DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

TEMA: DISEÑO E IMPEMENTACIÓN DE UN CONTROL INTELIGENTE PARA NAVEGACIÓN AUTÓNOMA DE DOS MINI-DRONES Y RECONSTRUCCIÓN DE OBJETOS 3D

AUTORES: CEVALLOS REA, JORDY GABRIEL

LOZA ALBUJA, CHRISTIAN SANTIAGO

DIRECTOR: ING. ERAZO SOSA, ANDRÉS SEBASTIÁN

SANGOLQUÍ

2019

CERTIFICADO DEL DIRECTOR

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**DISEÑO E IMPLEMENTACIÓN DE UN CONTROL INTELIGENTE PARA NAVEGACIÓN AUTÓNOMA DE DOS MINI-DRONES Y RECONSTRUCCIÓN DE OBJETOS 3D**" realizado por los señores ***CEVALLOS REA JORDY GABRIEL y LOZA ALBUJA CHRISTIAN SANTIAGO***, ha sido revisado en su totalidad, analizado por la herramienta de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustenten públicamente.

Sangolquí, 01 de julio del 2019

Ing. Andrés Erazo M.Sc.

C.C: 1720400082

AUTORÍA DE RESPONSABILIDAD

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORÍA DE RESPONSABILIDAD

Nosotros, ***CEVALLOS REA JORDY GABRIEL*** *y* ***LOZA ALBUJA CHRISTIAN SANTIAGO***, declaramos que el contenido, ideas y criterios del trabajo de titulación "***DISEÑO E IMPLEMENTACIÓN DE UN CONTROL INTELIGENTE PARA NAVEGACIÓN AUTÓNOMA DE DOS MINI-DRONES Y RECONSTRUCCIÓN DE OBJETOS 3D***" es de nuestra autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de las Fuerzas Armadas ESPE, respetado los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Consecuentemente el contenido de la investigación mencionada es veraz.

Sangolquí, 01 de julio del 2019

CEVALLOS REA
JORDY GABRIEL
C.C. 1003827126

LOZA ALBUJA
CHRISTIAN SANTIAGO
C.C. 1717267650

AUTORIZACIÓN

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

AUTORIZACIÓN

Nosotros, ***CEVALLOS REA JORDY GABRIEL*** *y* ***LOZA ALBUJA CHRISTIAN SANTIAGO***, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación ***"DISEÑO E IMPLEMENTACIÓN DE UN CONTROL INTELIGENTE PARA NAVEGACIÓN AUTÓNOMA DE DOS MINI-DRONES Y RECONSTRUCCIÓN DE OBJETOS 3D"*** cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Sangolquí, 01 de julio del 2019

CEVALLOS REA
JORDY GABRIEL
C.C. 1003827126

LOZA ALBUJA
CHRISTIAN SANTIAGO
C.C. 1717267650

DEDICATORIA

Dedico este logro a toda mi familia quienes de una u otra forma siempre me han apoyado para salir adelante y continuar frente a las adversidades hasta llegar a la meta, de manera muy especial a mi tía Gabriela Rea quien ha sido un ejemplo de esfuerzo, dedicación y perseverancia; a mi hermana quien ha sido mi principal fuente de inspiración, mi confidente, mi mejor amiga y por quien deseo seguir cosechando triunfos.

De sobremanera quiero dedicar este triunfo a mi madre Rebeca Rea quien con su inmenso amo, dedicación y enseñanzas ha sido el pilar fundamental de mi formación.

Jordy Gabriel Cevallos Rea

Al culminar esta etapa de titulación universitaria, este logro va dedicado con mucho cariño a mis padres Wilson y Guadalupe por su amor, trabajo y sacrificio a lo largo de mi vida, por guiarme y no dejarme caer en los momentos difíciles, ellos han sido la principal motivación para seguir adelante.

A mis hermanos Lupita y Wilson que siempre mostraron preocupación hacia mí con su apoyo incondicional, siendo ejemplo de superación.

Christian Santiago Loza Albuja

AGRADECIMIENTO

En primer lugar, quiero agradecer a Dios que me ha bendecido con salud, una familia extraordinaria y la determinación de cumplir mis objetivos.

Quiero agradecer a mi madre por estar siempre para mi inculcándome valores, por ser siempre esa guía corrigiendo mis pasos, por brindarme el apoyo incondicional, por su preocupación, por darme la posibilidad de cumplir mis metas y sobre todo por darme ese inmenso amor a lo largo de toda mi vida. Gracias por esto y mucha más mamita.

A mi hermana por estar siempre pendiente de mí, por enseñarme a mirar la vida diferente, por ser esa persona por quien me levanto día a día con ganas de superarme para poder cumplir un poco de lo que esperas de mí. Gracias por comprenderme y brindarme tu amor siempre hermanita.

A mi tía por ser una persona admirable en todo sentido de la palabra, por ser esa fuente de consejos y mano amiga que no duda un solo instante en entregar todo por otra persona por apoyarme con mis proyectos, decisiones y sobre todo por ser una segunda madre.

A mis tíos y primos, por ayudarme y darme esa palabra de aliento cuando lo he necesitado.

A mis amigos, compañeros de Universidad gracias por compartir tantos momentos inolvidables que llevare siempre presentes.

Para terminar, quiero agradecer a todos mis docentes de la escuela, colegio y Universidad de Fuerzas Armadas “ESPE”; por todos los conocimientos inculcados en mi a lo largo de mi formación académica; a nuestro director de tesis Ing. Andrés Erazo por ser el principal colaborador y guía para que este proyecto culminara con éxitos.

Jordy Gabriel Cevallos Rea

Doy gracias a Dios por brindarme la oportunidad de alcanzar mis metas.

A mis padres por el apoyo incondicional y su entrega infinita, por velar que nada me faltase e inculcar en mí el valor del trabajo.

A la Universidad de las Fuerzas Armadas ESPE y a los docentes que contribuyeron en mi formación profesional, de manera especial al tutor de tesis, Ing Andrés Erazo, quien nos ayudó con el desarrollo del proyecto.

A los amigos que he conocido en este tiempo, que de una u otra manera estuvieron a lo largo de este camino.

Christian Santiago Loza Albuja

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

GLOSARIO DE TÉRMINOS

API	Interfaz de programación de aplicaciones
DHCP	Dynamic Host Configuration Protocol
IP	Protocolo de Internet
IMU	Unidad de medición inercial
MVO	Registro de vuelo
PS4	PlayStation 4
Pycloud	Librería de Python para nube de puntos
ROS	Sistema operativo robótico
ROSCORE	Colección de nodos y programas de ROS
RViz	Visualizador de ROS
SAR	Servicio Aéreo de Rescate
SDK	Kit de desarrollo de software
SGBM	Algoritmo de coincidencias de semibloques
SLAM	Localización y mapeo simultaneo
Sockets	Un socket se define como el punto final en una conexión
UAV	Vehículo aéreo no tripulado
WLS	Filtro de mínimos cuadrados ponderados

RESUMEN

El presente trabajo de titulación tiene como objetivo la navegación de mini-drones para realizar una representación 3D de un objeto basado en fotografías de cámaras monoculares emulando el comportamiento de una cámara estéreo. En el desarrollo del presente proyecto se observó que el uso de un solo mini-dron logra mediante la toma de fotografías desplazadas, y con un procesamiento de fotografías posterior al vuelo, una reconstrucción del entorno deseado. Para lograr que el dron rodee al objeto tomando fotografías de forma autónoma se diseñó e implementó un sistema de estabilidad de vuelo fuzzy el cual trabaja en cascada con un control de trayectoria. Adicionalmente se optimizó el número de fotografías almacenadas mediante el reconocimiento de objetos. Para el control y supervisión por parte del usuario del sistema se desarrolló una interfaz gráfica y una visualización de trayectoria en el entorno RViz. La reconstrucción 3D del objeto se basó en el cálculo de disparidad entre dos fotografías consecutivas y la adición de profundidad a la imagen del objeto.

PALABRAS CLAVES:

- **CONTROL FUZZY**
- **NAVEGACIÓN AÉREA**
- **CÁMARA MONOCULAR**
- **DRON**
- **PROCESAMIENTO DE IMÁGENES**

ABSTRACT

The objective of the present degree work is the navigation of mini drones to make a 3D representation of an object based on photographs of monocular cameras emulating the behavior of a stereo camera. In the development of the present project, it was observed that the use of a single mini drone achieves a reconstruction of the desired environment through the taking of displaced photographs, and with a post – flight photograph processing. In order to make the drone surround the object and take pictures autonomously, a fuzzy flight stability system was designed and implemented, this system works in cascade with a trajectory control. Additionally, the number of stored photographs was optimized by object recognition. For the control and supervision by the user of the system, a graphical interface and a path visualization were developed in the Rviz environment. The 3D reconstruction of the object was based on the calculation of disparity between two consecutive photographs and the addition of depth to the image of the object.

KEYWORDS:

- **FUZZY CONTROL**
- **AERIAL NAVIGATION**
- **MONOCULAR CAMERA**
- **DRONE**
- **IMAGE PROCESING**

CAPÍTULO I

INTRODUCCIÓN

1.1 Antecedentes

La historia de los UAVS (Vehículo aéreo no tripulado por sus siglas en inglés) comenzó hace casi un siglo, y la era moderna de los UAV se remonta a casi cuatro décadas; debiendo destacar entonces que a menudo pasó desapercibido el hecho de que el desarrollo y el empleo de UAV están siendo perseguidos por más de 50 países en todo el mundo. Varias tendencias se pueden ver en las operaciones de UAV, donde estos están avanzando continuamente como la historia ha demostrado. Esto incluye el avance de las capacidades de carga útil y la incorporación de tecnología de baja observación, las capacidades de la aeronave en sí son cada vez mayores. En los primeros años de desarrollo de UAV, los vuelos de UAV duraron menos de 20 minutos. Ahora, los UAV pueden volar decenas de miles de millas y permanecer en el aire durante varios días.

Las tecnologías emergentes, como la inteligencia a bordo, las capacidades de "detección y evasión", las mejoras de SAR (Servicio Aéreo de Rescate), las imágenes multiespectrales, la detección de luz y la generación de imágenes por láser, los sistemas de detección nuclear, la autonomía del sensor, las fuentes de alimentación ligeras y eficientes, y el almacenamiento masivo de datos, denotan el futuro para UAVs. (Kendra, 2007)

En la actualidad, la popularidad del uso de drones para el ocio es principalmente para fotografía aérea y ha desembocado en el auge de una industria de “drones hobby” a precios muy asequibles. Otra categoría superior de drones profesionales o semiprofesionales a precios intermedios han sido orientados al sector de servicios como: agricultura de precisión, cartografía, minería, monitoreo de incendios, planificación urbanística y vigilancia, entre otros. Paralelo al avance técnico de los

drones y el incremento de aplicaciones y usuarios, se ha producido también un desarrollo considerable de software para manejar los sistemas más fácilmente y procesar los datos que se obtienen de modo más rápido y sencillo; en consecuencia, existe una gran variedad de drones, desde aeronaves de gran tamaño (denominados HALE o MALE UAVs), equipos medianos o pequeños que pueden transportarse en una maleta, y hasta micro UAVs que caben en la palma de la mano como consecuencia de la miniaturización tecnológica (Figura 1). En consonancia con la complejidad del sistema, el despliegue es más o menos complicado e implica desde la necesidad de aeropuertos para los grandes drones hasta la posibilidad de operar dentro de una habitación cerrada para los pequeños. (Mandujano, Mulero-Pázmany, & Rísquez-Valdepeña, 2017)

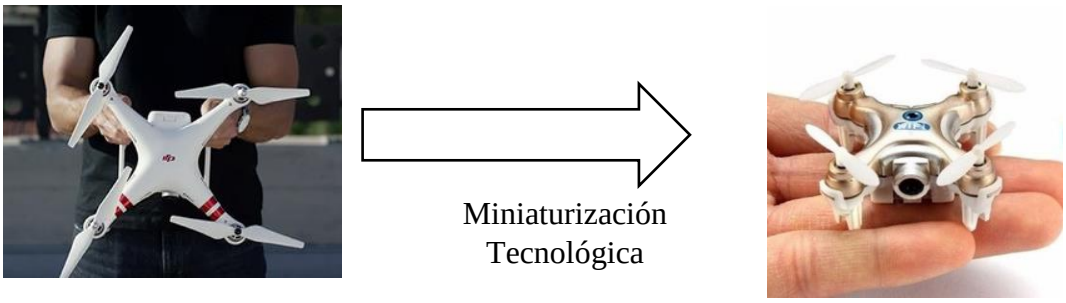


Figura 1. Miniaturización de la tecnología

En este momento, los drones forman una parte fundamental en la robótica la cual se ha inspirado en la naturaleza tomando en cuenta el comportamiento de las especies animales. Un comportamiento notable es la capacidad de cooperación para lograr un objetivo común; es así como se han desarrollado algoritmos para que los robots puedan cumplir con dicho objetivo. (Tayupanta, 2018)

Al mismo tiempo, para afrontar ciertas aplicaciones, la cooperación en grupos de varios robots se ha convertido en una necesidad, siendo una tendencia la investigación de sistemas que consideren la colaboración entre robots y sensores heterogéneos presentes en el entorno para

multitud de aplicaciones, como robótica de servicio en entornos urbanos, o monitorización de desastres. La razón fundamental es que estas aplicaciones involucran entornos dinámicos, con condiciones cambiantes para la percepción, entre otros. En la mayoría de las ocasiones, un único agente (por ejemplo, un robot o una cámara) no permite conseguir la robustez y eficacia necesarias; por lo que en estos casos, la cooperación de diferentes agentes (robots, sensores en el entorno, drones) puede ser muy relevante. (Archimowicz & Martinez, 2011)

Para lograr la navegación de robots se hace necesario controlar variables como: posición y rotación, mediante el uso de un sistema de control que involucre inteligencia, más aún cuando los robots operan de forma cooperativa. La inteligencia desde un punto de vista teórico es la que integra realimentación y conocimiento en el proceso de planificación y generación de acciones que conlleven alcanzar los objetivos de control; por lo tanto, el término inteligente está estrechamente ligado al de autónomo, ya que un sistema inteligente por lo general implica tener un alto grado de autonomía. Esencialmente, el control inteligente se enfoca en abordar el problema de control con técnicas de la inteligencia artificial, control lógico difusa; las cuales tratan de emular las funcionalidades inteligentes de los seres vivos en concreto, el proceso del razonamiento humano y seguir un conjunto de reglas basadas en experiencias respectivamente. La idea nace por la necesidad de encontrar e implementar mejores y más sofisticadas soluciones en el área del control, especialmente en aquellos sistemas complejos donde ciertos requerimientos no pueden ser alcanzados con el control convencional. (Ochoa, 2018)

Como consecuencia de la navegación robótica en entornos con obstáculos surgió un tema importante de investigación en la visión por computadora, siendo este la reconstrucción de objetos tridimensionales a partir de imágenes bidimensionales, el principal objetivo es utilizar las

transformaciones proyectivas de un par de imágenes para determinar mapas de profundidad (Tayupanta, 2018).

Para ello se debe seguir una secuencia de pasos que son: se determina puntos de interés entre el par de imágenes, posteriormente se busca correspondencia de los puntos, y finalmente se utiliza las matrices de rotación para describir el movimiento entre las imágenes para su posterior reconstrucción (Grandón-Pastén, Aracena-Pizarro, & Tozzi, 2007).

La capacidad de construir un mapa del entorno y localizarse en él es esencial para que un robot sea considerado como autónomo; por tal razón, el problema de localización y mapeado simultáneos "SLAM" (Simultaneous Localization and Mapping) ha sido estudiado con gran interés en los últimos años y llevado a la práctica en diversas aplicaciones tales como tareas de rescate, vehículos aéreos, o robots guías en museos.

Existen varios métodos de detección de puntos de interés como son (Ballesta, Gil, Reinoso, & Úbeda, 2010):

- Harris Corner Detector: El detector de esquinas de Harris es uno de los detectores de puntos de interés más comunes. El punto característico viene dado por aquel píxel que tenga los valores propios elevados en el cálculo de la matriz de momentos de segundo orden. Los puntos de Harris se utilizan para extraer marcas visuales en SLAM monocular.
- Harris-Laplace: Los puntos extraídos con Harris-Laplace son detectados mediante una función de Harris adaptada a escala y se seleccionan en el espacio de escalas por un operador Laplaciano.

- SUSAN: El detector SUSAN (Smallest Univalue Segment Assimilating Nucleus) funciona aplicando una máscara circular sobre el píxel. Un píxel es punto característico en función del número de píxeles similares al píxel central situados dentro de la máscara. SUSAN ha sido utilizado tradicionalmente en aplicaciones de reconocimiento de objetos.
- SIFT: El algoritmo SIFT (Scale-Invariant Feature Transform) es un detector y descriptor de puntos característicos. En este apartado nos centramos en SIFT como detector. Los puntos característicos son detectados en las imágenes mediante la función DoG (Difference of Gaussian) aplicada en el espacio de escala. Los puntos se seleccionan como los extremos locales de la función DoG. El algoritmo utilizado en tareas de reconocimiento de objetos. Recientemente, SIFT se ha utilizado en aplicaciones de SLAM visual.
- SURF: Las características del SURF (Speeded Up Robust Features) superan, según sus autores, a otros métodos existentes en términos de robustez, repetitividad y distinción de los descriptores. El método de detección está basado en la matriz Hessiana y depende de imágenes integrales para reducir el coste computacional. Al igual que con las características SIFT, nos centramos sólo en los puntos detectados cuando se evalúa como detector.

En la robótica es muy utilizado el software RVIZ. Este es un visualizador 3D de datos muy completo que nos brinda ROS que tiene como misión principal la de ser observador; por lo que este paquete de software libre permite ver el robot y responde en tiempo real a lo que le ocurre en el mundo real, generando datos tridimensionales que van desde modelos de robot accionados hasta

imágenes detectadas y nubes de puntos como se observa en la Figura 2. RVIZ se puede usar para mostrar lecturas de sensores, datos devueltos por la visión estereoscópica, hacer SLAM evitando obstáculos y además nos permite la incorporación de varios elementos de visualización como cámaras, sensores laser, etc. Proporcionando de esta manera una información muy necesaria sobre el estado y la visión del entorno robótico (Figura 2).

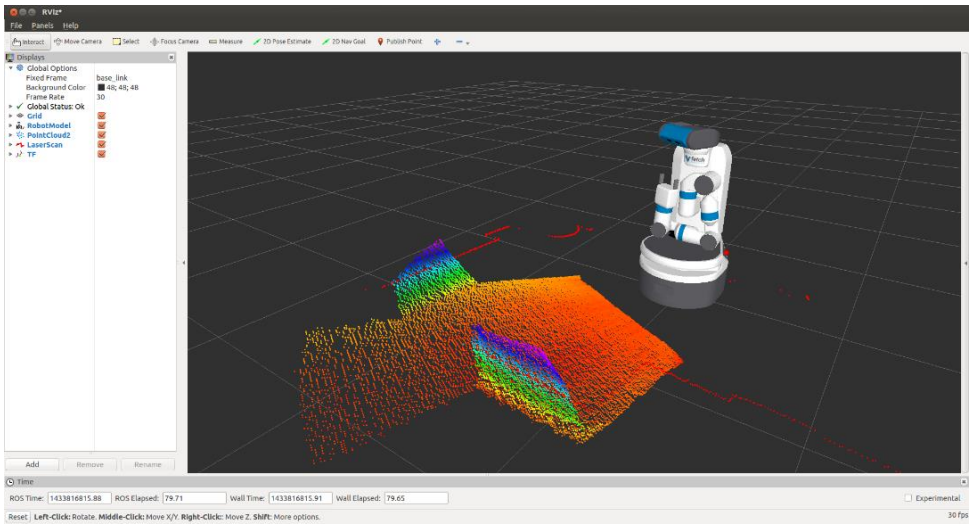


Figura 2.RViz, software libre para reconstrucción 3D mediante ROS

En la tesis de Tayupanta se realizó la comparación y análisis de descriptores aplicada a estereografía con lo cual se obtuvo un modelo aplicable a la reconstrucción 3D, por lo cual el presente proyecto se enfocará en la aplicación del mismo para la reconstrucción de objetos y control de navegación.

1.2 Justificación e importancia.

Es conocido el uso de vehículos terrestres para la obtención de mapas mediante el uso de las técnicas SLAM. Estos vehículos en ocasiones no proporcionan toda la facilidad cuando se desea

hacer un mapeo de un objeto que se encuentre frente a ellos ya que existen zonas inaccesibles o a su vez su construcción mecánica inhabilita ciertos movimientos para el seguimiento del objeto, por lo que ha crecido el uso de vehículos aéreos para la obtención de mapas usando diferentes técnicas. Del mismo modo, se puede utilizar estos vehículos para la obtención de imágenes de los objetos logrando reconstruirlos.

Según Vásquez, la investigación de nuevas técnicas como Héctor-Mapping para el mapeo de objetos y su reconstrucción 3D se ha visto necesaria para mejorar cada vez más el reconocimiento de entornos mediante el uso de estas plataformas (Vásquez, 2015). A su vez Tayupanta en su trabajo presenta la reconstrucción de entornos a través de cámaras monoculares mediante el algoritmo de redes neuronales convolutivas logrando la detección de objetos mas no una buena reconstrucción de los mismos por lo que recomienda el uso de otros algoritmos para realizar la reconstrucción (Tayupanta, 2018).

El uso de plataformas robóticas en la actualidad es tan cotidiano que algunas de ellas son imprescindibles en muchas aplicaciones, pero en varias ocasiones el uso de un único robot no puede lograr cumplir el objetivo por lo cual se utiliza la robótica colaborativa en donde dos o más robots se complementan para lograr la meta. En este proyecto se utilizará dos drones ya que cada uno de estos tienen cámaras monoculares y en conjunto estas dos cámaras nos permiten una visión estereoscópica.

El SLAM se aplica para resolver el problema de autonomía de robots móviles en entornos desconocidos para generar un mapa y el posicionamiento del robot mediante la combinación de técnicas de visión y algoritmos matemáticos, permitiendo que el robot pueda trabajar en tiempo real sobre el terreno, actualizando el mapa disponible y adecuando la trayectoria según su posición

en el mismo. Esta capacidad brinda a la robótica móvil la versatilidad requerida en diversas aplicaciones robóticas como misiones de exploración, rescate y mapeado.

Los mapas tridimensionales de entornos u objetos proporcionan al robot la completa información sobre el terreno en el que se mueve o a su vez sobre los objetos que tiene como obstáculos permitiendo tomar acciones sobre los mismos para cumplir la misión determinada. A su vez, la reconstrucción de objetos reales 3D tiene una atención significativa no solo como visión artificial sino como herramienta para diversas aplicaciones como: medición, fabricación, arqueología, robótica; logrando modelar a partir de imágenes imitando la capacidad de los seres humanos de observar un objeto en 3D cuando la imagen es en 2D.

El presente proyecto pretende mejorar los resultados obtenidos en investigaciones previas como es el trabajo de Tayupanta: "Navegación Autónoma de Mini-drones para Multirobots SLAM 3D en Entornos Estáticos con Algoritmos de Inteligencia de Enjambre" en el cual utiliza 3 mini-drones para realizar el SLAM 3D y una cámara exterior para la localización de los mini-drones, proyecto en el cual no se enfocó en la navegación. Se minimizará el uso de recursos utilizando la robótica cooperativa entre dos mini-drones con cámaras monoculares con una trayectoria basada en la localización referente a un objeto y un vuelo autónomo alrededor del mismo teniendo en cuenta que dichos drones volarán de forma paralela.

1.3 Alcance del proyecto

El proyecto tiene como finalidad la navegación de dos mini-drones alrededor de un objeto para realizar la reconstrucción del objeto en 3D. Entendiéndose en este proyecto como navegación al movimiento de los drones sin oscilaciones (estabilidad) y un desplazamiento circular (trayectoria).

Para lograr este fin se utilizará dos mini-drones Cheerson Cx-10w (Figura 3), estos permiten su control mediante WIFI por una aplicación Android; además, incorpora una cámara monocular de 0.3MP permitiendo su vuelo en primera persona y por último incorpora un giroscopio de 6 ejes. De ser necesario se utilizará drones con mejores características.

Figura 3.Mini-drone Cheerson Cx-10w.

Para lograr la estabilidad en la navegación se realizará un control que permita que el dron se eleve y mantenga una posición fija, manipulando los ángulos: Pitch, Yaw y Roll. El esquema de control (Figura 4).

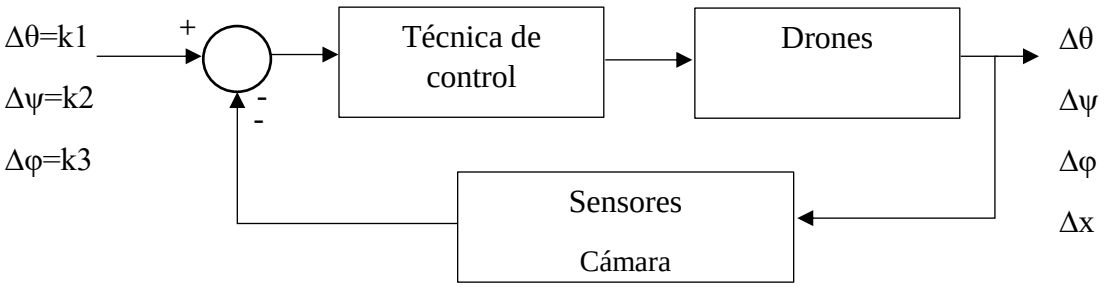


Figura 4.Diagrama de lazo de control para estabilidad

La técnica de control inteligente utilizada para la estabilidad del dron será fuzzy, redes neuronales u otra de este tipo que nos permita realizar un control sin un modelamiento matemático previo del dron.

Con la obtención previa de un vuelo estable se desarrollará un controlador de trayectoria que permita el vuelo paralelo de los dos drones, este es a una distancia constante y con un mismo desplazamiento; y una trayectoria circular, con un movimiento de traslación alrededor del objeto y un movimiento de rotación para que la cámara del dron siempre enfoque al objeto (Figura 5).

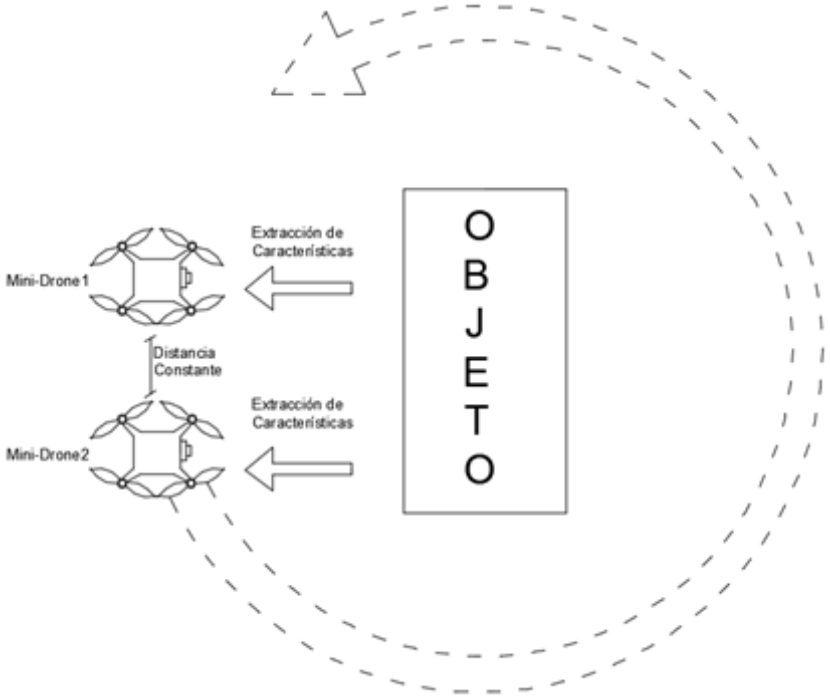


Figura 5.Diagrama de trayectoria

La técnica de control de trayectoria será al igual que en el control de estabilidad: fuzzy, redes neuronales u otra de este tipo; y permitirá manipular el espacio articular del dron (roll, pitch, yaw) para lograr que el dron se localice y se oriente en su espacio operacional (x, y, z).

En primera instancia se intentará realizar un control a lazo abierto que modificará en diferentes tiempos los ángulos (roll, pitch, yaw) permitiendo que el dron realice la trayectoria circular. Si la precisión del controlador en lazo abierto no es adecuada para la aplicación deseada, se

implementará una cámara externa para poder realizar la retroalimentación y corregir la trayectoria de los drones. Dicha cámara generaria un plano en “x” e “y” localizando a los drones en dicho plano mediante el uso de códigos ArcUco. Estos códigos a su vez nos permiten ver la localización en “z” mediante sus dimensiones, esta información en “x”, “y”, “z” será la realimentada para el control de trayectoria.

Para el trabajo de los dos controladores, estabilidad y trayectoria, se probarán dos métodos: en cascada (Figura 6), en el cual la salida del control de estabilidad será la entrada del control de trayectoria; y (pseudo) paralelo (Figura 7), en el cual los dos controladores trabajan de forma independiente en su lazo de control y a su vez de forma simultánea con tiempos de muestreo diferentes.

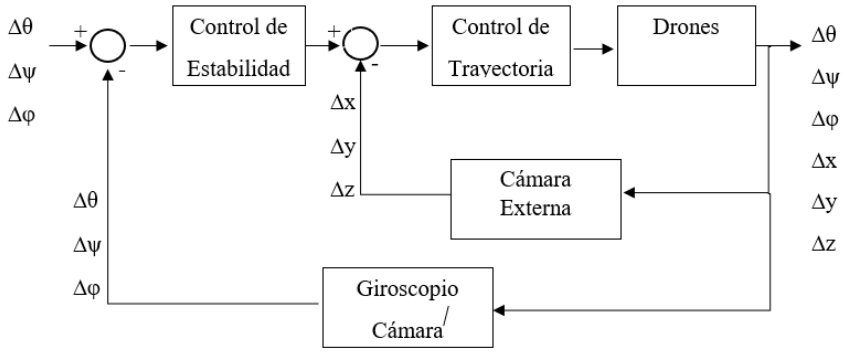


Figura 6. Diagrama de lazo de control en cascada

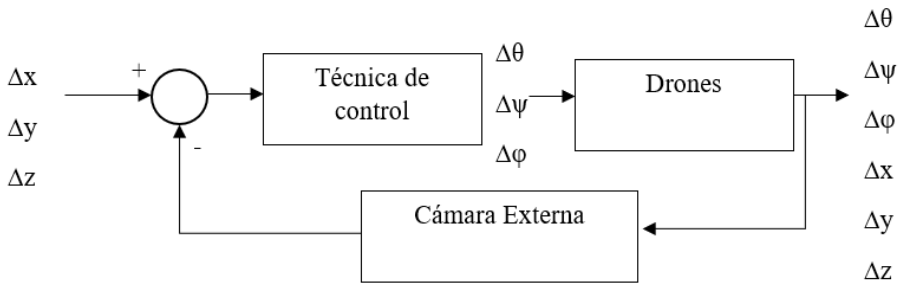


Figura 7. Diagrama de lazo de control pseudo-paralelo

Para el control de vuelo, así como la comunicación entre drones y PC se utilizará una tarjeta Raspberry Pi3 (Figura 8) por cada dron (Figura 9), ya que esta tiene incorporado módulos WIFI, Ethernet y otros que ayudan al enlace de los dispositivos requeridos en este proyecto y a su vez soporta sistemas operativos de software libre.

Figura 8.Tarjeta Raspberry Pi3

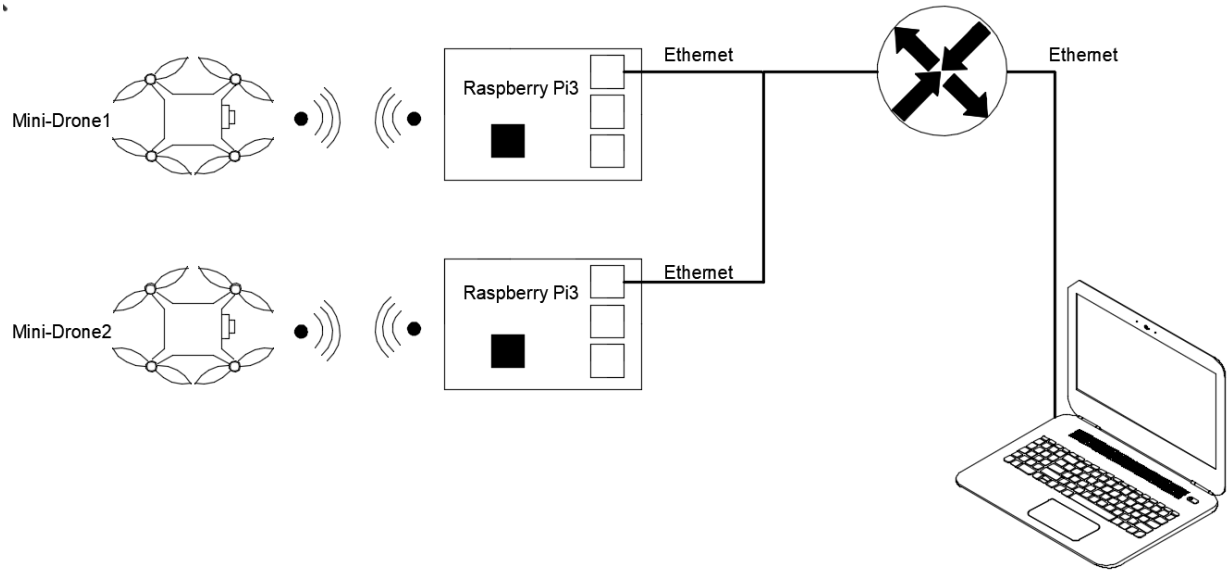


Figura 9.Configuración de la comunicación

Para cumplir con el objetivo de la detección y reconstrucción de objetos (Figura 10), se implementará una técnica de reconstrucción basada en el análisis de descriptores aplicada a estereografía en un entorno de trabajo ROS (Robot Operating System) y su herramienta de visualización y reconstrucción 3D RVIZ.

Se diseñará y desarrollará una interfaz en el computador para dar el inicio de vuelo, la visualización de las imágenes obtenidas por las cámaras y la reconstrucción 3D.

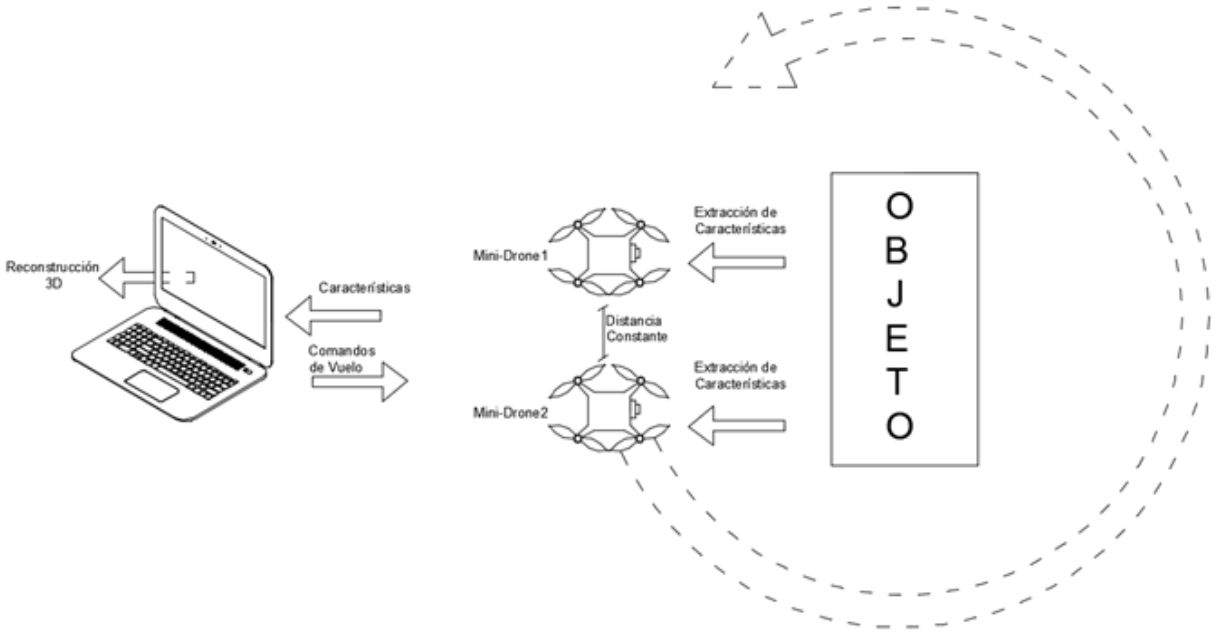


Figura 10. Diagrama de funcionamiento

Una vez estén todos los dispositivos funcionando, se llevarán a cabo las diferentes pruebas y se realizarán las diferentes comparaciones del objeto y su reconstrucción 3D.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar e implementar un sistema de navegación cooperativa de dos mini-drones alrededor de un objeto para realizar la reconstrucción del mismo en 3D.

1.4.2 Objetivo específico

- Implementar una técnica de control inteligente para la estabilidad de vuelo de los mini-drones.
- Implementar una técnica de control inteligente que permita el control de trayectoria alrededor de un objeto estático.
- Aplicar un modelo de reconstrucción 3D basado en el análisis de descriptores aplicada a estereografía.
- Elaborar una interfaz que permita dar el inicio de vuelo y observar la reconstrucción 3D realizada.
- Realizar un análisis de resultados y las pruebas físicas de funcionamiento.

CAPÍTULO II

FUNDAMENTOS TEÓRICOS

2.1 Dron Tello

El dron Tello (Figura 11) es fabricado por la empresa Ryze Tech con tecnología DJI e Intel, este es considerado un mini dron por sus pequeñas dimensiones y es ideal para volar en interiores. Cuenta con un control de vuelo de forma remota, utilizando wifi o bluetooth, mediante una aplicación para smartphones disponible para Android e iOS (RYZE, 2019).

Figura 11. Dron Tello de Dji

Este dron cuenta además con un módulo IMU (Unidad de Medición Inercial), que internamente consta con un giroscopio y un acelerómetro, además tiene en su exterior un sensor óptico e infrarrojo los cuales al volar la aeronave son utilizados. En condiciones de vuelo normal la aeronave utiliza el sensor óptico y cuando este falla cambia a modo attitude en el cual utiliza el sensor infrarrojo.

La aplicación (Figura 12) para el control de vuelo permite manejar el dron en primera persona gracias a la visualización de su cámara frontal de 5MP además de contar con despegue y aterrizaje autónomo, piruetas predefinidas y captura de fotos o videos.

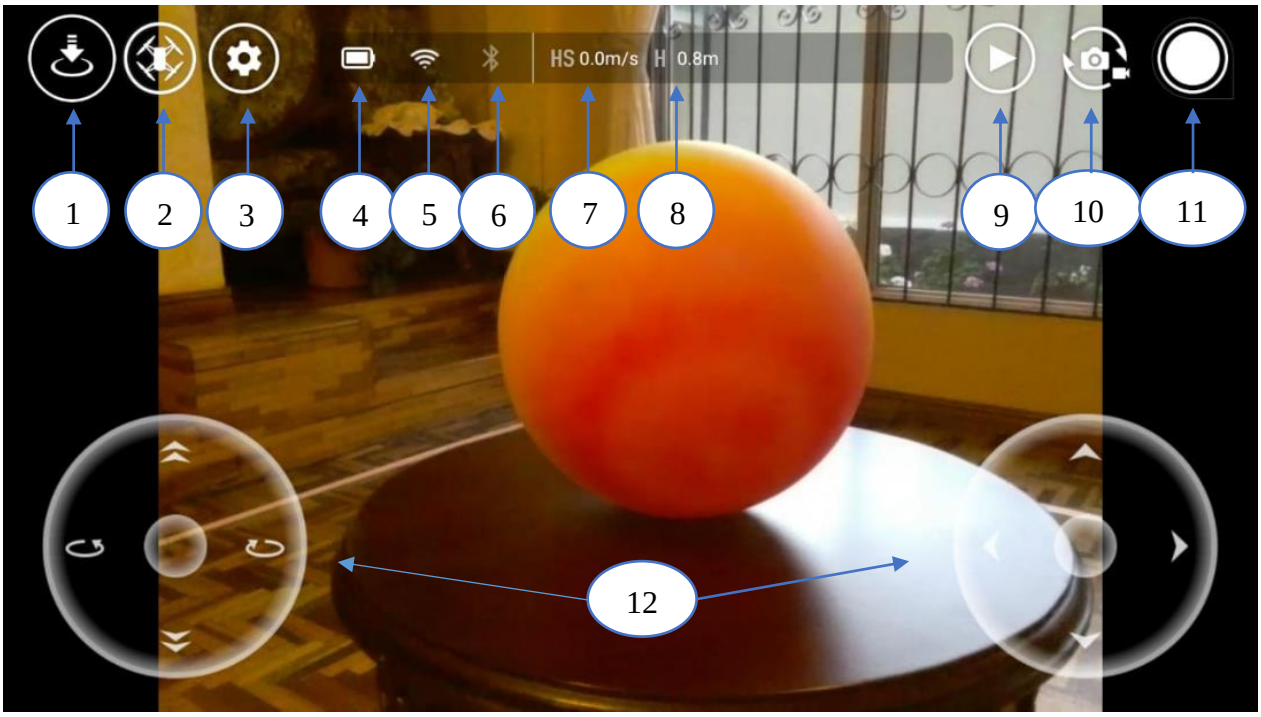


Figura 12. Aplicación móvil Tello

Esta aplicación cuenta con las siguientes funciones (Figura 12):

1. Despegue/aterrizaje automático
2. Modo de vuelo inteligente
3. Ajustes
4. Nivel de batería
5. Estado de Wi-Fi
6. Estado de Bluetooth
7. Velocidad de vuelo
8. Altitud de vuelo
9. Reproducción
10. Cambio entre foto y video

11. Botón de disparo/grabación

12. Palancas de control virtuales

2.1.1 Características técnicas

Las características técnicas del mini-dron se presentan a continuación en la Tabla 1.

Tabla 1
Especificaciones técnicas

Dimensiones	9,8x9,2x4,1 cm
Peso	87 g
Hélices	7,62 cm
Funciones incorporadas	Funciones incorporadas: buscador de rango, barómetro, LED, sistema de visión, Wi-Fi 802.11n de 2.4 GHz, vista en vivo de 720p
Cámara	5 MP
Máxima distancia de vuelo	100 m
Máxima velocidad	8 m/s
Tiempo de vuelo	13 min
Batería	1100 mAh 3.8V

Fuente: (RYZE, 2019)

2.2 Entorno de desarrollo para aplicaciones robóticas ROS

ROS es un acrónimo de Robot Operating System. Este es un software abierto utilizado para plataformas robóticas con herramientas, librerías, visualizadores, comunicación predefinidas para varios robots disminuyendo la dificultad para representar comportamientos robóticos por la reutilización de código y el desarrollo colaborativo que este sistema permite (Santos, Portugal, & Rocha, 2013).

Cuenta con varias distribuciones las cuales son paquetes que vienen con versiones diferentes, estas son (Distributions, 2018):

- ROS Box Turtle.
- ROS C Turtle
- ROS Diamondback

- ROS Electric Emyx
- ROS Fuerte Turtle
- ROS Groovy Galápagos
- ROS Índigo Igloo
- ROS Jade Turtle
- ROS Kinetic Kame
- ROS Lunar Loggerhead
- ROS Melodic Morenia

Estas se encuentran en orden ascendente por lo que la distribución más actual es la Melodic y es con la que se desarrolla el presente proyecto, para esto se necesita la versión de Ubuntu más actualizada y se utiliza la versión 18.04 LTS.

2.2.1 ROS Kinetic

La distribución Kinetic Kame de ROS, fue lanzada en mayo del 2016, está dirigida para la versión 16.04 de Ubuntu y otros sistemas operativos como Windows y OS X (Morenia, 2018). Esta distribución es utilizada para trabajar con un nodo central y poder compartir información de los nodos con diferentes distribuciones de ROS (Cashmore, y otros, 2015). Permite trabajar con la rapberry como un nodo del sistema.

2.2.2 ROS Melodic

La distribución Melodic Morenia de ROS es la más actual hasta el momento. Fue lanzada en mayo del 2018, y está dirigida para la versión 18.04 de Ubuntu y otros sistemas operativos como Windows y OS X (Morenia, 2018). Esta distribución es utilizada en el nodo central del sistema, es decir la PC, la cual manejara nodos como la cámara externa y parámetros del controlador.

Todas las distribuciones de ROS trabajan con un entono modular y distribuido, por lo que se tiene la ventaja de crear varios paquetes para que estos trabajen de forma simultánea; permitiendo el envío y recepción de mensajes diferentes en cada paquete de cada nodo (Cashmore, y otros, 2015).

2.3 Funcionamiento ROS

El funcionamiento de ROS se basa en cuatro recursos: Paquetes, Nodos, Tópicos y Servicios; propios de cada programa, y a su vez cuenta con un nodo master.

2.3.1 ROS máster

ROS tiene como uno de sus objetivos la ejecución de pequeños programas independientes y para esto necesita la comunicación entre diferentes nodos; por lo cual utiliza un nodo central o nodo master que permite la interacción entre los diferentes nodos. Para la ejecución de este nodo ROS master se utiliza el comando: roscore. Este nodo no toma ningún argumento y no necesita ser configurado por lo que solo necesita de la instalación previa de alguna distribución de ROS (O'Kane, 2018).

2.3.2 Paquetes

Un paquete de ROS es una colección coherente de archivos que contiene archivos ejecutables y archivos de soporte destinados a un propósito específico. Cada paquete está definido por un documento que es el archivo package.xml; manifiesto en el cual se encuentra detalles como: nombre, versión, mantenedor y dependencias (O'Kane, 2018).

2.3.3 Nodos

Los nodos son archivos ejecutables, en los que se lleva a cabo cálculos dentro y estos utilizan una biblioteca tipo cliente para realizar dos funciones: publicar o suscribir mensajes entre nodos. Para ejecutar cualquier nodo se utiliza el comando: rosrun “nombre del package” “nombre del

ejecutable". Sin embargo, el registro para transformase en nodo se encuentra en el mismo programa y no en la función rosrun; por lo que se podría correr el ejecutable del nodo ingresando en la carpeta en la que se encuentra y daría el mismo resultado (O'Kane, 2018).

2.3.4 Tópicos y mensajes

El mecanismo que utiliza ROS para la comunicación es enviar mensajes. Estos mensajes se organizan en los llamados Tópicos. Por ende, para la comunicación se necesita que un nodo sea publicador (Publisher) y escriba mensajes acerca de un tópico; el nodo que recibe el mensaje es un nodo subscrito (Subscribe) al tópico (O'Kane, 2018). El tópico es la línea de comunicación entre nodos que permiten el envío y recepción de mensaje entre estos **(*Figura 13*)**; esto se puede observar mediante el comando: rqt_graph.

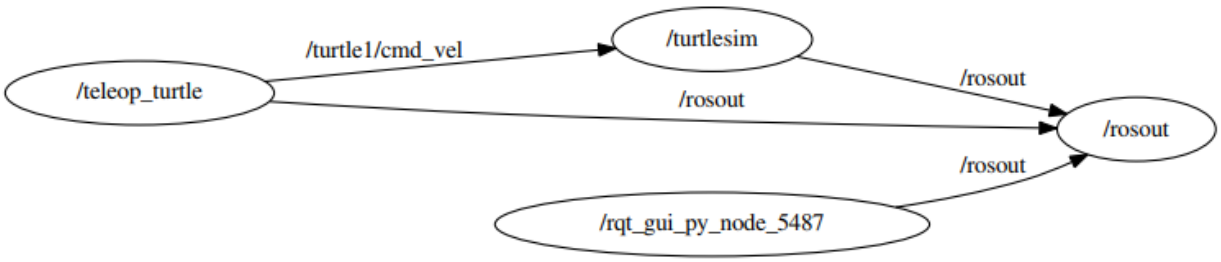


Figura 13. Herramienta rqt para ejemplo turtle_sim ROS.

En donde los nodos del que parte la flecha son nodos publicadores y los nodos al donde llegan las flechas son nodos subscriptores; el nombre de la flecha representa el tópico mediante el cual se están enviando los mensajes. Los nodos son representados por elipses.

2.3.5 Servicios

Los servicios son un tipo de arquitectura de ROS mediante el cual se envían dos tipos de mensajes, uno para la solicitud y otro de respuesta, por lo cual es una comunicación bilateral, sincrónica y de uno a uno a diferencia de un tópico el cual puede tener una comunicación de varios nodos publicadores y varios nodos subscriptores (tipo broadcast) (Cazorla, 2013).

2.3.6 RViz

ROS ofrece la visualización en 3D a través de su software RViz. El uso de este programa proporciona una simulación del entorno deseado y opera en diferentes sistemas operativos. Permite de esta manera una multitud de opciones en la visualización y los elementos que son utilizados para esto son: la adición y extracción de elementos que están en la escena, presentando marcadores, sistemas de referencia, nubes de puntos PCL, entre otros. (Macanás, 2014).

Otro valor agregado de esta herramienta es al momento de proyectar las trayectorias, ya que posibilita advertir el trazado que efectuará el robot real antes que lo realice, permite ver la posición, estado de sensores, trazado del movimiento de un punto a otro, hacer SLAM evitando obstáculos y otros.

2.4 Tipos de control de vuelo

El dron o UAV es definido como un vehículo aéreo no tripulado, con un control remoto o un control autónomo. Este es recuperable al final de vuelo por lo que se dice que es reutilizable (Sellali & Allali, 2017). Los Drones tienen integrados una unidad de cámara de alta visión, múltiples cámaras compactas para control de vuelo, giroscopio, sensor infrarrojo, GPS y un procesador para procesar imágenes de video e información de los diferentes sensores, todo esto destinado para controlar el vuelo de los mismos. (Yamamoto & Uchida, 2018).

2.4.1 Control de vuelo mediante control remoto, o rutas de vuelo.

Los vehículos aéreos no tripulados consisten en un sistema de varias hélices para su elevación y vuelo. Para el intercambio de datos entre máquinas, algunos drones han optado por incorporar una red IP inalámbrica, GPS y cámaras por lo que su control remoto puede utilizar diferentes tecnologías como radio frecuencia, bluetooth o Wireless mediante PC o teléfonos inteligentes. (Yamamoto & Uchida, 2018)

Los drones tienen un sistema de control de vuelo que les permite realizar desplazamientos y maniobras en el aire. Dichos movimientos son controlados desde tierra por una persona o el vuelo es realizado mediante un plan de vuelo preestablecido previamente al despegue; por lo que los drones con este tipo de vuelo tienden a mantener estas rutas prestablecidas, aunque existan perturbaciones. Debido a esto, no son sistemas autónomos de vuelo que puedan tomar decisiones acerca de su trayectoria de vuelo en función de las nuevas condiciones o perturbaciones existentes. A su vez, tampoco existe la posibilidad de comunicación para realizar una nueva ruta mientras está en vuelo. (Lozano, 2015)

2.4.2 Control de vuelo autónomo

En la actualidad varios temas de investigación en diversos lugares del mundo han tratado de desarrollar maquinas voladoras, drones, capaces de realizar misiones tanto con intervención del ser humano como sin esta (Sellali & Allali, 2017).

El control autónomo de las aeronaves no tripuladas permite incorporar planes de vuelo dinámicos mediante la planificación de vuelo a un alto nivel. Para esto se utiliza protocolos y tecnología modernos de control de tal manera que el sistema autónomo permita a la aeronave realizar un vuelo inteligente que pueda reaccionar ante perturbaciones variantes, y como es un vuelo autónomo el dron no requiere una supervisión constante de una persona, pudiendo modificar la trayectoria con el objetivo de cumplir su misión (Lozano, 2015).

Existen varias técnicas utilizadas para realizar un control autónomo de los drones, las cuales pueden ser implementadas individualmente o de forma conjunta para mejorar el rendimiento en plantas complejas, estos tipos de control pueden ser:

- Control PID: Es un tipo de control clásico, comúnmente utilizado en varios drones comerciales por su facilidad de implementación ya que no requiere una caracterización

exacta de la planta. Es un tipo de control utilizado generalmente para la estabilidad, aunque también permite el seguimiento de trayectorias. Este control está compuesto por 3 partes: Proporcional (P) la que se relaciona directamente con el tiempo que se demora en llegar a la señal de referencia, Integral (I) que se relaciona con perturbaciones externas y Derivativa (D) relacionada con la predicción del siguiente estado del sistema llevando más rápido al valor referencial (Vázquez, 2016). El esquema de control se muestra a continuación en donde generalmente se implementa un control PID para cada movimiento del dron (Figura 14).

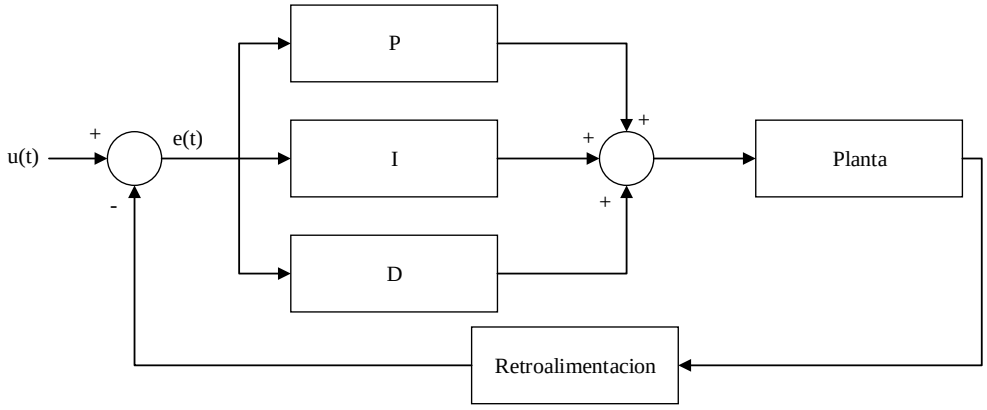


Figura 14. *Control PID*

Con la ecuación

$$u(t) = k_p e(t) + k_i \int^{t} e(\tau)d\tau + k_d \frac{de}{dt}$$

Donde:

u(t) es la referencia

e(t) es el error

kp es la constante proporcional

kp es la constante integral

kp es la constante derivativa

Como respuesta de un sistema P, PI, PD, PID se puede obtener varios comportamientos. En el peor de los casos un comportamiento inestable, el cual se da cuando la respuesta no tiene un valor definido y no se acerca al valor de referencia. De igual manera se puede tener un comportamiento estable en el que no siempre se llega al valor de referencia, pero si con una respuesta que oscila en torno a esta; por lo cual es una respuesta críticamente oscilante. Se puede obtener de igual manera una respuesta oscilatoria amortiguada en el que se logra corregir el error en un tiempo determinado. Y finalmente se puede obtener una respuesta amortiguada critica estable en la que no se produce oscilaciones y se llega al valor exacto de referencia (Palacios, 2009).

- Control Fuzzy: El control difuso o fuzzy es basado en el conocimiento de reglas y condiciones, por lo que es una técnica que busca expresar el conocimiento de tipo lingüístico cualitativo en lenguaje matemático basado en teoría de conjuntos difusos y sus funciones de pertenencia. Por esto es una técnica apropiada para problemas de comportamiento no lineal, logrando flexibilidad y excelentes resultados cuando los datos no son precisos (Palacios, 2009).

  El controlador difuso (Figura 15) se compone de cuatro bloques (Passino & Yurkovich, 1998):

  1. Base de reglas, contiene un conjunto de proposiciones lógicas de tipo “si-entonces”, que permiten establecer una relación entre entradas y salidas; cada una de estas es creada emulando la experiencia del operario en el proceso que se desea controlar.
  2. Un mecanismo de inferencia, que permite emular la toma decisiones basándose en el conocimiento del experto.

3. Una interfaz de fuzzificación, que convierte las entradas del controlador en información que el mecanismo de inferencia puede usar fácilmente para activar y aplicar reglas.
4. Una interfaz de defuzzificación, que convierte las conclusiones del mecanismo de inferencias en salidas reales del sistema de control.

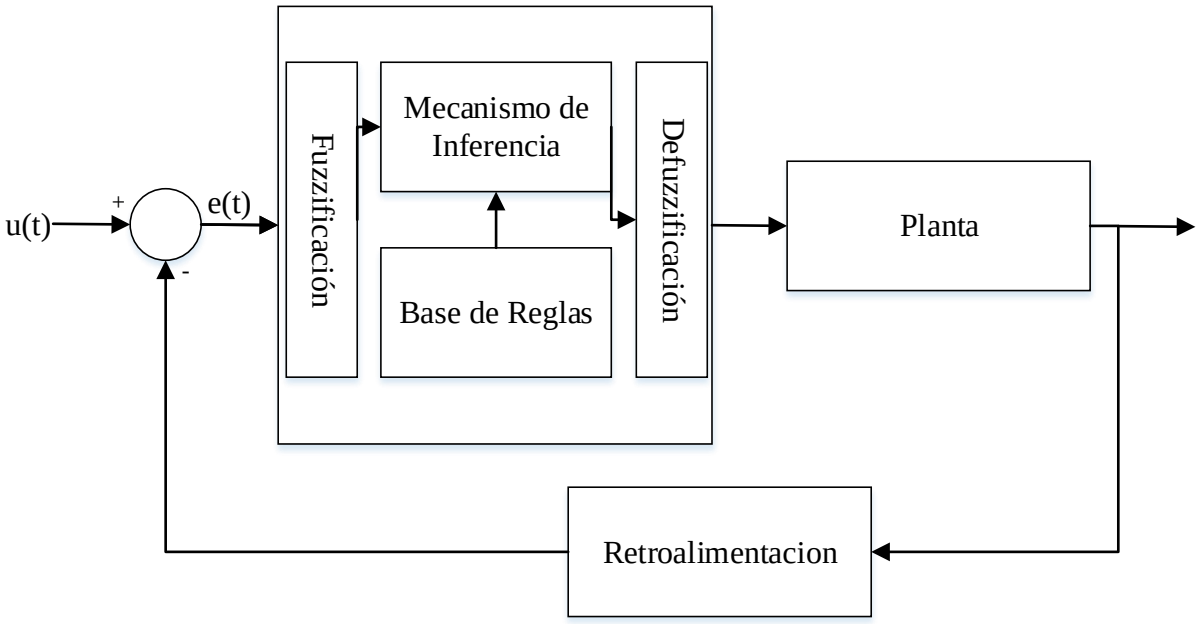


Figura 15. Control fuzzy

2.4.3 Derivada por aproximaciones polinomiales

Al querer obtener las variaciones del error (derivada del error) sin contar con una función matemática que defina el comportamiento de la planta, estas se realizan mediante aproximaciones polinomiales. Para esto se utiliza un conjunto de datos discretos y un punto en el cual se desea obtener su derivada. En la interpolación polinomial a más puntos en la evaluación de la derivada se obtiene una mayor exactitud.

Tabla 2
Fórmulas de derivación

Primera derivada	
Formula de diferencias finitas progresivas	Dos puntos: $f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}$ Tres puntos: $f'(x_0) = \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h}$
Formula de diferencias finitas centradas	Dos puntos: $f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}$ Tres puntos: $f'(x_0) = \frac{f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)}{12h}$
Formula de diferencias finitas regresivas	Dos puntos: $f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h}$ Tres puntos: $f'(x_0) = \frac{f(x_0 - 2h) - 4f(x_0 - h) + 3f(x_0)}{2h}$

2.5 Posición y orientación en el espacio (Rotación y traslación)

Las transformaciones de proyección de 3D a 2D emplean coordenadas homogéneas. Para puntos 2D cuenta con tres componentes y para puntos 3D es un vector de cuatro componentes (Gallardo, 2011):

2.5.1 Matrices de rotación

Las matrices de rotación muestran matemáticamente la rotación en un entorno de tres dimensiones tomando en cuenta el ángulo en el que se está girando (Figura 16) (Botero, 2005).

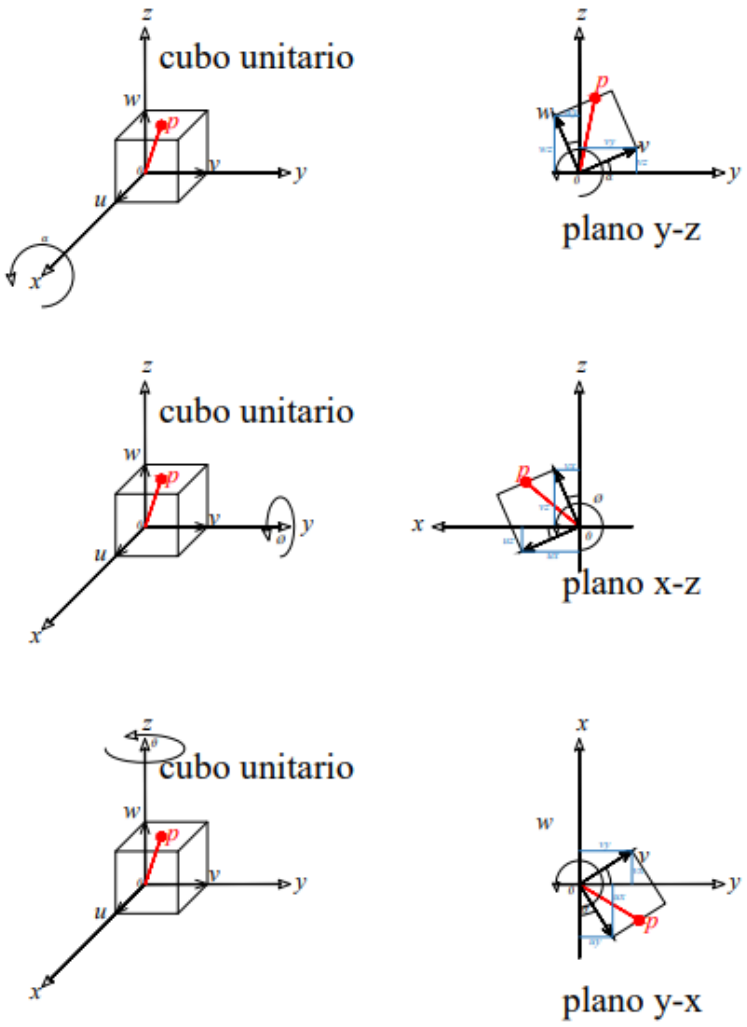


Figura 16. Rotación en ejes.

Cada matriz de rotación cumple con las siguientes propiedades:

- Ejes ortogonales
- Determinante igual a uno
- Normal de cualquier vector igual a uno, matriz unitaria.

La matriz de rotación es utilizada comúnmente en robótica dado que permite que especificado un ángulo un objeto rígido gire la misma cantidad de grados en una dirección. Esta se presenta de la siguiente manera:

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Dado que un objeto puede girar en torno a diferentes ejes se tiene dos marcos, un marco global el cual representa el punto donde comienza el robot o la posición inicial, y el marco referencial el que indica con respecto a que está girando. Por lo que para obtener las coordenadas de posición de un sistema se utilizan operaciones con las matrices de rotación. Las matrices de rotaciones básicas a un punto específico P(u,v,w) son (Botero, 2005):

$$R(x,\theta) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$R(y,\theta) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} * \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

$$R(z,\theta) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Estas pueden ser multiplicadas entre sí mediante una secuencia finita de giros, para determinar la rotación final respecto al sistema de referencia.

2.5.2 Ángulos de Euler

Los ángulos de Euler pueden mostrar la rotación y orientación mediante 3 ángulosϕ, θ, ψ; a diferencia de la matriz de rotación que requiere 9 elementos. Existen 2 formas (Garcia):

- Ángulos de Euler ZXZ; es una representación en la que se realizan giros sobre ejes previamente girados, es decir sobre un sistema nuevo.

- Roll, Pitch e Yaw; sistema utilizado en aeronáutica y robótica. En este se aplica los giros sobre los ejes del sistema fijo es decir XYZ. Dónde girar un ángulo ψ en torno a OX se denomina Yaw, girar un ángulo θ en torno a OY se denomina Pitch y girar un ángulo ϕ en torno a OZ se denomina Roll.

2.5.3 Matrices de traslación

La traslación permite mover objetos sin probar deformación en los mismos. Esto se da ya que cada punto del objeto se traslada a la misma dirección y la misma distancia. Esta es representada de la siguiente manera (ISA & Automatica):

$$Tras(p) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{x'} \\ p_{y'} \\ p_{z'} \\ 1 \end{bmatrix} = \begin{bmatrix} p_{x'} + x \\ p_{y'} + y \\ p_{z'} + z \\ 1 \end{bmatrix}$$

2.5.4 Matrices de proyección

La matriz de proyección es la que nos permite tener rotación y posición en una misma matriz, por lo que esta es la unión de una matriz 3x3 que muestra la rotación y una columna más de tres elementos que permite obtener la posición. Para normalizar la misma se añade una fila más logrando que sea una matriz cuadrada de 4x4.

En esta la fila añadida suele representar perspectiva y el escalado con matrices 1x3 y 1x1 respectivamente. En ciertas aplicaciones, la matriz de perspectiva es rellenada con 0 y el escalado con 1.

$$Proy(p) = \begin{bmatrix} Rotación & Rotación & Rotación & Posición\ x \\ Rotación & Rotación & Rotación & Posición\ y \\ Rotación & Rotación & Rotación & Posición\ z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.5.5 Cuaterniones

Hamilton descubrió los cuaterniones al tratar de extender los números complejos convirtiéndolos en números hipercomplejos, representando mayores dimensiones y siendo estos generados de manera análoga a los números complejos. El proceso se lo realiza añadiendo i, j, y k a los números reales tal que $i^2 = j^2 = k^2 = ijk = -1$. El cuaternion se puede representar usando matrices complejas de 2x2 o 4x4. (Mendomatica, 2010)

Por ejemplo, siendo el cuaternion $q = a + bi + cj + dk$

$$q = \begin{pmatrix} 1 - di & -b + ci \\ b + ci & a + di \end{pmatrix}$$

$$q = \begin{pmatrix} a & -b & d & -c \\ b & a & -c & -d \\ -d & c & a & -b \\ c & d & b & a \end{pmatrix}$$

Una ventaja al emplear cuaternios es el evitar la singularidad que se da al usar ángulos de Euler cuando el segundo movimiento angular es de 0°o 180°; dado que al suceder esto el primer y segundo ángulo quedan indefinidos perdiendo un grado de libertad, y conduciendo generalmente a un bloqueo del sistema. La redundancia de los cuaterniones permite que estos sean utilizados para representar orientaciones de objetos en espacios tridimensionales en gráficos computacionales.

La composición de rotaciones resulta de la multiplicación entre cuaternios y da como resultado un cuaternion que representa una rotación de valor theta sobre un eje r. Este está dado por:

$$Q = R(r, \theta) = \left(cos\left(\frac{\theta}{2}\right), r.sen\left(\frac{\theta}{2}\right) \right)$$

A su vez existe la composición para movimientos de rotaciones y traslaciones conjuntas; en donde se traslada un vector p y se rota Q a un vector r en un sistema inicial OXYZ, transformándolo en un sistema nuevo OUVW donde las nuevas coordenadas del vector r en relación al sistema OUVW son (Botero, 2005):

$$(0, r_{OUVW}) = Q.(0, r_{OXYZ}).Q^* + (0, p)$$

2.6 Visualización 3D

La representación de entornos y objetos en 3 dimensiones ha sido un problema de visión computacional en la actualidad ya que esta es utilizada en varias aplicaciones como planificación de movimiento de robots, modelado de escenas y realidad virtual; por lo que existen varios métodos que permiten la extracción de información espacial, uno de estos métodos se basa en cámaras estéreo (Revelo-Luna, Usama, & Flórez-Marulanda, 2012).

Esta representación de objetos reales en la memoria de un computador es definida como reconstrucción 3D. El objetivo principal de todos los algoritmos de representación 3D es que este sea capaz de realizar la conexión de puntos característicos del objeto en elementos de la superficie, la eficiencia de cada técnica o algoritmo se mide mediante la calidad de la reconstrucción (Grandón-Pastén, Arancena-Pizarro, & Tozzi, 2007).

2.6.1 Estereovisión

Al utilizar cámaras estéreo se tiene dos problemas: la correspondencia y la reconstrucción 3D. La correspondencia es obtener el mismo punto físico tanto en la cámara izquierda y derecha. Para esto es necesario que las imágenes tengan una relación en cuento a puntos característicos. La

reconstrucción abarca la obtención de coordenadas 3D de un punto en el espacio, tomando dos puntos correspondientes en las dos imágenes.

Para procesar la visión estereoscópica existen dos técnicas muy utilizadas: Disparidad por regiones y disparidad por extracción de características (Revelo-Luna, Usama, & Flórez-Marulanda, 2012).

2.6.2 Correspondencia Estéreo

La correspondencia estéreo puede ser solucionada mediante varios métodos. Uno de estos el flujo óptico, el cual define el movimiento del objeto en una secuencia de imágenes utilizando para ello la intensidad de los pixeles. Este método calcula la componente de velocidad de la dirección del gradiente de brillo (Revelo-Luna, Usama, & Flórez-Marulanda, 2012).

2.6.3 Profundidad y reconstrucción

Una vez definida la correspondencia, se debe encontrar el mapa de disparidad. Esto se refiere a la diferencia en la ubicación de un objeto en dos imágenes semejantes (Rambhia, 2013). La disparidad determina la coordenada z de cada pixel logrando así establecer tres coordenadas a partir de proyecciones en dos dimensiones, llegando a una reconstrucción 3D del entorno u objeto (Revelo-Luna, Usama, & Flórez-Marulanda, 2012). Para la visualización de la reconstrucción existe un entorno de trabajo llamado Jupyter Notebook, donde mediante código Python permite generar una imagen por capas, logrando representaciones gráficas con profundidad (Gwydion, 2018).

2.6.4 Reconocimiento de objetos

El reconocimiento de objetos es la tarea de encontrar e identificar automáticamente objetos en una imagen. Para esto Google ha creado una herramienta llamada TensorFlow que permite

desarrollar algoritmos de inteligencia artificial y Machine Learning para detectar y clasificar objetos en una imagen (Movetia, 2017).

Un sistema por lo general está constituido por (Flores, 2015):

- Adquisición de imagen: digitalización de una imagen.
- Pre-procesamiento: ajustar la imagen para una aplicación.
- Segmentación: separar los objetos de interés.
- Extracción de rasgos: describir cuantitativamente la naturaleza de los objetos.
- Clasificación de objetos: coloca una etiqueta a un objeto a partir de sus rasgos.

CAPÍTULO III

IMPLEMENTACIÓN CONTROL DE NAVEGACIÓN

3.1 Introducción

El proyecto inicialmente estipulaba la utilización de dos mini drones Cheerson Cx10W con los cuales se deseaba rodear un objeto para su reconstrucción en 3D. Dichos drones contaban con una cámara monocular de 0.3 MP y un giroscopio de 6 grados de libertad. Al trabajar con los drones se presentaron varias dificultades: una fue que al no ser un sistema libre se tuvo que decodificar la comunicación para el envío de ángulos de vuelo y la recepción del video de la cámara, y una segunda dificultad fue que al no ser accesibles los datos del giroscopio interno del dron no se conocía la posición y orientación; por lo cual se utilizó una cámara externa ubicada superior a los drones y apuntando hacia ellos para solventar estos limitantes. Esta cámara detectaba la posición y orientación de los drones gracias al reconocimiento de marcadores ArUco que fueron pegados sobre los drones.

Una desventaja que se presentó al trabajar con el sistema de cámara externa fue las variaciones en la lectura de los marcadores ArUco, que daban valores erráticos de posición y orientación. Para querer corregir este error se implementó un filtro Kalman que nos daba unos datos más estables gracias a su predicción, pero con un retardo de la posición real, por lo que los drones llegaban a la inestabilidad.

Una dificultad adicional e importante fue la duración de la batería; aproximadamente tres minutos en vuelo bajo y lento. Esto fue un inconveniente ya que no se podía realizar pruebas con facilidad y al momento de implementar un control de estabilidad el dron perdía altura antes de estabilizarse en la posición deseada, lo que hacía que perdiera el control.

Por estos motivos se decidió que para la continuación del proyecto se adquiriría un dron con mejores prestaciones, el cual luego de un análisis de prestaciones, precio, tamaño y tiempo de adquisición, se determinó un dron Tello DJI como la mejor opción. Al igual que el dron Cheerson Cx10W, es considerado mini dron para vuelos en interiores. El dron adquirido cuenta con un SDK libre lo que permite una comunicación sencilla, captura de valores de sensores además la duración de su batería rodea los 10 minutos.

La idea de la utilización de dos mini drones es la triangulación de imágenes logrando medir profundidad para lograr la reconstrucción de un objeto en 3 dimensiones, lo cual al realizar el proyecto se evidenció que se podía obviar un segundo dron ya que al recorrer el primer dron toma fotos del objeto en diferentes posiciones logrando triangular dichas fotografías y por consiguiente conocer la profundidad del objeto a replicar.

Para la navegación del mini-dron se encuentra implementado un control de estabilidad, el cual permite que el dron vuele sin oscilaciones. Además, se genera un control de desplazamiento en el que se predefine una trayectoria alrededor del objeto enfocando en dirección al mismo. Para el funcionamiento de estos controladores no es necesario dispositivos externos tan solo el dron, la computadora central y el objeto que se desea reconstruir. Para esto se generó un ambiente controlado de tres metros cuadrados (Figura 17).

Figura 17. Entorno del Sistema

Para la puesta en marcha de los dos controladores se utiliza los datos de movimiento, rotación y posición del dron. Estos datos son enviados hacia el computador central del sistema mediante una comunicaron Wi-fi. De la misma manera se envían los parámetros de vuelo para que este cumpla con una navegación estable y el seguimiento de trayectoria. En el presente proyecto se ha definido el eje de coordenadas para la rotación y traslación del dron como se observa en la Figura 18.

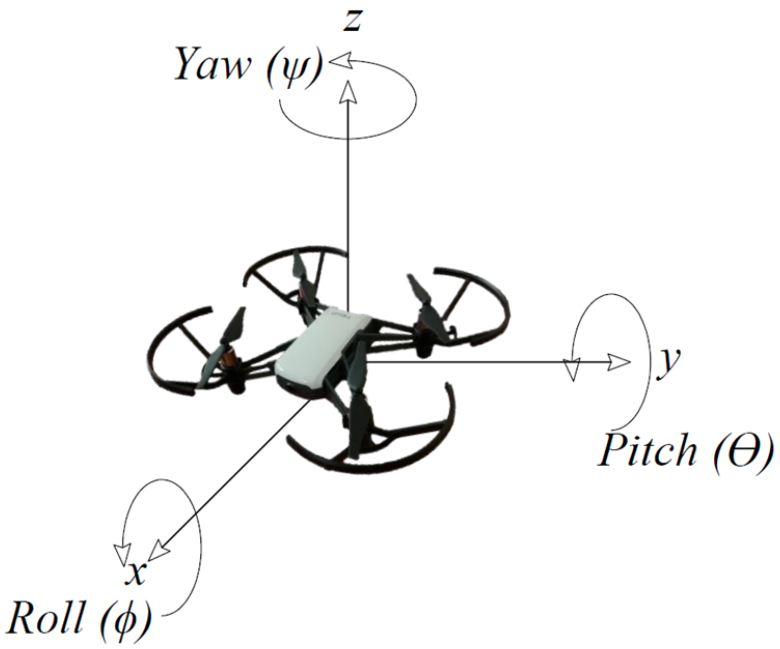


Figura 18. Ángulos de navegación Dron Tello

La reconstrucción del objeto se basa en la triangulación de fotografías tomadas por el dron en su recorrido alrededor del mismo. En cada una de estas se detecta puntos característicos y se compara con la siguiente fotografía, logrando encontrar los puntos característicos que coinciden y el desplazamiento de los mismos mediante flujo óptico, para así tener una aproximación de la profundidad del objeto y poderlo replicar en 3D.

3.2 Distribución de Red

El mini dron crea una red wifi para su comunicación, sin clave de acceso y con una IP DHCP única para cada dispositivo, a la cual se puede acceder en forma simultánea desde varios dispositivos. Sim embargo al conectarse más de un dispositivo controlador al mismo dron, existe prioridad de control al primer dispositivo conectado a su red, el resto de dispositivos conectados

no gobiernan el vuelo y reciben el video con un retardo. Al utilizar un mini dron la comunicación con la PC central será directa, punto a punto y mediante conexión wifi (Figura 19).

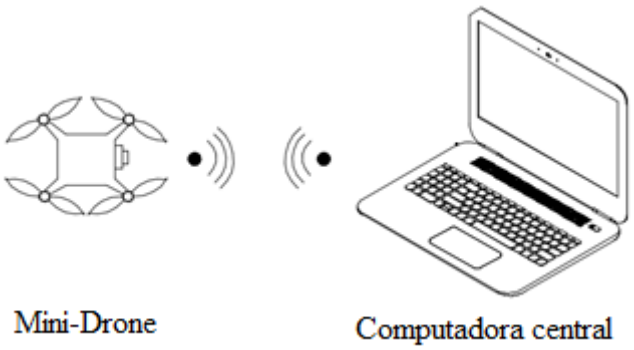


Figura 19. Distribución de Red

3.3 Comunicación Dron – PC

La PC central es el controlador del dron, enviando los parámetros de vuelo y recibiendo el video en formato h264. Este formato de video proporciona una alta calidad de imagen en relación a los pocos bits que envía, el cual se transfiere entre sockets hacia la computadora para su procesamiento.

Para establecer la comunicación con los drones se utilizó un SDK libre, el cual nos permite acceder a las imágenes provenientes de la cámara y a los valores de los sensores. La comunicación que se establece en el SDK es a través de sockets con la dirección IP 192.168.0.1 con el puerto 8889.

Se emplean dos hilos de comunicación, el primero para la obtención del video y el segundo para la obtención de los valores de los sensores.

Al correr el SDK para la comunicación se envían y reciben comando de inicialización del dron (Figura 20).

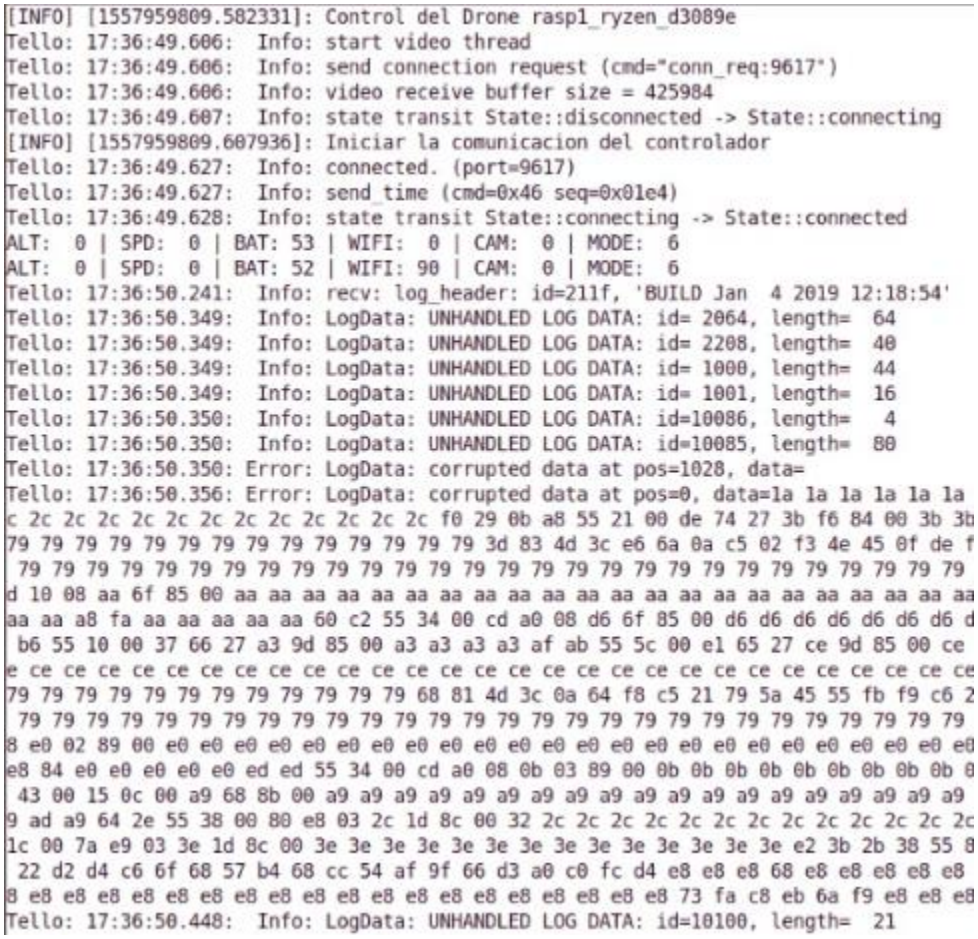


Figura 20. Mensajes para inicializar comunicación

Para que la conexión sea exitosa se debe tener previamente ejecutado el ROS-Master y a su vez los nodos ROS que habilitan los sockets para la recepción del video del dron en la computadora central.

3.4 Nodos ROS

El proyecto desarrollado en ROS contiene los nodos presentados en la Figura 21, mismos que necesitan del nodo principal ROSCORE para la suscripción y publicación de tópicos entre sí.

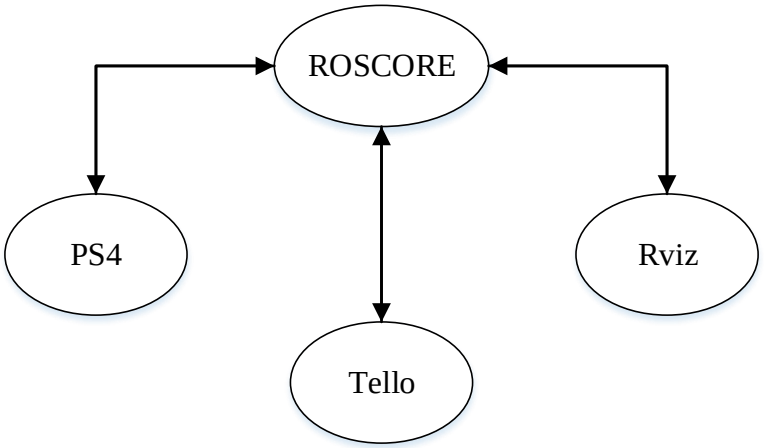


Figura 21. Topología nodos ROS

3.4.1 Nodo PS4

Este nodo permite realizar el control de vuelo del mini drone a través del control PS4, donde envía los valores de $pitch, roll, yaw$ y el valor del botón PS para ejecutar un paro de emergencia por el tópico $/cmd\_vel\_ps4\_(IDdelminidrone)/control$.

Como es un controlador estándar para el control de vuelo, se puede configurar para el control de los diferentes mini drones dependiendo del ID_Drone. A su vez este envía el diagnóstico de la palanca conectada verificando si esta se encuentra en correcto funcionamiento o no mediante el tópico $/control\_(IDdelminidrone)/diagnosticStatus$. En la Figura 22 se muestra la topología del nodo.

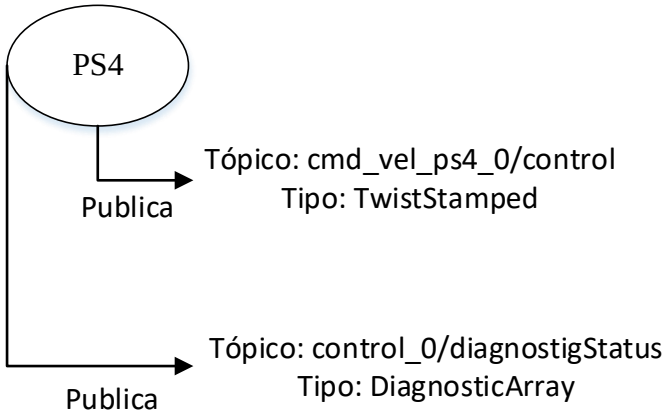


Figura 22.Topología nodo PS4

3.4.2 Nodo Tello

Este nodo se encuentra suscrito a los tópicos publicados por el nodo PS4. El nodo Tello es el encargado de correr los controladores de estabilidad y trayectoria, abrir el hilo de comunicación para obtener los datos del dron, y ejecutar la interfaz gráfica. A su vez este envía el status, el marcador, y la imagen de la cámara del dron mediante los tópicos publicadores mostrados en la Figura 23.

Además, este nodo pública mensajes de posición, orientación y el identificador del dron a toda la red mediante la función tf.

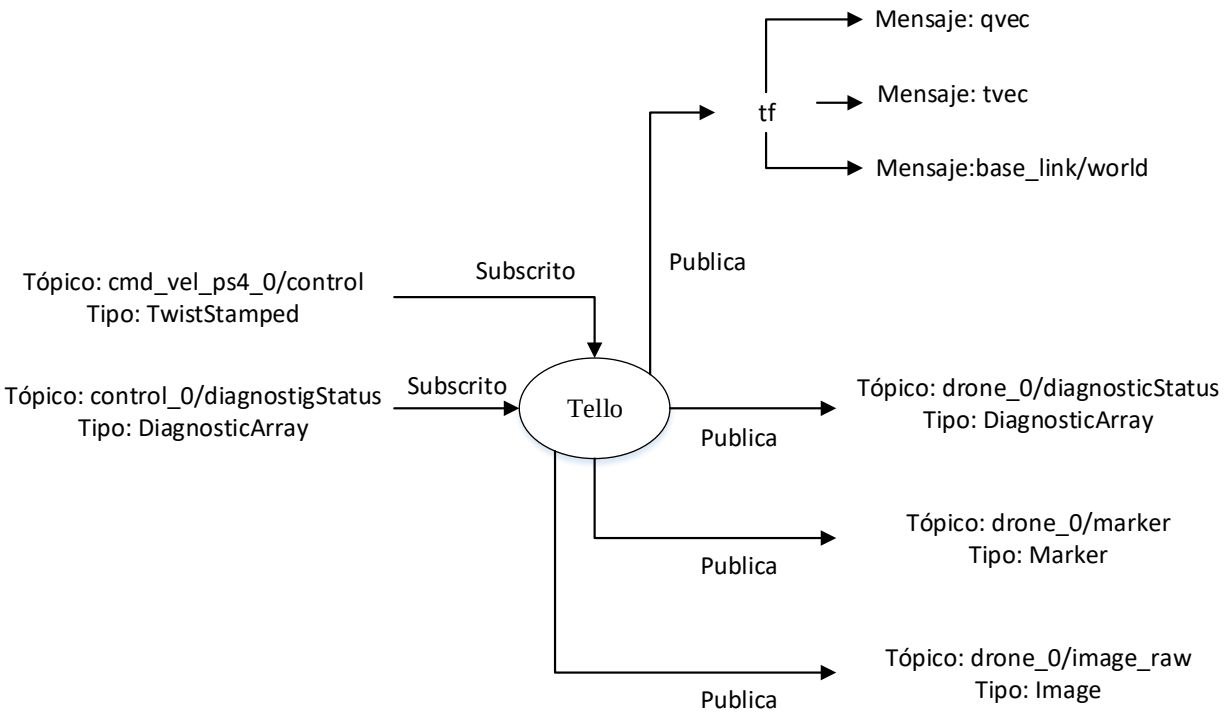


Figura 23. Topología nodo Tello

3.4.3 Nodo Rviz

Este nodo crea el entorno de simulación Rviz con todas las configuraciones iniciales publicadas en el nodo $/rviz/settings\_rviz$. Este nodo se encuentra suscrito a los tópicos publicados por el nodo Tello como muestra la Figura 24, para la visualización de la posición y orientación del dron en tiempo real.

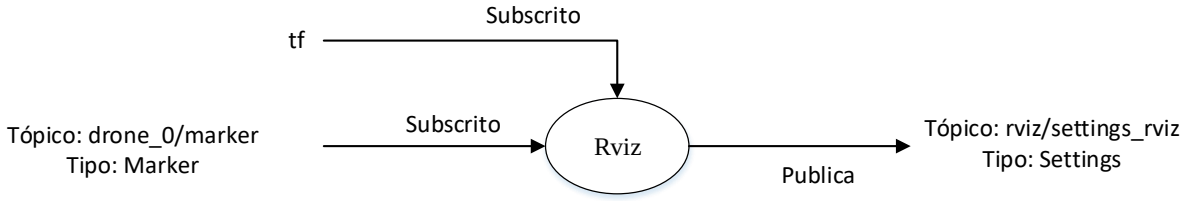


Figura 24. Topología nodo Rviz

3.4.4 Nodo camera_calibration

Este nodo es utilizado independientemente para calibrar y conseguir los parámetros intrínsecos de la cámara del dron, así como su matriz de distorsión; mismas que ayudan en el reconocimiento de objetos y localización de los mismos. Esta calibración arroja como resultado un archivo yml que contiene los parámetros mencionados de la cámara.

3.5 Archivo Launch

Para el lanzamiento de nodos mencionados anteriormente se tiene creado un archivo launch el cual permite que los nodos corran uno tras otro sin necesidad de ir corriéndolos uno por uno por línea de comandos.

Se tiene el archivo Ryze_d3089e.launch que es el encargado de ejecutar el nodo Tello, seguido de una espera de ocho segundos para ejecutar el archivo rviz.launch el cual ejecuta el nodo Rviz y carga el archivo URDF del dron para su posterior simulación. Por último, con una espera de cinco segundos ejecuta el archivo PS4.launch el cual ejecuta el nodo PS4.

3.6 Navegación manual PS4

3.6.1 Crear un paquete dentro del entorno

Dentro del entorno ROS debemos crear un espacio de trabajo el cual nos permite posteriormente comunicar nodos y publicar o suscribirse a tópicos. Para el caso del control manual se creó un paquete llamado control PS4, el cual utiliza los tipos de datos: std_msgs. En este se encuentran tipos de datos primitivos y mensajes básicos, y a su vez este paquete puede contener programas en Python y en lenguaje C. Dependiendo el lenguaje se especifica en la creación del paquete las dependencias rospy y roscpp respectivamente, por lo cual para la creación de esta se utilizó el comando:

catkin_create_pkg controlPS4 std_msgs rospy roscpp

3.6.2 Reconocimiento de palanca PS4

Para que el sistema operativo pueda reconocer la palanca se utilizó la librería pygame y a su vez el programa controller.py de (McLeod, 2017); mediante el cual se identificó cada uno de los botones de la palanca y se definió lo controles de la siguiente manera: análogo izquierdo “L3” para la manipulación de throttle y yaw, análogo derecho “R3” para roll y pitch, pulsador superior derecho “R1” para el despegue autónomo, botón superior izquierdo “L1” para el aterrizaje autónomo y el botón “PS” para cerrar los nodos (Figura 25).

Figura 25. Control PS4

El reconocimiento de los análogos comenzó identificando que para cada eje existía un número de axis diferente y los valores de estos variaban entre -1.00 y 1.00 dependiendo su posición, como se muestra en la Tabla 3.

Tabla 3

Movimiento y axis de análogos PS4

Pulsador / Análogos PS4	Axis/Button	Movimiento	Valores	Eje
L3	0	Izquierda - Derecha	-1.00 a 1.00	Yaw
L3	1	Arriba – Abajo	-1.00 a 1.00	Throttle
R3	3	Izquierda - Derecha	-1.00 a 1.00	Roll
R3	4	Arriba – Abajo	-1.00 a 1.00	Pitch
L1	4	High / Low	1 o 0	Aterrizaje
R1	5	High / Low	1 o 0	Despegar
PS	10	High / Low	1 o 0	Cerrar

Los valores entre los que varía la palanca son enviados hacia el dron sin linealizarlos ya que el SDK permite el seteo de los valores entre -1 y 1, mismos que son enviados por la palanca.

Debido a esto, la asignación de valores en el programa queda de la siguiente manera:

LEFT_Y=self.axis_data[1]

LEFT_X =self.axis_data[0]

RIGHT_Y=self.axis_data[4]

RIGHT_X=self.axis_data[3]

LAND=self.button_data[4]

TAKEOFF=self.button_data[5]

PS=self.button_data[10]

La asignación de los datos ya escalados por el SDK va a un nodo publicador “/cmd_vel_0”, el cual tiene tipos de datos twist y se divide en angular y lineal para diferenciar entre roll, pitch, yaw y throttle.

El computador central se suscribe al nodo para recibir los datos de los vectores, existe un vector lineal de la forma twist.linear y otro angular twist.angular. Cada dato de estos vectores es separado individualmente en los valores para el movimiento del dron y estos son enviados al dron mediante las funciones.

self.drone.set_throttle(twist.linear.x)

self.drone.set_yaw(twist.angular.z)

self.drone.set_pitch(twist.angular.y)
self.drone.set_roll(twist.angular.x)

Para el despegue y aterrizaje automático se debe asignar el valor de uno a la variable del vector twist.linear.y y twist.linear.z respectivamente.

despegue=self.drone.takeoff()
aterrizaje=self.drone.land ()

3.7 Obtención de posición y orientación

El proceso de ubicación del mini dron se logra por la información enviada por estos gracias a sus sensores internos. El dron Tello cuenta con un módulo IMU (Unidad de Medición Inercial), un sensor óptico y un sensor infrarrojo. Dentro de la IMU cuenta con un giroscopio de 6 ejes de libertad y un acelerómetro lo que nos permite conocer los ángulos de rotación ($roll, pitch, yaw$), velocidad y aceleración del dron; mientras que el sensor óptico nos indica el desplazamiento que tiene el dron en los tres ejes(x, y, z) a partir de su posición inicial.

Para la adquisición de los valores de orientación se realiza una petición al SDK. Estos datos son guardados en el vector qvec. Adicionalmente se guarda la velocidad lineal (acelerómetro) en el vector twist_linear y la velocidad angular (giroscopio) en el vector twist_angular.

qvec = [self.log_data.imu.q0, self.log_data.imu.q1, self.log_data.imu.q2, self.log_data.imu.q3]
twist_linear = [self.log_data.imu.acc_x, self.log_data.imu.acc_y, self.log_data.imu.acc_z]
twist_angular = [self.log_data.imu.gyro_x, self.log_data.imu.gyro_y, self.log_data.imu.gyro_z]

De manera similar a la obtención de la orientación se toma los datos de la posición. Se realiza la petición al SDK para adquirir los datos del registro MVO (registro de vuelo) que envía el dron

hacia el computador, decodifico los mismos logrando adquirir la posición en los tres ejes “x”, “y” y “z”, y los datos son guardados en el vector tvec (Krag, 2018).

tvec = [self.log_data.mvo.pos_x, -self.log_data.mvo.pos_y, -self.log_data.mvo.pos_z]

Con esto se tienen conjuntamente posición y orientación de la aeronave para la utilización de estos datos. Para el envió de parámetros de vuelo como: despegue, aterrizaje, movimiento y rotación se utiliza la librería TelloPy de (Hanyazou, 2019).

Los datos recibidos del dron en cuanto a rotación se encuentran en cuaternios. La posición en metros se comparte con el resto de nodos ROS emulando el comportamiento broadcast. Esta información es utilizada para la simulación el entorno Rviz en el computador central y a su vez para la implementación de los controladores.

Cabe aclarar que, al obtener los datos posteriores al despegue automático del dron, los valores de orientación se encuentran correctamente generados; sin embargo, el MVO envía valores aleatorios a las coordenadas de posición inicial, por esto en primera instancia las posiciones del resto de movimientos quedan desplazadas dependiendo de estos valores.

Para que la posición inicial del dron coincida con el centro del eje de coordenadas en “x” y “y”, y a su vez tenga la elevación en el eje “z” real, se obtuvo la medida de la altura con el despegue automático. Esta fue de 1.2 metros por lo que las posiciones iniciales del dron serian (0,0,1.2).

Para establecer estas posiciones como valores iniciales posterior al despegue del dron se esperó 10 segundos, tiempo estimado para que los datos aleatorios ya no varíen y así poderlos asignar a un vector, mismo que esta restado en todo momento de la posición adquirida. A su vez para coincidir en la altura del eje z, se suma el valor inicial de elevación del despegue automático.

tvec0[]=tvec[]

tvec1[]=[tvec[0]-tvec0[0] , tvec[1]-tvec0[1] , tvec[2]-tvec0[2]+1.2]

Donde tvec, tvec0 y tvec1son vectores de una fila por tres columnas que contienen las posiciones en x, y, z.

3.8 Visión Rviz del mini-dron

Es necesario observar en todo momento la posición y orientación del dron para tener un mayor control sobre los mismos desde la computadora central y a su vez desde el vuelo manual, por lo que al utilizar ROS para crear nodos de comunicación y demás acciones que el presente proyecto requiere, también se utiliza la herramienta RViz para la visualización y orientación del dron.

Para esto es necesario el reconocimiento, el cual se realiza mediante la información otorgada por el dron: posición (x,y,z), los ángulos de giro en cuaternios, y la velocidad y aceleración; todo con respecto a su posición inicial. Para esto se le asigna al dron un base_link, mismo que será su identificador en el entorno de simulación obtenido una posición relativa al eje de coordenadas fijo "world" (Figura 26).

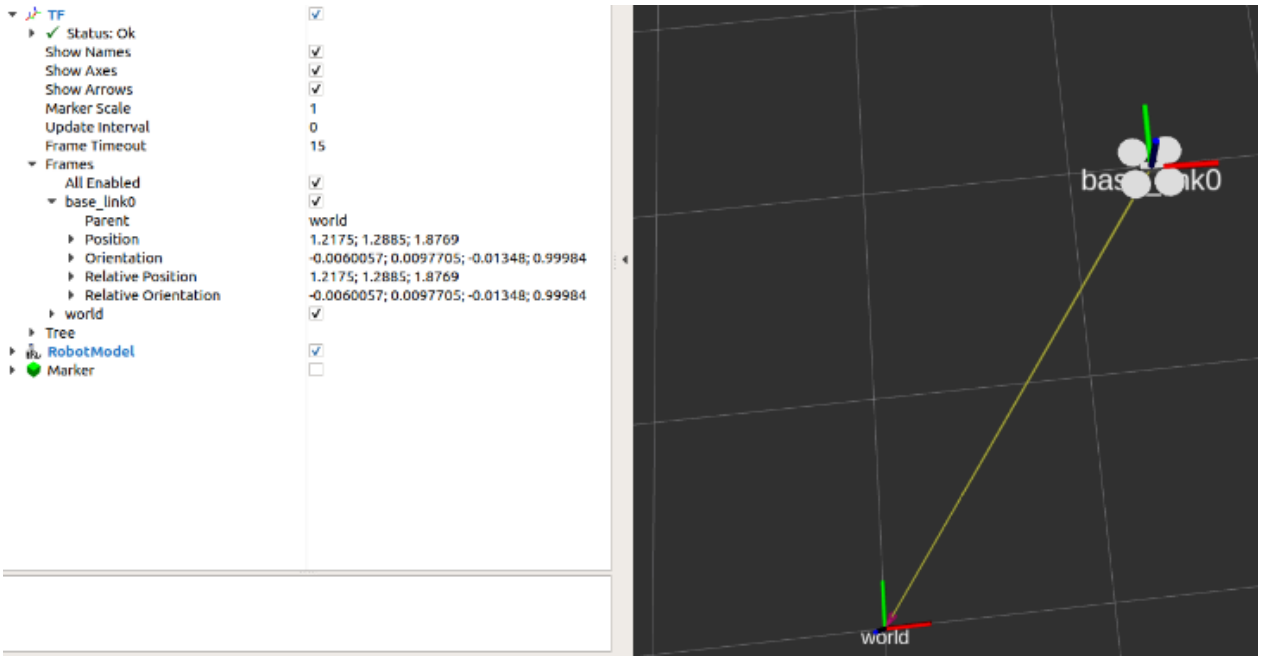
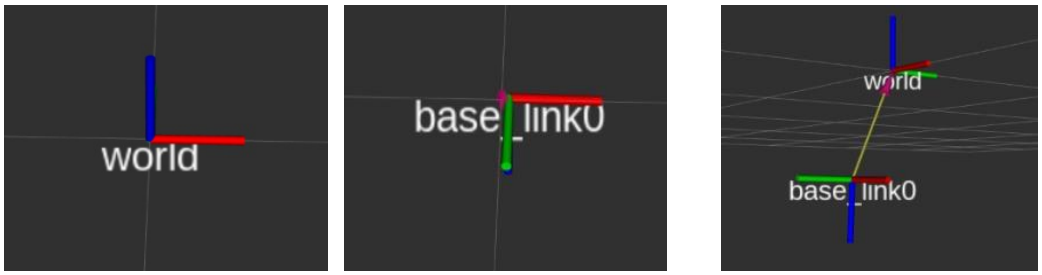


Figura 26. Posición y orientación en RViz

3.8.1 Rotación del dron en RViz

En un inicio al simular directamente los valores de posición obtenidos desde el dron en el entorno RViz con respecto al origen predefinido, se mostraban de forma errada tanto en orientación de los ejes “y” y “z” como en rotación del ángulo “yaw” (Figura 27), solo coincidiendo en la orientación del eje x, por lo que al elevarse el dron en la simulación este se mostraba bajo el plano.



a) b) c)

Figura 27. a) Origen predefinido; b) Orientación inicial del dron; c) Dron respecto al origen

Para corregir el error de orientación de los ejes se multiplico los valores de posición en “y” y “z” por menos uno (Figura 28). Con esto el dron se ubica sobre el plano y en posición correcta respecto al origen; pero con el ángulo de giro “Yaw” aun sobre el eje “x”.

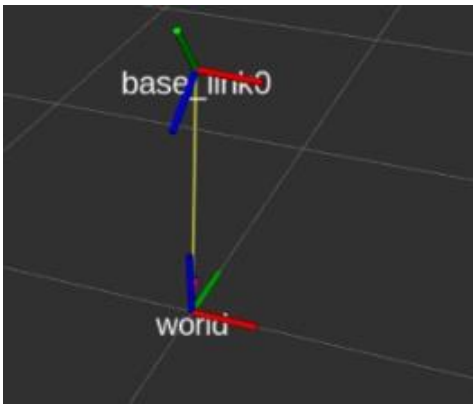


Figura 28. Dron sobre el origen.

Se transformó los ángulos de rotación tomados del dron de cuaternios a ángulos de Euler mediante la función quaternium_from_euler para poder cambiar el giro de “Yaw” del eje “x” hacia

el eje “z”, con lo que los ejes “z” y “x” quedaron invertidos; por lo que se roto el eje de coordenadas del dron 180º entorno a “y” logrando así tener la posición y orientación real del dron respecto al origen en la simulación (Figura 29).

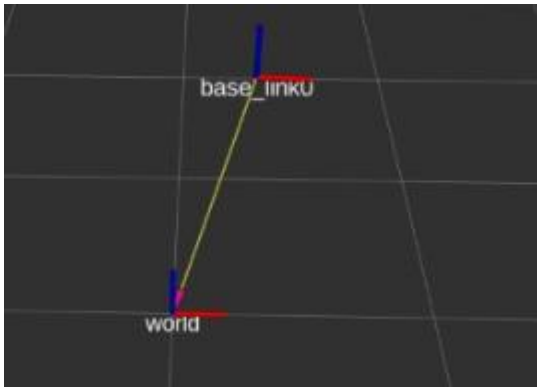


Figura 29. Posición y orientación final del dron respecto al origen

3.8.2 Archivos URDF

Para una visualización del dron en el entorno Rviz se crea un archivo URDF (Universal Robot Description Format), el cual describe los elementos físicos del robot y permite un modelo visual del mismo (Figura 30).

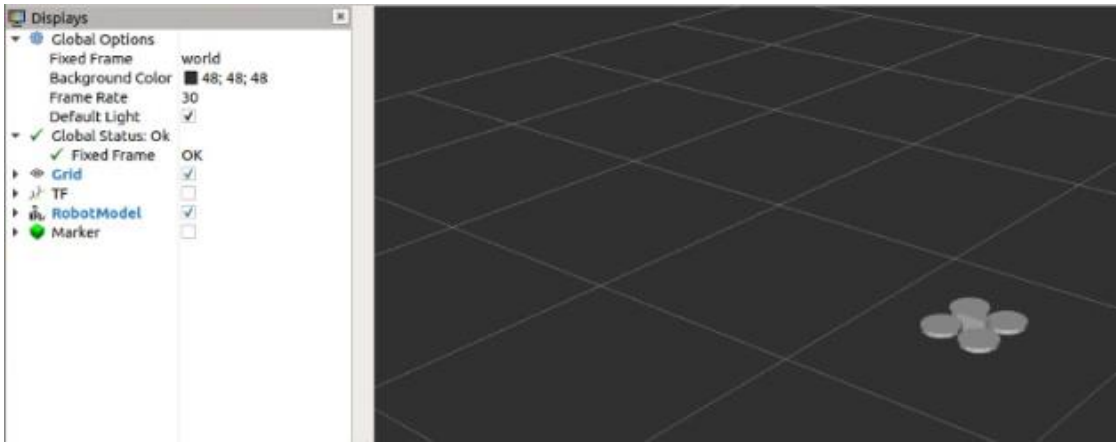


Figura 30.Simulación URDF del dron.

Para la simulación del dron Tello en Rviz se creó un archivo URDF que permita una geometría visual del mismo; con 4 cilindros de 0.01 m de largo y un radio de 0.045 m simulando cada una de las hélices, y dos prismas rectangulares de 0.07 m de largo, 0.02m de alto y 0.02m de profundidad simulando la carcasa del dron que contiene la cámara. Cada elemento del dron se encuentra simulado en color gris, este se determina por su vector rgba en el mismo archivo.

Ya creado todos los elementos a utilizarse se coloca la posición de cada uno de estos para que simulen gráficamente al dron. Para que el movimiento de la simulación responda de igual manera que el dron real, se asigna a este archivo urdf el base_link del dron recibiendo la información de rotación y posición.

3.8.3 Marcadores

Los marcadores en Rviz son comúnmente utilizados para crear visualizaciones personalizadas con texto o figuras. En el presente proyecto se utiliza los mismos para visualizar la trayectoria realizada por el dron (Figura 31).

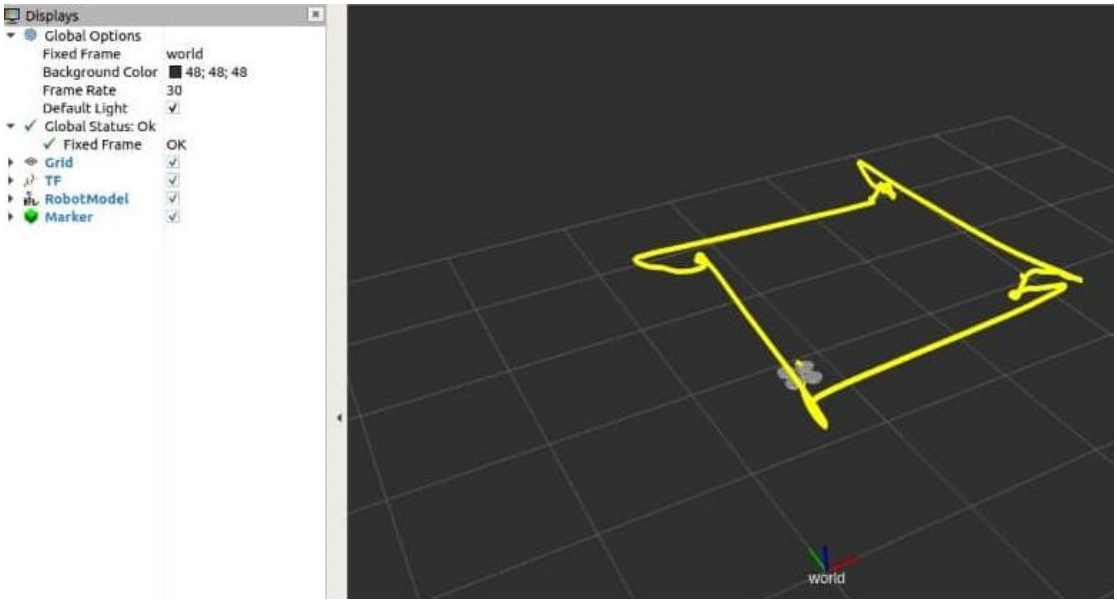


Figura 31. Simulación de trayectoria con marcador.

Para la visualización de la trayectoria tomada por el dron se debe saber la posición actual en un momento dado, y al utilizar una línea también es necesario tener un conjunto de puntos por los que el dron se ha desplazado; este conjunto de puntos se almacena y se publica en un nodo.

Para que el seguimiento mediante el marcador sea una línea se utiliza el objeto del marcador de tipo line_strip, una escala de 0.03 que determina el ancho de la línea y el color en rgba.

3.9 Control de Estabilidad Fuzzy

Es una técnica de control inteligente basada en la experiencia de un experto en el proceso. Se utilizó esta técnica ya que no se cuenta con un modelado matemático del dron y a su vez el acceso a los parámetros que se envían a los actuadores es de forma conjunta mediante ángulos que varían la posición y orientación del dron. Para la implementación de este controlador en el presente proyecto se utilizó la librería “skfuzzy.control” misma que permite crear los conjuntos universos tanto para antecedentes como consecuentes, y a su vez el ingreso del conjunto de reglas para la toma de decisiones del controlador.

Para la estabilidad del dron se realiza un sistema de control (Figura 32), en el cual se envía una posición determinada en x, y, z, y yaw en la que el dron permanece sin moverse. Para esto se realiza una lectura de su posición actual y se compara con la enviada. La técnica de control fuzzy es la encargada de comparar y si es necesario corregir la posición enviando señales de control a los ángulos de navegación del dron logrando un control frente a perturbaciones de posición y rotación.

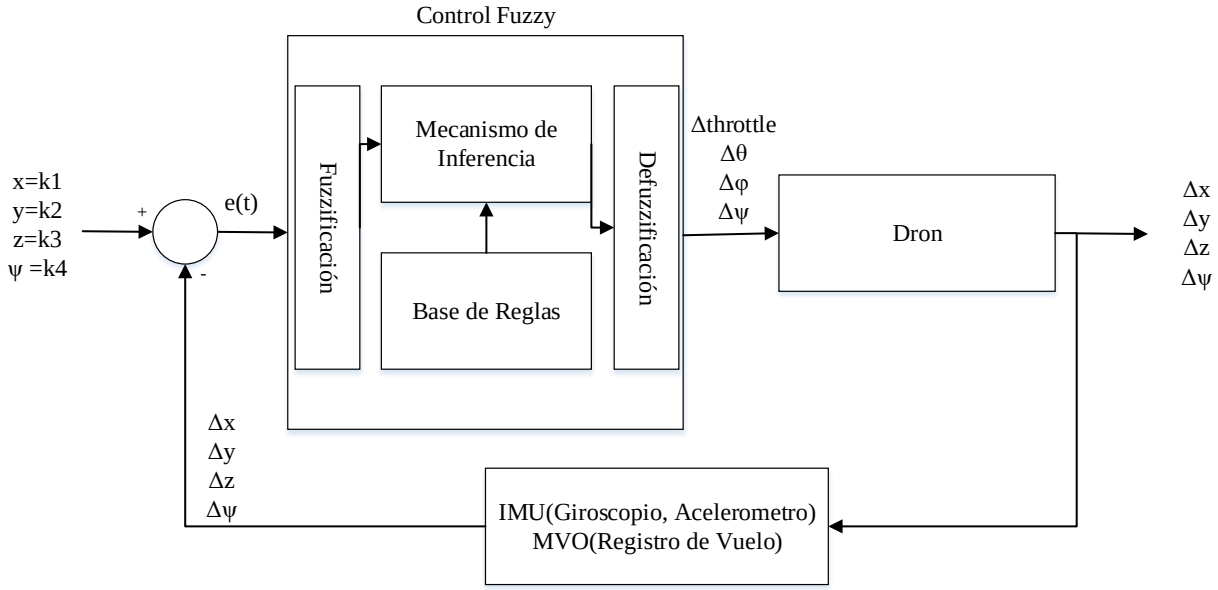


Figura 32.Esquema de control de estabilidad

3.9.1 Valores Lingüísticos

Para la interpretación de las reglas se tiene siete posibles antecedentes para el error, y se tiene cinco consecuentes para la salida del controlador, mismos que se muestran en la Tabla 4:

Tabla 4
Valores Lingüísticos

Valores Lingüísticos	
PG	Positivo Grande
PM	Positivo Mediano
PP	Positivo Pequeño
ZE	Cero
NP	Negativo Pequeño
NM	Negativo Mediano
NG	Negativo Grande

3.9.2 Control fuzzy P

El control Fuzzy P implementado en el proyecto es independiente para la posición en "x", "y", "z" y orientación "Yaw", pero funciona simultáneamente. Tiene como entrada de cada controlador el error en estas posiciones y el error en orientación. Este controlador al contar con una sola entrada es brusco generando un sobre impulso alto y su precisión para llegar al setpoint cuenta con un error relativamente alto y constante en estado estacionario.

3.9.2.1 Base de Reglas

Para la creación de las reglas se tuvo en cuenta:

$$e = set\_point - valor\_actual$$

Por lo que para x:

- Cuando la posición actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron esta atrás de la posición deseada.
- Cuando la posición actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra delante de la posición deseada.

Con estas explicaciones, la base de reglas para el error en la posición x será interpretada como se muestra en la Tabla 5:

Tabla 5

Valores para controlador de X

	$\boldsymbol{e(t)}$	$\boldsymbol{out}$
PG	Posición muy atrás	Fuerza grande adelante
PP	Posición pequeña atrás	Fuerza pequeña adelante
ZE	Posición deseada	Sin fuerza
NP	Posición pequeña adelante	Fuerza pequeña atrás
NG	Posición muy adelante	Fuerza grande atrás

Las reglas implementadas para el error en x modifican el ángulo de giro pitch y son:

- **Si** e es NG **entonces** out es NG
- **Si** e es NM **entonces** out es NP
- **Si** e es NP **entonces** out es NP
- **Si** e es ZE **entonces** out es ZE
- **Si** e es PP **entonces** out es PP
- **Si** e es PM **entonces** out es PP
- **Si** e es PG **entonces** out es PG

La Tabla 6 muestra las reglas para el desplazamiento en x.

Tabla 6

Base de reglas para x

e	NG	NM	NP	ZE	PP	PM	PG
out	NG	NP	NP	ZE	PP	PP	PG

Para el controlador en "y" se tuvo en cuenta que:

- Cuando la posición actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron se encuentra al lado derecho de la posición deseada.
- Cuando la posición actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra al lado izquierdo de la posición deseada.

Con estas explicaciones la base de reglas para el error en la posición "y" será interpretada como se muestra en la Tabla 7:

Tabla 7
Valores para controlador de Y

	$e(t)$	out
PG	Posición muy a la derecha	Fuerza grande hacia la derecha
PM	Posición media a la derecha	-
PP	Posición pequeña a la derecha	Fuerza pequeña hacia la derecha
ZE	Posición deseada	Sin fuerza
NP	Posición pequeña a la izquierda	Fuerza pequeña hacia la izquierda
NM	Posición media a la izquierda	-
NG	Posición muy a la izquierda	Fuerza grande hacia la izquierda

Las reglas implementadas para el error en "y" modifican el ángulo de giro roll y son:

- **Si** e es NG **entonces** out es PG
- **Si** e es NM **entonces** out es PP
- **Si** e es NP **entonces** out es PP
- **Si** e es ZE **entonces** out es ZE
- **Si** e es PP **entonces** out es NP
- **Si** e es PM **entonces** out es NP
- **Si** e es PG **entonces** out es NG

La Tabla 8 muestra las reglas para Y.

Tabla 8
Reglas para Y

e	NG	NM	NP	ZE	PP	PM	PG
out	PG	PG	PP	ZE	PP	NG	NG

Para el controlador en "z" se tuvo en cuenta que:

- Cuando la posición actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron se encuentra bajo de la posición deseada.
- Cuando la posición actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra sobre posición deseada.

Con estas explicaciones la base de reglas para el error en la posición "z" será interpretada como se muestra en la Tabla 9:

Tabla 9
Valores para el controlador de Z

	$e(t)$	out
PG	Posición muy abajo	Fuerza grande hacia arriba
PM	Posición media abajo	-
PP	Posición pequeña abajo	Fuerza pequeña hacia arriba
ZE	Posición deseada	Sin fuerza
NP	Posición pequeña arriba	Fuerza pequeña hacia abajo
NM	Posición media arriba	-
NG	Posición muy arriba	Fuerza grande hacia abajo

Las reglas implementadas para el error en "z" modifican throttle y responde a las mismas reglas que el controlador de "x" que se muestran en Tabla 6.

Para el controlador en "yaw" se cambió los valores que enviaba el dron, ya que estos en un principio cuando la cámara se encontraba en dirección del eje x el valor era de 180º y su vez variaba en sentido horario decreciendo en valores positivos y en sentido antihorario en valores negativos, por lo que antes de realizar el controlador se colocó el ángulo cero en dirección al eje x y a su vez el ángulo de giro en sentido anti horario crece hasta el valor de 360º. Para realizar las reglas del controlador se tomó en cuenta:

- Cuando el error es menor que -180º se le suma 360º
- Cuando el erro es mayor que 180º se le resta de 360º
- Cuando la orientación actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron se encuentra rotado en sentido horario de la posición deseada.
- Cuando la orientación actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra rotado en sentido antihorario de la posición deseada.

Con estas explicaciones la base de reglas para el error en la orientación se muestra en la Tabla 10:

Tabla 10
Valores para el controlador de Yaw

	$e(t)$	out
PG	Orientación muy a la derecha	Fuerza grande hacia la derecha
PM	Orientación media a la derecha	-
PP	Orientación pequeña a la derecha	Fuerza pequeña hacia la derecha
ZE	Orientación deseada	Sin fuerza
NP	Orientación pequeña a la izquierda	Fuerza pequeña hacia la izquierda
NM	Orientación media a la izquierda	-
NG	Orientación muy a la izquierda	Fuerza grande hacia la izquierda

Las reglas implementadas para el error en "yaw" modifican el ángulo y responde a las mismas reglas que el controlador de Y que se muestran en Tabla 8.

3.9.2.2 Mecanismo de inferencia

El proceso o mecanismo de inferencia utilizado por la librería previamente mencionada es mediante el método de mínimo-máximo o también conocido como Mandami, e involucra dos pasos:

- Las premisas de todas las reglas son comparadas con las entradas del controlador verificando que regla debe ser utilizada en la situación actual, a este proceso generalmente se lo llama "coincidencia". En este paso primero se cuantifica el significado de cada una de las premisas, con lo cual a cada entrada se le identifica el nivel de pertenencia en cada regla. Seguidamente para la cuantificación de la operación lógica "Y" se selecciona el menor valor de estos. En el caso del proyecto en cuestión al utilizar funciones triangulares simétricas, solo una entrada podrá superponer dos funciones de pertenencia, por lo cual se podrá obtener un máximo de 2 reglas a la vez con diferentes grados de pertenencia.
- El segundo paso del mecanismo de inferencia permite obtener un rango de valores de salida mediante las recomendaciones o reglas a utilizar, seleccionadas en el paso anterior.

3.9.2.3 Interfaz de fuzzificación

Como antecedentes o entradas del sistema se tiene al error, siendo los valores que puede tomar este entre -3m y 3m por el espacio de trabajo, como se muestra en la Tabla 11.

Tabla 11

Universo de valores de las entradas

	$e(t)$ (m)
PG	$2.00 < e$
PM	$1.00 < e < 3.00$
PP	$0.00 < e < 2.00$
ZE	$-1.00 < e < 1.00$
NP	$0.00 < e < -2.00$
NM	$-1.00 < e < -3.00$
NG	$-2.00 < e$

El valor de las funciones de pertenencia para la entrada del error se calcula como se muestra en la Tabla 12.

Tabla 12

Funciones de pertenencia del error

	$e(t)$ (m)	
PG	$\mu_{PG} = e - 2.00, si\ 2.00 < e < 3.00$	$\mu_{PG} = 1\ , si\ e > 3.00$
PM	$\mu_{PM} = e - 1.00, si\ 1.00 < e < 2.00$	$\mu_{PM} = 3.00 - e, si\ 2.00 < e < 3.00$
PP	$\mu_{PP} = e + 0.00, si\ 0.00 < e < 1.00$	$\mu_{PP} = 2.00 - e, si\ 1.00 < e < 2.00$
ZE	$\mu_{ZE} = e + 1.00, si - 1.00 < e < 0.00$	$\mu_{ZE} = 1.00 - e, si\ 0.00 < e < 1.00$
NP	$\mu_{NP} = e + 2.00, si - 2.00 < e < -1.00$	$\mu_{NP} = 0.00 - e, si - 1.00 < e < 0.00$
NM	$\mu_{NM} = e + 3.00, si - 3.00 < e < -2.00$	$\mu_{NM} = -1.00 - e, si - 2.00 < e < -1.00$
NG	$\mu_{NG} = 1\ , si\ e < -3.00$	$\mu_{NG} = -2.00 - e, si - 3.00 < e < -2.00$

Los conjuntos difusos para cada valor lingüístico se crean mediante la función “automf” que pertenece a la librería previamente mencionada, obteniendo un conjunto para cada eje de traslación y la rotación en yaw (Figura 33).

La función automf crea conjuntos difusos a partir de las funciones de pertenecía mencionadas anteriormente, estos son triangulares y simétricos entre si, dependiendo el número de posibles entradas lingüísticas.

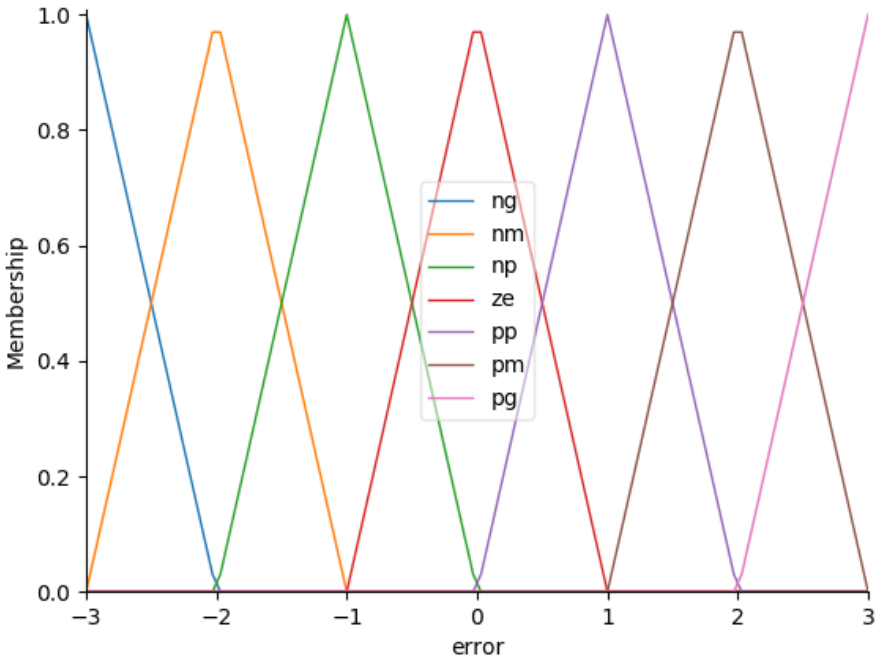


Figura 33. Conjuntos difusos del error en las posiciones.

De la misma manera que se calculó los valores de pertenencia para las posiciones en “x”, “y” y “z”, se realiza para yaw, con la diferencia que los rangos del error varían de -pi a pi (Figura 34).

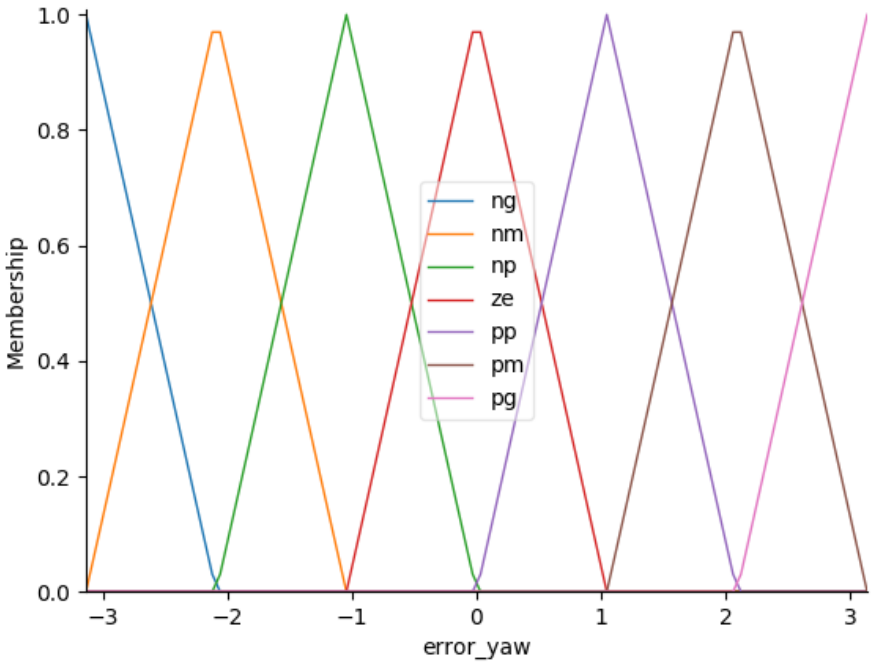


Figura 34. Conjuntos difusos del error en la orientación

3.9.2.4 Interfaz de defuzzificación

En la interfaz de defuzzificación se transforma los valores lingüísticos a valores numéricos, mismos que son enviados al dron. Los valores de trabajo de los ángulos de navegación son de -1 a 1. Estos se establecen como rango de valores de salida indicados en la Tabla 13.

Tabla 13

Universo de valores de la salida

	out
PG	$0.50 < out$
PP	$0.00 < out < 1.00$
ZE	$-0.50 < out < 0.50$
NP	$-1.00 < out < 0.00$
NG	$out < -0.50$

La misma función que crea las funciones de pertenencia para las entradas permite crear dichas funciones para la salida (Figura 35).

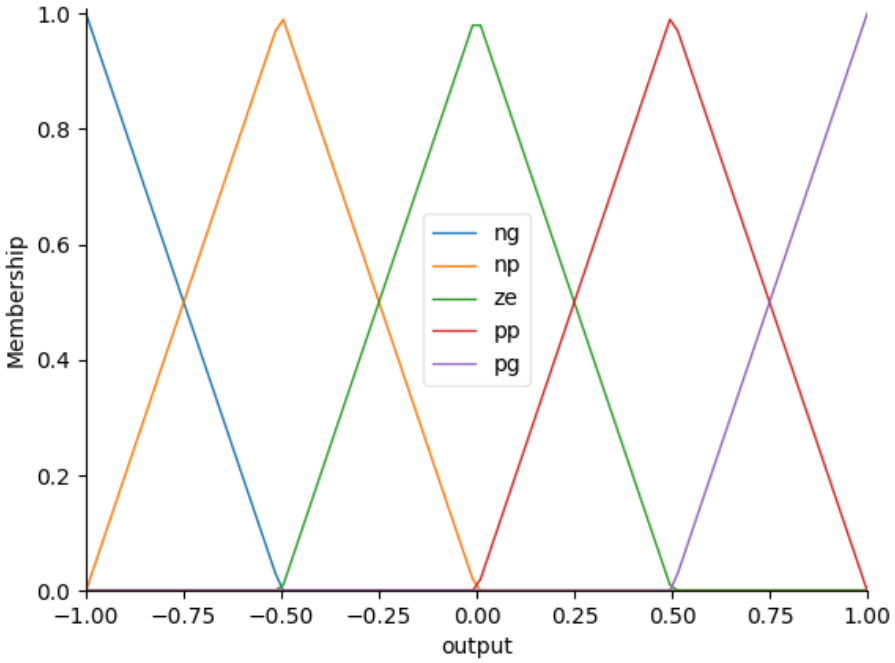


Figura 35. Conjuntos difusos de salida

En este paso se opera con todos los conjuntos difusos recomendados por el mecanismo de inferencia para poder determinar un valor fijo. Esto se realiza mediante el método de centro de gravedad “COG” en el cual primero se calculan los centros de cada uno de estos conjuntos a los cuales se les denomina con “b”, y posteriormente se calcula el área bajo la curva de los conjuntos; todos estos limitados en el nivel de pertenencia como muestra la Figura 36.

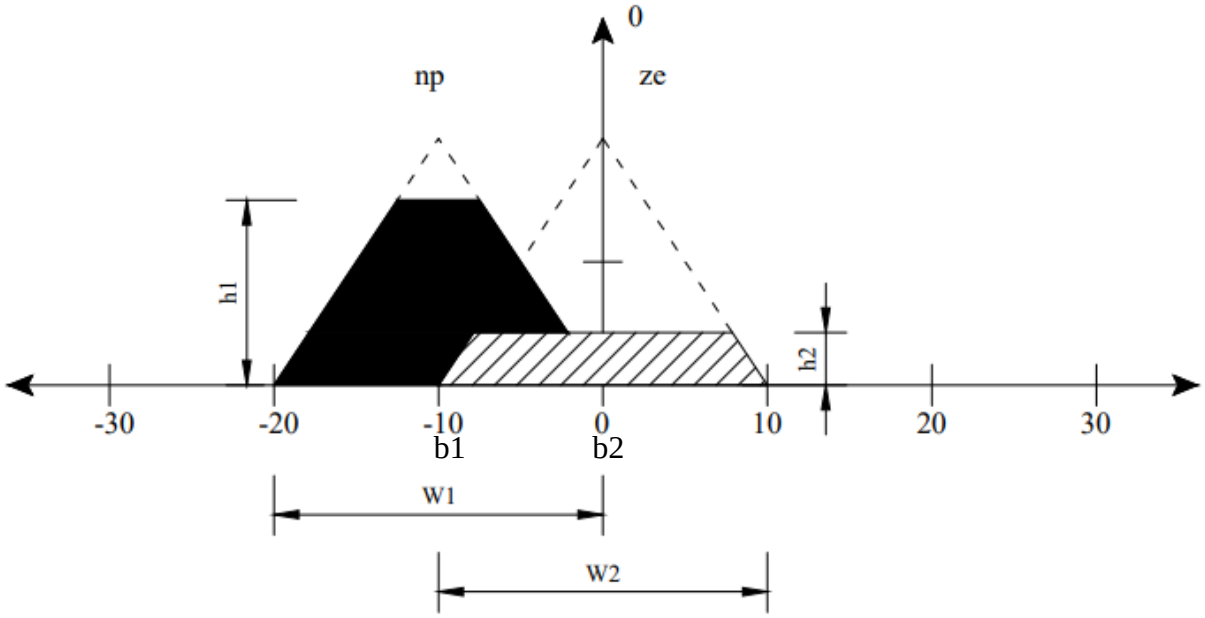


Figura 36. Conjuntos difusos con funciones triangulares simétricas

De este modo la fórmula para calcular el valor de salida es:

$$u^{crisp} = \frac{\sum_i b_i \int u_{(i)}}{\sum_i \int u_{(i)}}$$

La integral en el caso de tener funciones de pertenencia triangulares simétricas de un ancho w y altura recortada en h como muestra la Figura 36, se puede calcular mediante:

$$w(h - \frac{h^2}{2})$$

3.9.3 Controlador fuzzy PD

La parte derivativa de un controlador es aplicable en procesos rápidos y tiene la ventaja de reducir el sobre impulso y el tiempo de estabilización. Por esta razón se utilizó al error y la derivada del mismo como otra entrada de retroalimentación para mejorar el rendimiento del controlador. En el caso de la estabilidad se controla la posición en “x, y, z”, por lo tanto, se tiene como variables de entrada del controlador al error de todas estas posiciones y sus respectivas variaciones,

obteniéndose un total de 8 entradas. Las salidas del controlador son los ángulos: roll, pitch, yaw y throttle, mismos que varían la posición y orientación.

Para la implementación de la derivada fue escogida la fórmula de tres puntos y centrada, de entre seis formulas probadas. Para esta se necesita un conjunto de cinco datos, por lo cual se creó un buffer FIFO del tamaño requerido guardando los datos y logrando trabajar con ellos para determinar la derivada en el punto central a lo largo de tiempo. El buffer creado contiene la posición y el tiempo.

$$Buffer = \begin{bmatrix} X_0 & X_1 & X_2 & X_3 & X_4 \\ Y_0 & Y_1 & Y_2 & Y_3 & Y_4 \\ Z_0 & Z_1 & Z_2 & Z_3 & Z_4 \\ t_0 & t_1 & t_2 & t_3 & t_4 \end{bmatrix}$$

Por lo que se calcula la derivada de las posiciones ubicadas en columna central y para la obtención de h se realiza un promedio de todos los tiempos existentes en el buffer.

$$t_{prom} = \frac{\sum_{i=0}^{4} t_i}{5}$$

$$h = t_{prom}$$

3.9.3.1 Base de reglas

Para la creación de las reglas de la posición en “x” se consideró:

- Cuando la posición actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron se encuentra al lado derecho de la posición deseada.
- Cuando la posición actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra al lado izquierdo de la posición deseada.
- Cuando la variación de la posición del dron es negativa este va hacia adelante.
- Cuando la variación de la posición del dron es positiva este va hacia atrás.

Con estas explicaciones, para la implementación de la base de reglas en "x" se tomó siete posibles antecedentes tanto para el error y la derivada del mismo. A su vez se tomó el mismo número de consecuentes para la salida del controlador, que son interpretados como se muestra en la Tabla 14:

Tabla 14

Valores para controlador PD de X

	$\boldsymbol{e(t)}$	$\boldsymbol{\frac{de(t)}{dt}}$	$\boldsymbol{out}$
PG	Posición muy atrás	Velocidad rápida hacia atrás	Fuerza grande adelante
PM	Posición media atrás	Velocidad media hacia atrás	Fuerza media adelante
PP	Posición pequeña atrás	Velocidad pequeña hacia atrás	Fuerza pequeña adelante
ZE	Posición deseada	Sin cambio de velocidad	Sin fuerza
NP	Posición pequeña adelante	Velocidad pequeña hacia adelante	Fuerza pequeña atrás
NM	Posición media adelante	Velocidad mediana hacia adelante	Fuerza media atrás
NG	Posición muy adelante	Velocidad rápida hacia adelante	Fuerza grande atrás

Las reglas implementadas para el error en "x" y su variación modifican el ángulo de giro pitch y son:

- **Si** e es NG **y** e' es NG **entonces** out es NG
- **Si** e es NG **y** e' es NM **entonces** out es NG
- **Si** e es NG **y** e' es NP **entonces** out es NM
- **Si** e es NG **y** e' es ZE **entonces** out es NG
- **Si** e es NG **y** e' es PP **entonces** out es NG
- **Si** e es NG **y** e' es PM **entonces** out es NM
- **Si** e es NG **y** e' es PG **entonces** out es NP
- **Si** e es NM **y** e' es NG **entonces** out es NG

- **Si** e es NM **y** e' es NM **entonces** out es NM
- **Si** e es NM **y** e' es NP **entonces** out es NP
- **Si** e es NM **y** e' es ZE **entonces** out es NM
- **Si** e es NM **y** e' es PP **entonces** out es NM
- **Si** e es NM **y** e' es PM **entonces** out es NP
- **Si** e es NM **y** e' es PG **entonces** out es NP
- **Si** e es NP **y** e' es NG **entonces** out es NM
- **Si** e es NP **y** e' es NM **entonces** out es NP
- **Si** e es NP **y** e' es NP **entonces** out es NP
- **Si** e es NP **y** e' es ZE **entonces** out es NP
- **Si** e es NP **y** e' es PP **entonces** out es NP
- **Si** e es NP **y** e' es PM **entonces** out es NP
- **Si** e es NP **y** e' es PG **entonces** out es NP
- **Si** e es ZE **y** e' es NG **entonces** out es NP
- **Si** e es ZE **y** e' es NM **entonces** out es NP
- **Si** e es ZE **y** e' es NP **entonces** out es ZE
- **Si** e es ZE **y** e' es ZE **entonces** out es ZE
- **Si** e es ZE **y** e' es PP **entonces** out es ZE
- **Si** e es ZE **y** e' es PM **entonces** out es PP
- **Si** e es ZE **y** e' es PG **entonces** out es PP
- **Si** e es PP **y** e' es NG **entonces** out es PP
- **Si** e es PP **y** e' es NM **entonces** out es PP

- **Si** e es PP **y** e' es NP **entonces** out es PP
- **Si** e es PP **y** e' es ZE **entonces** out es PP
- **Si** e es PP **y** e' es PP **entonces** out es PP
- **Si** e es PP **y** e' es PM **entonces** out es PP
- **Si** e es PP **y** e' es PG **entonces** out es PM
- **Si** e es PM **y** e' es NG **entonces** out es PP
- **Si** e es PM **y** e' es NM **entonces** out es PP
- **Si** e es PM **y** e' es NP **entonces** out es PM
- **Si** e es PM **y** e' es ZE **entonces** out es PM
- **Si** e es PM **y** e' es PP **entonces** out es PP
- **Si** e es PM **y** e' es PM **entonces** out es PM
- **Si** e es PM **y** e' es PG **entonces** out es PG
- **Si** e es PG **y** e' es NG **entonces** out es PP
- **Si** e es PG **y** e' es NM **entonces** out es PM
- **Si** e es PG **y** e' es NP **entonces** out es PG
- **Si** e es PG **y** e' es ZE **entonces** out es PG
- **Si** e es PG **y** e' es PP **entonces** out es PM
- **Si** e es PG **y** e' es PM **entonces** out es PG
- **Si** e es PG **y** e' es PG **entonces** out es PG

Por lo que en la Tabla se muestran las reglas para el ángulo pitch.

Tabla 15
Reglas control PD de ángulo pitch

e' / e	NG	NM	NP	ZE	PP	PM	PG
NG	NG	NG	NG	NG	NM	NM	NM
NM	NG	NG	NM	NM	NM	NM	NM
NP	NG	NM	NM	NP	NP	NP	NP
ZE	NP	NP	ZE	ZE	ZE	PP	PP
PP	PP	PP	PP	PP	PM	PM	PG
PM	PM	PM	PM	PM	PM	PG	PG
PG	PP	PM	PM	PG	PG	PG	PG

Para la creación de las reglas de la posición en "y" se consideró:

- Cuando la posición actual del dron es menor al valor set point, el error será positiva entendiéndose que el dron esta atrás de la posición deseada.
- Cuando la posición actual del dron es mayor al valor set point, el error será negativo entendiéndose que el dron se encuentra delante de la posición deseada.
- Cuando la variación de la posición del dron es negativa este va hacia la derecha.
- Cuando la variación de la posición del dron es positiva este va hacia la izquierda.

Con estas explicaciones, se tomó los mismos antecedentes y consecuentes que en el error en "x" y son interpretados como se muestra en la Tabla 16:

Tabla 16
Valores para controlador PD de Y

$e(t)$		$\frac{de(t)}{dt}$	out
PG	Posición muy a la derecha	Velocidad rápida hacia izquierda	Fuerza grande hacia la derecha

CONTINUA

PM	Posición media a la derecha	Velocidad media hacia izquierda	Fuerza media hacia la derecha
PP	Posición pequeña a la derecha	Velocidad pequeña hacia izquierda	Fuerza pequeña hacia la derecha
ZE	Posición deseada	Sin cambio de velocidad	Sin fuerza
NP	Posición pequeña a la izquierda	Velocidad pequeña hacia derecha	Fuerza pequeña hacia la izquierda
NM	Posición media a la izquierda	Velocidad mediana hacia derecha	Fuerza media hacia la izquierda
NG	Posición muy a la izquierda	Velocidad rápida hacia derecha	Fuerza grande hacia la izquierda

Las reglas implementadas para el error en "y" y su variación modifican el ángulo de giro roll y son:

- **Si** e es NG **y** e' es NG **entonces** out es PG
- **Si** e es NG **y** e' es NM **entonces** out es PG
- **Si** e es NG **y** e' es NP **entonces** out es PM
- **Si** e es NG **y** e' es ZE **entonces** out es PG
- **Si** e es NG **y** e' es PP **entonces** out es PG
- **Si** e es NG **y** e' es PM **entonces** out es PM
- **Si** e es NG **y** e' es PG **entonces** out es PP
- **Si** e es NM **y** e' es NG **entonces** out es PG
- **Si** e es NM **y** e' es NM **entonces** out es PM
- **Si** e es NM **y** e' es NP **entonces** out es PP
- **Si** e es NM **y** e' es ZE **entonces** out es PM
- **Si** e es NM **y** e' es PP **entonces** out es PM
- **Si** e es NM **y** e' es PM **entonces** out es PP
- **Si** e es NM **y** e' es PG **entonces** out es PP

- **Si** *e* es NP **y** *e′* es NG **entonces** *out* es PM
- **Si** *e* es NP **y** *e′* es NM **entonces** *out* es PG
- **Si** *e* es NP **y** *e′* es NP **entonces** *out* es PP
- **Si** *e* es NP **y** *e′* es ZE **entonces** *out* es PP
- **Si** *e* es NP **y** *e′* es PP **entonces** *out* es PP
- **Si** *e* es NP **y** *e′* es PM **entonces** *out* es PP
- **Si** *e* es NP **y** *e′* es PG **entonces** *out* es PP
- **Si** *e* es ZE **y** *e′* es NG **entonces** *out* es PP
- **Si** *e* es ZE **y** *e′* es NM **entonces** *out* es PP
- **Si** *e* es ZE **y** *e′* es NP **entonces** *out* es ZE
- **Si** *e* es ZE **y** *e′* es ZE **entonces** *out* es ZE
- **Si** *e* es ZE **y** *e′* es PP **entonces** *out* es ZE
- **Si** *e* es ZE **y** *e′* es PM **entonces** *out* es NP
- **Si** *e* es ZE **y** *e′* es PG **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es NG **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es NM **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es NP **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es ZE **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es PP **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es PM **entonces** *out* es NP
- **Si** *e* es PP **y** *e′* es PG **entonces** *out* es NM
- **Si** *e* es PM **y** *e′* es NG **entonces** *out* es NP

- **Si** e es PM **y** e' es NM **entonces** out es NP
- **Si** e es PM **y** e' es NP **entonces** out es NM
- **Si** e es PM **y** e' es ZE **entonces** out es NM
- **Si** e es PM **y** e' es PP **entonces** out es NP
- **Si** e es PM **y** e' es PM **entonces** out es NM
- **Si** e es PM **y** e' es PG **entonces** out es NG
- **Si** e es PG **y** e' es NG **entonces** out es NP
- **Si** e es PG **y** e' es NM **entonces** out es NM
- **Si** e es PG **y** e' es NP **entonces** out es NG
- **Si** e es PG **y** e' es ZE **entonces** out es NG
- **Si** e es PG **y** e' es PP **entonces** out es NM
- **Si** e es PG **y** e' es PM **entonces** out es NG
- **Si** e es PG **y** e' es PG **entonces** out es NG

En la Tabla 17 se muestran las reglas para el ángulo pitch.

Tabla 17

Reglas control PD de ángulo roll

e' / e	**NG**	**NM**	**NP**	**ZE**	**PP**	**PM**	**PG**
NG	PM	PM	PG	PG	PG	PG	PG
NM	PM	PM	PM	PM	PG	PG	PG
NP	PP	PP	PP	PP	PP	PM	PG
ZE	NP	NP	ZE	ZE	ZE	PP	PP
PP	NG	NM	NP	NP	NP	NP	NP
PM	NG	NG	NG	NM	NM	NM	NM
PG	NG	NG	NG	NG	NG	NM	NM

3.9.3.2 Mecanismo de inferencia

El proceso o mecanismo de inferencia utilizado por la librería previamente mencionada en el diseño del controlador P, realiza los mismos pasos, con la diferencia de que al ser dos entradas se puede obtener un máximo de cuatro reglas a la vez.

3.9.3.3 Interfaz de fuzzificación

Como antecedentes o entradas del sistema se tiene al error y la velocidad con la que este varia, siendo los valores que puede tomar el error entre -3 y 3m, y los valores de la derivada del error entre -1 y 1. El universo de estas entradas se encuentra definido en la Tabla 18.

Tabla 18

Universo de valores de las entradas controlador PD

	$\boldsymbol{e(t)}$	$\boldsymbol{\frac{de(t)}{dt}}$
PG	$2.00 < e$	$0.66 < e' < 1.00$
PM	$1.00 < e < 3.00$	$0.33 < e' < 1.00$
PP	$0.00 < e < 2.00$	$0.00 < e' < 0.67$
ZE	$-1.00 < e < 1.00$	$-0.33 < e' < 0.33$
NP	$0.00 < e < -2.00$	$-0.67 < e' < 0.00$
NM	$-1.00 < e < -3.00$	$-1.00 < e' < -0.33$
NG	$-2.00 < e$	$-1.00 < e' < -0.66$

Las funciones de pertenencia para la entrada del error son las mismas que en el caso del control P, que se muestran en la Tabla 12, y las funciones de pertenencia para la entrada de la derivada del error se muestran en la Tabla 19.

Tabla 19

Funciones de pertenencia de la derivada del error

	$\frac{de(t)}{dt}$	
PG	$\mu_{PG} = \frac{e' - 0.66}{0.33}, si\ 0.66 < e' < 1.00$	$\mu_{PG} = 1\ , si\ e' > 1.00$
PM	$\mu_{PM} = \frac{e' - 0.33}{0.33}, si\ 0.33 < e' < 0.66$	$\mu_{PM} = \frac{1 - e'}{0.33}, si\ 0.66 < e' < 1.00$
PP	$\mu_{PP} = \frac{e' - 0.00}{0.33}, si\ 0.00 < e' < 0.33$	$\mu_{PP} = \frac{0.66 - e'}{0.33}, si\ 0.33 < e' < 0.66$
ZE	$\mu_{ZE} = \frac{e' + 0.33}{0.33}, si - 0.33 < e' < 0.00$	$\mu_{ZE} = \frac{0.33 - e'}{0.33}, si\ 0.00 < e' < 0.33$
NP	$\mu_{NP} = \frac{e' + 0.66}{0.33}, si - 0.66 < e' < -0.33$	$\mu_{NP} = \frac{0 - e'}{0.33}, si - 0.33 < e' < 0.00$
NM	$\mu_{NM} = \frac{e' + 1.00}{0.33}, si - 1.00 < e' < -0.66$	$\mu_{NM} = \frac{-0.33 - e'}{0.33}, si - 0.66 < e' < -0.33$
NG	$\mu_{NG} = 1\ , si\ e' < -1.00$	$\mu_{NG} = \frac{-0.66 - e'}{0.33}, si - 1 < e' < -0.66$

Los conjuntos difusos para cada valor lingüístico se crean al igual que en el control P para el error y el error en yaw. Estos se encuentran representados en (Figura 33) y (Figura 34) respectivamente; mientras que para a entrada de la derivada se muestra en la (Figura 37).

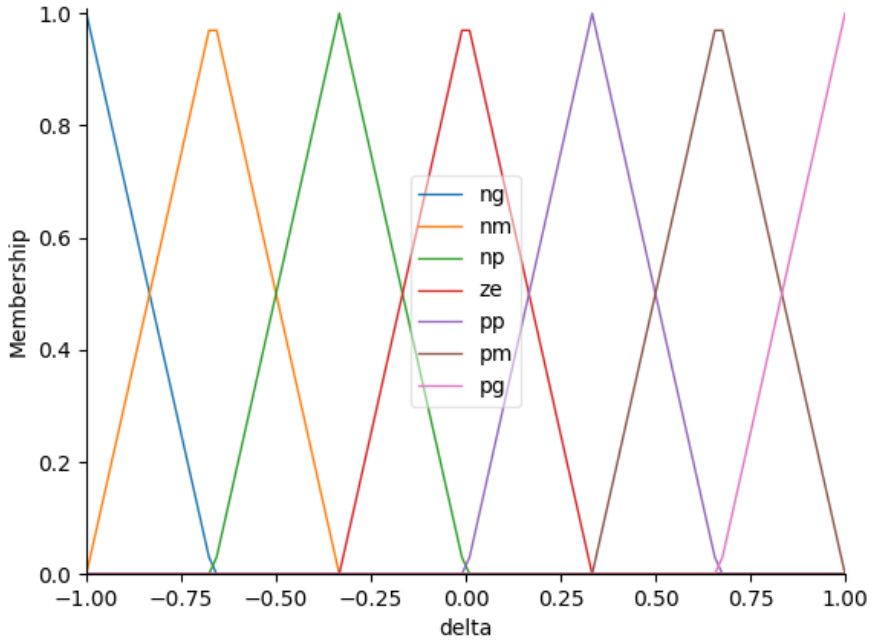


Figura 37. Conjuntos difusos de la derivada del error.

3.9.3.4 Interfaz de defuzzificación

La interfaz de defuzzificación es similar a la del control P con la diferencia que se aumentan dos valores lingüísticos ubicados entre valores pequeños y grandes, para obtener una mayor precisión. Los valores de salida se indican en la Tabla 13.

Tabla 20

Universo de valores de la salida

	out
PG	$0.66 < out$
PM	$0.33 < out < 1.00$
PP	$0.00 < out < 0.66$
ZE	$-0.33 < out < 0.33$
NP	$-0.66 < out < 0.00$
NM	$-1.00 < out < -0.33$
NG	$out < -0.66$

Por lo que los conjuntos difusos quedan representados por la Figura 38.

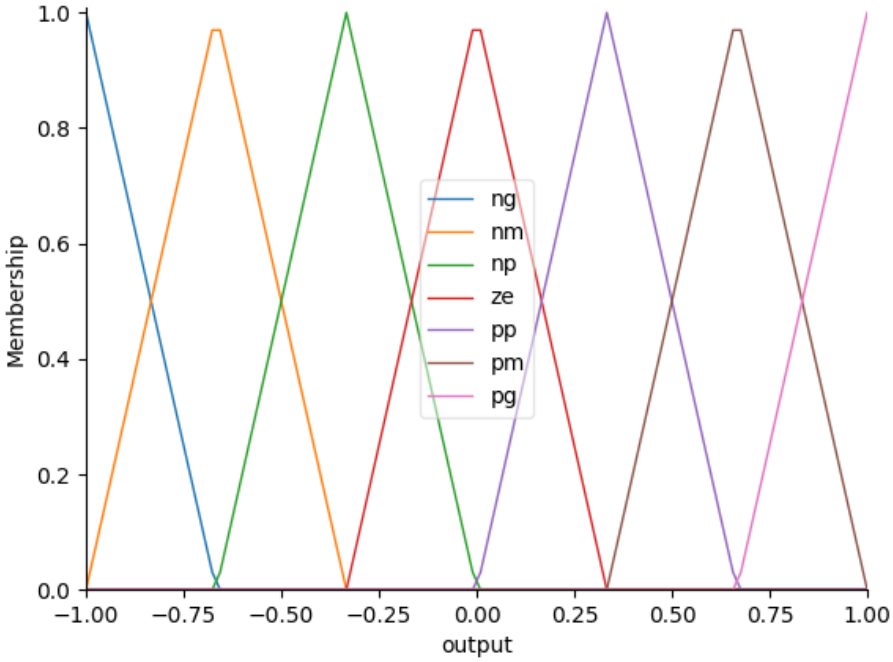


Figura 38. Conjuntos difusos de salida controlador PD

Para que la toma de decisiones en tiempo real sea rápida y no requiera un proceso de calcular cual es la salida para cada combinación de entradas, se realizó una simulación previa del controlador obteniendo un conjunto de datos de salida representados en una superficie de control, misma que se encuentra representada en la Figura 39 para el controlador de pitch y throttle, y en el Figura 40 para los controladores de roll y yaw.

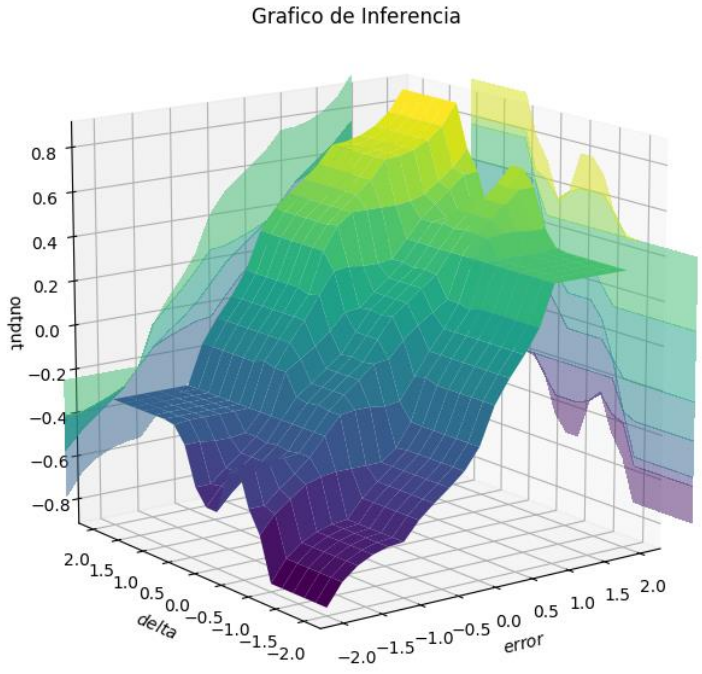


Figura 39. Gráfico de inferencia controlador pitch y throttle.

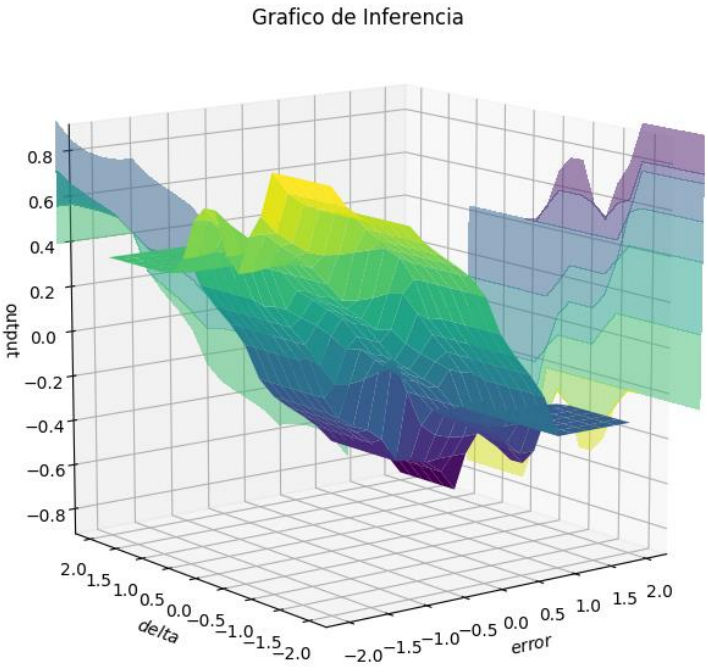


Figura 40. Gráfico de inferencia controlador roll y yaw.

3.10 Control de Trayectoria

El presente proyecto tiene como finalidad la reconstrucción de un objeto por lo cual la trayectoria debe ser entorno al mismo y enfocándolo. Se definió dos tipos de trayectorias, una cuadrada y una hexagonal. Para ambas trayectorias se realiza un desplazamiento que puede ser en el eje "x" como en "y" a una distancia constante definida por el usuario, tomando en cuenta que el objeto se encuentra en el centro. Para la generación automática de la trayectoria el algoritmo calcula los ángulos de separación de los puntos dependiendo si es un cuadrado o un hexágono, dividiendo los ángulos de forma simétrica y calculando los puntos "n" a donde se debe dirigir el dron.

$$\theta_i = \frac{360}{n} \cdot i$$

$$x_i = r\cos(\theta_i)$$

$$y_i = r\,sen(\theta_i)$$

Una vez conocidos los puntos el dron se dirige en traslación al primer punto y cuando el error de la posición es menor a 0.1m el dron comienza a rotar para enfocar al punto de despegue donde se encuentra el objeto. Cuando el error en rotación es menor a 0.2rad el dron se traslada al siguiente punto. Este procedimiento se repite dependiendo el número de puntos definidos en la trayectoria (Figura 41 y Figura 42); terminando el dron en el primer punto.

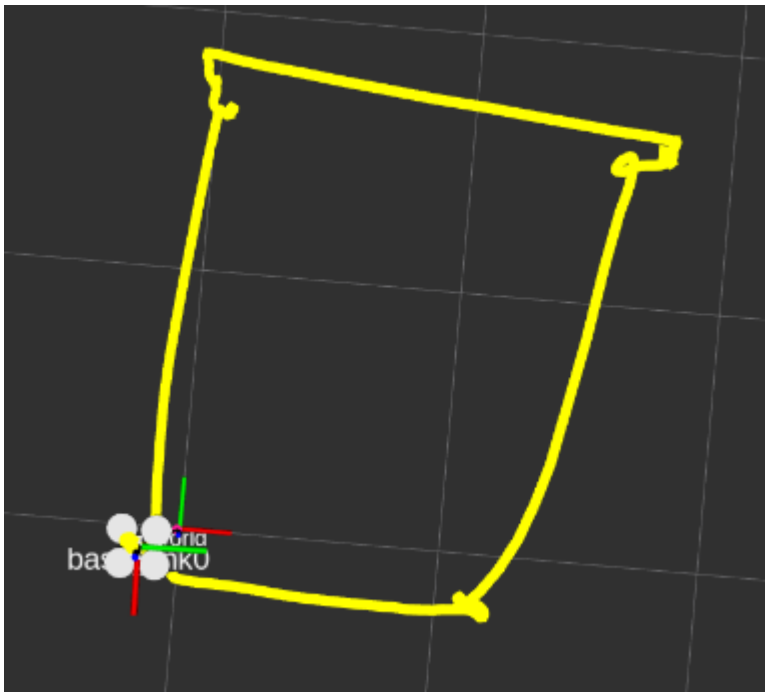


Figura 41. Trayectoria cuadrada con r =0.8

Figura 42. Trayectoria hexagonal con r = 0.9

Para que a lo largo de la trayectoria el dron pueda tener un control de estabilidad y actuar frente a perturbaciones, se tomó en cuenta que al rotar en cada punto lo que el dron conoce como error en x y error en y va cambiando según el ángulo de rotación, por lo que se tuvo que multiplicar el error por la inversa de la matriz de rotación en el eje z; encontrando así nuevos vectores de error que entran a los controladores de estabilidad.

$$\begin{bmatrix} e_x \\ e_y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -sen(\theta) & 0 & 0 \\ sen(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} e_x \\ e_y \\ 0 \\ 1 \end{bmatrix}$$

De tal manera el nuevo error rotado se debe comparar con su posición real en el eje de coordenadas correspondiente, por lo que el control de trayectoria y estabilidad trabajan en cascada como se muestra en el esquema de la Figura 43.

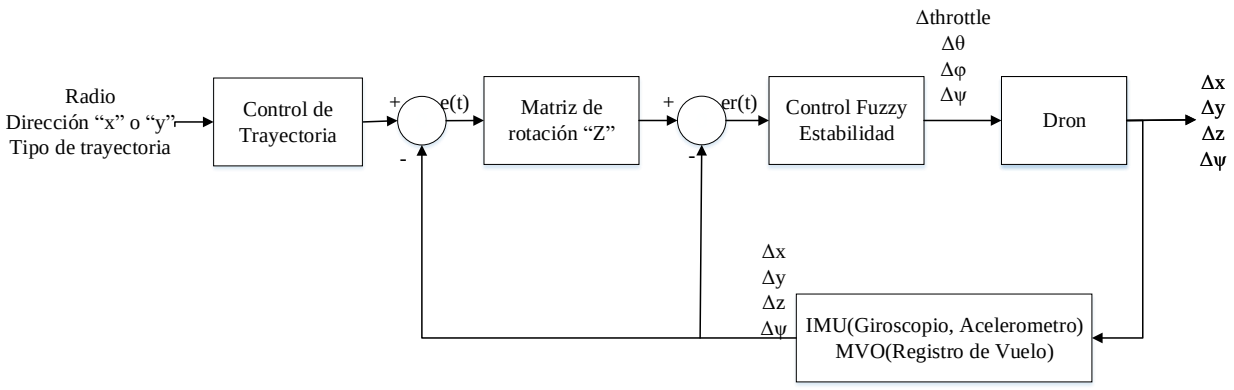


Figura 43. Esquema de control de trayectoria

3.11 Interfaz Grafica

Para el envío de los parámetros de vuelo hacia el dron y la recepción de la cámara se ha realizado la interfaz que se muestra en Figura 44, la cual consta de 9 check box, 4 botones y 5 casillas de texto.

Dicha interfaz fue realizada mediante la librería Tkinter de OpenCV y se inicializa al correr el comando *roslaunch Ryze_d3089e.launch* en la terminal del computador central; el cual también abre el entorno de RViz para la visualización de la trayectoria que sigue el dron.

Los botones:

- Start: carga la información de los check box, casillas de texto e inicializa el vuelo.
- Quit: cierra la interfaz, guarda las fotos recopiladas y abre el gráfico de errores y salidas del controlador.
- Land: Aterriza el dron
- Takeof: despegue del dron

Los tipos de vuelo del dron son:

- Manual: el dron volara con un control PS4 conectado mediante cable a la PC.
- Automático: puede ser de estabilidad para mandar al dron a un punto o de trayectoria para volar el dron alrededor de un punto. Si es de estabilidad se deberá ingresar la referencia en “x, y, z, yaw” y escoger el tipo de controlador entre P o PD. Mientras si es de trayectoria se debe escoger entre cuadrado o hexágono, el tipo de controlador, el eje de desplazamiento entre “x” o “y” y llenar el radio.

La interfaz también muestra valores obtenidos del dron como porcentaje de batería, posición, velocidad, altitud y la señal de wifi.

Figura 44.Interfaz de vuelo

CAPITULO IV

IMPLEMENTACION DE RECONSTRUCCION 3D

4.1 Calibración cámara externa

La calibración de la cámara se realiza para obtener los parámetros intrínsecos de la misma, los cuales son: las distancias focales, las coordenadas del centro de la cámara y los coeficientes de distorsión. Estos se reconocen mediante un nodo en ROS llamado camera_calibration, el cual es comúnmente utilizado para cámaras monoculares y utiliza como herramienta principal el enfoque a una imagen de tablero de ajedrez 8x6 (Figura 45), con lo cual permite obtener los parámetros de la cámara en matrices de: distorsión, proyección y rectificación.

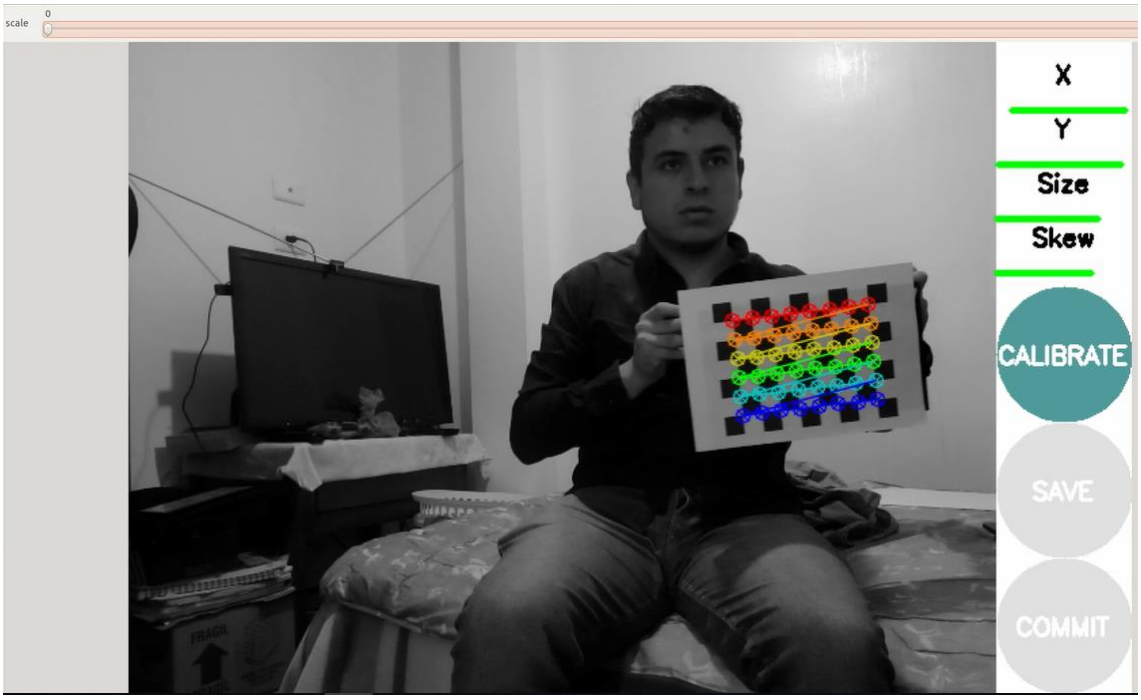


Figura 45. Calibración cámara del dron

4.1.1 Parámetros de la cámara

Para la cámara utilizada se tiene la siguiente matriz de parámetros intrínsecos:

$$K = \begin{bmatrix} 1429.233876 & 0 & 668.392476 \\ 0 & 1431.179753 & 405.542988 \\ 0 & 0 & 1 \end{bmatrix}$$

Los valores de distorsión obtenidos mediante la calibración son:

$$d = [0.062458 \quad 0.031584 \quad 0.008856 \quad 0.003121 \quad 0.0]$$

La matriz de rectificada de distorsión o matriz de proyección es:

$$K_c = \begin{bmatrix} 1458.054321 & 0 & 670.279330 \\ 0 & 1456.422363 & 409.567702 \\ 0 & 0 & 1 \end{bmatrix}$$

4.2 Identificación del objeto

Para lograr la identificación del objeto en una imagen se utilizó la API TensorFlow la cual tiene la característica de ser de código abierto y permite crear redes de aprendizaje profundo que resuelve el problema de la detección. Existen modelos ya entrenados o se puede construir un modelo propio. En el presente proyecto se seleccionó un modelo ya entrenado “mobilenet_v1_coco_2018_01_28”, el cual es un conjunto de datos de detección, segmentación y subtitulado de objetos a gran escala que entrega un archivo binario con extensión .pb que contiene tanto la topología como los pesos de la red entrenada para dicha tarea. En esta colección se identifican 90 clases u objetos, con su respectivo porcentaje de similitud al ser detectados Figura 46.

Figura 46. Detección de objetos

Ya con la identificación del objeto y al enmarcarlo en un cuadro delimitador se toma la posición de los vértices para calcular una aproximación de la distancia que hay entre la cámara y el objeto Figura 47. Esto depende del ancho de dicho recuadro y la granularidad (enfoque de la luz al objeto). La fórmula para calcular la distancia es:

$$d = (1 - (x_{max} - x_{min}))^4$$

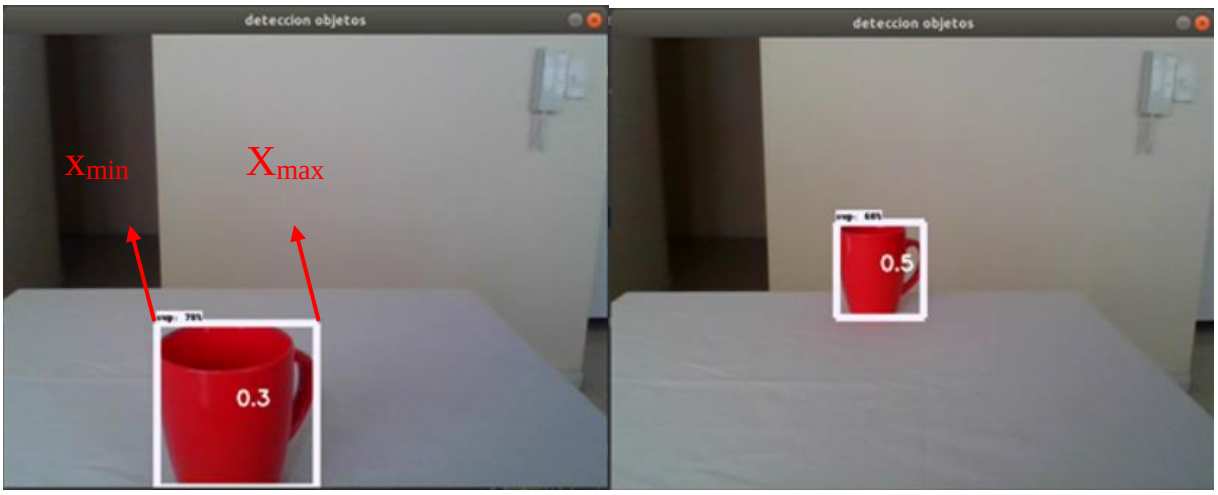


Figura 47. Distancia del objeto a la cámara

Como la identificación de objetos se realiza en tiempo real durante el vuelo del dron, se puede observar la distancia que existe entre el dron y el objeto.

4.3 Captura de imagen

Para realizar la captura de imágenes se verifico el objeto al que se desea realizar la reconstrucción y gracias a la identificación de objetos se guarda las imágenes donde exista un 35% de similitud con el objeto deseado, descartando así las fotografías que toma el dron sin enfocar al objeto. De esta manera se reduce significativamente el número de imágenes guardadas durante la trayectoria del dron en un 50%.

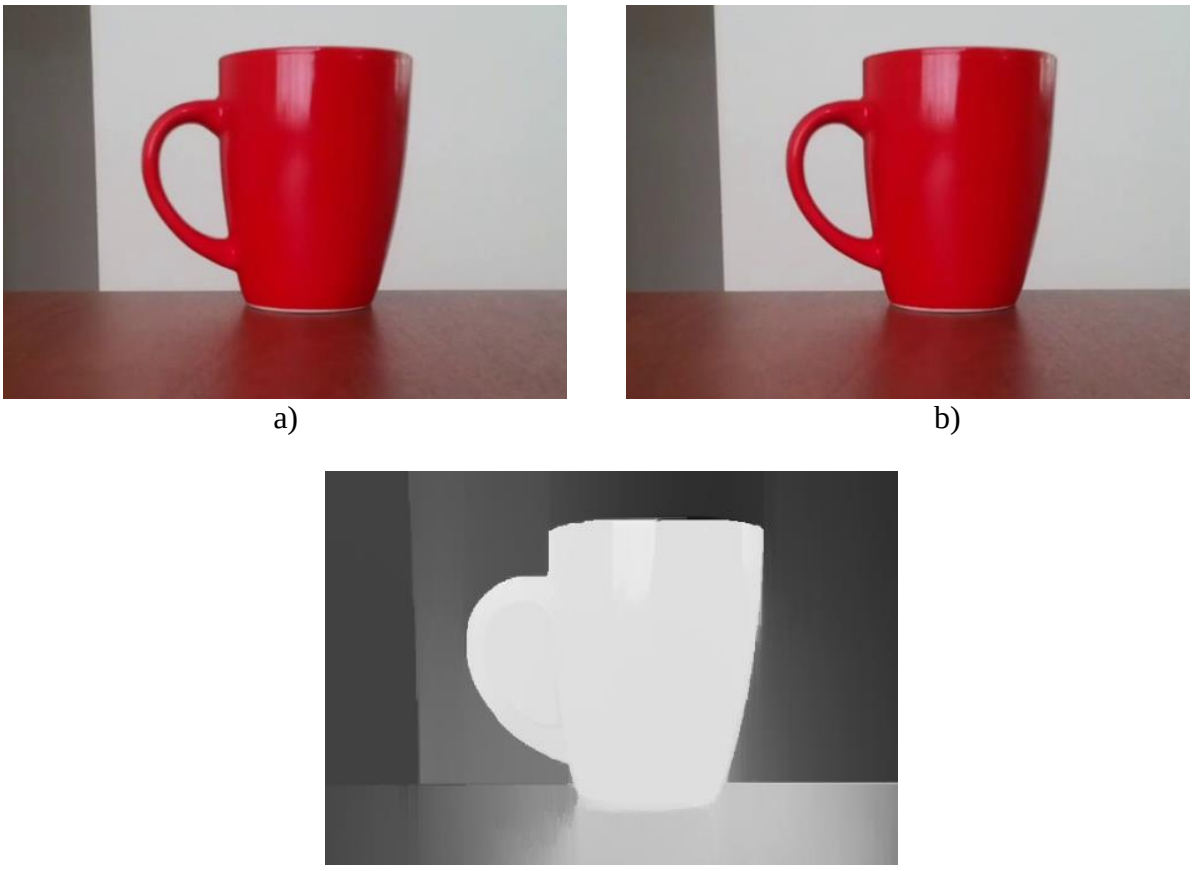
4.4 Correlación de imágenes

Para encontrar la correlación de imágenes el sistema emula el comportamiento de visión estéreo mediante dos imágenes tomadas con la cámara monocular del dron. Esto quiere decir que toma dos imágenes consecutivas, encuentra puntos característicos existentes en ambas imágenes y calcula la distancia que se movieron estos puntos de una imagen a otra.

4.5 Mapa de profundidad

Para realizar el mapa de profundidad se calcula la disparidad entre un par de imágenes logrando obtener la posición en “x”, “y” y “z” de cada pixel. Esto es posible mediante la correlación ya realizada de todos los puntos en cada imagen. La disparidad de un pixel es igual al valor de desplazamiento que lleva a la suma mínima de diferencias al cuadrado para cada uno de estos (Rambhia, 2013). De esta forma se construye un mapa de densidad de disparidades (Figura 48).

Para calcular el mapa de disparidad se utilizó una función en OpenCV llamada SterioSGBM que significa algoritmo de coincidencias de semi-bloques, el cual es una modificación del algoritmo de correspondencia estéreo propuesto en (Hirschmuller, 2008).



a) b)

c)

Figura 48. a) Imagen izquierda. b) Imagen derecha. c) Mapa de disparidad

En este mapa de profundidad realizado se puede observar en escala de grises los elementos según su distancia a la cámara, es decir el elemento que se encuentra en un tono más oscuro está más alejado.

Además, se utilizó un filtro WLS (filtro de mínimos cuadrados ponderados) el cual utiliza dos mapas de disparidad, izquierda y derecha; permitiendo obtener una imagen de profundidad completa y libre de orificios (Samartzidis, 2017).

4.6 Reconstrucción 3D

Para la reconstrucción 3D se obtiene primeramente los datos de todos los pixeles de la imagen capturada por el dron, posición (x,y) y color (r,g,b) creando una matriz de 691200x5, ya que es una imagen de 960x720. A esta matriz se le agrega una columna con el dato de profundidad z que proviene del mapa de profundidad. Con esto se reconstruye el objeto a color y en sus 3 dimensiones.

Para la visualización se graficó todos los pixeles de la nueva matriz, pero al ser demasiados y algunos tienen valores de profundidad infinitos, el costo computacional sobrepaso las prestaciones del computador utilizado en el proyecto, por lo que se utilizó el entorno JupyterNotebook y la función pycloud para mostrar la visualización del objeto en 3 dimensiones, como se muestra en la Figura 49.

Figura 49. Visualización reconstrucción 3D

CAPITULO V

PRUEBAS Y RESULTADOS

5.1 Pruebas de estabilidad control fuzzy P

5.1.1 Prueba de desplazamiento

Para esta prueba se realizó 10 ensayos en las mismas condiciones en un espacio controlado, enviando al dron a un punto fijo x=2.0, y=1.5 y z=2.0 y con el controlador fuzzy P. Los resultados de la prueba se presentan en la Tabla 21 y fueron tomados cuando el dron se ha desplazado y se mantiene estable en una posición.

Tabla 21

Desplazamiento con control P

Set Point (x=2.0, y=1.5 y z=2.0)							
# Prueba	Distancias Real						Tiempo de ejecución (s)
	x(m)	Error en x (%)	y(m)	Error en y (%)	z(m)	Error en z (%)	
1	2,25	12,50	2,10	40,00	2,10	5,00	21,7
2	2,70	35,00	1,90	26,67	1,80	10,00	22
3	2,30	15,00	1,80	20,00	1,80	10,00	21,1
4	2,15	7,50	1,85	23,33	2,05	2,50	25
5	2,50	25,00	2,00	33,33	2,00	0,00	20
6	2,65	32,50	1,75	16,67	2,15	7,50	23,5
7	2,38	19,00	2,00	33,33	2,10	5,00	21,2
8	2,19	9,50	1,80	20,00	1,90	5,00	22,1
9	2,20	10,00	2,20	46,67	1,95	2,50	24,3
10	2,80	40,00	1,70	13,33	1,70	15,00	21
Promedio	2,41	20,60	1,91	27,33	1,96	6,25	22,19

Dentro del tiempo de ejecución consta el tiempo que tarda el dron en recibir los comandos de vuelo, el despegue y estabilización en el punto inicial y el tiempo de vuelo hasta llegar al punto definido. Cabe recalcar que el tiempo de vuelo aumentara si el set point es mayor. Se observo que el tiempo de estabilización del sistema es igual al tiempo de vuelo, y este es alrededor de 8 segundos para las distancias de esta prueba.

El error en estado estacionario para la posición en “x” es de 0,41m que corresponde a un 20.5%; en la posición en “y” es de 0,41m que corresponde a un 27% de error; y en “z” es de 0,04m que

corresponde a un 2% de error. La Figura 50 corresponde al ensayo 5 y se muestra el error y el comportamiento del controlador

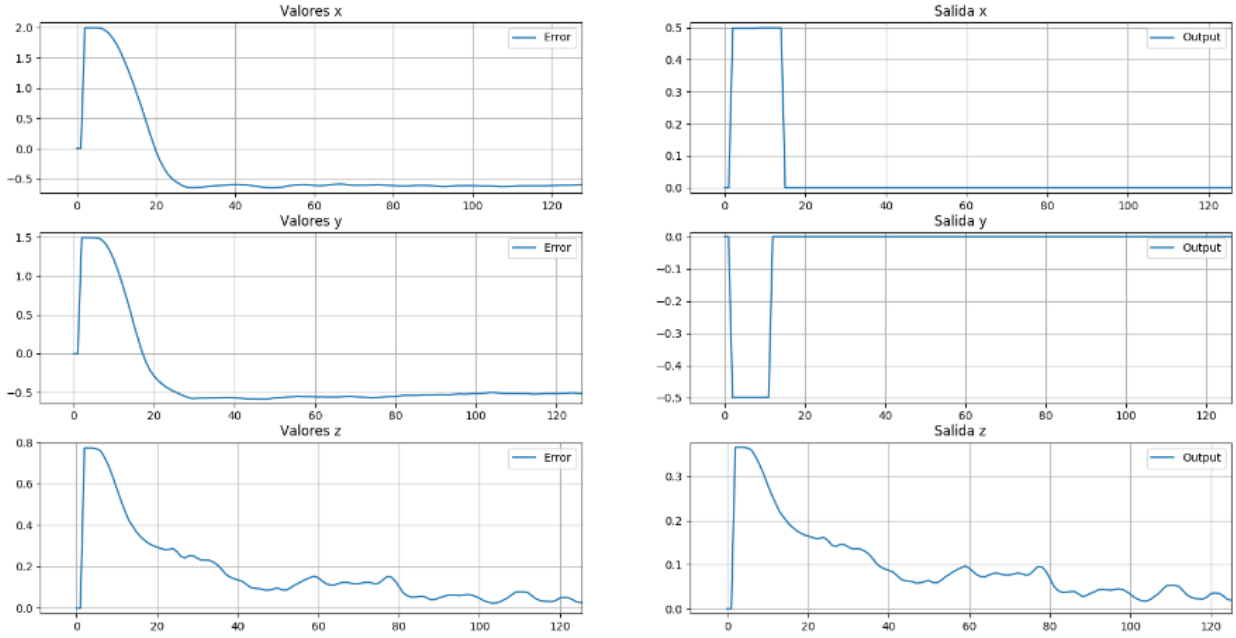


Figura 50. Prueba desplazamiento controlador P

Resultado:

La tendencia del error en el desplazamiento de “x” y “y” es sobrepasar el set point, y se debe a la inercia del movimiento cuando se deja de enviar una señal de control; el error en “z” es mínimo y este comienza con un desface de 1,2 que es la altura de vuelo en su despegue. El controlador P para el desplazamiento es brusco y no responde bien a errores menores a 0,6m.

5.1.2 Prueba frente a perturbaciones

Para esta prueba se repitió un ensayo de la prueba anterior con la diferencia que cuando el dron estaba en estado estable se le ejerció una fuerza moviendo al dron de dicha posición. La señal de error y la salida de control se pueden observar en la Figura 51.

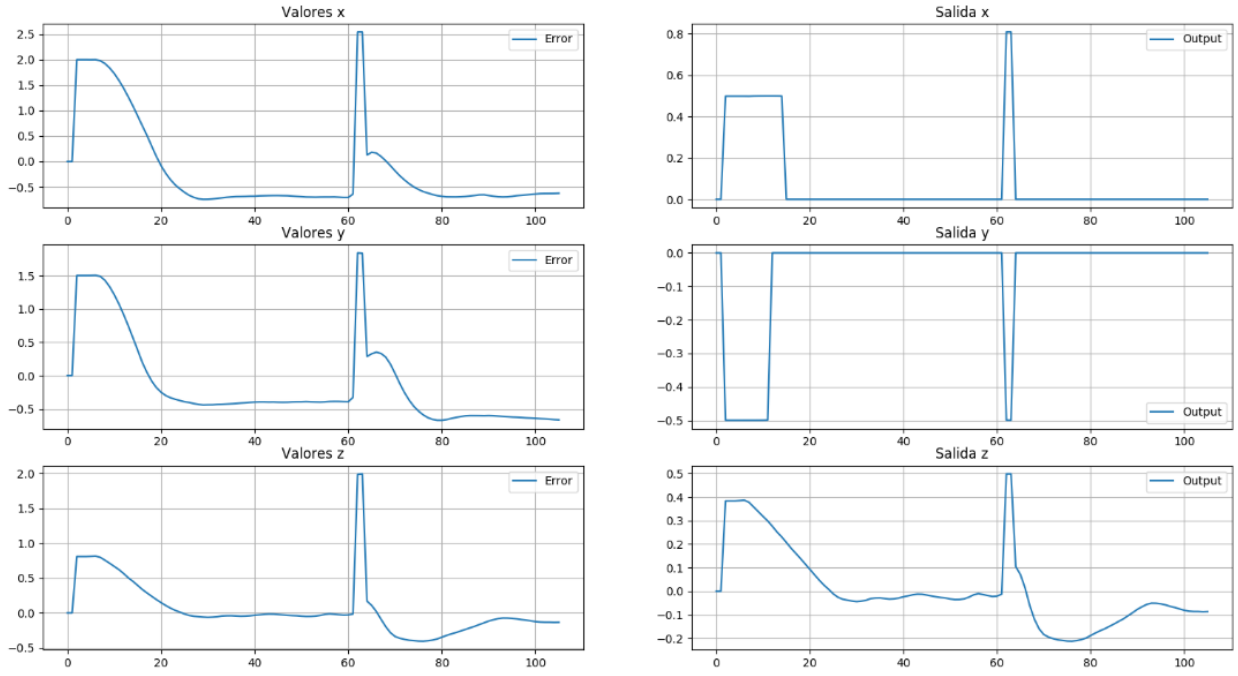


Figura 51. Prueba frente a perturbaciones controlador P

Resultado:

En la figura se puede observar que el control P de estabilidad responde a perturbaciones, pero mantiene el error en estado estacionario y da un sobre impulso alto.

5.1.3 Prueba de orientación

Para esta prueba se realizó el despegue del dron y se le envió un giro de 0.78 radianes en su posición inicial. Una vez estable se le aplico una perturbación. En la Figura 52 muestra el error inicial de 0.78 radianes, la estabilización, el error frente a la perturbación y la señal de control frente a estos errores.

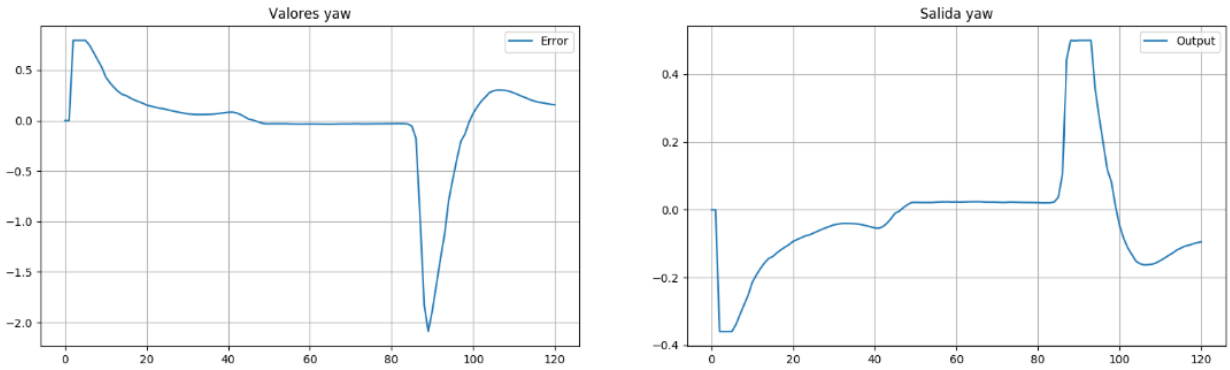


Figura 52. Prueba orientación controlador P

Resultado:

El controlador de orientación o ángulo yaw tiene una mayor presión que los controladores P de desplazamiento. Presenta un error en estado estacionario cercano al cero y responde positivamente a perturbaciones externas.

5.2 Pruebas de estabilidad control fuzzy PD

5.2.1 Pruebas de derivadas

Para el proyecto realizado se ha probado con seis fórmulas de derivación numérica que se muestran en 2, tomando para las pruebas una función seno y estimando el error en cada una de estas.

Al utilizar la fórmula de derivación progresiva de dos puntos (Figura 53) en la función seno se obtiene un error alto equivalente al valor de 4.0 dado que utiliza solo el punto en el que se desea calcular la derivada y el punto siguiente, entendiéndose que “h” es la diferencia en el tiempo entre estos puntos. Aumentando el número de puntos utilizados a tres (Figura 54) el error sigue presente, pero disminuye a 0.035.

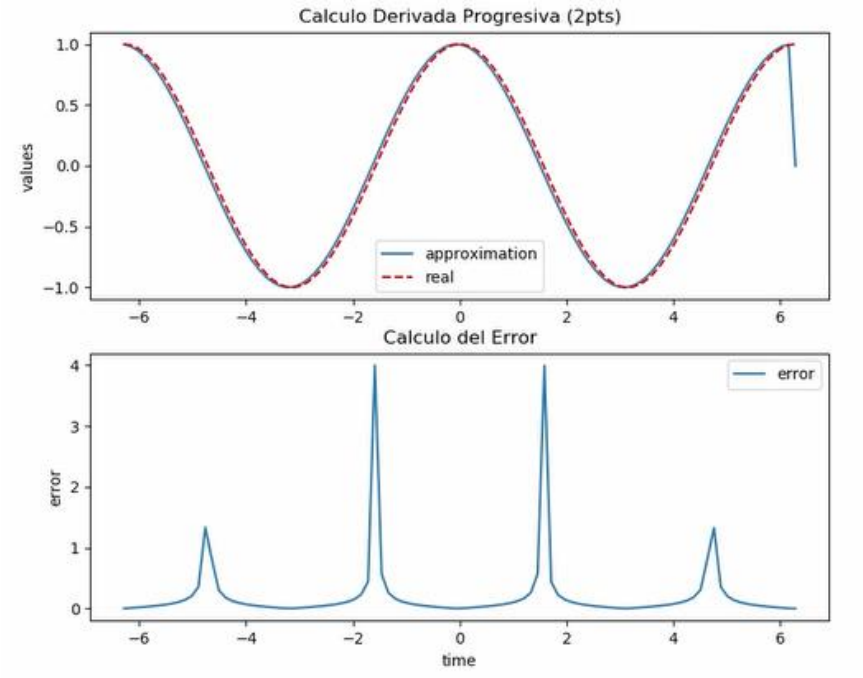


Figura 53. Derivada progresiva de 2 puntos

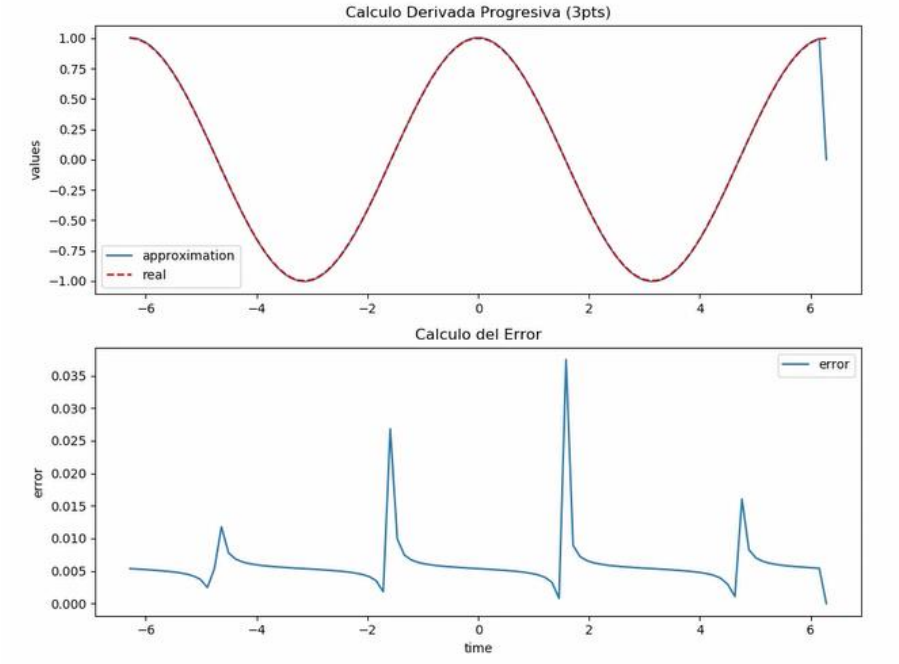


Figura 54. Derivada progresiva de 3 puntos

Al utilizar la fórmula de derivación centrada de dos puntos (Figura 55) en la función seno se obtiene un error inicial con un valor de 0.5, dado que esta forma de derivación necesita un punto anterior, el cual no se puede obtener en un principio, y un punto posterior. En un instante de tiempo cuando ya existen datos previos el error converge a cero rápidamente. Aumentando el número de puntos utilizados a tres (Figura 56) se tiene la misma tendencia con la diferencia que el instante de tiempo para que converja el error es mayor, pero a su vez es mucho más estable al utilizar más puntos.

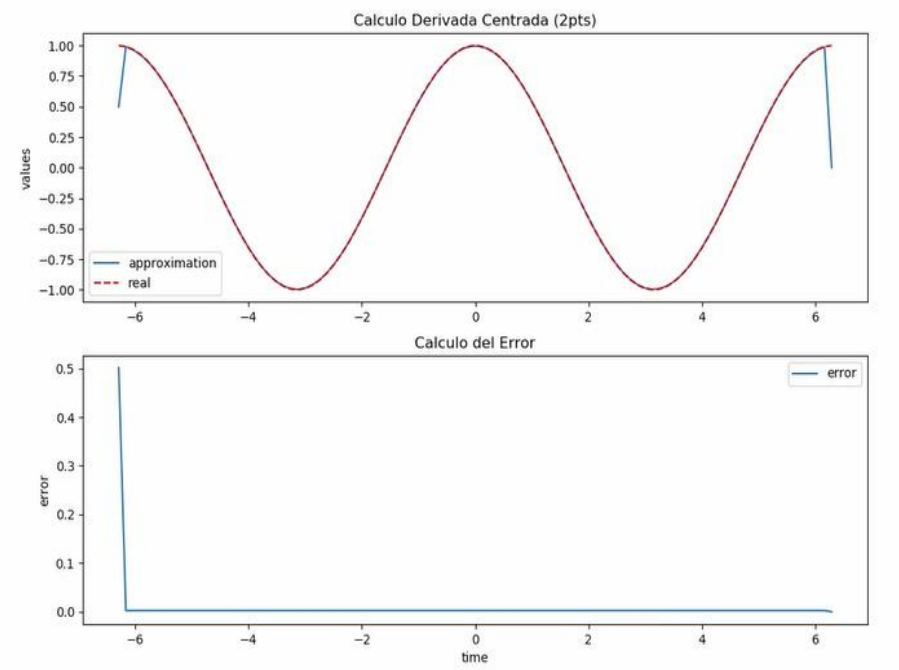


Figura 55. Derivada centrada de 2 puntos

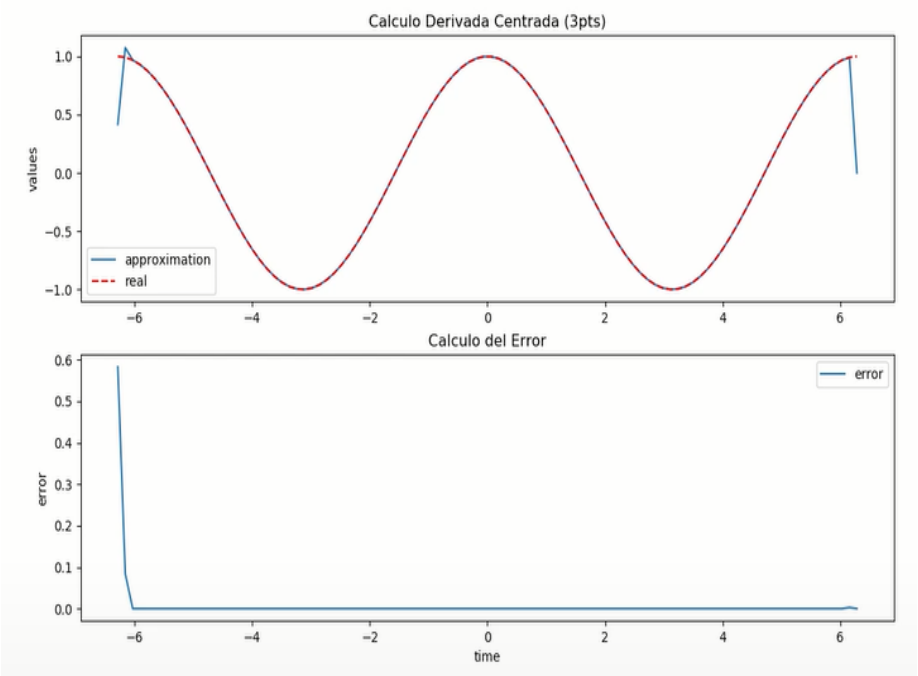


Figura 56. Derivada centrada de 3 puntos

Al utilizar la fórmula de derivación regresiva de dos puntos (Figura 57) en la función seno se obtiene un error alto equivalente al mismo error presente en la derivación progresiva del mismo número de puntos. Esto se debe a que utiliza solo el punto en el que se desea calcular la derivada y el punto anterior. Aumentando el número de puntos utilizados a tres (Figura 58) el error sigue presente, aunque de magnitud menor; su tiempo inicial para converger o llegar a cero incrementa.

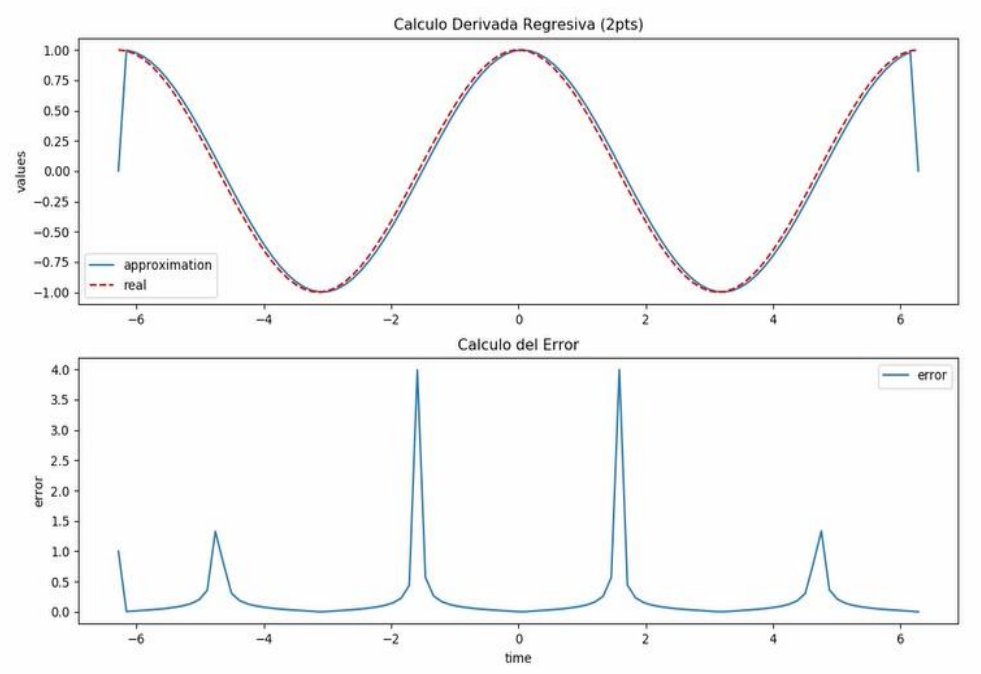


Figura 57. Derivada regresiva de 2 puntos

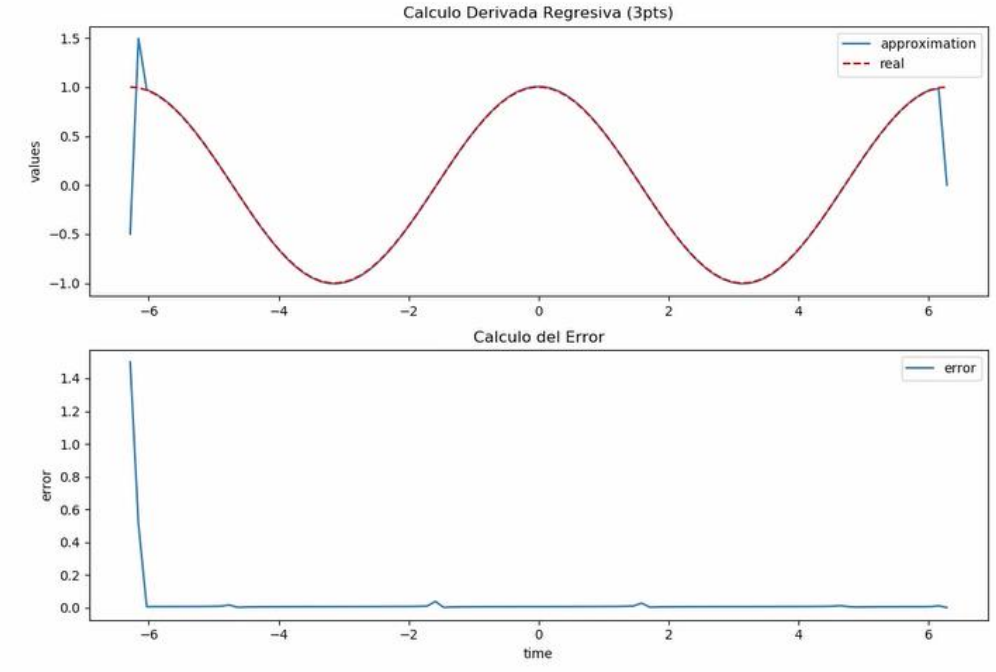


Figura 58. Derivada regresiva de 3 puntos

Resultado:

Se escoge la derivada de tres puntos centrada ya que el error de esta converge rápidamente a cero a comparación de las demás implementadas y también ocupa menos recursos que otras derivadas centradas con un mayor número de puntos.

5.2.2 Prueba de desplazamiento

Para esta prueba se realizó 10 ensayos en las mismas condiciones, enviando al dron a un punto fijo x=2.0, y=1.5 y z=2.0 y con el controlador fuzzy PD. Los resultados de la prueba se presentan en la Tabla 22 y fueron tomados cuando el dron se ha desplazado y se mantiene estable en una posición. Esta prueba es sin perturbaciones externas.

Tabla 22.
Desplazamiento con control PD

Set Point (x=2.0, y=1.5 y z=2.0)							
# Prueba	Distancias Real						Tiempo de ejecución (s)
	x(m)	Error en x (%)	y(m)	Error en y (%)	z(m)	Error en z (%)	
1	2,12	6,00	1,60	6,67	2,10	5,00	25,7
2	2,05	2,50	1,65	10,00	1,72	14,00	26
3	1,94	3,00	1,40	6,67	1,80	10,00	25,1
4	2,10	5,00	1,55	3,33	2,05	2,50	29
5	1,92	4,00	1,53	2,00	1,80	10,00	24
6	2,02	1,00	1,45	3,33	2,05	2,50	27,5
7	1,95	2,50	1,56	4,00	2,00	0,00	25,2
8	2,08	4,00	1,44	4,00	1,85	7,50	26,1
9	2,10	5,00	1,60	6,67	1,65	17,50	28,3
10	2,02	1,00	1,64	9,33	1,70	15,00	25
Promedio	2,03	3,40	1,54	5,60	1,87	8,40	26,19

Al igual que en la prueba de desplazamiento con controlador P el tiempo de ejecución es la suma del tiempo que tarda el dron en recibir los comandos de vuelo, el despegue y estabilización en el punto inicial, y el tiempo de vuelo hasta llegar al punto definido. Cabe recalcar que el tiempo de vuelo aumentara si el set point es mayor. El tiempo de estabilización del sistema es igual al tiempo de vuelo, y este es alrededor de 12 segundos para las distancias de esta prueba.

El error en estado estacionario para la posición en “x” es de 0,10m que corresponde a un 5%; en la posición en “y” es de 0.1m que corresponde a un 6.7% de error; y en “z” es de 0,05m que corresponde a un 2,5% de error. La Figura 59 corresponde al ensayo 7 y se muestra el error, su derivada y el comportamiento del controlador

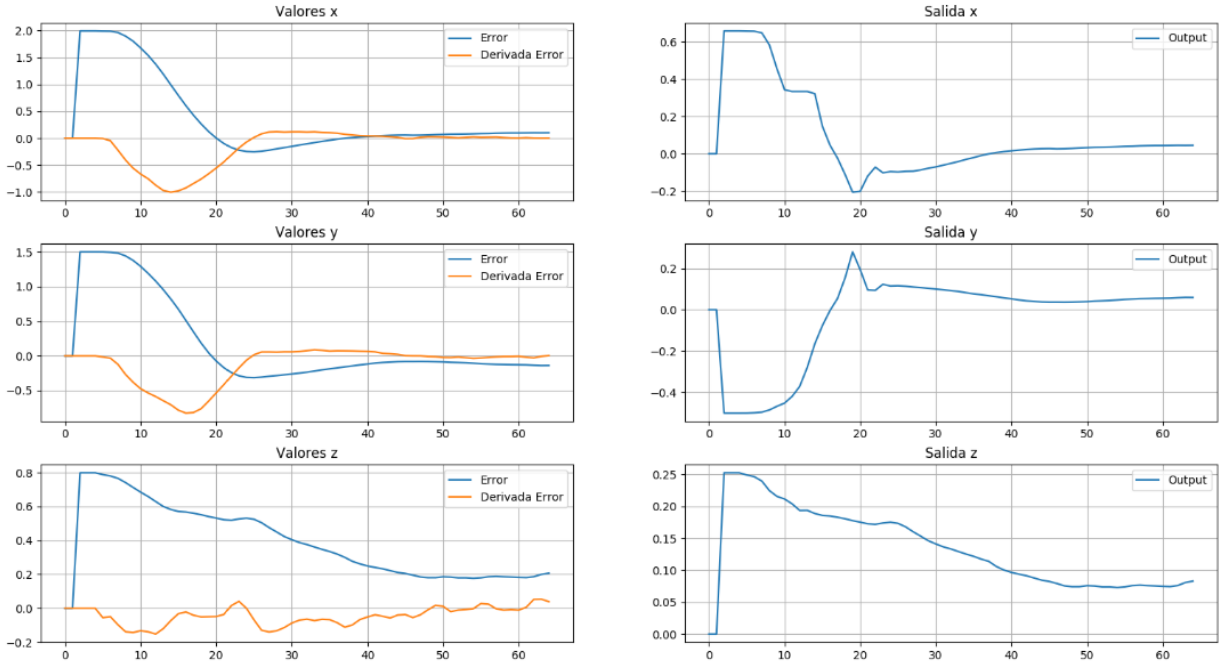


Figura 59. Prueba desplazamiento controlador P

Resultado:

El sobre impulso del controlador PD es mínimo por lo que el tiempo de estabilización aumenta en relación al controlador P, mientras que el error en estado estacionario se reduce significativamente en los tres ejes de desplazamiento aproximándose a cero. El controlador PD para el desplazamiento tiene una respuesta suave y responde a errores pequeños alrededor de 0.1m.

5.2.3 Prueba frente a perturbaciones

Para esta prueba se repitió un ensayo de la prueba anterior con la diferencia que cuando el dron estaba en estado estable se le ejerció una fuerza moviendo al dron de dicha posición. La señal de error y la salida de control se pueden observar en la Figura 60.

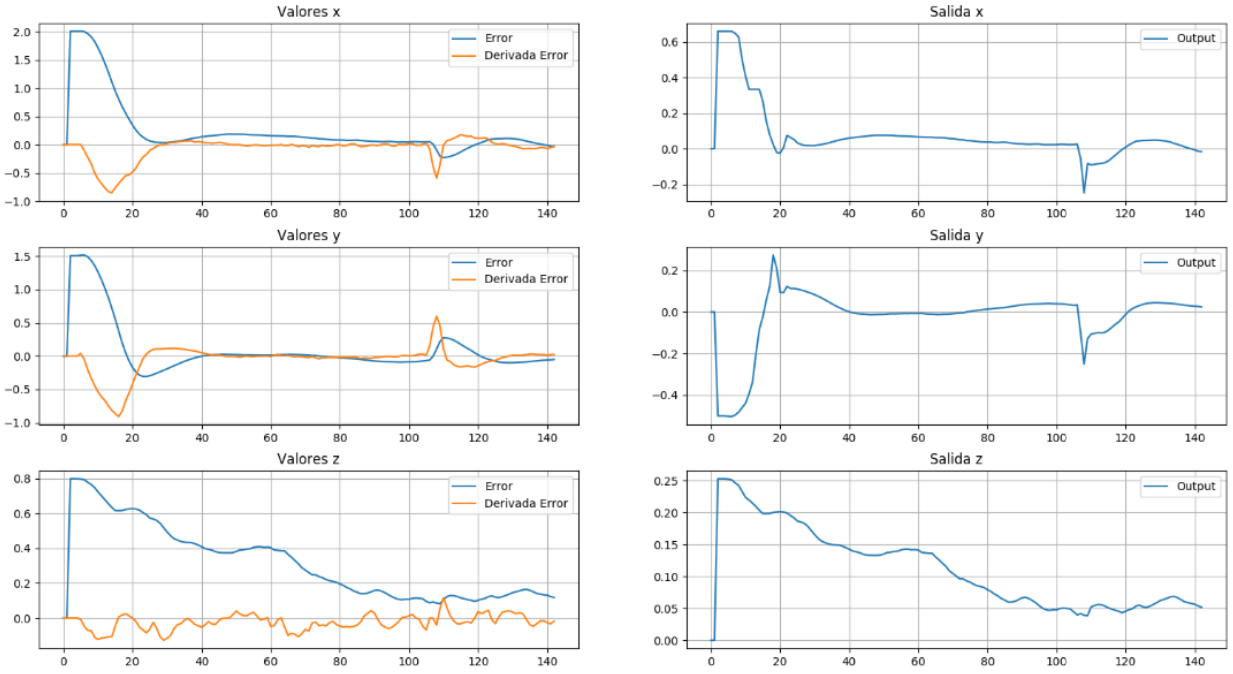


Figura 60. Prueba frente a perturbaciones controlador PD

Resultado:

En la figura se puede observar que el control PD de estabilidad responde a perturbaciones externas aun estas sean mínimas. El error en estado estacionario se acerca al cero y el controlador no tiene sobre impulsos altos por lo que el tiempo de estabilización es rápido.

5.2.4 Prueba de orientación

Para esta prueba se realizó el despegue del dron y se le envió un giro de 0.78 radianes en su posición inicial. Una vez estable se le aplicó una perturbación. En la Figura 61 se muestra el error inicial de 0.78 radianes, la estabilización, el error frente a la perturbación, la derivada de estos errores, y la señal de control frente a los errores y su variación. Se debe indicar que los errores en las posiciones “x,y,z” al utilizar este control para la orientación se ven afectados.

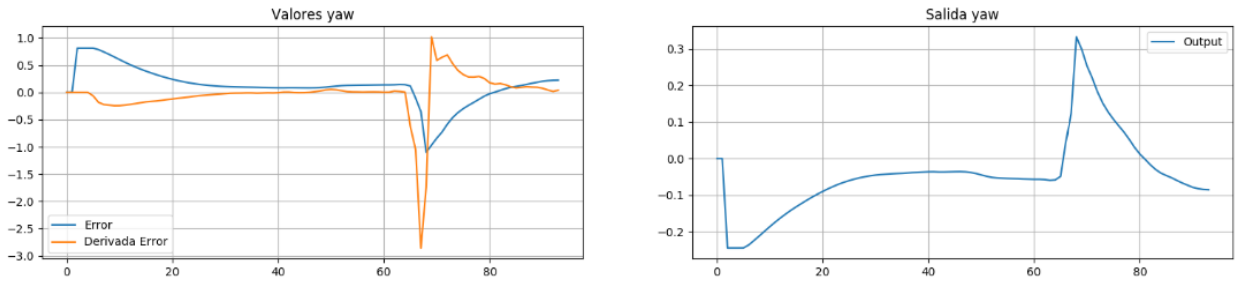


Figura 61. Prueba orientación controlador PD

Resultado:

El controlador de orientación o ángulo yaw tiene una precisión similar al control de orientación P, obteniéndose un error en estado estacionario cercano al cero y respondiendo positivamente a perturbaciones externas.

5.3 Pruebas de trayectoria control fuzzy P

5.3.1 Prueba trayectoria cuadrada desplazamiento en x

Para esta prueba se envió al dron un radio de 1.5m y una tolerancia al error de 0.7, esto debido al error en estado estacionario que tiene este controlador en la prueba de estabilidad. La visualización en Rviz de su desplazamiento se puede observar en Figura 62.

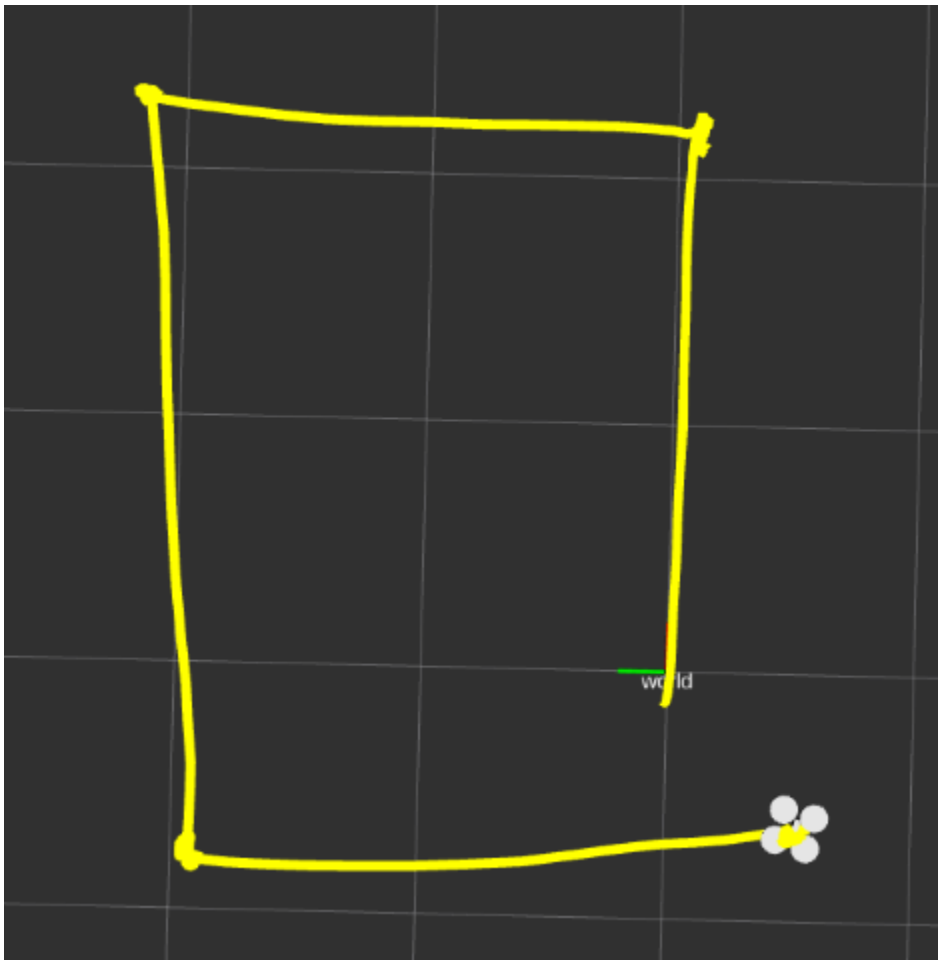


Figura 62. Prueba control P trayectoria cuadrada desplazamiento en x

El comportamiento del controlador se muestra en la Figura 63. Se puede observar que al realizar el desplazamiento en “x”, es en este eje donde se presenta el cambio de set point.

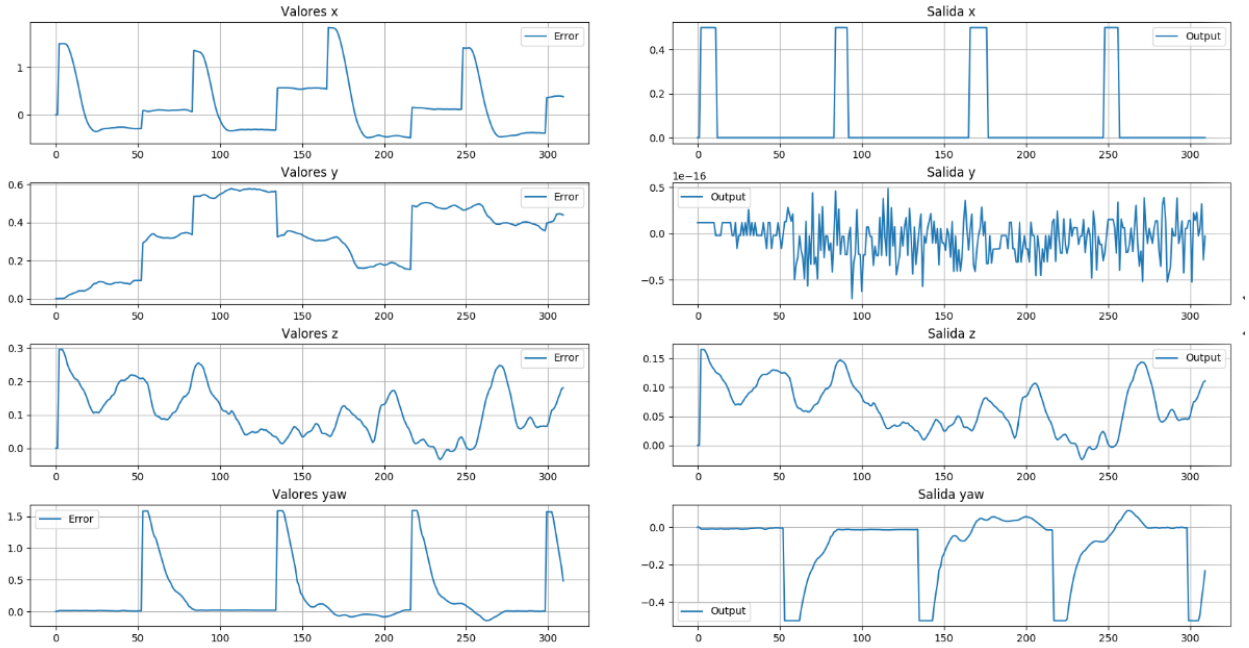


Figura 63. Controlador P trayectoria cuadrada desplazamiento en x

Resultados:

El dron sigue la trayectoria, pero al no actuar frente a errores pequeños no corrige su posición, por lo que sale de un punto y llega a otro, lo cual es la acumulación de errores al desplazase; dando como resultado un cuadrado más grande de lo enviado.

A su vez se observa que el error en estado estacionario en la posición es mayor a 0.5m mientras que en rotación el error tiende a cero.

El tiempo en el que el dron realiza la trayectoria es de 70 segundos hasta llegar a la posición final.

5.3.2 Prueba trayectoria hexagonal desplazamiento en y

Para esta prueba se envió al dron un radio de 1.2m y una tolerancia al error de 0.7, al igual que la trayectoria cuadrangular. La visualización en Rviz de su desplazamiento se puede observar en la Figura 64.

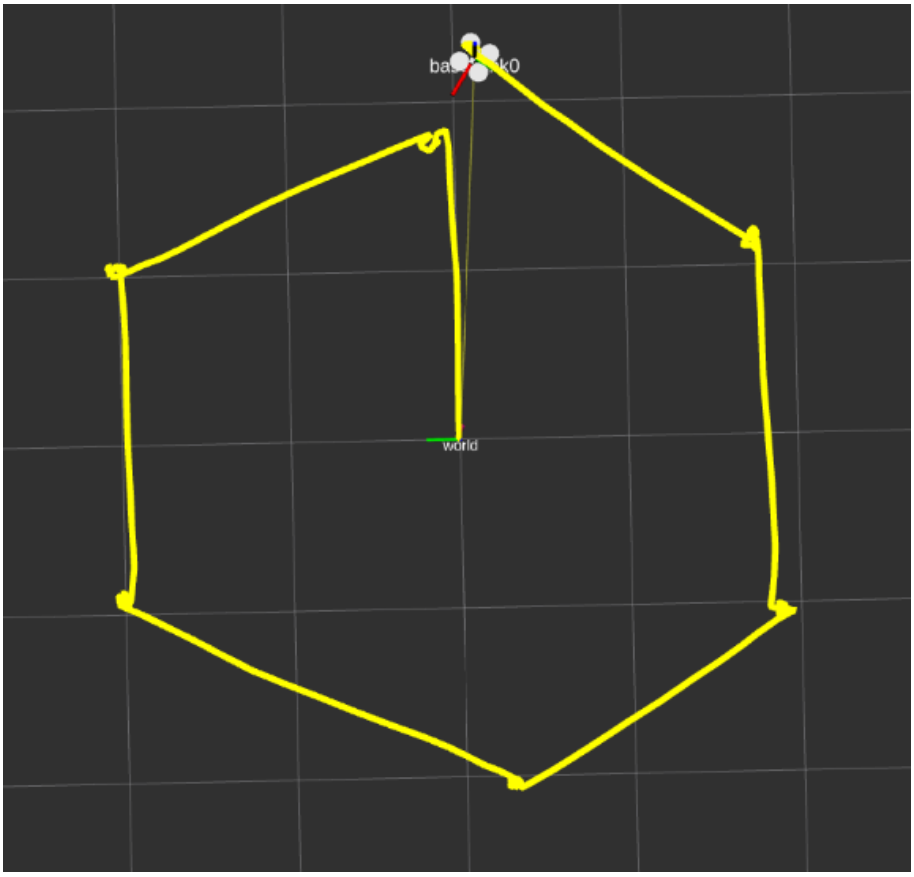


Figura 64. Prueba control P trayectoria hexagonal desplazamiento en y

El comportamiento del controlador se muestra en la Figura 65. Se puede observar que, al realizar el desplazamiento en “y”, es en este eje donde se presenta el cambio de set point.

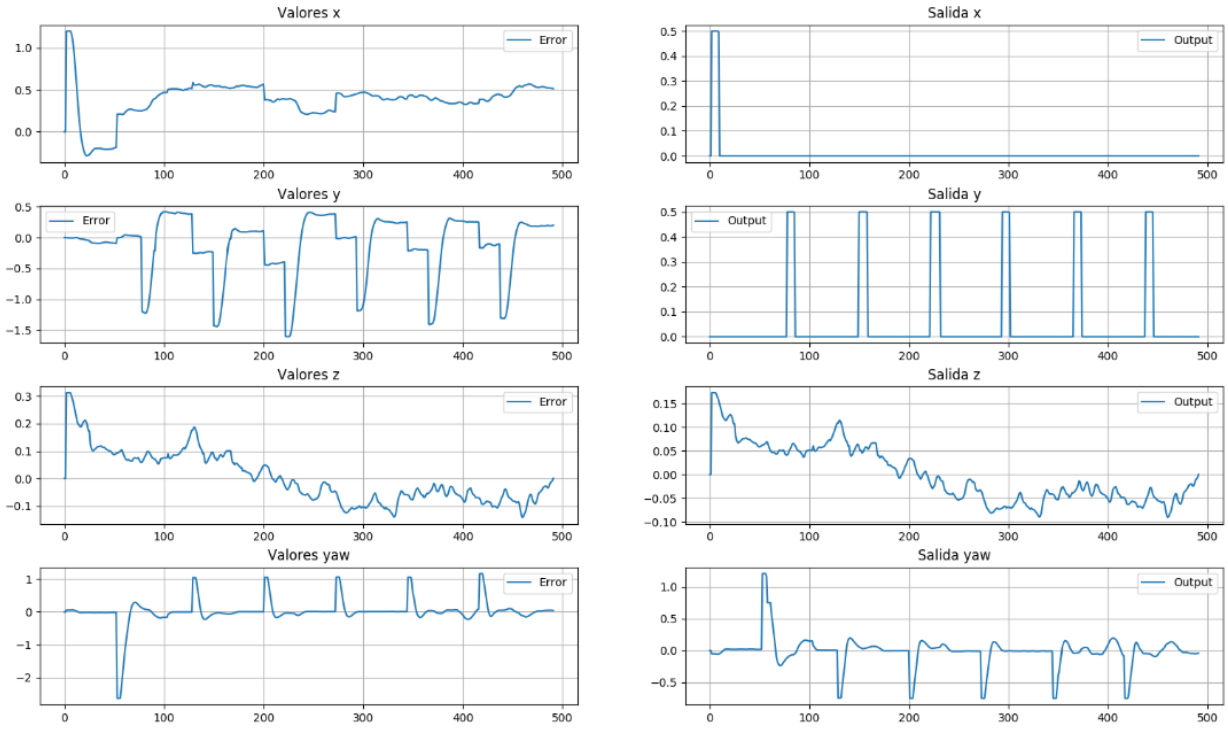


Figura 65. Controlador P trayectoria hexagonal desplazamiento en y

Resultados:

El dron sigue la trayectoria, pero al no actuar frente a errores pequeños no corrige su posición, por lo que sale de un punto y llega a otro; lo cual es la acumulación de errores al desplazase, dando como resultado un hexágono más grande de lo enviado.

A su vez se observa que el error en estado estacionario en la posición es mayor a 0.3m mientras que en rotación el error tiende a cero.

El tiempo en el que el dron realiza la trayectoria es de 100 segundos hasta llegar a la posición final.

5.4 Pruebas de trayectoria control fuzzy PD

5.4.1 Prueba trayectoria cuadrada desplazamiento en x

Para realizar la prueba se envió un radio de 1.5m, de esta manera el dron realizo la trayectoria mostrada en la Figura 66. También se puede observar que en cada esquina el dron corrige su posición a diferencia que en el controlador P, ya que cuenta con una mayor precisión y se encuentra validado que su aproximación a un punto debe ser menor a 0.1m para girar y desplazarse al punto siguiente. Se observa que el dron consigue llegar al punto de partida y con el radio indicado.

Figura 66. Prueba control PD trayectoria cuadrada desplazamiento en x

El comportamiento del controlador se muestra en la Figura 67. Se puede observar que al realizar el desplazamiento en “x”, y es en este eje donde se presenta el cambio de set point.

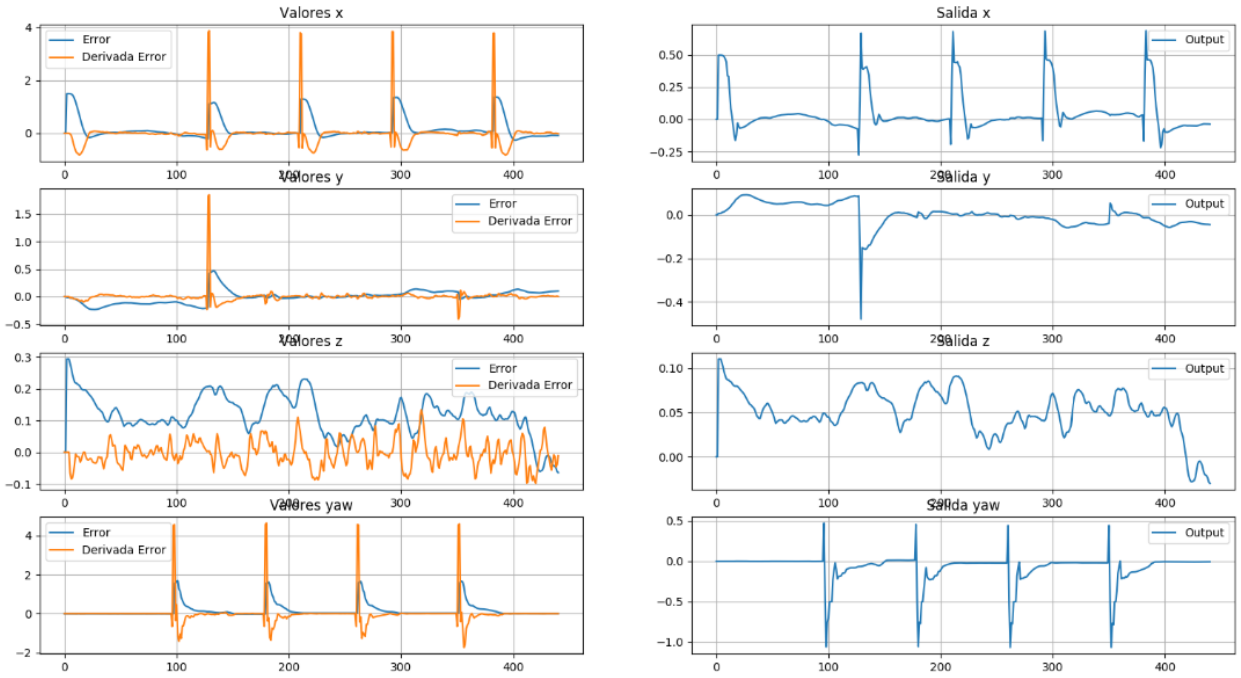


Figura 67. Controlador PD trayectoria cuadrada desplazamiento en x

Resultados:

El dron sigue la trayectoria y el resultado es un cuadrado del tamaño deseado ya que este controlador actúa frente a errores pequeños logrando un error en estado estacionario cercano a cero.

El tiempo en el que el dron realiza la trayectoria es de 90 segundos hasta llegar a la posición final, el cual es 20 segundos más que el controlador P.

5.4.2 Prueba trayectoria hexagonal desplazamiento en y

Para realizar la prueba se envió un radio de 1.2m, de esta manera el dron realizo la trayectoria mostrada en la Figura 68. Al igual que en la trayectoria cuadrangular el dron corrige la posición en los vértices y sus validaciones para cambio de set point son iguales por lo que consigue llegar al punto de partida y con el radio indicado.

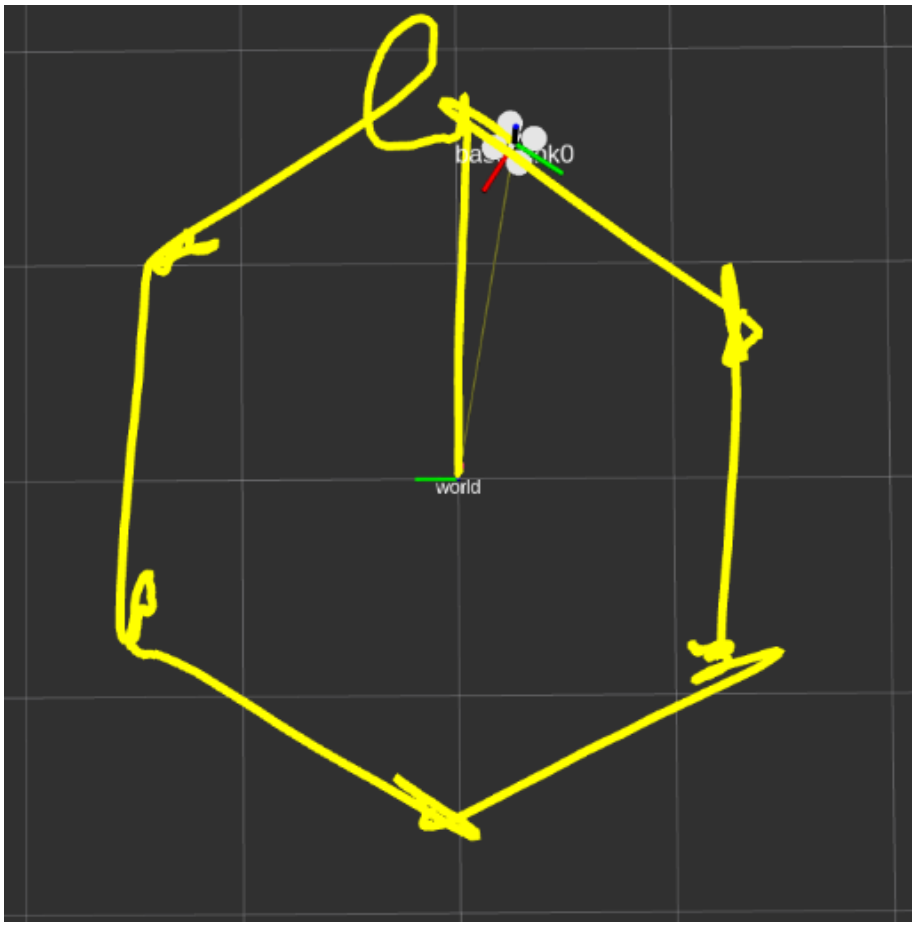


Figura 68. Prueba control PD trayectoria hexagonal desplazamiento en y

El comportamiento del controlador se muestra en la Figura 69. Se puede observar que al realizar el desplazamiento en “y”, es en este eje donde se presenta el cambio de set point.

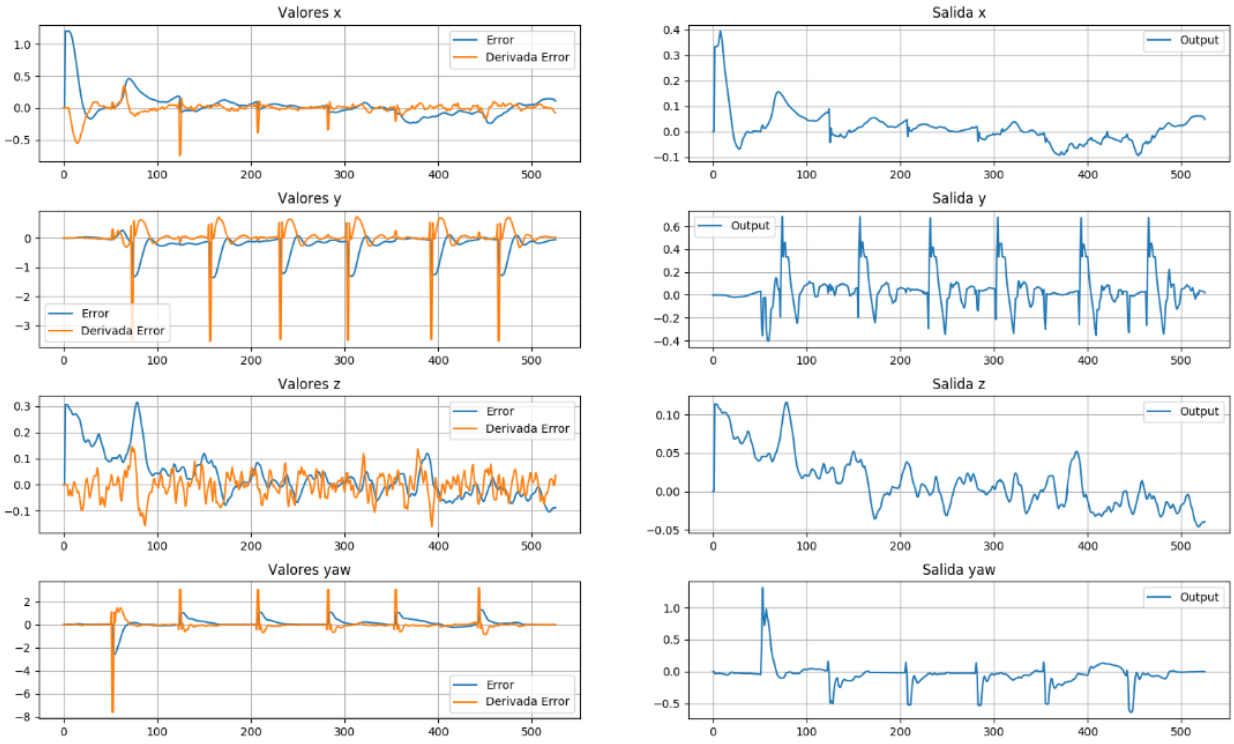


Figura 69. Controlador PD trayectoria hexagonal desplazamiento en y

Resultados:

El dron sigue la trayectoria y el resultado es un hexágono del tamaño deseado, ya que este controlador actual frente a errores pequeños logrando un error en estado estacionario cercano a cero.

El tiempo en el que el dron realiza la trayectoria es de 110 segundos hasta llegar a la posición final, el cual es 10 segundos más que el controlador P.

5.5 Prueba reconocimiento de distancia

Para esta prueba se ubicó al dron a un punto fijo y se varia la posición del objeto (una botella) con incrementos de 5 centímetros entre mediciones (Figura 70). Se comparó con la distancia detectada del algoritmo de identificación Figura 71 para calcular su error.

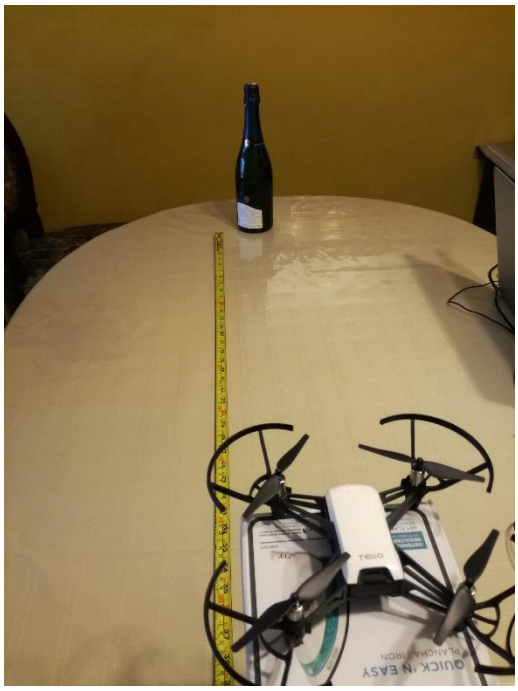


Figura 70. Medición distancia

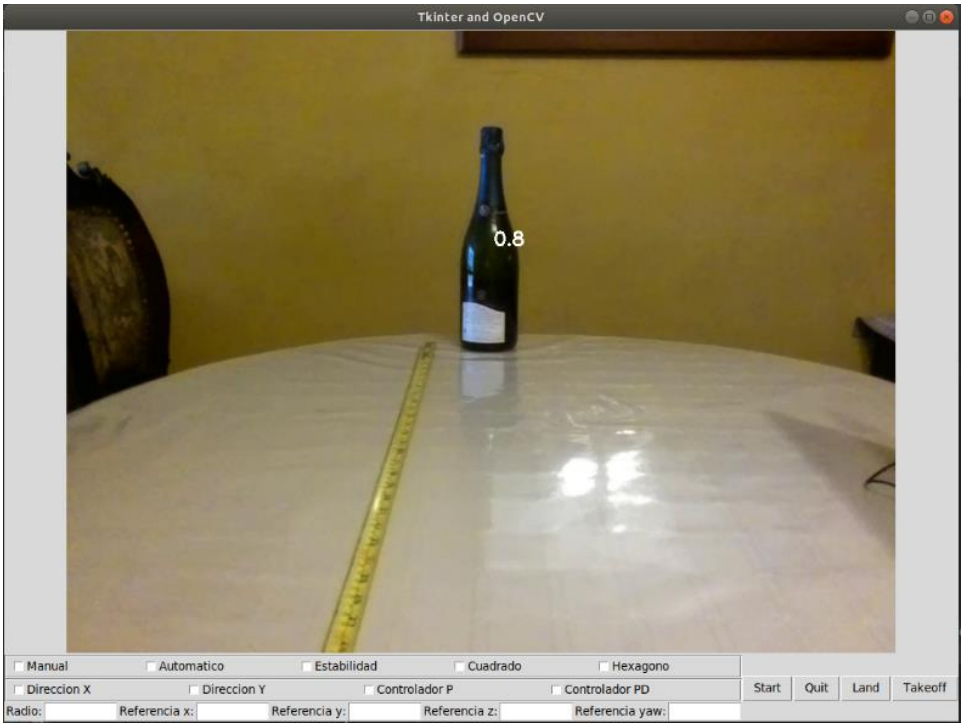


Figura 71. Distancia detectada

La distancia real, la distancia detectada y el error entre ambas mediciones se muestran en la Tabla 23.

Tabla 23

Mediciones de distancia dron - objeto

Distancia real (cm)	Distancia detectada (cm)	Error (%)
10	10	0,0
15	20	25,0
20	20	0,0
25	30	16,7
30	30	0,0
35	40	12,5
40	40	0,0
45	50	10,0
50	50	0,0
55	50	10,0
60	60	0,0
65	60	8,3
70	60	16,7
75	60	25,0
80	70	14,3
85	70	21,4
90	70	28,6
95	70	35,7
100	70	42,9
105	80	31,3
110	80	37,5
115	80	43,8

Resultado:

Debido a que el algoritmo de identificación detecta distancias de 10cm a 100cm con pasos de 10cm no se tiene una buena precisión en cuanto a la distancia del objeto al dron; notándose que el error a distancias mayores a 70cm es mayor que a distancias menores.

Ya que para la toma de fotografías el dron debe identificar el objeto, este no puede volar más alejado de 115cm; por lo que entra dentro del rango de trabajo del algoritmo.

5.6 Prueba reconstrucción 3D

En esta prueba se desplazó al dron frente al objeto a una distancia de 80cm. La Figura 72 muestra el par de imágenes que simula una cámara estéreo. La Figura 73 muestra el mapa de profundidad realizado por el algoritmo implementado, en el cual denota con tonos blancos a objetos más

cercanos. En la Figura 74 se muestra la reconstrucción realizada por el entorno JupyterNotebook utilizado para la visualización 3D.

a) b)

Figura 72.a) Imagen izquierda. b) Imagen derecha.

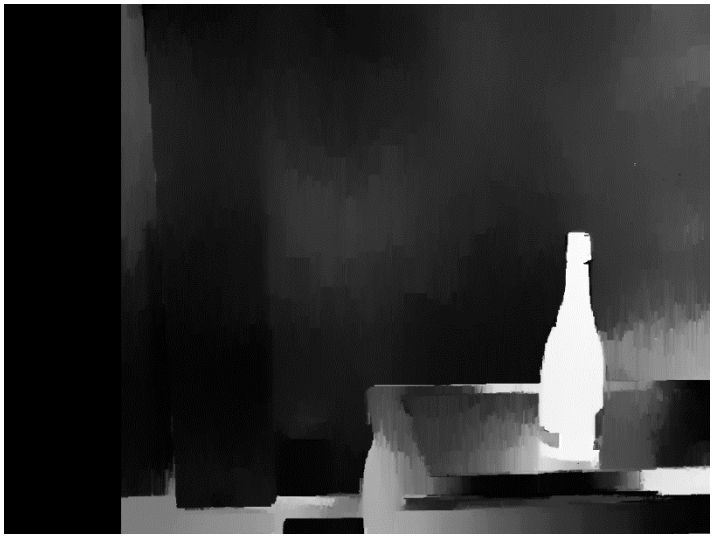
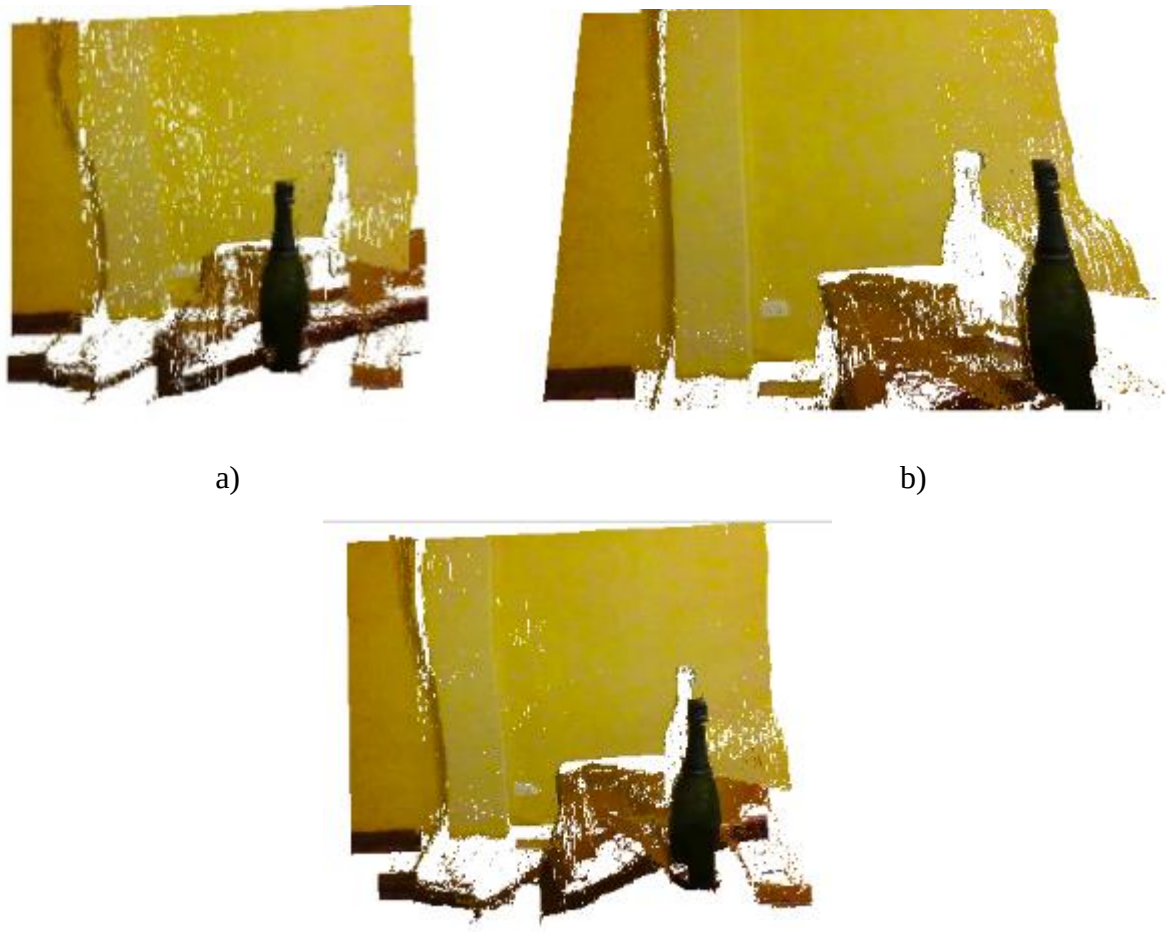


Figura 73. Mapa de disparidad



a) b)

c)

Figura 74. Visualización reconstrucción 3D

Se repitió la prueba cambiando la distancia entre el dron y el objeto a 1.15m. La Figura 75 muestra el par de imágenes que simulan una cámara estéreo, la Figura 76 muestra el mapa de profundidad realizado por el algoritmo implementado, en el cual denota con tonos blancos a objetos más cercanos. En la Figura 77 se muestra la reconstrucción realizada por el entorno JupyterNotebook utilizado para la visualización 3D.

a) b)

Figura 75. a) Imagen izquierda. b) Imagen derecha.

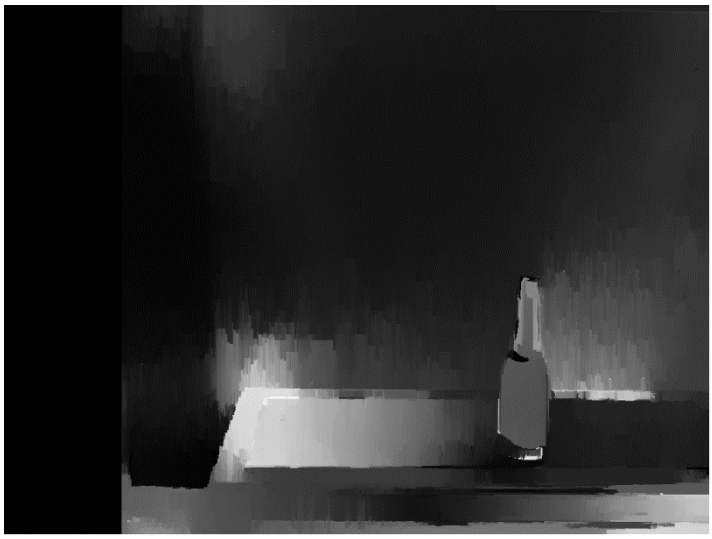
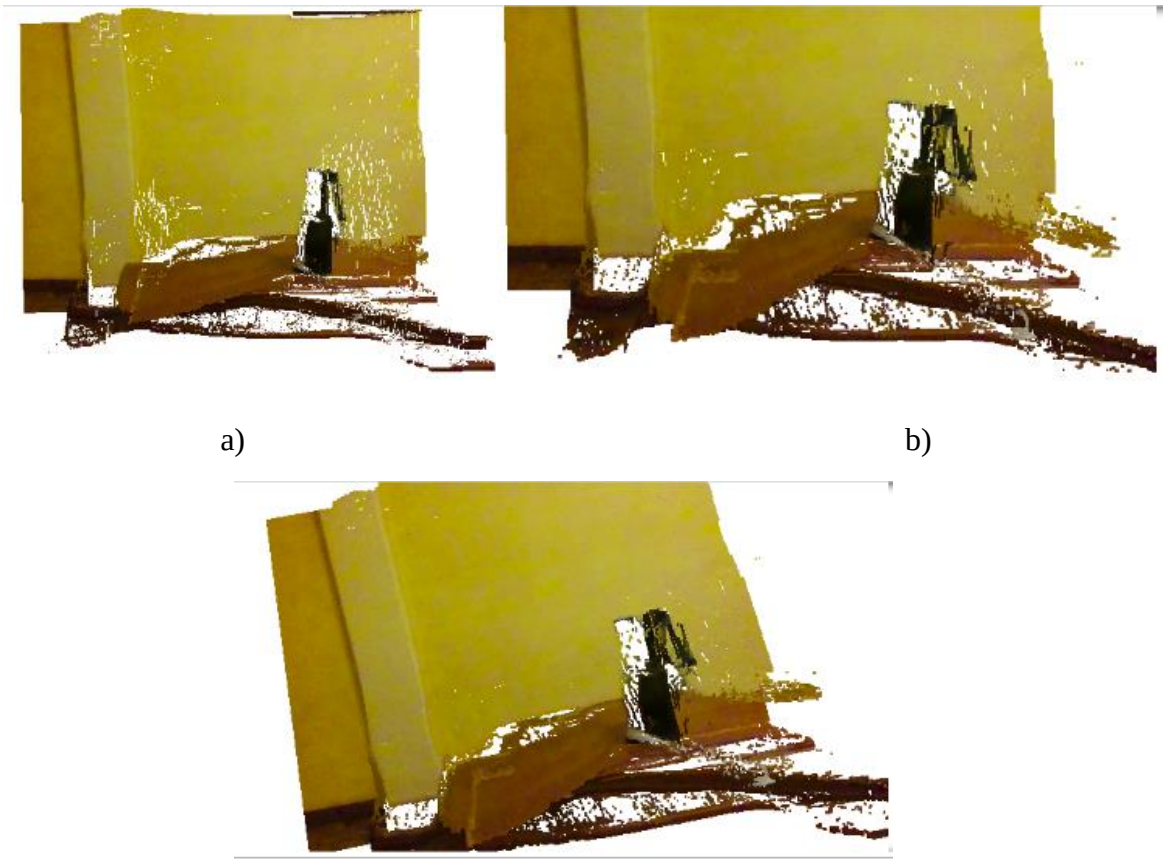


Figura 76. Mapa de disparidad



a) b)

c)

Figura 77. Visualización reconstrucción 3D

5.7 Trabajos Futuros

Para trabajos posteriores se propone que el control de vuelo y la reconstrucción del objeto trabajen simultáneamente en tiempo real mediante la implementación de una cámara estéreo en el dron Tello o utilizando otro dron que tenga este tipo de cámara.

Se propone que la visualización del objeto sea mostrada con una nube de puntos en el simulador Rviz, con un computador de mejores prestaciones y utilizando algoritmos para la segmentación de pixeles que pertenezcan al objeto para su reconstrucción 3D.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

3.1 Conclusiones

En el presente trabajo se ha demostrado el funcionamiento de una técnica de control inteligente para la navegación de drones basado en la estabilidad, permitiendo seguir una trayectoria predefinida y la toma de fotografías de un objeto estático en un entorno controlado para su posterior reconstrucción 3D.

Se comparó los controladores fuzzy de tipo P y PD para la estabilidad de vuelo del mini dron Tello a través del diseño e implementación de un controlador independiente para cada ángulo de giro, obteniendo en el tipo P un tiempo de estabilización menor y un error en estado estacionario de 0.5m; mientras que en tipo PD el error es menor a 0.1m para los ángulos roll, pitch y throttle. En el ángulo yaw el comportamiento del controlador tipo P y PD es similar con un error menor a 0.2 rad.

El control de trayectoria implementado se encarga del seguimiento de puntos de una ruta predefinida trabajando con una configuración en cascada con el control de estabilidad como muestra el esquema de la Figura 43, obteniendo el desplazamiento autónomo del dron alrededor del objeto sin tener grandes desviaciones en la trayectoria y respondiendo a perturbaciones externas.

Los resultados de las pruebas de trayectoria evidenciaron que para diferentes trayectorias y ejes de movimiento las respuestas de los controladores son constantes; en el controlador tipo P se obtiene una trayectoria de mayores dimensiones sin llegar el punto deseado debido a la suma de errores entre puntos mientras que en el controlador PD cumple con la trayectoria establecida.

Utilizando el reconocimiento de objetos se logra una aproximación de la distancia existente entre el dron y el objeto; a su vez limita el almacenamiento de fotografías cuando el objeto enfocado no supera el 35% de similitud con el objeto deseado por lo que reduce significativamente la cantidad de imágenes almacenadas.

La aplicación de disparidad para calcular la profundidad permite obtener las 3 dimensiones de cada pixel a partir de dos imágenes monoculares desplazadas entre sí, emulando una imagen de tipo estéreo, el resultado de la reconstrucción del objeto es la unión de características de la última imagen a color con su respectivo mapa de profundidad.

La interfaz gráfica desarrollada permite escoger el tipo de controlador, trayectoria y dimensión de la misma, esta a su vez muestra el video capturado por el dron, la distancia al objeto y los datos de los sensores internos; facilitando la manipulación de parámetros de vuelo y conjuntamente con la visualización de la trayectoria en el entorno RViz permite una mayor supervisión por parte del usuario.

3.2 Recomendaciones

Utilizar el entorno de desarrollo ROS para la creación de proyectos con drones para que estos sean modulares, escalables en tiempo de ejecución y numero de drones utilizados.

Seleccionar el tipo de controlador fuzzy dependiendo la aplicación, si esta requiere un menor tiempo de ejecución un controlador tipo P y si se requiere un menor error en posición un controlador tipo PD.

En la realización de pruebas de vuelo utilizar protectores en las hélices del dron evitando daños físicos en el dron y su entorno.

Para la aplicación del proyecto el entorno de trabajo debe tener una buena iluminación, no presentar grandes perturbaciones externas y la distancia de vuelo del dron al objeto debe ir de 0.80 m a 1.15m.

BIBLIOGRAFÍA.

Archimowicz, Y., & Martinez, S. (2011). *EdRo - Un enjambre de robots.* Uruguay: Universidad de la República.

Ballesta, M., Gil, A., Reinoso, O., & Úbeda, D. (2010). Análisis de Detectores y Descriptores de Características Visuales en SLAM en Entornos Interiores y Exteriores. *Revista Iberoamericana de Automática e Informatica Industrial*, 68-80.

Botero, A. J. (2005). *Descripciones y transformaciones espaciales.*

Buckl, K. (08 de Marzo de 2005). *Universidad Tecnica de München*. Obtenido de www.campar.in.tum.de/Chair/KalmanFilter

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Carrera, A., & Carreras, M. (08 de 04 de 2015). *ROSPlan: Planning in the Robot Operating System*. Obtenido de https://www.aaai.org/ocs/index.php/ICAPS/ICAPS15/paper/view/10619

Cazorla, A. G. (2013). *ROS: Robot Operating System.* Cartagena: Universidad Politécnica de Cartagena.

Ciano, O. (13 de Noviembre de 2016). *GitHub*. Obtenido de https://github.com/Otacon/wifi_china_drone_controller

Distributions, R. (22 de 11 de 2018). *ROS.org*. Obtenido de http://wiki.ros.org/Distributions

Doxygen. (22 de Diciembre de 2017). *OpenCV*. Obtenido de https://docs.opencv.org/3.4.0/d5/dae/tutorial_aruco_detection.html

Flores, W. G. (3 de Junio de 2015). *Centro de investigación y estudios avanzados de IPN*. Obtenido de https://www.tamps.cinvestav.mx/~wgomez/toptamps/presentacion.pdf

Garcia, C. (s.f.). Herramientas Matemáticas para la localización espacial.

Grandón-Pastén, N., Aracena-Pizarro, D., & Tozzi, C. L. (2007). Reconstrucción de objeto 3D a partir de imágenes calibradas. *Ingeniare. Revista Chilena de Ingeniería*, 158-168.

Grandón-Pastén, N., Arancena-Pizarro, D., & Tozzi, C. L. (2007). Reconstrucción de objeto 3d a partir de imágenes calibradas. *Ingeniare*, 158-168.

Gwydion, M. (18 de Enero de 2018). *Adictos al trabajo*. Obtenido de https://www.adictosaltrabajo.com/2018/01/18/primeros-pasos-con-jupyter-notebook/

Hanyazou. (29 de Marzo de 2019). *GitHub*. Obtenido de https://github.com/hanyazou/TelloPy

Hirschmuller, H. (2008). Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Inelligence*, 328-341.

ISA, & Automatica, I. d. (s.f.). Fundamentos matematicos.

Jurado, G., & Salinas, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Patter Recognition*, 47 (6), 2280-2292 .

Kam, H. C., & Yu, Y. K. (2018). An Improvement on ArUco Marker for Pose Tracking Using Kalman Filter. *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*.

Kendra, C. (2007). The Silent Force Multiplier: The History and Role of UAVs in Warfare. *IEEE Aerospace Conference* (págs. 1-7). Big Sky, MT, USA: IEEE.

Krag. (2018 de Julio de 2018). *GitHub*. Obtenido de https://github.com/Kragrathea/TelloLib

Lozano, V. M. (2015). *Sistema de control que permite el vuelo autónomo de drones* . Universidad de Alicante.

Macanás, J. V. (2014). *Interacción entre webcam y brazo robot para el posicionamiento del efector final.* Cartagena: Universidad Politécnica de Cartagena.

Mandujano, S., Mulero-Pázmany, M., & Rísquez-Valdepeña, A. (2017). Drones: Una nueva tecnología para el estudio y monitoreo de hábitats. *Agro Productividad*, 79-84.

McLeod, C. (08 de Julio de 2017). *GitHub*. Obtenido de https://gist.github.com/claymcleod/028386b860b75e4f5472

Mendomatica. (Octubre de 2010). *Cuaterniones.* Obtenido de www.mendomatica.mendoza.edu.ar: http://www.mendoza.edu.ar/wp-content/uploads/2017/04/TEMAS-DE-MATEM%C3%81TICA-Cuaterniones.pdf

Morenia, R. M. (14 de 08 de 2018). *ROS.org*. Obtenido de http://wiki.ros.org/melodic

Movetia. (27 de Junio de 2017). Obtenido de https://blog.movetia.com/tensorflow-deteccion-objetos-google/

Ochoa, R. L. (2018). *Sistema para la Ejecución de Trayectorias.* Salamanca: Universidad de Guanajuato.

O'Kane, J. M. (2018). *A Gentle Introduction to ROS.* Independently published.

Palacios, F. M. (2009). *Controlador Fuzzy de un Quadrotor*. Madrid: Universidad Complutense de Madrid.

Passino, K., & Yurkovich, S. (1998). *Fuzzy Control.* Ohio: addison-wesley.

Rambhia, J. (29 de Marzo de 2013). Obtenido de https://jayrambhia.com/blog/disparity-mpas?fbclid=IwAR3ivTZXE7YYOttfQhuTIYbUZ8nORATn6hHSOWYqlFlI_NeytC3hZG7dWQQ

Revelo-Luna, D. A., Usama, F. D., & Flórez-Marulanda, J. F. (2012). *Reconstrucción 3D de escenas mediante un sistema de visión estéreo basado en extraccion de características y desarrollado en OpenCV.* Popayán, Colombia: Universidad de Cauca.

RYZE. (2019). *Ryze robotics*. Recuperado el 2 de Mayo de 2019, de https://www.ryzerobotics.com/tello

Samartzidis, T. (09 de Julio de 2017). Obtenido de http://timosam.com/python_opencv_depthimage?fbclid=IwAR0p2_XeZbeqxmODFg-Pzq4LQIXsB41X0-62SJqT-4MiwjkD1uFXIyy3sZo

Santos, J. M., Portugal, D., & Rocha, R. P. (2013). An evaluation of 2D SLAM techniques available in Robot Operating System. *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 1-6.

Sellali, B. B., & Allali, A. (2017). Neuro — Fuzzy methods coupled to operational PID, to improve the flight parameters of a drone. *2017 18th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA).* Monastir, Tunez.

Taipe, D. K. (2017). *Sistema de Localización Indoor y Outdoor para un.* Instituto politécnico de leiria.

Tayupanta, W. E. (2018). *Navegación Autónoma de Mini-drones para Multirobots SLAM 3D en Entrnos Estáticos con Algoritmos de Inteligencia de Ejambre.* Sangolquí: ESPE.

Vásquez, Á. S. (2015). *Desarrollo de un sistema de SLAM para el robot AR.Drne en entorno ROS.* Madrid: Universidad de Alcalá.

Vázquez, I. B. (2016). *Análisis y comparacion de rendimiento en algoritmos de control para seguimieno de trayectorias aplicados en un drone cuadricóptero.* México: Instituto Politécnico Nacional.

Yamamoto, N., & Uchida, N. (2018). Improvement of Image Processing for a Collaborative Security Flight Control System with Multiple Drones. *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA).* Krakow: Polonia.

ANEXOS