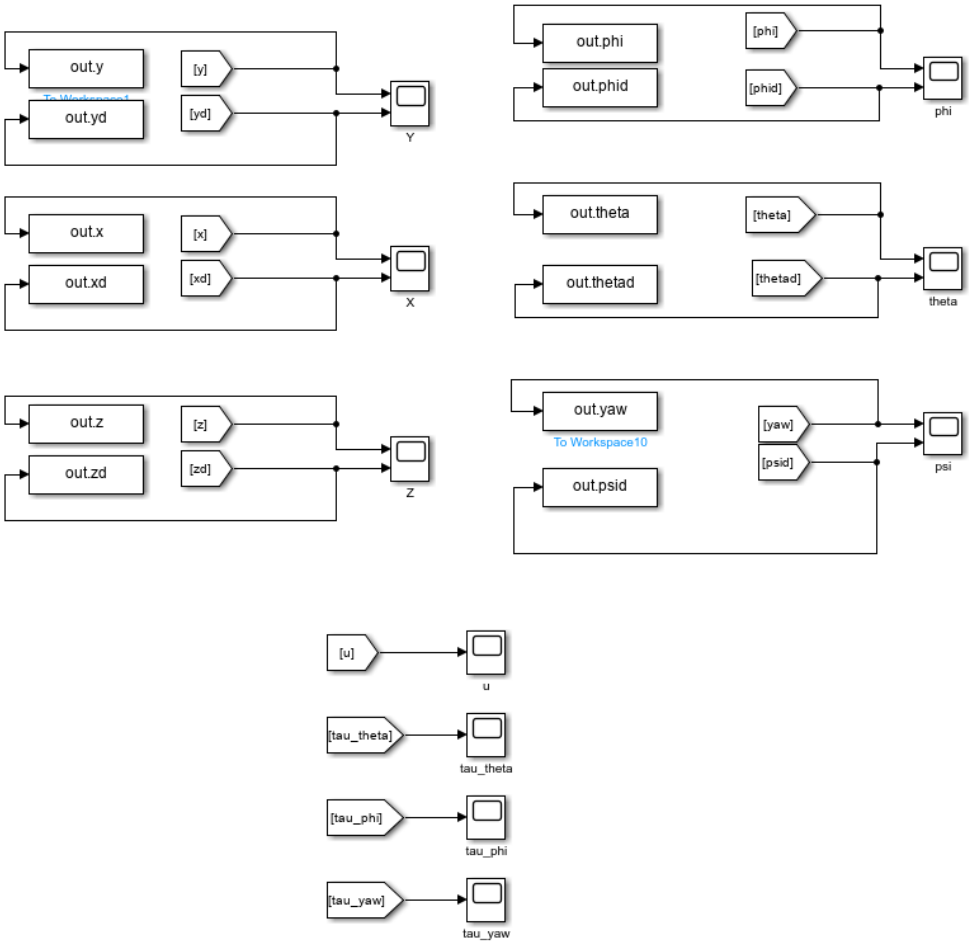

Figura 6.2: Diagrama a bloques movimientos en traslación y rotación.

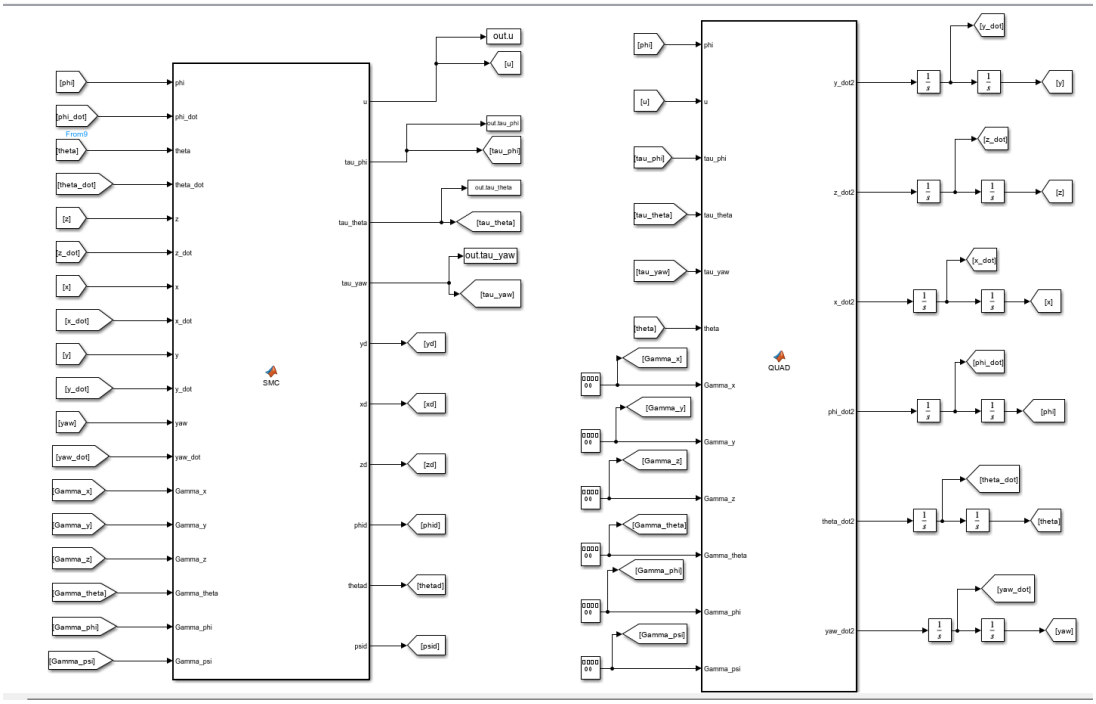

Figura 6.3: Diagrama a bloques de variables de entrada del modelo.

Cabe destacar que con la ayuda de la simulación se redujo las posibilidad de errores de navegación ya que las simulaciones se contaban con ubicación en GPS para mejorar la trayectoria del UAV y sus actividades a ejecutar.

6.1.4. Resultados de la plataforma de simulación

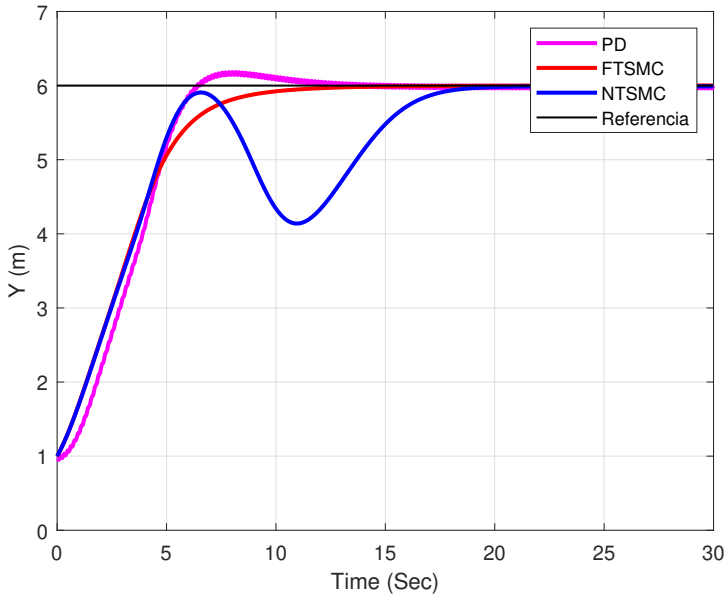

Figura 6.4: Respuesta de desplazamiento del algoritmo FTSMC comparada con el control NTSMC y el control PD en el eje y.

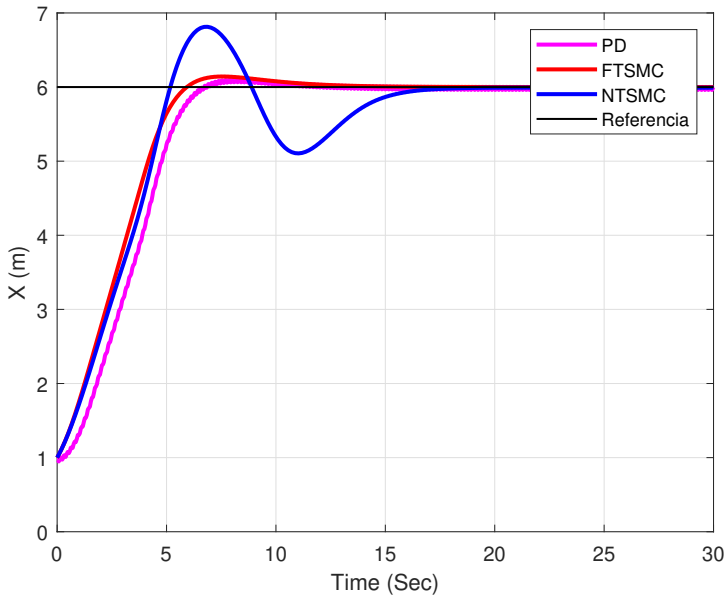

Figura 6.5: Respuesta de desplazamiento FTSMC comparada con el control NTSMC y el control PD en el eje x.

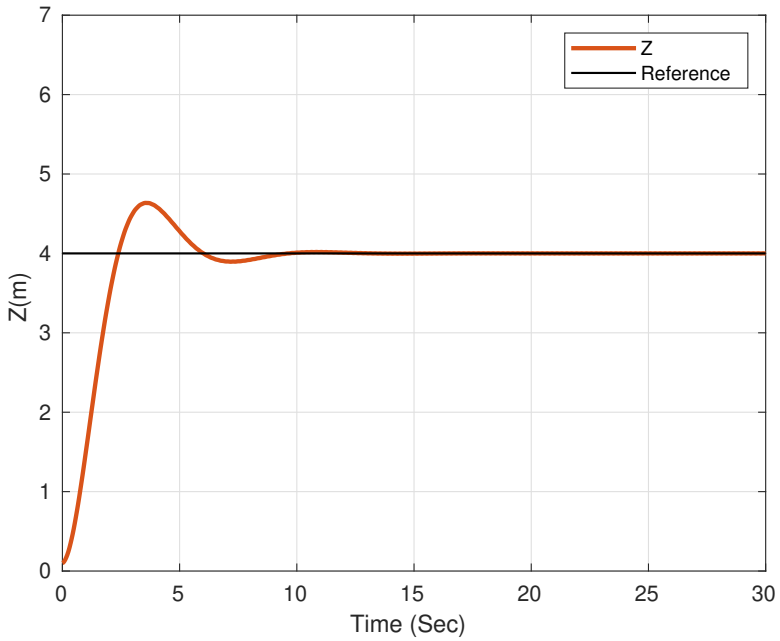

Figura 6.6: Respuesta de altitud z.

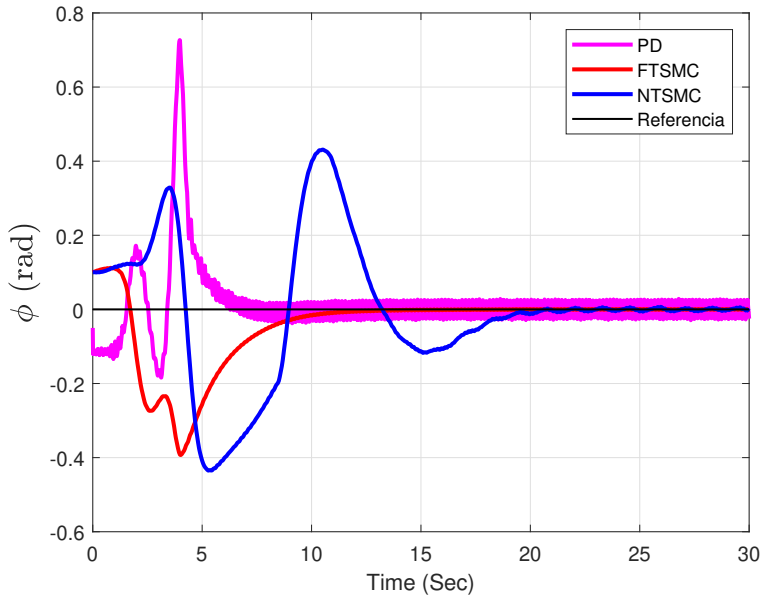

Figura 6.7: Comportamiento del ángulo de balanceo (ϕ) utilizando las dos estrategias de control propuestas (FTSMC vs NTSMC vs PD).

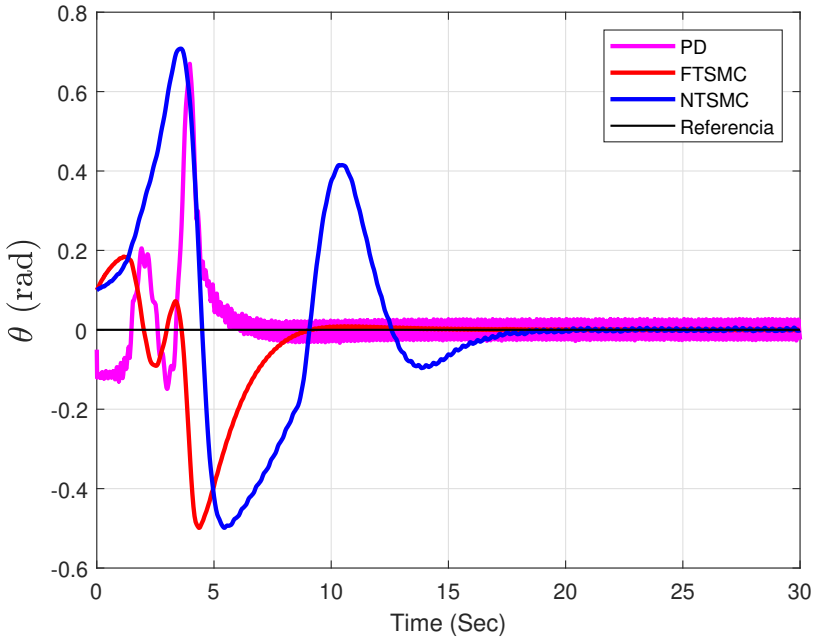

Figura 6.8: Comportamiento del ángulo de cabeceo (θ) utilizando las dos estrategias de control propuestas (FTSMC vs NTSMC vs PD).

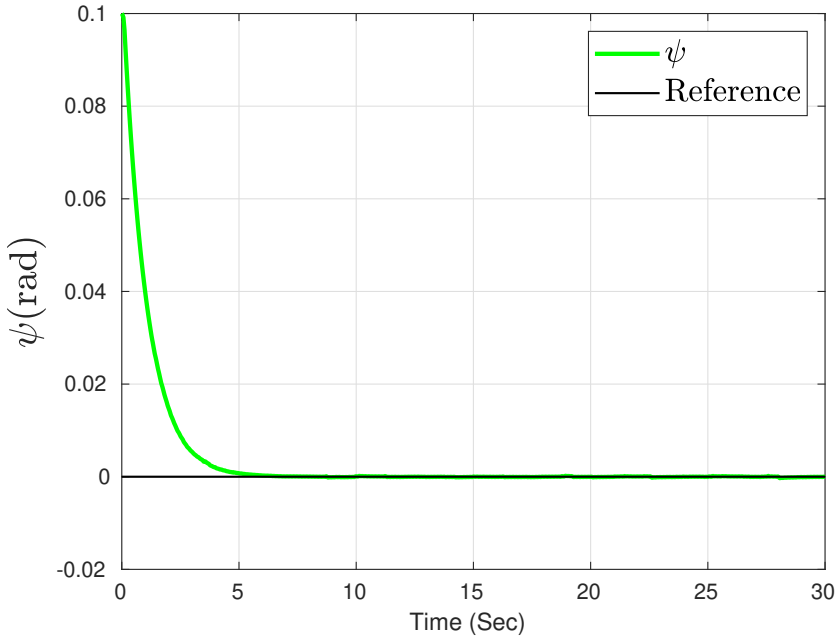

Figura 6.9: Comportamiento del ángulo ψ.

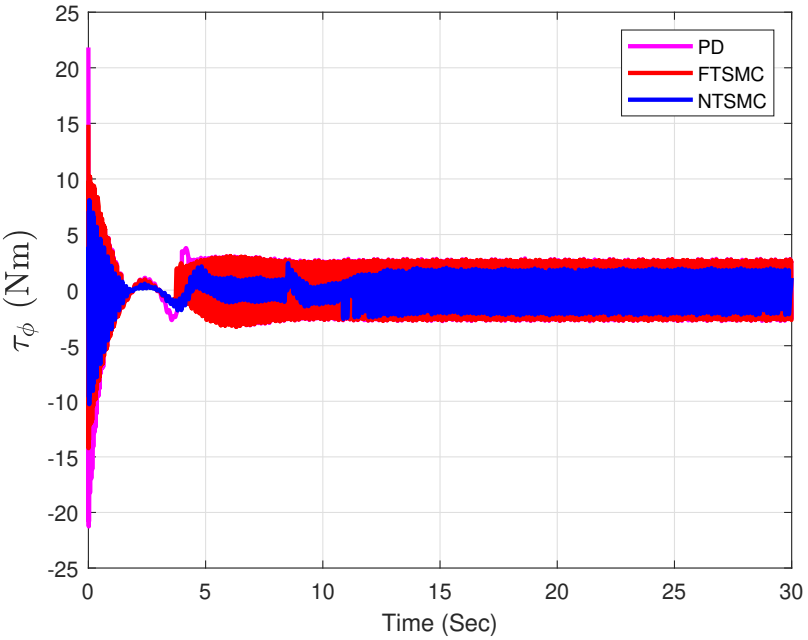

Figura 6.10: Comportamiento τ_ϕ utilizando las dos estrategias de control propuestas (FTSMC vs NTSMC vs PD).

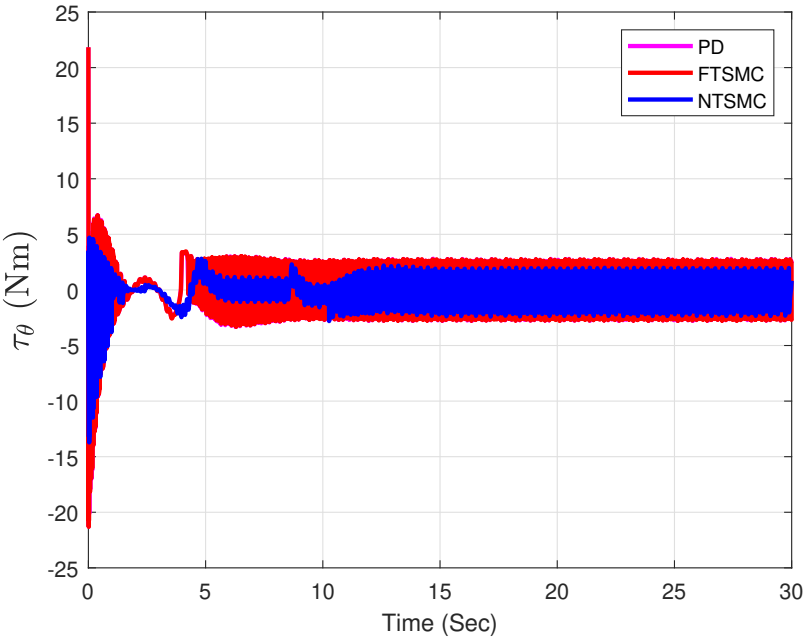

Figura 6.11: Comportamiento τ_θ utilizando las dos estrategias de algoritmo propuestas (FTSMC vs NTSMC vs PD).

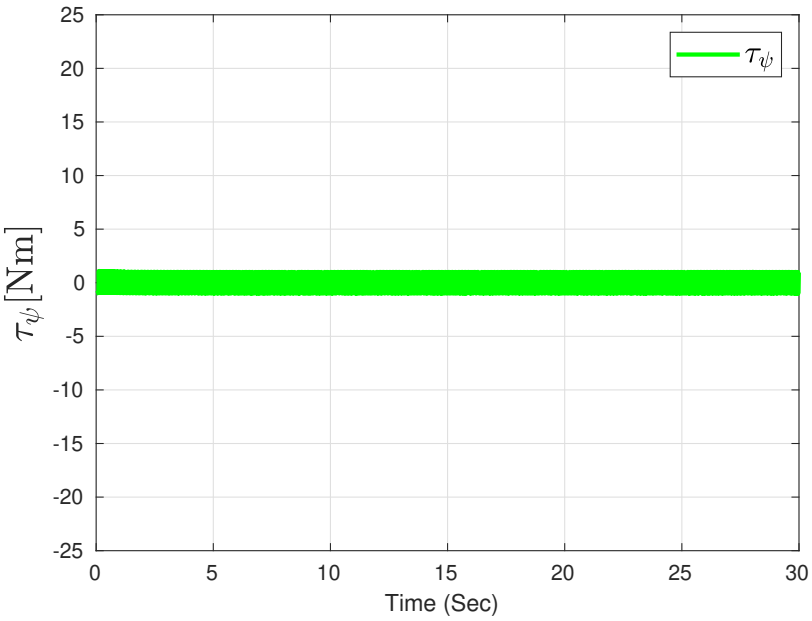

Figura 6.12: Respuesta de comportamiento τ_ψ.

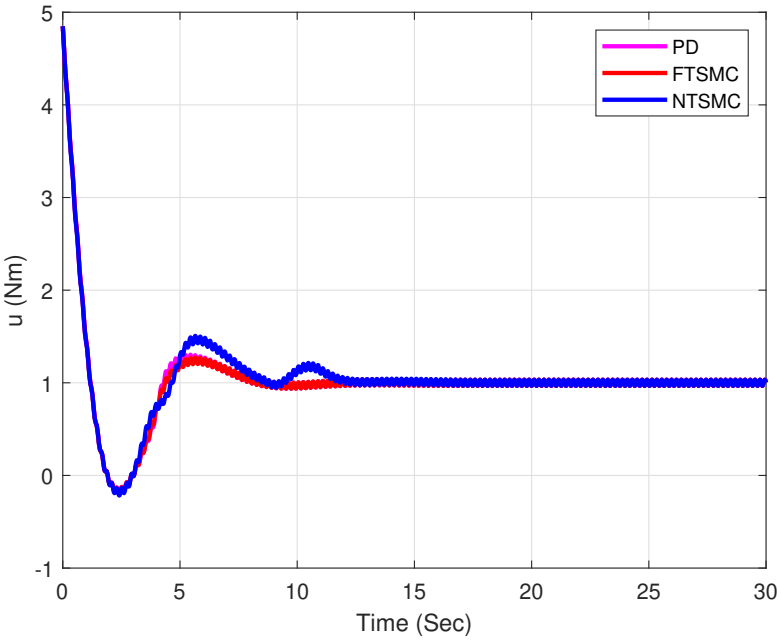

Figura 6.13: Respuesta del comportamiento de la entrada de control (u) para los controladores FTSMC, NTSMC y PD.

6.2. Simulación de la Autonomía del UAV

En esta sección se presenta un enfoque integral para la simulación de la autonomía de un UAV en la detección de edificios derrumbados utilizando el entorno de Gazebo [65] , esto combinara varias tecnologías avanzadas para permitir que un UAV pueda detectar de manera autónoma edificios que se han derrumbado. El proceso comienza con la simulación en Gazebo, donde el UAV puede practicar y perfeccionar sus habilidades de detección en escenarios virtuales antes de enfrentar situaciones reales.

Para la percepción visual, el UAV utiliza la cámara estereoscópica ZED [67] , lo que le permite evaluar la profundidad y reconocer con precisión los edificios derrumbados. La información recopilada se procesa y analiza utilizando el marco de trabajo ROS (Robot Operating System) [68] , que facilita la creación y ejecución de software de control de UAVs.

La potencia de procesamiento necesaria para esta tarea es proporcionada por Jetson Nano, una computadora de abordo especialmente diseñada para aplicaciones de inteligencia artificial en tiempo real. La comunicación con el UAV se establece a través de los protocolos MAVLink y MAVROS, lo que permite enviar instrucciones y recibir datos en tiempo real.

Sin embargo, el algoritmo de desarrollo es YOLO (You Only Look Once) [69] , una avanzada red neuronal de reconocimiento de imágenes que funciona como los “Ojos inteligentes” del UAV. YOLO permite identificar edificios derrumbados en imágenes en tiempo real con gran precisión.

6.2.1. Simulación en Gazebo

Es un software altamente ajustable y adaptable, permitiendo a los usuarios simular diversos elementos como edificaciones, individuos y objetos. Además, ofrece la posibilidad de simular herramientas de detección como sensores, cámaras y radares, impulsando la innovación y creación de nuevas capacidades. Las simulaciones son ejecutables en tiempo real, brindando a los programadores la oportunidad de observar la respuesta del código

en el momento. Este proyecto se centrará exclusivamente en la simulación de un UAV realista y así evaluar las distintas funciones integradas en el UAV.

Este conjunto de herramientas incluye, además, una variedad de nodos que, en combinación con una biblioteca de ROS, posibilitan la programación y el control del vuelo del UAV. A través de los nodos, se pueden personalizar y afinar una amplia gama de parámetros de vuelo, lo que facilita la realización de pruebas de navegación y la optimización del comportamiento del UAV en distintas situaciones y escenarios.

Durante la fase inicial del proyecto, se consideraron múltiples modelos de UAV proporcionados por ROS, seleccionando finalmente paquetes específicos que incluyen una representación genérica de un UAV en tres dimensiones con sus respectivas especificaciones técnicas.

Este modelo base llamado Iris Quadcopter se ha personalizado para integrar un sensor de visión de alta capacidad, diseñado para la identificación efectiva de elementos y barreras en el entorno, una herramienta esencial para operaciones de mapeo y detección. Además, está equipado con cuatro motores dinámicos que le confieren la agilidad necesaria para maniobrar en diferentes direcciones.

Ahora en la Figura 6.14 se visualiza un diagrama de arquitectura de sistema o de flujo de datos en ROS (Robot Operating System). Muestra la relación entre diferentes nodos y topics en la simulación que involucra Gazebo, MAVROS (el puente entre ROS y autopilotos MAVLink), y una cámara. El paquete seleccionado también engloba varios nodos y bibliotecas proporcionados por ROS, que facilitan la tarea de programar y dirigir el vuelo del UAV. Con estos recursos, es posible afinar los parámetros de vuelo y ejecutar la navegación de forma efectiva.

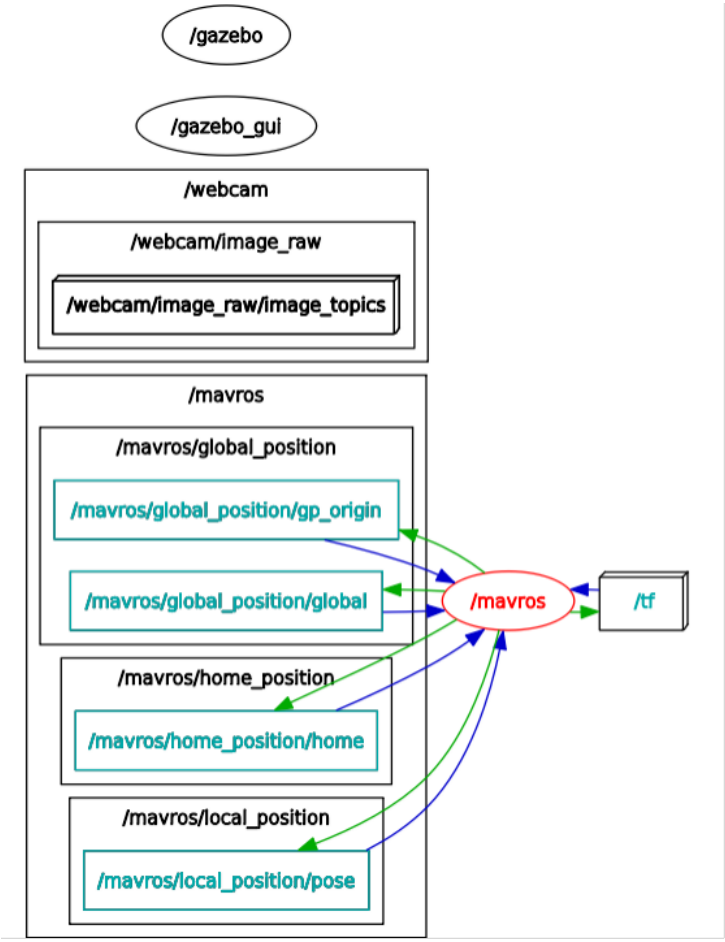

Figura 6.14: Diagrama de Nodos y Tópicos ROS

El diagrama proporcionado ilustra la arquitectura del sistema y el flujo de datos en un entorno ROS, incluyendo la interacción con el simulador Gazebo y MAVROS. A continuación, se describe cada uno de los componentes y su interconexión dentro del sistema:

- **/gazebo** y **/gazebo_gui** representan los nodos relacionados con el simulador Gazebo, manejando la lógica de simulación y la interfaz gráfica de usuario, respectivamente.
- **/webcam** es un nodo que gestiona la captura de datos de una cámara, real o simulada.
  - **/webcam/image_raw** publica imágenes crudas capturadas por la cámara.
  - **/webcam/image_raw/image_topics** puede ser un subgrupo de topics que publican diferentes aspectos de la imagen cruda.

- **/mavros** es el nodo principal para la comunicación entre ROS y el sistema de autopilotaje MAVLink.
  - **/mavros/global_position** incluye topics relacionados con la posición global del UAV.
    - **/mavros/global_position/gp_origin** podría indicar el origen geográfico para la misión actual.
    - **/mavros/global_position/global** transmite la posición global en términos de latitud, longitud y altitud.
  - **/mavros/local_position** proporciona datos de la posición local del dron.
    - **/mavros/local_position/pose** publica la pose del dron en un marco de referencia local.
  - **/mavros/home_position** gestiona la posición de inicio del dron.
    - **/mavros/home_position/home** publica la ubicación de inicio utilizada para el regreso al hogar.
- **/tf** es un paquete de ROS que mantiene el seguimiento de múltiples marcos de referencia, permitiendo la transformación entre marcos de posición global y local.

Un aspecto beneficioso de emplear un modelo genérico es que los principios y el código generado en la fase de simulación pueden ser aplicados a UAVs físicos actuales, permitiendo una transición fluida hacia aplicaciones prácticas. De este modo, se logra una implementación que mantiene cierto grado de independencia con respecto al hardware específico que podría adoptarse más adelante. La Figura 6.15 ofrece una visión del modelo de cuadricóptero en el ambiente de simulación Gazebo, que ha sido adaptado y optimizado para las exigencias específicas de esta investigación en UAVs autónomos y análisis de daños en estructuras.

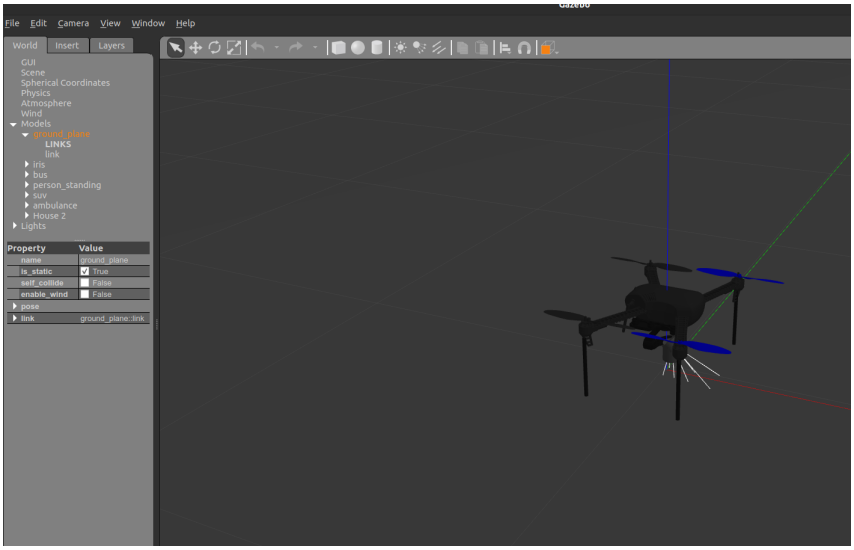

Figura 6.15: Cuadrirotor empleado para la simulación realizando pruebas de autonomía en Gazebo.

En la Figura 6.16 se presenta un entorno simulado donde se recrea un escenario urbano caracterizado por edificios dañados, escombros y vehículos diversos. El propósito principal de esta simulación es llevar a cabo una serie de pruebas de vuelo autónomas. Mediante la implementación de herramientas avanzadas como ROS (Robot Operating System), MAVROS, MAVLINK, Ardupilot y el sistema de detección de objetos YOLO (You Only Look Once), se busca mejorar los algoritmos de navegación y control del vehículo aéreo no tripulado.

La integración de estas tecnologías permitirá minimizar los posibles errores durante las pruebas de vuelo, aumentando así la precisión y seguridad en operaciones de vuelo autónomo. Esta meticulosa configuración de simulación es esencial para validar la interacción entre el UAV y su entorno antes de su aplicación en el mundo real.

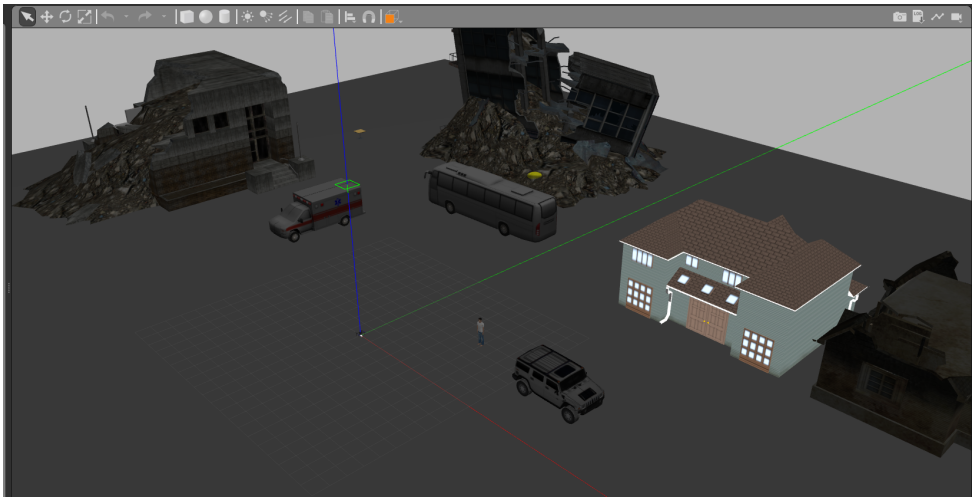

Figura 6.16: Panorámica del entorno.

La Figura 6.17 ilustra los resultados obtenidos de las pruebas realizadas en la simulación, destacando la capacidad del UAV para ejecutar un despegue (Takeoff) e implementar los modos de vuelo de ArduPilot sin inconvenientes. Esta prueba, en particular, se lleva a cabo de manera autónoma, con el propósito inicial de localizar a una persona.

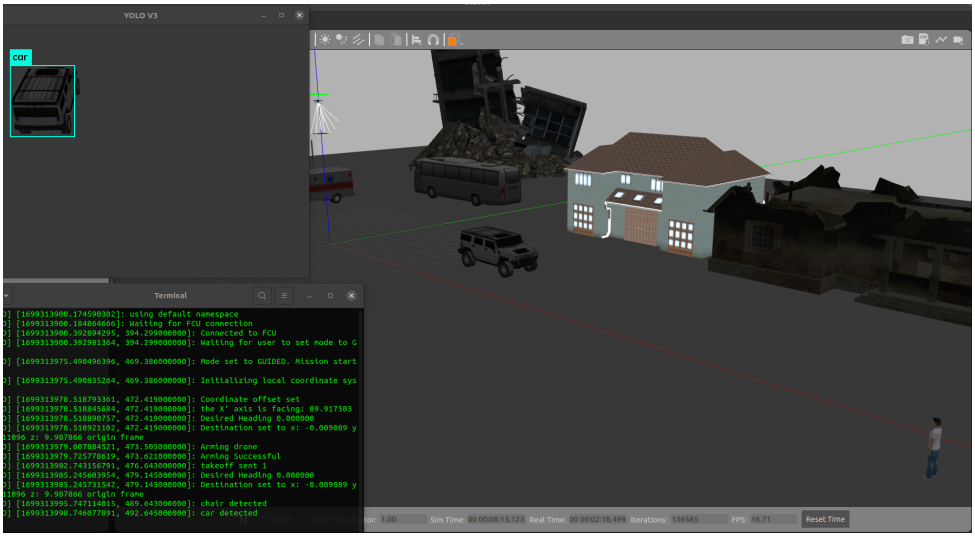

Figura 6.17: Vuelo autónomo.

6.2.2. Detección de la clase “Edificio”

Se visualiza la Figura 6.18 en la Figura proporcionada se observa la interfaz de una simulación de un UAV en Gazebo, en la parte superior izquierda donde se muestra la

percepción del sensor de visión, que a través del algoritmo YOLO, identifica y sigue al objetivo.

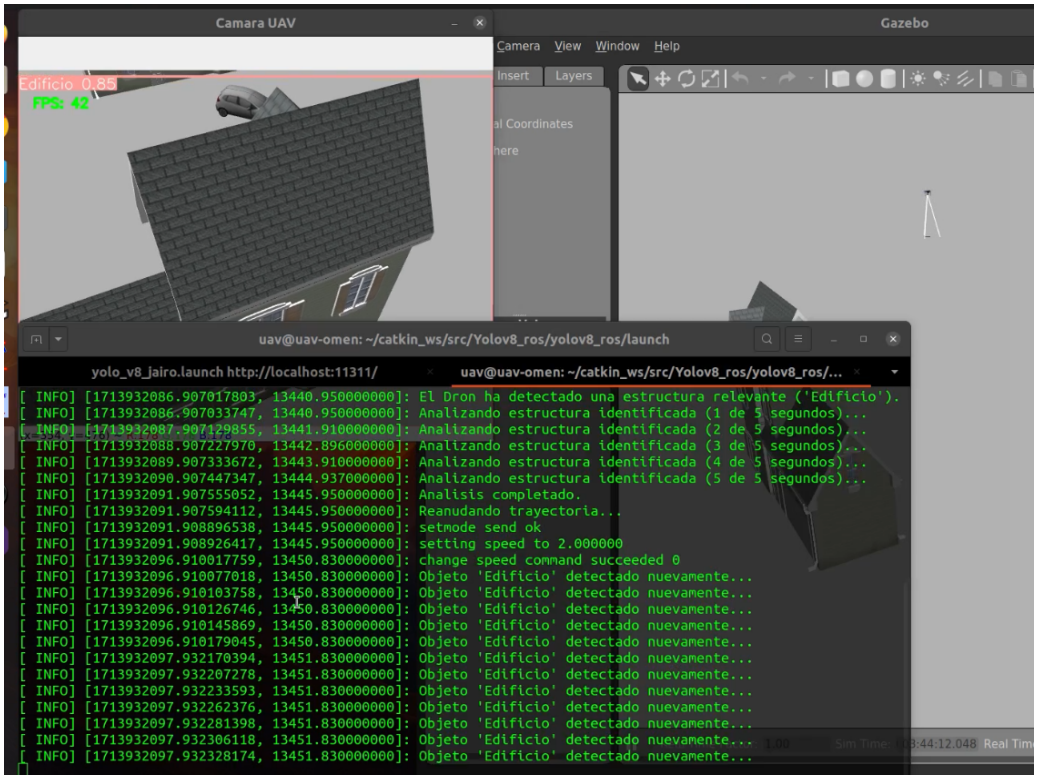

Figura 6.18: Detección del objetivo “Edificio”.

Se muestra un edificio identificado con una probabilidad del 0.85. La terminal ROS indica múltiples mensajes de detección y análisis de estructuras.

A continuación se describen los fragmentos clave del código que explican los resultados observados en la imagen.

Función detection_callback en la Clase “Edificio”

Esta función se ejecuta cada vez que se recibe un mensaje de detección de bounding boxes. Verifica si el objeto detectado es un edificio y maneja el modo del dron (cambio a "BRAKE", espera, análisis y reanudación) dependiendo de si es una nueva detección significativa.

```
1 void detection_callback(const yolov8_ros_msgs::BoundingBoxes::
```

```
    ConstPtr& msg) {geometry_msgs::Point current_position =
    get_current_location();
    // Suponiendo que esta es la ubicacion actual del UAV
    for (const auto& box : msg->bounding_boxes) {
        if (box.Class == "Edificio") {
            float distance_from_last = hypot(current_position.x -
            last_detection_position.x, current_position.y -
            last_detection_position.y);
            if (first_detection || distance_from_last >
            DETECTION_DISTANCE
            _THRESHOLD) {
                first_detection = false;
                last_detection_position = current_position;
                // Actualizar la  ltima  posici n de detecci n
                ROS_INFO("Edificio detectado", distance_from_last);
                if (set_mode("BRAKE") == 0) {
                    ROS_INFO("El Dron ha detectado una estructura.");
                    sleep(5); // Detener  5 segundos en modo BRAKE
                    ROS_INFO("Analizando estructura identificada ");
                    sleep(5); // Esperar 5 segundos
                    ROS_INFO("Analisis completado.");
                    ROS_INFO("Reanudando trayectoria programada");
                    set_mode("GUIDED");
                    set_speed(4.0);
                    // Restablecer velocidad a 2.0 m/s
                } else {
                    ROS_ERROR("Fallo al cambiar a modo BRAKE.");
                }
            } else {
                ROS_INFO("Edificio detectado nuevamente...",

                distance_from_last);
            }
            break;
        }
    }
}
```

```
int main(int argc, char** argv)
{
    ros::init(argc, argv, "gnc_node");
    ros::NodeHandle gnc_node("~");
    ros::Subscriber sub = gnc_node.subscribe
    ("/yolov8/BoundingBoxes",
    10, detection_callback);
    init_publisher_subscriber(gnc_node);
    wait4connect();
    set_mode("GUIDED");
    arm();
    ROS_INFO("Dron armado y en modo Autonomo.");
    float ascent_speed = 0.5; // m/s, ajustar segun sea necesario
    set_speed(ascent_speed);
    float alt_lat = 19.51;
    float alt_lon = -99.12;
    float alt_alt = 10;
    takeoff_global(alt_lat, alt_lon, alt_alt);
    ROS_INFO("Despegando...");
    sleep(10); // Esperar 10 segundos despues del despegue
    float target_lat = 19.512202;
    float target_lon = -99.129115;
    float target_alt = 20;
    set_speed(5.0); // Velocidad para dirigirse al primer waypoint
    set_destination_lla_raw(target_lat, target_lon, target_alt, 0);
    ROS_INFO("Detectando estructuras...");
    sleep(5);
    ros::Rate rate(1.0);
    while (ros::ok()) {
        ros::spinOnce(); // Procesa callbacks
        set_destination_lla_raw(second_target_lat, second_target_lon,
        second_alt, 0);
        ROS_INFO("Detectando estructuras...");
        rate.sleep();
    }
    return 0;
}
```

La terminal muestra múltiples mensajes que indican el progreso de la detección y análisis de estructuras:

```
:El Dron ha detectado una estructura relevante ('Edificio').
:Analizando estructura identificada (1 de 5 segundos)...
:Analizando estructura identificada (2 de 5 segundos)...
:Analizando estructura identificada (3 de 5 segundos)...
:Analizando estructura identificada (4 de 5 segundos)...
:Analizando estructura identificada (5 de 5 segundos)...
:Analisis completado.
:Reanudando trayectoria programada...
:Objeto 'Edificio' detectado nuevamente...
```

6.2.3. Detección de la clase “Derrumbado”

Estos mensajes confirman las acciones del código, mostrando la detección inicial del edificio, el análisis de la estructura y la detección continua del mismo edificio.

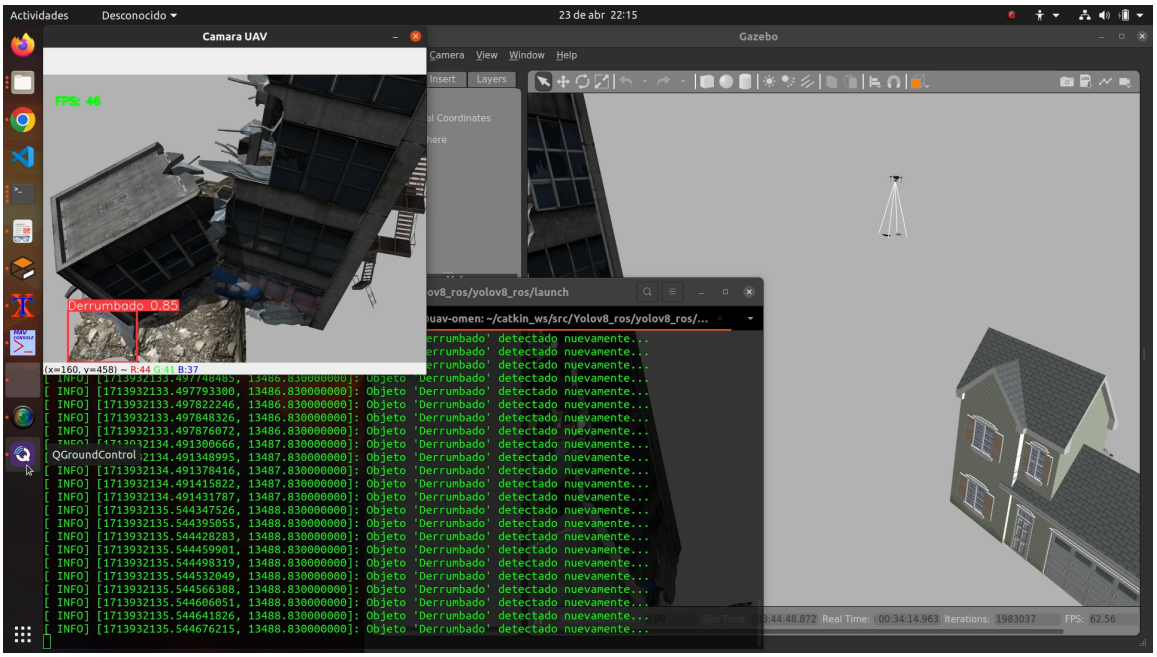

Figura 6.19: Detección del objetivo “Derrumbado”.

Posteriormente el UAV realiza la inspección para la detección de la siguiente clase a detectar "Derrumbado” como se muestra en la Figura 6.19, en la ventana de la cámara del

UAV se muestra un edificio derrumbado identificado con una probabilidad del 0.85. La terminal ROS indica múltiples mensajes de detección y análisis de esta nueva estructura.

Función detection_callback en la Clase “Derrumbado”

Esta función se ejecuta cada vez que se recibe un mensaje de detección de bounding boxes. Verifica si el objeto detectado es un edificio derrumbado y maneja el modo del dron (cambio a "BRAKE", espera, análisis y reanudación) dependiendo de si es una nueva detección significativa.

```
void detection_callback(const yolov8_ros_msgs::BoundingBoxes::
ConstPtr& msg) {geometry_msgs::Point current_position =
    get_current_location();
    // Obtiene la ubicacion actual del UAV
    for (const auto& box : msg->bounding_boxes) {
        if (box.Class == "Derrumbado") {
            float distance_from_last = hypot(current_position.x -
            last_detection_position.x,
            current_position.y - last_detection_position.y);
            if (first_detection || distance_from_last >
            DETECTION_DISTANCE_THRESHOLD) {
                first_detection = false;
                last_detection_position = current_position;
                // Actualizar la ultima posici n de detecci n
                ROS_INFO("Derrumbado detectado", distance_from_last);
                if (set_mode("BRAKE") == 0) {
                    ROS_INFO("El UAV ha detectado una estructura
                    derrumbada.");
                    sleep(5); // Detener 5 segundos en modo BRAKE
                    ROS_INFO("Analizando estructura identificada ");
                    sleep(5); // Esperar 5 segundos
                    ROS_INFO("Analisis completado.");
                    ROS_INFO("Reanudando trayectoria programada...");
                    set_mode("GUIDED");
                    set_speed(4.0); // Restableciendo velocidad a 4m/s
```

```
26                  } else {
27                      ROS_ERROR("Fallo al cambiar a modo BRAKE.");
28                  }
29              } else {
30                  ROS_INFO("Derrumbado detectado nuevamente...",
31                  distance_from_last);
32              }
33              break;
34          }
35      }
36  }
```

Posteriormente se inicializa el nodo ROS, se establece un subscriber para el tema `/yolov8/BoundingBoxes`, y se dirige el UAV a su destino inicial donde se produce la detección del edificio derrumbado.

La terminal muestra múltiples mensajes que indican el progreso de la detección y análisis de estructuras:

```
1  : Objeto 'Derrumbado' detectado nuevamente...
2  : Objeto 'Derrumbado' detectado nuevamente...
3  : Objeto 'Derrumbado' detectado nuevamente...
4  : Objeto 'Derrumbado' detectado nuevamente...
5  : Objeto 'Derrumbado' detectado nuevamente...
```

Estos mensajes confirman las acciones del código, mostrando la detección inicial del edificio derrumbado, el análisis de la estructura y la detección continua.

6.2.4. Detección de la clase “Víctima”

La Figura 6.20 se muestra en la parte superior izquierda las detecciones realizadas por el modelo. En la interfaz de la cámara del UAV, se pueden observar varias detecciones de víctimas: la primera víctima está etiquetada con un cuadro delimitador naranja y

una confianza de 0.69, y la segunda víctima está etiquetada con un cuadro delimitador naranja y una confianza de 0.65. Estas detecciones indican que el algoritmo ha identificado correctamente las víctimas en el escenario de desastre. Los cuadros delimitadores y los niveles de confianza proporcionan una evaluación visual de la precisión del modelo en la detección de víctimas.

El objetivo de esta simulación es evaluar el rendimiento del modelo de detección de objetos en un entorno realista y dinámico. La utilización de un UAV para capturar imágenes desde diferentes ángulos y alturas permite una evaluación exhaustiva de las capacidades del algoritmo. La simulación muestra es capaz de detectar víctimas con un nivel de confianza razonable en un escenario complejo. Las detecciones visualizadas en la interfaz de la cámara del UAV confirman la capacidad del algoritmo para identificar correctamente las víctimas, aunque la precisión de las detecciones puede variar dependiendo de la posición y el ángulo de la cámara. En general, los resultados de la simulación proporcionan una visión clara del rendimiento del algoritmo propuesto en la tarea de detección de víctimas en un entorno de desastre.

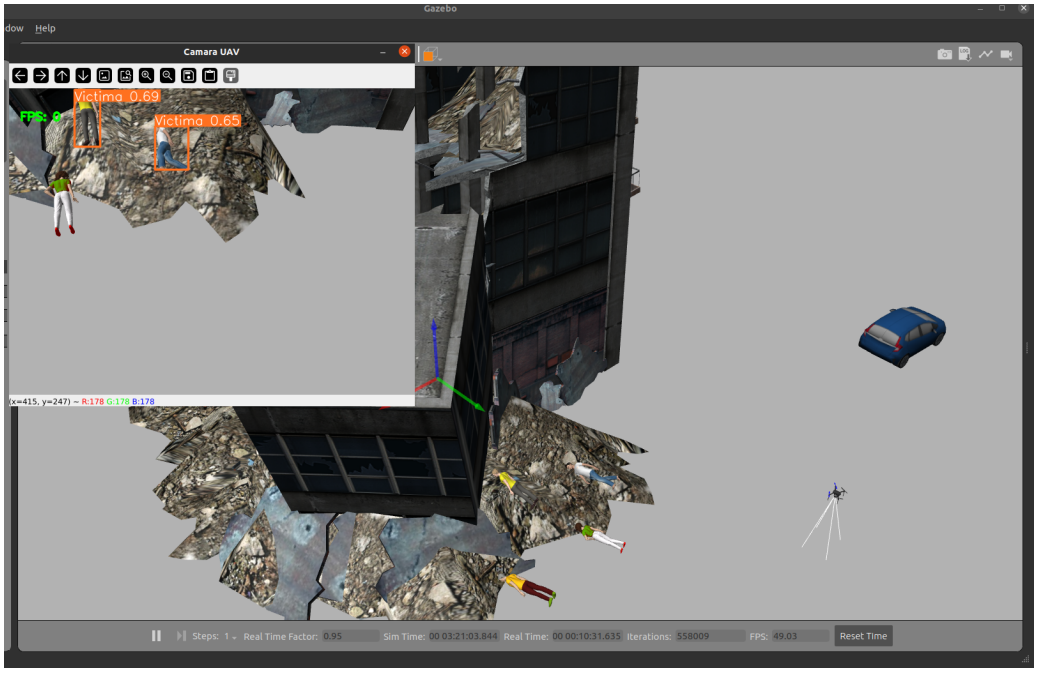

Figura 6.20: Detección del objetivo “Victima”.

Continuando se muestra un fragmento del código utilizado en la simulación para manejar las detecciones de estructuras colapsadas y edificios, así como para identificar víctimas:

```
for (const auto& box : msg->bounding_boxes) {
    if (box.Class == "Edificio" || box.Class == "Derrumbado") {
        float distance_from_last = hypot(current_position.x
                                         - last_detection_position.x,
                                         current_position.y
                                         - last_detection_position.y);
        if (first_detection || distance_from_last >
            DETECTION_DISTANCE_THRESHOLD) {
            first_detection = false;
            last_detection_position = current_position;
            // Actualizar la ultima posicion de deteccion
            ROS_INFO("DETECTADO %s A %f METROS", box.Class.c_str(),
            distance_from_last);
            if (set_mode("BRAKE") == 0) {
                if (box.Class == "Derrumbado") {
                    ROS_INFO("ESTRUCTURA DERRUMBADA,
                    IDENTIFICADO VICTIMAS");
                    set_destination_lla_raw(current_position.x,
                    current_position.y, 5, 0); // Bajar a 5 metros
                    sleep(5); // Esperar 5 segundos
                    ROS_INFO("DETECTANDO VICTIMAS");
                    sleep(5); // Esperar otros 5 segundos
                } else {
                    ROS_INFO("EL DRON HA DETECTADO UNA ESTRUCTURA.");
                    sleep(5); // Detener 5 segundos en modo BRAKE
                    ROS_INFO("ANALIZANDO ESTRUCTURA IDENTIFICADA...");
                    sleep(5); // Esperar 5 segundos
                    ROS_INFO("ANALISIS COMPLETADO.");
                }
            }
        }
    }
}
```

La Figura 6.21 muestra la interfaz de Gazebo y QGroundControl, que son utilizados para planificar y monitorear la trayectoria del UAV durante las prácticas físicas. Se han tomado en cuenta las coordenadas de una zona específica donde se encuentran varias construcciones. Estas coordenadas se utilizan para acercar las prácticas lo más posible a resultados reales. En la interfaz se observa una vista satelital de la zona de pruebas, con una trayectoria planificada indicada por una flecha roja. La implementación de estas coordenadas permite una evaluación más precisa y contextualizada del rendimiento del algoritmo en condiciones reales.

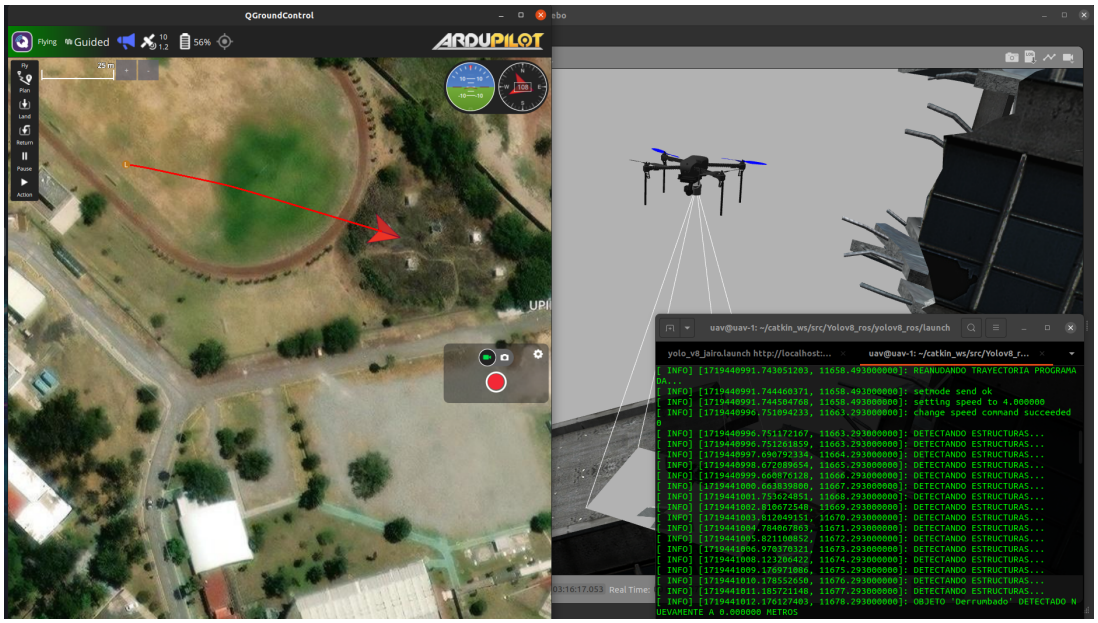

Figura 6.21: Localización en GPS de la Simulación.

Video de Simulación

El siguiente video https://youtu.be/AEYHW3TXjNM, documenta una simulación en la que el UAV equipado con el sistema de visión realiza una inspección de edificios en un entorno controlado. La simulación fue ejecutada en el entorno Gazebo, con una interfaz de consola que muestra la interacción y control del UAV a través de comandos ROS (Robot Operating System). A continuación se detallan las fases y observaciones clave:

- **Inicio y Preparación del UAV**: En las primeras imágenes, se observa la conexión y preparación del UAV en el entorno de simulación, mostrando mensajes de estado en la consola. Estos mensajes indican que el dron se arma y cambia a modo autónomo. También se configuran las coordenadas de destino y la velocidad de ascenso.
- **Despegue y Vuelo**: El UAV despega y comienza su misión de inspección. La consola registra el cambio de velocidad y ajustes adicionales conforme el dron se desplaza hacia el área objetivo para la inspección.
- **Detección de Edificios Intactos**: Durante el vuelo, el sistema de visión detecta y clasifica estructuras intactas. Las detecciones son marcadas en la vista de la cámara del UAV con la etiqueta .Edificioz una puntuación de confianza. Cada detección se acompaña de mensajes en la consola indicando que se ha identificado un objeto tipo .Edificio".
- **Detección de Edificios Derrumbados**: Al llegar a áreas donde se encuentran estructuras afectadas, el sistema detecta edificios derrumbados, etiquetándolos como "Derrumbado". Esta detección se refleja en la consola y en la vista de la cámara del UAV, con una caja de delimitación roja y una puntuación de confianza.
- **Interfaz de Control y Monitoreo**: La última captura muestra la interfaz de QGroundControl, donde se monitorea el progreso y trayectoria del UAV sobre un mapa en tiempo real. La vista muestra el movimiento y posición del UAV en el entorno simulado.

Al finalizar la inspección, el sistema puede generar un reporte con el conteo de edificios intactos, derrumbados, y potenciales víctimas detectadas en la simulación. Este video

ilustra de forma clara cómo el UAV utiliza algoritmos de visión para clasificar y cuantificar daños en estructuras en un entorno simulado, proporcionando una solución eficaz para la evaluación de daños en situaciones de desastre.

7

Plataforma de Desarrollo: Resultados experimentales

El desarrollo del UAV fue una consecuencia directa de mejoras iterativas, centradas especialmente en optimizar el manejo del peso y la integración de los dispositivos electrónicos. A medida que el proyecto progresaba, se realizaron ajustes cruciales que resultaron en un rendimiento superior del UAV. Estos refinamientos incluyeron la redistribución cuidadosa de la masa para lograr un equilibrio óptimo, así como la reorganización de la instrumentación electrónica para maximizar la eficiencia operativa y la confiabilidad en vuelo. Cada cambio estuvo guiado por el compromiso de perfeccionar tanto la funcionalidad como la eficacia aerodinámica del cuadricóptero, asegurando que cada componente electrónico no sólo estuviera correctamente alojado, sino que también contribuyera a la estabilidad general y al manejo del UAV.

7.1. Desarrollo e Instrumentación del UAV

A lo largo de la fase de desarrollo del UAV, se llevaron a cabo modificaciones significativas en el diseño para integrar un sistema avanzado de visión artificial. Este desarrollo incluyó la incorporación de la cámara estereoscópica ZED Mini, la cual se colocó estratégicamente para optimizar la detección y el procesamiento de imágenes en tiempo real.

Integración de la ZED Mini

La ZED Mini se instaló en la parte inferior de la estructura del UAV. Esta posición fue seleccionada deliberadamente para proporcionar un campo de visión despejado y adecuado, vital para las operaciones de mapeo y navegación autónoma. El diseño tenía que garantizar que la adición de este dispositivo no afectara negativamente el centro de gravedad del dron ni comprometiera su estabilidad en vuelo.

Posicionamiento del Controlador de Vuelo Pixhawk

El controlador de vuelo Pixhawk, siendo un componente crítico para la estabilización y el control del UAV, se montó en el centro de inercia de la aeronave. Este posicionamiento central es crucial para el rendimiento óptimo de los algoritmos de control y para mantener una respuesta equilibrada a las entradas de control en todas las condiciones de vuelo.

Alojamiento del Sistema Embebido Jetson Nano

La integración del sistema embebido Jetson Nano planteó desafíos adicionales. Este poderoso procesador de datos se ubicó en la parte central del UAV para facilitar la gestión de cables y el acceso a otros componentes electrónicos. La centralización del Jetson Nano es fundamental para minimizar la latencia en la comunicación con otros elementos del sistema, como sensores y actuadores.

Rediseño para Protección y Calibración

Para abordar la complejidad de proteger los dispositivos electrónicos vitales y ofrecer un acceso conveniente para la calibración y el mantenimiento, se diseñó un componente protector especial. Este mecanismo no solo salvaguarda el hardware de daños potenciales sino que también proporciona un método práctico para asegurar el UAV durante las pruebas de calibración y ajustes previos al vuelo.

Aumento de Carga Útil

Gracias a la potencia de empuje de los motores, fue posible añadir estos dispositivos sin impactar de manera significativa la capacidad de carga útil del dron. A pesar de la preocupación inicial sobre el peso adicional, el diseño final demostró que la plataforma podía manejar el equipo de visión artificial y el sistema de procesamiento de datos sin comprometer su funcionalidad o tiempo de vuelo.

Este rediseño meticuloso asegura que el UAV no solo está equipado con la tecnología de vanguardia necesaria para la visión y procesamiento avanzados sino que también retiene la agilidad y la estabilidad necesarias para las operaciones de campo. Con estos avances, el UAV está bien preparado para llevar a cabo tareas complejas en entornos dinámicos y desafiantes, abriendo nuevas posibilidades en el ámbito de la navegación autónoma y la detección de daños en tiempo real, en la Figura 7.1se muestra la explosión del ensamble en SolidWorks.

La versión más reciente del UAV presenta unas dimensiones compactas de 50 cm x 50 cm, medida que no incluye las hélices. La altura del UAV, medida desde el suelo hasta la cámara, alcanza los 17.5 cm, proporcionando una estabilidad y visibilidad óptima. Tras la incorporación del sistema de visión, el peso total del UAV instrumentado se sitúa en 2.7 kg, un equilibrio adecuado que permite la maniobrabilidad sin comprometer la capacidad de carga.

Figura 7.1: Desarrollo y ensamble en CAD del UAV

La Figura 7.2 que se adjunta ilustra el ensamblaje de todos los componentes previamente descritos. Actualmente, el proyecto está en una fase de pruebas físicas en las que se están llevando a cabo experimentos cruciales. Estos están enfocados en evaluar la capacidad del UAV para navegar de manera autónoma siguiendo trayectorias determinadas a través de la detección realizada por nuestro algoritmo.

Figura 7.2: Construcción e Instrumentación del UAV

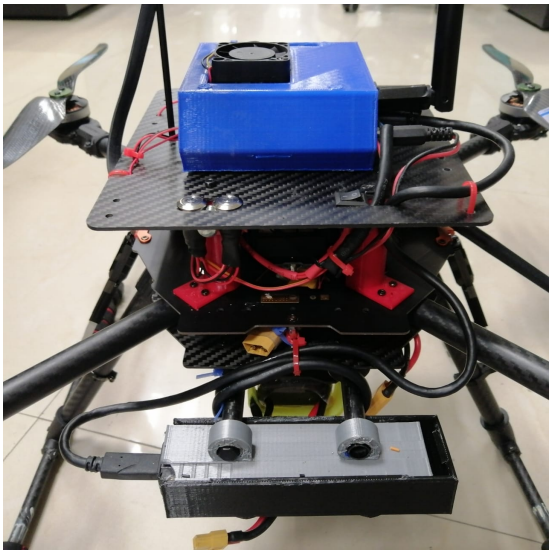

Figura 7.3: Estructura interna del UAV

7.2. Ejecución del Algoritmo de Visión

El diagrama de flujo presentado describe el proceso completo desde la obtención del archivo .py hasta la implementación y validación del algoritmo en un sistema embebido Jetson Nano. A continuación, se detalla cada etapa clave representada en el diagrama de la Figura 7.4.

a) **Inicio**: El proceso comienza en Inicio, el cual marca el punto de partida de la configuración y preparación para ejecutar el algoritmo de detección en el sistema.

b) **Obtención del archivo .py**: Este paso implica la obtención del archivo de código fuente en formato `.py`, que contiene el algoritmo de detección. Este archivo es crucial ya que contiene las instrucciones que el sistema utilizará para detectar y clasificar objetos en el entorno.

c) **Configuración de la simulación**: En este paso, se realiza la configuración del entorno de simulación, donde se ajustan parámetros relevantes, tales como el entorno virtual, los modelos a analizar, y las configuraciones necesarias para ejecutar el algoritmo en condiciones simuladas. Este entorno permite validar el comportamiento del sistema antes de implementarlo en un entorno físico.

d) **Ejecución de simulación del Algoritmo**: Una vez configurada la simulación, se procede a ejecutar el algoritmo en el entorno simulado. Esto permite observar cómo responde el sistema y verificar que el algoritmo funciona correctamente en condiciones controladas.

e) **Captura de datos y análisis**: Durante la simulación, se recopilan datos de la ejecución del algoritmo. Estos datos incluyen información sobre los objetos detectados y su clasificación. En este punto, se analizan los resultados obtenidos para determinar la precisión y efectividad del algoritmo.

f) **¿Objetos detectados?**: Este es un punto de decisión que evalúa si el algoritmo ha detectado algún objeto en el entorno. Si no se detectan objetos, el proceso vuelve al paso de ejecución de simulación para realizar ajustes o reintentar la detección. Si se detectan objetos, el proceso continúa hacia la siguiente fase.

g) **Transferencia del modelo a Jetson Nano**: Cuando se confirma que el algoritmo detecta objetos de manera efectiva, el modelo se transfiere al dispositivo Jetson Nano, que es un procesador especializado para aplicaciones de inteligencia artificial en entornos físicos. Este paso marca la transición de un entorno simulado a un entorno de prueba real.

h) **Instalación de librerías necesarias**: Una vez que el modelo está en el Jetson Nano, se instalan todas las librerías necesarias para ejecutar el algoritmo. Estas librerías pueden incluir herramientas de procesamiento de imágenes, paquetes de aprendizaje profundo, y otros recursos necesarios para la operación del modelo.

i) **Configuración de la Cámara Zed Mini**: En este paso, se configura la cámara Zed Mini, que será utilizada por el sistema para capturar imágenes del entorno en tiempo real.

j) **Implementación del algoritmo**: Después de la configuración de los componentes, el algoritmo se implementa en el sistema operativo del UAV y comienza a ejecutarse en el Jetson Nano. En este punto, el algoritmo está listo para procesar las imágenes capturadas y realizar detecciones en tiempo real.

k) **Pruebas y validación en el entorno real**: El último paso es realizar pruebas en un entorno real. En esta etapa, el UAV opera con el algoritmo de detección en condi-

ciones de campo, permitiendo observar cómo responde en situaciones reales. Los resultados de esta fase determinan si el sistema es efectivo y está listo para su despliegue final.

l) **Fin**: Una vez completadas las pruebas y validaciones, el proceso llega a su fin. En este punto, el sistema está listo para ser utilizado en aplicaciones de inspección y detección de daños estructurales en entornos reales.

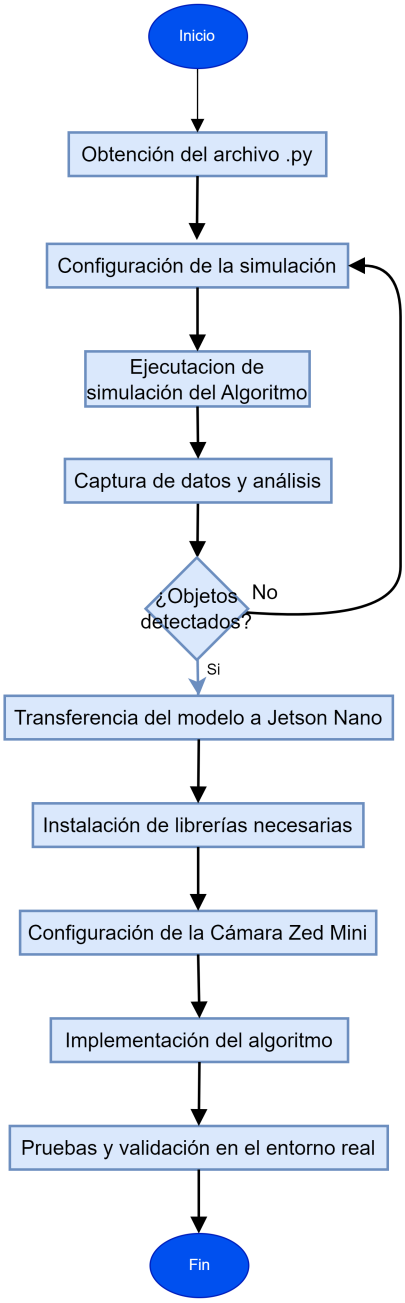

Figura 7.4: Sistema de Visión

7.2.1. Transferencia del Algoritmo al UAV

En esta sección se presenta el sistema de visión Figura 7.5 que se aplicará para una combinación de la Jetson Nano, una computadora de placa única que proporciona capacidad de procesamiento en tiempo real y alta eficiencia, y la Pixhawk, una controladora de vuelo de código abierto para drones que permite un control de vuelo preciso y flexible. Esta combinación proporciona al sistema UAV la capacidad de procesar y analizar datos visuales en tiempo real mientras mantiene el vuelo del UAV.

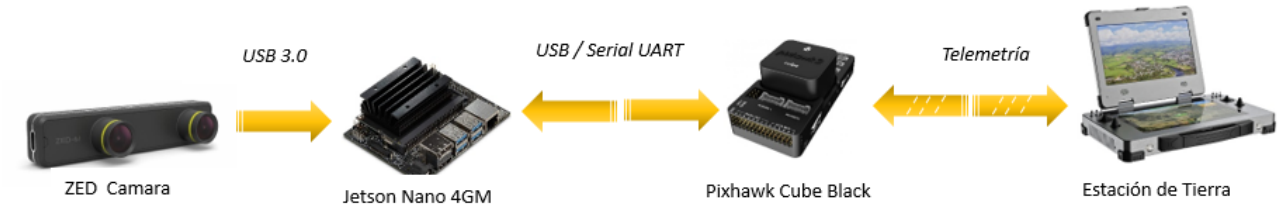

Figura 7.5: Sistema de Visión

Al comunicar una Jetson Nano a una Pixhawk, se pueden obtener varias ventajas, como por ejemplo:

- Procesamiento de alta velocidad: la Jetson Nano es capaz de procesar grandes cantidades de datos en tiempo real, lo que permite una mayor velocidad de procesamiento y una mejor respuesta a situaciones críticas.
- Análisis de datos en tiempo real: al conectar la Jetson Nano a la Pixhawk, es posible procesar y analizar los datos de diferentes sensores en tiempo real, lo que permite tomar decisiones más precisas y adaptativas en situaciones cambiantes.
- Mejora en la precisión del control: al utilizar la Jetson Nano para el procesamiento de datos y análisis, se puede mejorar la precisión del control de la Pixhawk, lo que se traduce en una mayor estabilidad y precisión en el movimiento del vehículo.

- Integración de visión computacional: la Jetson Nano es capaz de ejecutar algoritmos de visión artificial, lo que puede permitir a la Pixhawk reconocer objetos, realizar seguimiento de objetivos o incluso realizar tareas autónomas.

En resumen, al conectar una Jetson Nano a una Pixhawk, se puede lograr un procesamiento de datos más rápido y preciso, lo que puede ser útil para la aplicación que se requiere de un control más preciso y análisis en tiempo real.

En el contexto de la visión computacional, la integración del Pixhawk abre nuevas posibilidades en términos de navegación autónoma, detección de objetos, seguimiento visual y mapeo en tiempo real. La capacidad de procesar imágenes a bordo de la aeronave y tomar decisiones en tiempo real permite la ejecución de tareas más complejas y sofisticadas, mejorando la autonomía y eficiencia de los sistemas robóticos.

Al utilizar el Pixhawk en combinación con algoritmos de visión computacional, se pueden desarrollar aplicaciones como la detección y seguimiento de objetos en movimiento, la inspección visual de infraestructuras, el mapeo tridimensional de entornos y la detección de obstáculos. [52]

Es importante describir que el dispositivo del Pixhawk muestran los conceptos fundamentales del control de vuelo y la navegación autónoma, así como los aspectos clave de la programación y configuración del Pixhawk. También se abordarán las técnicas y algoritmos utilizados para el procesamiento de imágenes y la extracción de información relevante.

La Pixhawk es un controlador de vuelo de código abierto utilizado en muchos drones y sistemas de vehículos aéreos no tripulados (UAV) para controlar la estabilidad, la navegación y la orientación del vehículo, la combinación del Pixhawk y la visión computacional representa un paso importante en la evolución de los sistemas robóticos. Permite la ejecución de tareas más complejas y sofisticadas, mejorando la autonomía y eficiencia de los VANTs y aviones autopilotados. [53]

La Jetson Nano [54] ofrece una capacidad de procesamiento de alto rendimiento con un consumo de energía eficiente, lo que la convierte en una opción ideal para aplicaciones de visión computacional en tiempo real.

La integración de la Jetson Nano con la plataforma de control de vuelo Pixhawk amplía aún más las posibilidades de aplicaciones de visión computacional en sistemas robóticos aéreos [55] [56]. El Pixhawk, conocido por su confiabilidad y flexibilidad, se encarga del control y estabilización del UAV, mientras que la Jetson Nano se ocupará del procesamiento avanzado de imágenes y la ejecución de algoritmos de visión computacional en tiempo real.

La combinación de la Jetson Nano y el Pixhawk permite desarrollar aplicaciones sofisticadas, como la detección y seguimiento de objetos, el mapeo y la navegación autónoma en tiempo real. Esta integración es especialmente relevante en escenarios que requieren una toma de decisiones rápida y precisa, como la inspección visual de infraestructuras, la búsqueda y rescate, la monitorización medioambiental y la seguridad [58] [59].

Además como se ha mencionado la obtención de imágenes se implementó con un sensor de visión la cámara ZED Mini [60] ésta es una herramienta esencial en este proyecto de visión computacional, que se enfoca en la detección de edificios derrumbados desde una altura específica. La ZED Mini, conocida por su capacidad de captura estereoscópica de alta resolución, juega un papel fundamental en la recopilación de datos tridimensionales precisos y en la creación de un entorno de visión en tiempo real que permite identificar estructuras colapsadas de manera eficiente y confiable.

En este contexto, la funcionalidad de la cámara ZED Mini es multidimensional. En primer lugar, gracias a su capacidad de captura estereoscópica, la cámara es capaz de generar imágenes y mapas de profundidad tridimensionales, lo que permite a los algoritmos de visión computacional calcular la distancia y la posición de los objetos con una gran precisión. Esta característica es esencial para determinar si un edificio ha colapsado y evaluar la magnitud del daño.

Además, la ZED Mini proporciona una vista en tiempo real desde una altura específica, lo que es especialmente valioso para la detección de edificios derrumbados, ya que permite al algoritmo de análisis discernir entre estructuras intactas y aquellas que han sufrido un daño significativo desde un ángulo y una perspectiva específicos.

7.2.2. Implementación en Jetson Nano

Una vez que el modelo esté entrenado y validado, se implementa en la Jetson Nano para realizar detección en tiempo real. La siguiente figura muestra los requisitos específicos del sistema para ejecutar el algoritmo, destacando las instalaciones de librerías requeridas para su correcta operación. El sistema está basado en una NVIDIA Jetson Nano con Jetpack 4.6, que incluye varias bibliotecas y herramientas esenciales como CUDA, OpenCV, TensorRT, VPI, VisionWorks, Vulkan y cuDNN. Estas librerías proporcionan el soporte necesario para las operaciones de procesamiento de imágenes, inferencia de modelos de aprendizaje profundo y optimización de rendimiento en la Jetson Nano.

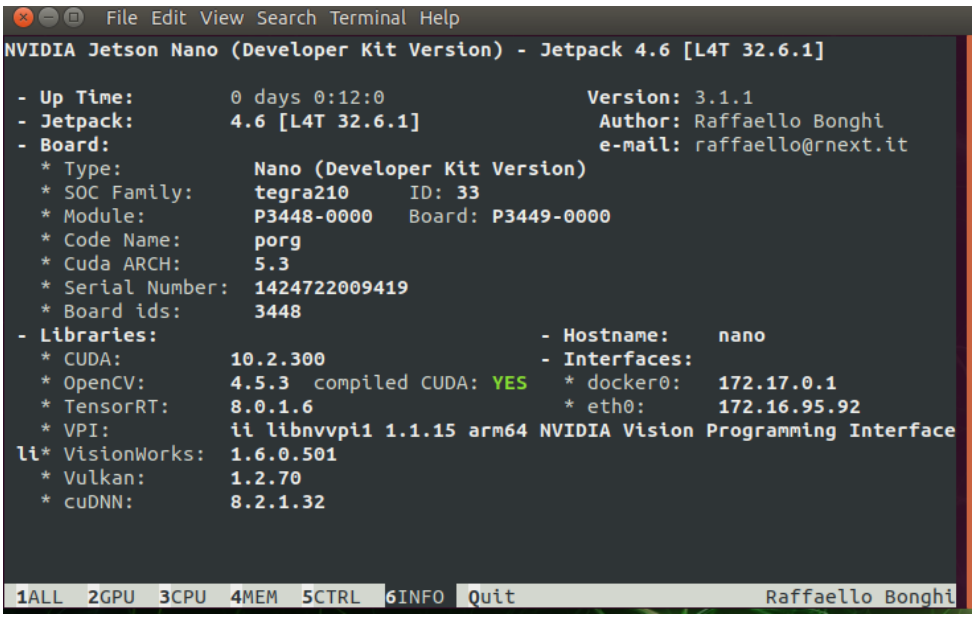

Figura 7.6: Requisitos del Sistema para la Ejecución del Algoritmo

7.3. Resultado de Algoritmo de Visión

7.3.1. Sistema de Algoritmo No Abordo

Tras la exitosa construcción e instrumentación del UAV, se diseñó e implementó un plan de vuelo sobre pequeñas estructuras para evaluar la efectividad del algoritmo de visión computacional que había sido probado anteriormente. Durante este vuelo, se capturó una serie de imágenes aéreas, entre ellas una destacada que fue procesada por el algoritmo aclarando que esta ejecución fue de forma no abordo.

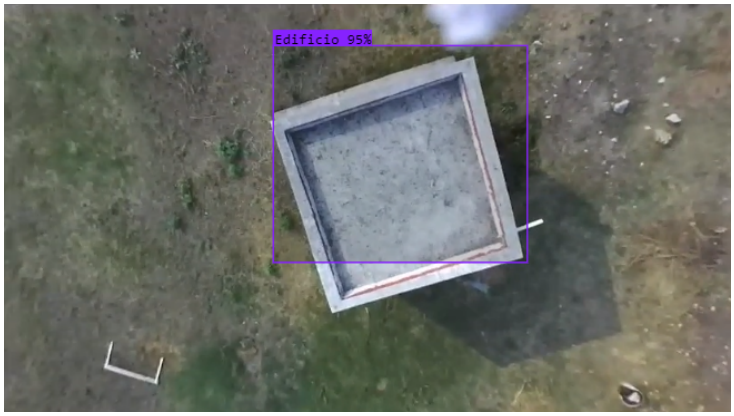

Figura 7.7: Detección de un edificio con un 95 % de confianza.

Como se muestra en la Figura 7.7, el algoritmo identificó un edificio, marcándolo con un cuadro delimitador de color púrpura y un texto superpuesto que indica “Edificio 95 %” , lo que refleja la confianza del algoritmo en la precisión de la detección. Esta alta confianza demuestra la capacidad del modelo para identificar correctamente estructuras complejas, validando la implementación del sistema en el UAV y la eficacia del algoritmo en condiciones reales.

Las siguientes Figuras 7.8 y 7.9 proporcionadas muestran resultados de un algoritmo, con la misma tarea de identificar edificios desde una perspectiva aérea diferente. En cada una de las imágenes, se puede ver la aplicación del algoritmo con las siguientes características:

En la Figura 7.8 , se observa una estructura con un cuadro delimitador púrpura y un

texto que indica “Edificio 98 %”. Esto muestra que el algoritmo ha detectado un edificio con un 98 % de confianza.

Figura 7.8: Detección de un edificio con un 98 % de confianza

La Figura 7.9 también presenta una estructura similar con un cuadro delimitador púrpura alrededor y un texto que dice “Edificio 91 %”, lo que refleja una confianza del 91 % en la detección del edificio por parte del algoritmo.

Figura 7.9: Detección de una estructura con un 91 % de confianza.

La Figura 7.10 muestra otra vista aérea capturada de lo que parece ser una estructura parcialmente colapsada, el algoritmo ha identificado esta estructura como Derrumbado con un 97 % de confianza, lo que indica un alto nivel de certeza en la detección de daños. El cuadro delimitador rojo marca específicamente la simulación de escombros o las partes derrumbadas, diferenciándolos claramente del resto de la estructura que parece estar intacta.

Figura 7.10: Confirmación de la detección de daños de la estructura con un 97 %.

La habilidad del UAV para detectar y cuantificar el daño estructural con alta precisión es crucial para evaluar la integridad de las construcciones afectadas por desastres y para priorizar las acciones de respuesta y recuperación.

7.3.2. Sistema del Algoritmo Abordo

Como paso siguiente en este proyecto, el algoritmo se adaptará para funcionar en tiempo real a bordo del UAV. Esto implicará la ejecución de la red neuronal directamente en el sistema embebido de la aeronave, permitiendo así la detección y el análisis instantáneo de estructuras durante el vuelo. La implementación en tiempo real no solo es un avance técnico considerable sino que también amplía las aplicaciones prácticas del UAV en misiones de monitoreo y reconocimiento, proporcionando información valiosa inmediatamente después de la captura de las imágenes.

Figura 7.11: Pruebas físicas del algoritmo abordo

Las pruebas de implementación del algoritmo se llevaron a cabo integrándolo al sistema de visión, específicamente en la Jetson Nano. Se diseñó una trayectoria automática utilizando QGroundControl [65]. Es importante destacar que la ejecución del algoritmo fue continua desde el inicio de la trayectoria, sin intervención manual en el sistema autónomo. En la Figura 7.12 se ilustran los resultados obtenidos en tiempo real, mostrando la detección exitosa de un edificio con un 89 % de confiabilidad.

Figura 7.12: Detección de un edificio con un 89 % de confianza.

La Figura 7.13, muestra una imagen capturada por el UAV que ha sido analizada mediante un algoritmo avanzado de visión computarizada. Este algoritmo ha sido entrenado específicamente para reconocer ciertos patrones como indicativos de daños estructurales, interpretando las cajas dentro de la imagen como tales daños. Se ha conseguido identificar un edificio afectado, catalogándolo como derrumbado con un nivel de confianza del 37 %. Se está trabajando en la mejora del sistema de entrenamiento, incrementando el volumen y la variedad de imágenes capturadas para fortalecer la precisión y fiabilidad del proceso de detección.

Figura 7.13: Detección de un edificio con un 78 % de confianza.

Durante el entrenamiento de la red neuronal , se monitorean varias métricas clave para evaluar el rendimiento del modelo en el conjunto de validación. A continuación se presenta un análisis de estas métricas a lo largo de las épocas de entrenamiento esto es con respecto a la Figura 7.14 .

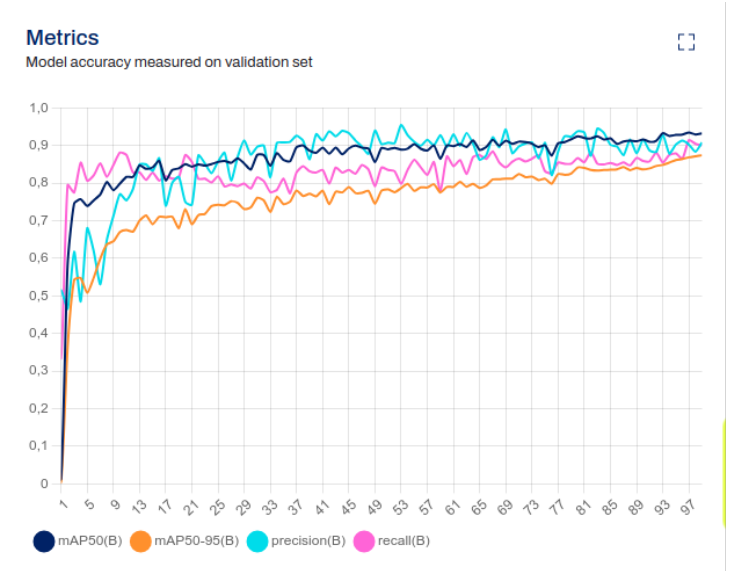

Figura 7.14: Precisión del modelo medida

- **mAP50 (B)**: representa la Mean Average Precision a un umbral de Intersect over Union (IoU) del 50 % para la clase B. Es una medida estándar de la precisión en la detección de objetos, evaluando qué tan precisamente el modelo identifica y localiza objetos relevantes.
- **mAP50-95 (B)**: es una métrica que promedia el mAP calculado en varios umbrales de IoU, desde 50 % hasta 95 %, en incrementos del 5 %, proporcionando una visión más comprensiva del rendimiento del modelo a través de diferentes niveles de precisión en la localización de objetos.
- **Precisión (B)**: calcula la proporción de detecciones correctas entre todas las predicciones positivas para la clase B, con una puntuación más alta indicando menos falsos positivos.
- **Recall (B)**: mide la capacidad del modelo para detectar todos los objetos relevantes para la clase B, con un valor más alto indicando que el modelo identifica la mayoría de los objetos de interés.

El gráfico ilustra que las métricas mejoran significativamente en las primeras épocas y luego alcanzan una estabilidad. Este comportamiento sugiere que el modelo llega a un punto de convergencia en su rendimiento, donde aprendizajes adicionales proporcionan mejoras marginales, potencialmente indicando el inicio de un sobreajuste.

Figura 7.15: Precisión del modelo medida

Esta Figura 7.15 muestra dos gráficos relacionados con el entrenamiento de un modelo de aprendizaje profundo para la detección de objetos.

El gráfico "**Box Loss**"muestra la diferencia entre las cajas delimitadoras predichas por el modelo y las cajas delimitadoras verdaderas (también conocidas como "ground truth") a lo largo de las iteraciones de entrenamiento y validación.

El eje X representa las iteraciones o épocas del entrenamiento, que se indican con números del 0 al más de 70. El eje Y representa la magnitud de la pérdida de las cajas (“Box Loss”), que varía entre 0 y 2.1.

Dos líneas representan los conjuntos de datos: La línea azul (“train”) indica el “Box Loss” en el conjunto de entrenamiento. La línea naranja (“val”) representa el “Box Loss” en el conjunto de validación. A lo largo del entrenamiento, ambas líneas muestran una tendencia decreciente, lo que indica que el modelo está aprendiendo correctamente a predecir las cajas delimitadoras. El objetivo es minimizar esta pérdida para que las predicciones del modelo coincidan lo más cercanamente posible con las anotaciones verdaderas.

El gráfico “**Class Loss**” se centra en la precisión de las clases detectadas por el modelo en cada detección.

Al igual que el primer gráfico, el eje X representa las iteraciones o épocas del entrenamiento, marcadas del 0 al 98 aproximadamente. El eje Y muestra la magnitud de la pérdida de clases (“Class Loss”), la cual comienza cerca de 45 y desciende. Nuevamente, hay dos líneas representando el conjunto de entrenamiento (azul) y el de validación (naranja). En este caso, la “Class Loss” disminuye rápidamente al principio, lo que sugiere que el modelo está mejorando su capacidad de clasificar correctamente las clases de los objetos detectados. La meta es reducir al máximo esta pérdida, lo que indicaría una alta precisión en la clasificación.

En general, la disminución en ambas métricas de pérdida a lo largo de las iteraciones es un signo positivo de que el modelo está mejorando su rendimiento tanto en la localización como en la clasificación de los objetos.

7.3.3. Reporte de Cuantificación de Daños

El algoritmo desarrollado en esta tesis se basa en el uso de UAVs para la detección y clasificación de daños en estructuras, incluyendo la identificación de edificios intactos, edificios derrumbados y víctimas. Este enfoque permite una evaluación rápida y detallada del estado de las estructuras, proporcionando datos valiosos para los equipos de rescate y recuperación.

El UAV, equipado con la cámara Zed Mini comunicado con la Jetson Nano y utilizando una red neuronal de visión computacional, realiza vuelos autónomos sobre las áreas afectadas, capturando imágenes en tiempo real. Estas imágenes son procesadas por el algoritmo para detectar y clasificar los diferentes tipos de daños. Durante la inspección, el UAV genera un reporte detallado que incluye el conteo de víctimas, edificios intactos y edificios derrumbados, proporcionando una visión clara y concisa del estado del área inspeccionada.

A continuación se presenta el código encargado de generar el reporte de inspección al finalizar el vuelo del UAV:

```
void generate_report() {
    std::ofstream report_file("inspection_report.txt");
    if (report_file.is_open()) {
        report_file << "Reporte de Cuantificacion de Danos:\n";
        report_file << "-----------------------------------\n";
        report_file << "Numero de Victimas Detectadas: " << victim_count << "\n";
        report_file << "Numero de Edificios Intactos Detectados: " <<
            intact_building_count << "\n";
        report_file << "Numero de Edificios Derrumbados Detectados: " <<
            collapsed_building_count << "\n";
        report_file.close();
        ROS_INFO("REPORTE GENERADO: inspection_report.txt");
    } else {
        ROS_ERROR("NO SE PUDO ABRIR EL ARCHIVO DE REPORTE.");
    }
}
```

Este código implementa una función que genera un archivo de texto denominado ‘inspection report.txt.‘ Este archivo contiene un resumen de la inspección realizada por el UAV, incluyendo el número de víctimas detectadas, el número de edificios intactos y el número de edificios derrumbados. La función ‘generate report‘ se llama al finalizar la inspección, asegurando que se registren los resultados obtenidos durante el vuelo.

La implementación de este sistema de cuantificación de daños representa un avance significativo en la automatización y eficiencia de las inspecciones post-catástrofe, permitiendo una respuesta más rápida y efectiva a las emergencias.

7.3.4. Señales del UAV abordo

Posteriormente al finalizar las pruebas se han obtenido las señales que se presentaron durante el vuelo, la gráfica de la Figura 7.16 muestra la evolución temporal de los ángulos de orientación (pitch, roll y yaw) de un UAV durante un período de 60 segundos.

Los tres paneles representan cómo varían estos ángulos a lo largo del tiempo, indicando eventos significativos en los segundos 20 y 40 que afectan temporalmente la orientación del UAV, seguida de una estabilización del sistema.

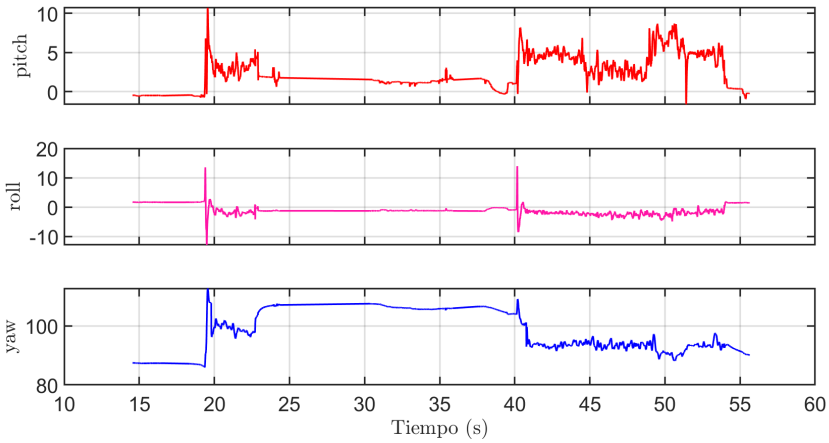

Figura 7.16: Señales de ejes rotacionales

Al igual en la siguiente gráfica de la Figura 7.17 se presenta las señales de los ejes de traslación X,Y,Z, durante un período de 60 segundos. En el eje Y, se observa un incremento significativo que alcanza hasta 50 metros alrededor de los 45 segundos, seguido de una rápida disminución. En el eje X, la posición desciende gradualmente desde el inicio, con una notable disminución a partir de los 40 segundos, alcanzando alrededor de -5 metros. En el eje Z, se presentan dos picos importantes, el primero alrededor de los 20 segundos alcanzando 7 metros y el segundo alrededor de los 45 segundos llegando a 10 metros, seguidos de una estabilización. Estos cambios reflejan maniobras específicas del UAV que afectan su posición en los diferentes ejes.

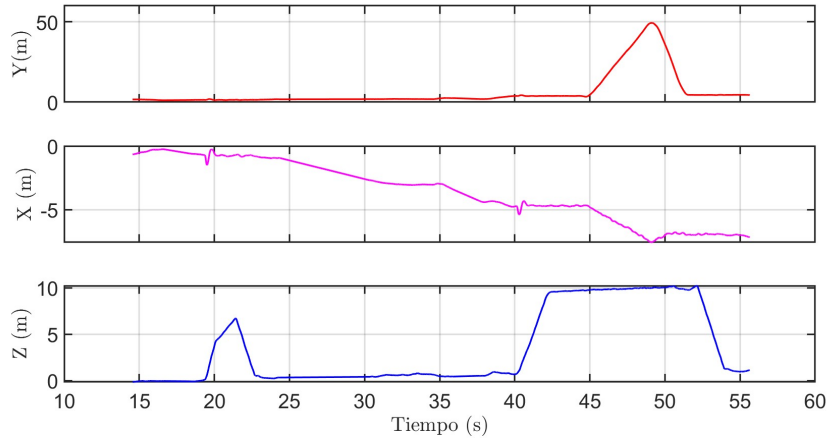

Figura 7.17: Señales de ejes tranlacionales

Video en tiempo real

El siguiente video muestra una de las pruebas en tiempo real realizadas con el sistema de visión integrado a bordo del UAV, esto fue con el objetivo de evaluar la efectividad del algoritmo en la detección de las clases ya previamente entrenadas. En él se observa cómo el sistema identifica, de manera práctica, estructuras afectadas en un entorno real. Para más información, puede ver el video en https://youtu.be/msa07GevKVY.

8

Conclusión

En este trabajo se presenta un algoritmo de control basado en un controlador de modo deslizante de tiempo fijo que utiliza la dinámica de errores no lineales en las trayectorias (x, y) de un cuadricóptero. Se emplea el modelo dinámico reducido del vehículo aéreo. Para mejorar el rendimiento de la dinámica del sistema de traslación, se utiliza una superficie de deslizamiento no lineal para reducir el tiempo de convergencia y el vehículo aéreo sigue la posición deseada de x_d,y_d y z_d. Además, se calculan los movimientos de traslación de los ejes $X - Y$ y de rotación (ϕ_d, θ_d).

También se utiliza una función de Lyapunov para demostrar la estabilización de cada eje del UAV; suponemos que entre los ejes existe un acoplamiento que desaparece con el algoritmo robusto propuesto.

Se realizan resultados de simulación numérica para cuantificar el rendimiento del control, que muestran que el algoritmo de control diseñado mejora el tiempo de respuesta de convergencia, la precisión y la robustez, a la vez que supera al controlador NTSMC y reduce el chattering.

Por otro lado, la tesis destaca la implementación de un sistema de visión computacional avanzado que permite al UAV realizar una inspección de los daños a través del análisis de imágenes y datos capturados. Esta funcionalidad se potencia mediante el uso de algoritmos de inteligencia artificial, específicamente con una red neuronal convolucional, que facilitan la detección y clasificación de elementos críticos en el entorno, mejorando así la eficacia de las operaciones de respuesta ante desastres.

El aspecto innovador de la tesis se evidencia en la aplicación de inteligencia artificial para la percepción activa, permitiendo al UAV no solo navegar sino también identificar y evaluar daños en estructuras o terrenos afectados por desastres naturales. La integración de la visión computacional permite al UAV procesar y analizar datos en vuelo, para realizar una cuantificación precisa de los daños. Esta capacidad no solo mejora la eficiencia y la efectividad de las respuestas ante desastres, sino que también potencia la toma de decisiones basada en datos precisos y oportunos. En conclusión, la tesis representa un importante avance en la tecnología de UAVs, demostrando que el control robusto, visión computacional e inteligencia artificial pueden revolucionar la respuesta a desastres naturales. Este trabajo sienta las bases para futuras investigaciones que podrían expandir aún más las capacidades y el alcance de los UAVs en diversas aplicaciones, mejorando significativamente la eficiencia y efectividad de las operaciones de emergencia y rescate.

8.1. Recomendaciones para trabajos futuros.

Para los futuros trabajos, no sólo se enfocarán en mejorar la precisión en la medición de la posición y actitud del UAV, sino que también se propone en utilizar la visión computacional para la detección y monitoreo de desastres naturales. Reconociendo la importancia de la respuesta temprana y la obtención de información precisa en situaciones de desastre, se busca integrar capacidades de visión computacional avanzada en el sistema de sensores del UAV.

La visión computacional se refiere al campo de estudio que permite a las máquinas comprender y analizar imágenes o secuencias de video. Al aprovechar esta tecnología, el UAV podría ser capaz de detectar y reconocer patrones asociados con diferentes tipos de desastres naturales, como incendios forestales, inundaciones, terremotos o deslizamientos de tierra.

En el caso de los incendios forestales, por ejemplo, la visión computacional podría ayudar al UAV a identificar áreas afectadas por el fuego mediante el análisis de cambios en la temperatura y en los patrones de calor detectados en imágenes térmicas. Esto permitiría una respuesta más rápida y precisa de los equipos de emergencia, facilitando la localización de focos de incendio y la evaluación de la propagación.

Para la detección de inundaciones, el UAV equipado con visión computacional podría analizar imágenes capturadas desde diferentes altitudes y ángulos para identificar áreas inundadas. Al combinar esta información con datos de sensores adicionales, como los sistemas de medición de nivel de agua, el UAV podría proporcionar una visión general precisa de las áreas afectadas y contribuir a los esfuerzos de rescate y mitigación.

Además de la detección de desastres naturales, la visión computacional también podría utilizarse para tareas de evaluación de daños después de un evento catastrófico. El UAV podría capturar imágenes detalladas de áreas afectadas y utilizar algoritmos de visión computacional para analizar y clasificar los daños en infraestructuras, edificios y carreteras. Esto permitiría a los equipos de rescate y recuperación priorizar los esfuerzos de recuperación y asignar recursos de manera más eficiente.

En conclusión, en futuros trabajos se pretende no solo mejorar la precisión en la me-

dición de la posición y actitud del UAV mediante el uso de una variedad de sensores, como visión artificial y LiDAR, sino también aprovechar la visión computacional para la detección y monitoreo de desastres naturales. La integración de estas tecnologías avanzadas en el UAV podría tener un impacto significativo en la respuesta a desastres y en la evaluación de daños, mejorando la capacidad de los equipos de emergencia para tomar decisiones informadas y salvar vidas en situaciones críticas.

9

Apéndice A

En el desarrollo de este proyecto sea publicado un artículo de investigación, cuya referencia es la siguiente:

Olguin-Roque J, Salazar S, González-Hernandez I, Lozano R. A Robust Fixed-Time Sliding Mode Control for Quadrotor UAV. Algorithms. 2023; 16(5):229. https://doi.org/10.3390/a16050229

algorithms

MDPI

Article

A Robust Fixed-Time Sliding Mode Control for Quadrotor UAV

Jairo Olguin-Roque, Sergio Salazar, Iván González-Hernandez and Rogelio Lozano *

UMI-LAFMIA, CINVESTAV, Ciudad de Mexico 07360, Mexico
* Correspondence: rogelio.lozano@cinvestav.mx

Abstract: This paper proposes a robust algorithm based on a fixed-time sliding mode controller (FTSMC) for a Quadrotor aircraft. This approach is based on Lyapunov theory, which guarantees system stability. Nonlinear error dynamics techniques are used to achieve accurate trajectory tracking in the presence of disturbances. The performance of the FTSMC is compared with the typical non-singular terminal sliding mode controller (NTSMC) to evaluate its effectiveness. The numerical results show that FTSMC is more efficient than the typical NTSMC in disturbance reduction.

Keywords: Quadrotor aircraft; FTSMC; NTSMC; robust control; trajectory tracking control

1. Introduction

UAVs (Unmanned Aerial Vehicles), also known as drones, have revolutionized the way in which various operations are carried out in different fields of industry. The ability of UAVs to fly at high altitudes and in hard-to-reach areas, combined with advanced sensor and camera technologies, allows inspection and surveillance tasks to be carried out with greater precision and safety [1,2]. In the field of precision agriculture, UAVs can be used to obtain detailed information on crop conditions, enabling more efficient land and water management [3,4].

Moreover, UAVs are also used to create detailed maps of specific terrains and geographical areas, which is of great help in activities such as infrastructure planning and natural resource management [5,6]. In natural disaster situations, UAVs can be used to carry out rapid and accurate assessments of damage and the needs of affected communities [7].

Furthermore, in the field of air transport of payloads, UAVs offer an economical and safe alternative for the delivery of supplies and materials to remote or inaccessible areas. In addition, the versatility and efficiency of UAVs make them indispensable tools in various areas of industry and society [8–11]. The following are just a few examples.

Proportional-Integral-Derivative (PID) control is a control technique commonly used in industry. Its implementation requires several iterative tunings of its parameters. In a recent study [12], a nonlinear PID controller for total thrust and torques applied to follow a desired trajectory of the Quadrotor aircraft was proposed.

This robust controller is highly adaptable to different types of UAVs and offers a promising approach for controlling aerial vehicles that require fast, accurate, and robust responses in challenging and dynamic situations.

On the other hand, the work in [13] compared the tracking capability of three different controllers in the design of a flight controller for a V-tail Quadrotor aircraft. The controllers compared were proportional derivative (PD), PID and sliding mode controller (SMC). According to their simulation results, the PD controller presented a steady-state error that was not corrected, which could be problematic in applications requiring high precision. In contrast, the SMC allowed the Quadrotor aircraft to quickly converge to the desired point.

In this context, controllers based on sliding modes present an inherent phenomenon called chattering, which is a problem for engineers because it wears out and reduces the useful life of the actuators in a given process [14]. This effect is characterized by rapid oscillations and high frequency in the control signal. In addition, this phenomenon can

check for updates

Citation: Olguin-Roque, J.; Salazar, S.; González-Hernandez, I.; Lozano, R. A Robust Fixed-Time Sliding Mode Control for Quadrotor UAV. *Algorithms* **2023**, *16*, 229. https://doi.org/10.3390/a16050229

Academic Editors: Mircea-Bogdan Radac and Frank Werner

Received: 17 February 2023
Revised: 19 March 2023
Accepted: 20 April 2023
Published: 29 April 2023

Figura 9.1: Articulo Publicado

9.1. Código Autonomía del UAV

```
#include <gnc_functions.hpp>
#include <cmath>
#include <unistd.h> // para sleep()
#include <ros/ros.h>
#include <yolov8_ros_msgs/BoundingBoxes.h>
#include <geometry_msgs/Point.h>
#include <fstream> // para escribir el archivo de resultados

geometry_msgs::Point last_detection_position; // Almacena la posicion del ultimo edificio detectado
bool first_detection = true; // Indica si es la primera deteccion
const float DETECTION_DISTANCE_THRESHOLD = 13.0; // Distancia minima para considerar una nueva deteccion, en metros
float previous_altitude = 20.0; // Altitud anterior al descenso, inicializada a la altitud de crucero

int victim_count = 0;
int intact_building_count = 0;
int collapsed_building_count = 0;

void detection_callback(const yolov8_ros_msgs::BoundingBoxes::ConstPtr& msg) {
    geometry_msgs::Point current_position = get_current_location(); // Suponemos que esta funcion esta definida para obtener la ubicacion actual del dron

    for (const auto& box : msg->bounding_boxes) {
        if (box.Class == "Edificio" || box.Class == "Derrumbado" || box.Class == "Victima") {
            float distance_from_last = hypot(current_position.x - last_detection_position.x,
                                             current_position.y - last_detection_position.y);
            if (first_detection || distance_from_last > DETECTION_DISTANCE_THRESHOLD) {
                first_detection = false;
                last_detection_position = current_position; // Actualizar la ultima posicion de deteccion
                ROS_INFO("DETECTADO %s A %f METROS", box.Class.c_str(), distance_from_last);
                if (set_mode("BRAKE") == 0) {
                    if (box.Class == "Derrumbado") {
                        collapsed_building_count++;
                        ROS_INFO("ESTRUCTURA DERRUMBADA, IDENTIFICADO VICTIMAS Y SOBREVIVIENTES...");
                        previous_altitude = current_position.z; // Guardar la altitud anterior
```

```
34                          set_destination_lla_raw(current_position.x, current_position.y,
                                5, 0); // Bajar a 5 metros
35                          sleep(5); // Esperar 5 segundos
36                          ROS_INFO("DETECTANDO VICTIMAS");
37                          sleep(5); // Esperar otros 5 segundos
38                          // Subir a la altitud anterior
39                          set_destination_lla_raw(current_position.x, current_position.y,
                                previous_altitude, 0);
40                          ROS_INFO("SUBIENDO A LA ALTITUD ANTERIOR Y REANUDANDO INSPECCION
                                ...");
41                          sleep(5); // Esperar a que suba a la altitud anterior
42                      } else if (box.Class == "Edificio") {
43                          intact_building_count++;
44                          ROS_INFO("EL DRON HA DETECTADO UNA ESTRUCTURA.");
45                          sleep(5); // Detener durante 5 segundos en modo BRAKE
46                          ROS_INFO("ANALIZANDO ESTRUCTURA IDENTIFICADA...");
47                          sleep(5); // Esperar 5 segundos mientras se muestra el mensaje
48                          ROS_INFO("ANALISIS COMPLETADO.");
49                      } else if (box.Class == "Victima") {
50                          victim_count++;
51                          ROS_INFO("VICTIMA DETECTADA.");
52                      }
53                      ROS_INFO("REANUDANDO TRAYECTORIA PROGRAMADA...");
54                      set_mode("GUIDED");
55                      set_speed(4.0); // Restablecer la velocidad a 4.0 m/s para continuar
56                  } else {
57                      ROS_ERROR("FALLO AL CAMBIAR A MODO BRAKE.");
58                  }
59              } else {
60                  ROS_INFO("OBJETO '%s' DETECTADO NUEVAMENTE A %f METROS", box.Class.c_str
                        (), distance_from_last);
61              }
62              break;
63          }
64      }
65  }
66
67  void generate_report() {
68      std::ofstream report_file("inspection_report.txt");
69      if (report_file.is_open()) {
70          report_file << "Reporte de Cuantificacion de Danos:\n";
71          report_file << "-----------------------------------\n";
72          report_file << "Numero de Victimas Detectadas: " << victim_count << "\n";
73          report_file << "Numero de Edificios Intactos Detectados: " <<
                intact_building_count << "\n";
74          report_file << "Numero de Edificios Derrumbados Detectados: " <<
                collapsed_building_count << "\n";
```

```
75          report_file.close();
76          ROS_INFO("REPORTE GENERADO: inspection_report.txt");
77      } else {
78          ROS_ERROR("NO SE PUDO ABRIR EL ARCHIVO DE REPORTE.");
79      }
80  }
81
82  int main(int argc, char** argv)
83  {
84      ros::init(argc, argv, "gnc_node");
85      ros::NodeHandle gnc_node("~");
86      ros::Subscriber sub = gnc_node.subscribe("/yolov8/BoundingBoxes", 10,
            detection_callback);
87
88      init_publisher_subscriber(gnc_node);
89      wait4connect();
90      set_mode("GUIDED");
91      arm();
92      ROS_INFO("DRON ARMADO Y EN MODO AUTONOMO.");
93
94      float ascent_speed = 0.5;
95      set_speed(ascent_speed);
96
97      float alt_lat = 19.51;
98      float alt_lon = -99.12;
99      float alt_alt = 10;
100     takeoff_global(alt_lat, alt_lon, alt_alt);
101     ROS_INFO("DESPEGANDO...");
102     sleep(10); // Esperar 10 segundos despues del despegue
103
104     float target_lat = 19.512202;
105     float target_lon = -99.129115;
106     float target_alt = 20;
107     set_speed(5.0); // Velocidad para dirigirse al primer waypoint
108     set_destination_lla_raw(target_lat, target_lon, target_alt, 0);
109     ROS_INFO("DETECTANDO ESTRUCTURAS...");
110     sleep(5);
111
112     float second_target_lat = 19.512102;
113     float second_target_lon = -99.128833;
114     float second_alt = 20;
115
116     set_speed(4); // Ajustar la velocidad a 4.0 m/s para el segundo waypoint
117     ros::Rate rate(1.0);
118     while (ros::ok()) {
119         ros::spinOnce(); // Procesa callbacks
120         set_destination_lla_raw(second_target_lat, second_target_lon, second_alt, 0);
```

```
121             ROS_INFO("DETECTANDO ESTRUCTURAS...");
122             rate.sleep();
123         }
124         generate_report(); // Generar el reporte al finalizar la inspeccion
125         return 0;
126     }
```

Bibliografía

[1] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 4 2020.

[2] Ebtehal Turki Alotaibi, Shahad Saleh Alqefari, and Anis Koubaa. Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7:55817–55832, 2019.

[3] Pasquale Daponte, Luca De Vito, Luigi Glielmo, Luigi Iannelli, Davide Liuzza, Francesco Picariello, and Giuseppe Silano. A review on the use of drones for precision agriculture. *IOP Conference Series: Earth and Environmental Science*, 275, 5 2019.

[4] Chitranjan Singh, Rahul Mishra, Hari Prabhat Gupta, and Preti Kumari. The internet of drones in precision agriculture: Challenges, solutions, and research opportunities. *IEEE Internet of Things Magazine*, 5:180–184, 5 2022.

[5] DJI. Matrice 100 quadcopter for developers. https://www.dji.com/mx/matrice100/info/. [Accedido el 30 de mayo de 2023].

[6] Rojas-Estrada M.A. Jimenez-Lizarraga J.Olguin-Roque, M.A. Ochoa-Villegas. Uso de un vant como un sistema para la adquisición de información de Áreas construidas en terrenos de zonas metropolitanas para usos catastrales. *Congr. Int. en Ing. Electrónica. Mem. ELECTRO*, 39:141–146, 10 2017.

[7] Ali Arjomandi-Nezhad, Moein Moeini-Aghtaie, Mahmud Fotuhi-Firuzabad, and Farrokh Aminifar. Budget-constrained drone allocation for distribution system damage assessment. *IET Smart Grid*, 5:42–50, 2 2022.

[8] Bei Wu, Ruyu Fu, Jundao Chen, Junhui Zhu, and Rongfang Gao. Research on natural disaster early warning system based on uav technology. *IOP Conference Series: Earth and Environmental Science*, 787, 6 2021.

[9] Viswa N. Sankaranarayanan, Spandan Roy, and Simone Baldi. Aerial transportation of unknown payloads: Adaptive path tracking for quadrotors. pages 7710–7715, 2020.

[10] Viswa N. Sankaranarayanan, Rishabh D. Yadav, Rahul K. Swayampakula, Sourish Ganguly, and Spandan Roy. Robustifying payload carrying operations for quadrotors under time-varying state constraints and uncertainty. *IEEE Robotics and Automation Letters*, 7(2):4885–4892, 2022.

[11] Farid Golnaraghi and Benjamin C Kuo. *Automatic control systems*. McGraw-Hill Education, 2017.

[12] Jose J Castillo-Zamora, Karla A Camarillo-Gomez, Gerardo I Perez-Soto, and Juvenal Rodriguez-Resendiz. Comparison of pd, pid and sliding-mode position controllers for v–tail quadcopter stability. *Ieee Access*, 6:38086–38096, 2018.

[13] Rajko Svečko, Dušan Gleich, and Andrej Sarjaš. The effective chattering suppression technique with adaptive super-twisted sliding mode controller based on the quasi-barrier function; an experimentation setup. *Applied Sciences*, 10(2), 2020.

[14] Yufei Liang, Dong Zhang, Guodong Li, and Tao Wu. Adaptive chattering-free pid sliding mode control for tracking problem of uncertain dynamical systems. *Electronics*, 11(21), 2022.

[15] Ihsan Ullah and Hailong Pei. Fixed time disturbance observer based sliding mode control for a miniature unmanned helicopter hover operations in presence of external disturbances. *IEEE Access*, 8:73173–73181, 2020.

[16] Ashkan Parsa, Ahmad Kalhor, and Mohammadali Amiri Atashgah. Backstepping-sliding mode control performance enhancement using close loop identification for quadrotor trajectory tracking. In *2017 5th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pages 286–291, 2017.

[17] Iván González-Hernández, Sergio Salazar, A. E. Rodríguez-Mata, Filiberto Muñoz-Palacios, Ricardo López, and Rogelio Lozano. Enhanced robust altitude controller via integral sliding modes approach for a quad-rotor aircraft: Simulations and real-time results. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 88:313–327, 12 2017.

[18] Boussad Abci, Gang Zheng, Denis Efimov, and Maan El Badaoui El Najjar. Robust altitude and attitude sliding mode controllers for quadrotors. volume 50, pages 2720–2725. Elsevier B.V., 7 2017.

[19] Xingling Shao, Jun Liu, Huiliang Cao, Chong Shen, and Honglun Wang. Robust dynamic surface trajectory tracking control for a quadrotor uav via extended state observer. *International Journal of Robust and Nonlinear Control*, 28:2700–2719, 5 2018.

[20] Tianpeng Huang, Deqing Huang, Zhikai Wang, and Awais Shah. Robust tracking control of a quadrotor uav based on adaptive sliding mode controller. *Complexity*, 2019, 2019.

[21] Moussa Labbadi and Mohamed Cherkaoui. Robust adaptive nonsingular fast terminal sliding-mode tracking control for an uncertain quadrotor uav subjected to disturbances. *ISA Transactions*, 99:290–304, 4 2020.

[22] Hamid Hassani, Anass Mansouri, and Ali Ahaitouf. A new robust adaptive sliding mode controller for quadrotor uav flight. pages 1–6, 2020.

[23] Kyunghyun Lee, Sangkyeum Kim, Seongwoo Kwak, and Kwanho You. Quadrotor stabilization and tracking using nonlinear surface sliding mode control and observer. *Applied Sciences*, 11(4), 2021.

[24] Ahmed Eltayeb, Mohd Fuaad Rahmat, Mohd Ariffanan Mohd Basri, and Magdi S. Mahmoud. An improved design of integral sliding mode controller for chattering attenuation and trajectory tracking of the quadrotor uav. *Arabian Journal for Science and Engineering*, 45:6949–6961, 8 2020.

[25] Fahimeh Shiravani, Patxi Alkorta, Jose Antonio Cortajarena, and Oscar Barambones. An enhanced sliding mode speed control for induction motor drives. *Actuators*, 11(1), 2022.

[26] Iván González-Hernández, Sergio Salazar, Rogelio Lozano, and Oscar Ramírez-Ayala. Real-time improvement of a trajectory-tracking control based on super-twisting algorithm for a quadrotor aircraft. *Drones*, 6(2), 2022.

[27] Bo Li, Wenquan Gong, Yongsheng Yang, Bing Xiao, and Dechao Ran. Appointed fixed time observer-based sliding mode control for a quadrotor uav under external disturbances. *IEEE Transactions on Aerospace and Electronic Systems*, 58(1):290–303, 2022.

[28] Hamid Hassani, Anass Mansouri, Ali Ahaitouf, and L. Fortuna. Robust finite-time tracking control based on disturbance observer for an uncertain quadrotor under external disturbances. 2022, jan 2022.

[29] Li Yu, Guang He, Xiangke Wang, and Shulong Zhao. Robust fixed-time sliding mode attitude control of tilt trirotor uav in helicopter mode. *IEEE Transactions on Industrial Electronics*, 69(10):10322–10332, 2022.

[30] Jiaqi Zheng, Lei Song, Lingya Liu, Wenbin Yu, Yiyin Wang, and Cailian Chen. Fixed-time sliding mode tracking control for autonomous underwater vehicles. *Applied Ocean Research*, 117:102928, 2021.

[31] Kaikai Deng, Dong Zhao, Qiaoyue Han, Shuyue Wang, Zihan Zhang, Anfu Zhou, and Huadong Ma. Geryon: Edge assisted real-time and robust object detection on drones via mmwave radar and camera fusion. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(3), sep 2022.

[32] Danu Kim, Jeongkyung Won, Eunji Lee, Kyung Ryul Park, Jihee Kim, Sangyoon Park, Hyunjoo Yang, and Meeyoung Cha. Disaster assessment using computer vision and satellite imagery: Applications in detecting water-related building damages. *Frontiers in Environmental Science*, 10, 2022.

[33] Lixia Gong, Chao Wang, Fan Wu, Jingfa Zhang, Hong Zhang, and Qiang Li. Earthquake-induced building damage detection with post-event sub-meter vhr terrasar-x staring spotlight imagery. *Remote Sensing*, 8(11), 2016.

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016.

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. 25, 2012.

[36] Bo Pang, Erik Nijkamp, and Ying Nian Wu. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics*, 45(2):227–248, 2020.

[37] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 2021): Developing Global Digital Economy after COVID-19.

[38] Kushagra Srivastava, Dhruv Patel, Aditya Kumar Jha, Mohhit Kumar Jha, Jaskirat Singh, Ravi Kiran Sarvadevabhatla, Pradeep Kumar Ramancharla, Harikumar Kandath, and K. Madhava Krishna. Uav-based visual remote sensing for automated building inspection. 2022.

[39] Endre Erős, Martin Dahl, Kristofer Bengtsson, Atieh Hanna, and Petter Falkman. A ros2 based communication architecture for control in collaborative and intelligent automation systems. *Procedia Manufacturing*, 38:349–357, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing.

[40] Billie F. Spencer, Vedhus Hoskere, and Yasutaka Narazaki. Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering*, 5(2):199–222, 2019.

[41] Hafiz Suliman Munawar, Fahim Ullah, Amirhossein Heravi, Muhammad Jamaluddin Thaheem, and Ahsen Maqsoom. Inspecting buildings using drones and computer vision: A machine learning approach to detect cracks and damages. *Drones*, 6(1), 2022.

[42] Kushagra Srivastava, Dhruv Patel, Aditya Kumar Jha, Mohhit Kumar Jha, Jaskirat Singh, Ravi Kiran Sarvadevabhatla, Pradeep Kumar Ramancharla, Harikumar Kandath, and K. Madhava Krishna. Uav-based visual remote sensing for automated building inspection. In Leonid Karlinsky, Tomer Michaeli, and Ko Nishino, editors,

Computer Vision – ECCV 2022 Workshops, pages 299–316, Cham, 2023. Springer Nature Switzerland.

[43] Mohammad Kakooei and Yasser Baleghi. Fusion of vertical and oblique images using intra-cluster-classification for building damage assessment. *Computers and Electrical Engineering*, 105:108536, 2023.

[44] Tushar Agrawal, Suraj, and Merin Meleet. Classification of natural disaster using satellite drone images with cnn using transfer learning. In *2021 International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)*, pages 1–5, 2021.

[45] DroneDeploy. Inspection. https://help.dronedeploy.com/hc/en-us/articles/1500004862021-Data-Sharing/. [Accedido el 30 de mayo de 2023].

[46] David Culler. Software. https://www.therobotreport.com/utm-patents-precisionhawk-air-traffic-control-drones-aircraft/. [Accedido el 30 de mayo de 2023].

[47] Kespry. Kespry and dronebase announce partnership to expand bring your own drone program to insurance and multi site mining and aggregates enterprises. https://www.worldconstructiontoday.com/news/kespry-and-dronebase-announce-partnership-to-expand-bring-your-own-drone-program-to-insurance-and-multi-site-mining-and-aggregates-enterprises/. [Accedido el 30 de mayo de 2023].

[48] Rogelio Lozano. *Unmanned aerial vehicles: Embedded control*. John Wiley & Sons, 2013.

[49] Emmanuel Moulay, Vincent Léchappé, Emmanuel Bernuau, and Franck Plestan. Robust fixed-time stability: Application to sliding-mode control. *IEEE Transactions on Automatic Control*, 67(2):1061–1066, 2022.

[50] Vadim I. Utkin. *Sliding Mode Control: Mathematical Tools, Design and Applications*, pages 289–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[51] Ardupilot. The cube black. https://ardupilot.org/copter/docs/common-thecube-overview.html/. [Accedido el 30 de mayo de 2023].

[52] Arif Mehmood Zaman and Jaho Seo. Development of an autonomous flying excavator. *Engineering Proceedings*, 24(1), 2022.

[53] Jun Yang, Arun Geo Thomas, Satish Singh, Simone Baldi, and Ximan Wang. A semi-physical platform for guidance and formations of fixed-wing unmanned aerial vehicles. *Sensors*, 20(4), 2020.

[54] NVIDIA. Jetson nano developer kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit/. [Accedido el 30 de mayo de 2023].

[55] Philippe Martin Wyder, Yan-Song Chen, Adrian J. Lasrado, Rafael J. Pelles, Robert Kwiatkowski, Edith O. A. Comas, Richard Kennedy, Arjun Mangla, Zixi Huang, Xiaotian Hu, Zhiyao Xiong, Tomer Aharoni, Tzu-Chan Chuang, and Hod Lipson. Autonomous drone hunter operating by deep learning and all-onboard computations in gps-denied environments. *PLOS ONE*, 14(11):1–18, 11 2019.

[56] Rune Hylsberg Jacobsen, Lea Matlekovic, Liping Shi, Nicolaj Malle, Naeem Ayoub, Kaspar Hageman, Simon Hansen, Frederik Falk Nyboe, and Emad Ebeid. Design of an autonomous cooperative drone swarm for inspections of safety critical infrastructure. *Applied Sciences*, 13(3), 2023.

[57] IntelRealSense. Depth camera d435. https://www.intelrealsense.com/depth-camera-d435//. [Accedido el 30 de mayo de 2023].

[58] Saad M. S. Mukras and Hanafy M. Omar. Development of a 6-dof testing platform for multirotor flying vehicles with suspended loads. *Aerospace*, 8(11), 2021.

[59] Ardupilot. Integration of ardupilot and vio tracking camera for gps-less localization and navigation. https://discuss.ardupilot.org/t/gsoc-2019-integration-of-ardupilot-and-vio-tracking-camera-for-gps-less-localization-and-navigation/42394/. [Accedido el 30 de mayo de 2023].

[60] Ardupilot. Meet zed mini, the world's first camera for mixed-reality.

[61] Dillon Reis, Jordan Kupec, Jacqueline Hong, and Ahmad Daoudi. Real-time flying object detection with yolov8, 2023.

[62] ultralytics. ultralytics yolov8. https://github.com/ultralytics/ultralytics?ref=blog.roboflow.com. [Accedido el 19 de septiembre de 2023].

[63] IPN3. Sanas8 dataset. https://universe.roboflow.com/ipn3/sanas8, aug 2023. visited on 2023-09-19.

[64] Yong Feng, Xinghuo Yu, and Zhihong Man. Non-singular terminal sliding mode control of rigid manipulators. *Automatica*, 38(12):2159–2167, 2002.

[65] Open Robotics. Gazebo. https://gazebosim.org/home, March 2023. visited on 2023-03-19.

[66] QGroundControl. Qgroundcontrol. https://docs.qgroundcontrol.com/master/en/, March 2023. visited on 2023-03-19.

[67] StereoLabs. Zed mini. [Accedido el 20 de Marzo de 2024].

[68] Robot Operating System. Ros. [Accedido el 20 de Marzo de 2024].

[69] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. 2016.