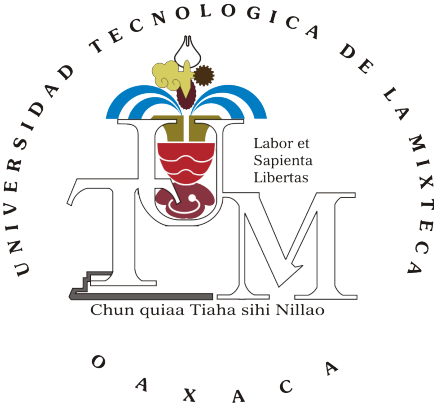
UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA



"CONTROL DIFUSO DEL QUADROTOR *AR.DRONE 2.0*® PARA EL SEGUIMIENTO AUTÓNOMO DE TRAYECTORIAS"

TESIS PARA OBTENER EL TÍTULO DE:

INGENIERO EN MECATRÓNICA

PRESENTA:

ARTURO ALONSO CASANOVA DE LOS SANTOS

DIRECTOR:

DR. FELIPE DE JESÚS TRUJILLO ROMERO

HUAJUAPAN DE LEÓN, OAXACA, MÉXICO, FEBRERO DE 2015

Resumen

En esta tesis se propone el desarrollo de un controlador difuso para estabilizar el funcionamiento de un quadrotor en el seguimiento de trayectorias tanto en simulación como en la plataforma real.

Un quadrotor es un helicóptero con cuatro rotores, lo cual permite tener una mayor estabilidad que un helicóptero normal. El quadrotor posee seis grados de libertad, tres que definen su posición: altura, desplazamiento vertical y desplazamiento horizontal; y otros tres que definen la orientación del cuerpo: *Roll*, *Pitch* y *Yaw*.

Para el control se utiliza un sistema difuso; este sistema se basa en la lógica difusa para procesar las señales provenientes del quadrotor y generar una respuesta. Para ello, se desarrollaron 4 controladores que tienen como entradas: la altura y los tres ángulos de orientación para cada uno de ellos. Como salida se obtienen los cambios de velocidad en los desplazamientos y en los ángulos de orientación.

Para la implementación del control tanto para la plataforma real como la simulación se utiliza el Toolbox de *Fuzzy Logic* y *Real-Time Windows Target* de Matlab y Simulink. Además, se creó una interfaz gráfica para su manipulación, en la cual se tiene acceso a las diferentes trayectorias de la base de datos para evaluación del seguimiento con el quadrotor.

El sistema debe ser capaz de seguir trayectorias simples como son: la elevación vertical, líneas rectas, curvas suaves, círculos, polígonos regulares, entre otras. Estas trayectorias están en una base de datos y de ser necesario se pueden agregar otras nuevas. Finalmente se comparó el desempeño del seguimiento de trayectorias utilizando el controlador difuso y un controlador proporcional, encontrando que el sistema difuso es más estable y se acerca más a la trayectoria deseada.

Abstract

In this thesis a fuzzy controller is proposed to stabilize the operation of a quadrotor in the trajectory tracking in both simulation and real platform.

A quadrotor is an helicopter with four rotors, which allows greater stability than a normal helicopter. The quadrotor has six degrees of freedom, three defining his position: height, vertical displacement and horizontal displacement; and three defining the orientation of the body: *Roll*, *Pitch* and *Yaw*.

The control is achieved with a fuzzy system; this system is based on fuzzy logic to process the signals from the quadrotor and generate a response. For this, 4 drivers were developed, whose inputs are: the height and the three angles of orientation for each one. As outputs are the rate changes of the movement and the orientation angles.

Implementation of the controller for both the real platform and the simulation, the *Fuzzy Logic* and *Real-Time Windows Target* of Matlab and Simulink Toolbox is used. In addition, a graphical interface for manipulation was created. With this interface is possible to have access to different trajectories for path tracking evaluation of the quadrotor.

The system must be able to follow simple trajectories such as: vertical lifting, straight lines, smooth curves, circles, regular polygons and more. These paths are in a database and if necessary you can add new ones. Finally the path tracking performance was compared using the fuzzy controller and a proportional controller, finding that the fuzzy system is more stable and quite close to the desired path.

Índice

Índice de figuras

Capítulo 1

Introducción

En la actualidad existen muchos problemas en las actividades diarias que resultan laboriosas o difíciles para el ser humano. Problemas que son posibles resolver con los avances tecnológicos.

Los robots permiten resolver varios problemas fundamentales relacionados con razonamiento automatizado, manipulación de objetos y localización, entre otros. Según la Federación Internacional de Robótica (IFR por sus siglas en inglés) un robot puede ser definido como: ''*un mecanismo de acción programable en dos o más ejes, con un grado de autonomía, moviéndose dentro de su entorno*''. La autonomía en este contexto se entiende como la capacidad de un robot de realizar tareas asignadas sin intervención humana [1].

De manera más general, un robot, o sistema robótico, se resume en un conjunto de 3 partes principales, como lo son los sensores, los actuadores y un sistema de control donde:

- ✓ **Los sensores** son aquellos que captan una percepción del entorno y estado de su estructura mecánica: visión, tacto, audición, proximidad, giros, desplazamientos, velocidades, etc. Los sensores pueden ser internos y externos.
- ✓ **Los actuadores** son sistemas electromecánicos que generan fuerza y par para producir movimiento: sistemas eléctricos, mecánicos o neumáticos.
- ✓ **Los sistemas de control** aseguran el funcionamiento correcto de los movimientos (bucles de control), planificiación (trayectorias), etc.

Existen muchas maneras de clasificar a los robots, de acuerdo a su: estructura, función, arquitectura, movilidad, el medio de desarrollo, entre otras. Considerando la clasificación con base en su estructura los podemos dividir en:

- ✓ *Poliarticulados*: son fijos y pueden mover sus elementos terminales en un determinado espacio de trabajo con un número limitado de grados de libertad.
- ✓ *Móviles*: tienen la capacidad de desplazarse, basados en ruedas o algúna plataforma de propulsión. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores.
- ✓ *Androides*: reproducen total o parcialmente la forma y el comportamiento del ser humano.

- ✓ *Zoomórficos*: imitan los sistemas de locomoción de los seres vivos, los cuales pueden ser caminadores o no caminadores. Los no caminadores están basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de un movimiento relativo de rotación o de ser aéreos o acuáticos, estos tienen un sistema de propulsión adecuado. Los caminadores son mono o multípedos capaces de desplazarse en su entorno.
- ✓ *Híbridos*: aquellos de difícil clasificación cuya estructura se sitúa en la combinación de algunas de las anteriores, esto se puede apreciar en la figura 1.1.

a b c

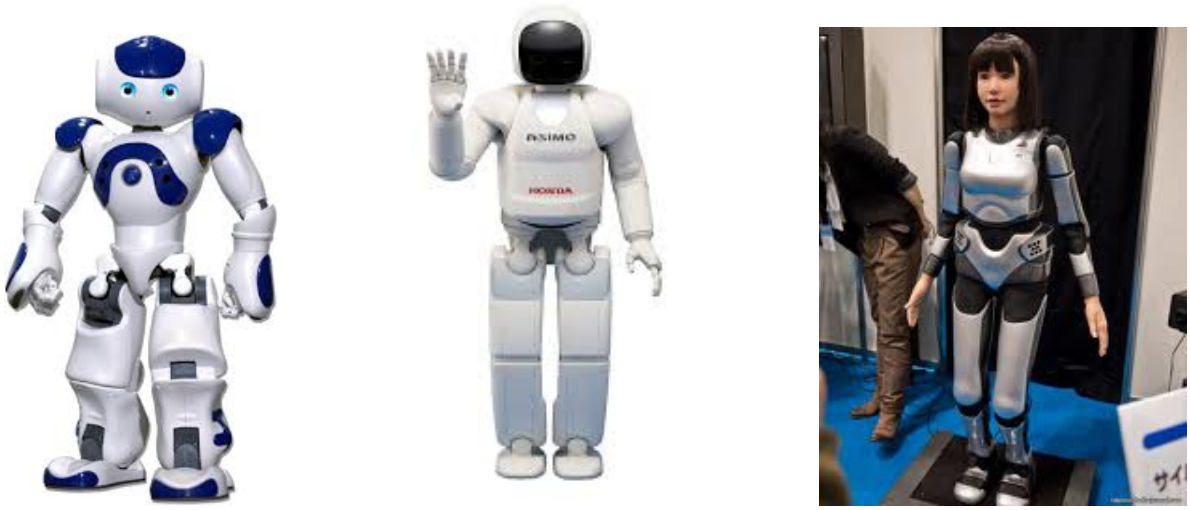
Figura 1.1: Robots Híbridos: *(a) Robot humanoide con ruedas* ; *(b) Robot Quadricóptero con ruedas*; *(c) Robot poliarticulado con orugas*

De estos ejemplos, los androides intentan reproducir el comportamiento del ser humano total o parcialmente, como son el NAO [2], el ASIMO [3] y el HRP-4C [4] (figura 1.2), este último considerado como ginoide, es decir, un robot humanoide con apariencia femenina. Los zoomórficos que imitan los sistemas de locomoción de los seres vivos, por ejemplo el Dragonfly [5], el Smartbird [6] y el AIBO [7] los cuales se pueden observar en la figura 1.3.

En la figura 1.4 se pueden observar algunos tipos de robots móviles, los cuales utilizan diferentes métodos de desplazamiento. Por ejemplo, en (a) es un robot omnidireccional que utiliza 3 omniruedas las cuales le permiten desplazarse en cualquier dirección, en (b) el robot móvil hace uso de orugas para su desplazamiento y en (c) se utiliza un método de desplazamiento similar al de las serpientes o gusanos para el movimiento del robot.

Es importante tomar en cuenta la aplicación de los robots, para poder saber cuál es la finalidad deseada y entender el comportamiento en el entorno.
Por lo tanto, clasificándolos con base en su aplicación los robots se pueden dividir en:

- ✓ *Industriales*: son destinados a realizar de forma automática procesos de fabricación y de manipulación
- ✓ *Seguridad y espacio*: son destinados a realizar misiones de seguridad civil o militar en tierra, aire y mar, e incluso misiones espaciales.
- ✓ *Servicio*: son destinados a la aplicación en los dominios de la vida como en entornos de salud, rehabilitación y lúdicos. Además, también se puede considerar su uso para aplicaciones en donde hay un ambiente peligroso para los seres vivos.



a b c

Figura 1.2: Robots Humanoides: *(a) Robot NAO*; *(b) Robot ASIMO*; *(c) Robot HRP-4C*

a b c

Figura 1.3: Robots Zoomórficos: *(a) Robot Dragonfly*; *(b) Robot Smartbird*; *(c) Robot AIBO*

Un robot de servicio, es un robot que opera semi o totalmente autónomo, para realizar servicios útiles para el bienestar de los seres humanos, con exclusión de las operaciones de fabricación [1]. Las características principales que se buscan en un robot de servicio son: inteligencia, flexibilidad y capacidad de realizar el trabajo que se le va a asignar. Estas características son enfocadas para que el robot realice tareas en entornos de interiores.

En la robótica móvil un robot debe tener la suficiente inteligencia para reaccionar y tomar decisiones basándose en observaciones de su entorno, sin suponer que este entorno sea perfectamente conocido. La autonomía de un robot móvil se basa en el sistema de navegación, en el cual se incluyen tareas de planificación, percepción y control. En los robots móviles, el problema de planificación, en el caso más general, puede descomponerse en: planificación global de la misión, de la ruta, de la trayectoria y, finalmente, evitar obstáculos no esperados. Un ejemplo de este tipo de robots móviles en donde se ponen en práctica estas tareas para realizar una navegación autónoma son los automóviles sin conductor (Driverless), que consisten en el desarrollo de autos que tengan la tecnología necesaria para circular de manera autónoma (figura 1.5). La primer

a b c

Figura 1.4: Robots Móviles: *(a) Robot omnidireccional* ; *(b) Robot con Orugas*; *(b) Robot gusano*

licencia para un coche autónomo fue expedida en mayo de 2012 en el estado Norteamericano de Nevada. Esta licencia fue para un Toyota Prius modificado con la tecnología experimental driverless de Google [8].

Figura 1.5: Automóvil Driverless

En un robot para interiores, la misión podría consistir en determinar a qué habitación hay que desplazarse, mientras que la ruta establecería el camino desde la posición inicial a una posición en la habitación, definiendo puntos intermedios de paso. Los robots móviles pueden desviarse de la ruta debido a la acumulación de imprecisiones mecánicas y de control [9], siendo éste un problema muy común en el seguimiento de trayectorias. La mayoría de los trabajos de investigación en esta área, tratan de reducir esta problemática mejorando el sistema de control.

En la clasificación mostrada en la figura 1.6 se aprecian los 3 principales medios en los cuales los robots móviles pueden desarrollar sus actividades. Dentro de dicha área se encuentran diferentes tipos de vehículos. Sin embargo, la presente tesis se enfoca en el uso de una variedad de helicóptero: Quadrotor.

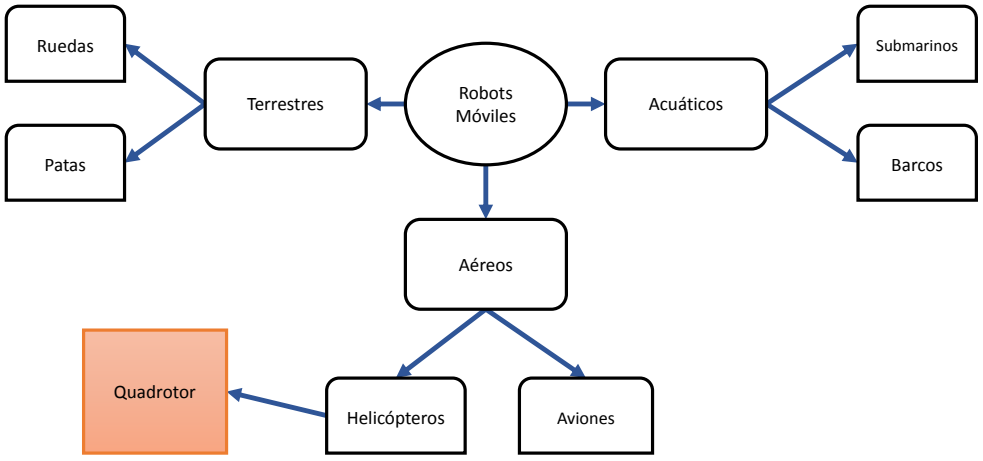


Figura 1.6: Tipos de Robots

1.1. Planteamiento del Problema

Los vehículos aéreos no tripulados están siendo de gran ayuda a los humanos para la realización de tareas tanto de entretenimiento, investigación, transporte, monitoreo y localización, entre otras. Para ello, los robots necesitan de técnicas de control, además de una cierta autonomía para su funcionamiento y el buen desarrollo de las tareas.

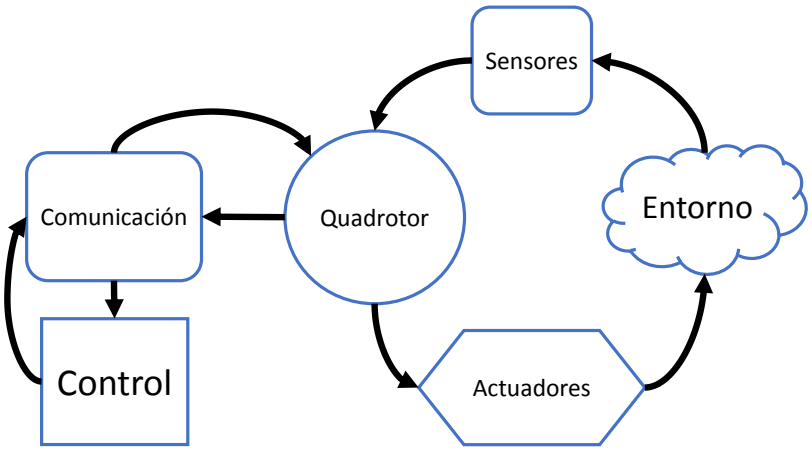


Figura 1.7: Esquema general del sistema de control del Quadrotor

Es así como, mediante el análisis de estas necesidades y las diferentes áreas de aplicación para los robots móviles, en este proyecto se controla el quadrotor Ar.Drone2.0 para que sea capaz

de seguir trayectorias específicas de manera autónoma. Por lo cual, fue necesario la implementación de un módulo de control.

El diagrama a bloques de la figura 1.7 es, de manera simplificada, el sistema implementado en este proyecto de tesis. En este diagrama se muestra cómo el quadrotor interactúa con el entorno y mediante el dispositivo de control es posible realizar el seguimiento de trayectorias de manera autónoma. El sistema obtiene la información del entorno por medio de los sensores externos, de manera que los datos obtenidos se toman como variables de entrada del módulo de control, mediante comunicación WiFi entre el quadrotor y una PC. El control regresa las variables de salida al sistema, con lo que modifica la velocidad angular de sus motores para producir algún movimiento, y estos a su vez influyen en su ubicación con respecto al entorno.

Esta tesis se enfoca en el desarrollo de una aplicación de software basado en el SDK (Software Development Kit) del fabricante, mediante la cual se realiza el control difuso para el seguimiento de las trayectorias. Para llevar a cabo ésto, se generaron un conjunto de módulos que son: (1) control, (2) comunicación, (3) trayectorias. Los cuales, en conjunto, sirven para el control del quadrotor y además, estos módulos de control se podrán utilizar para otra plataforma con características similares. El control se lleva a cabo mediante un algoritmo basado en lógica difusa y se utiliza el software Matlab-Simulink [10], para validar dicho control, tanto en simulación como en la plataforma experimental.

1.2. Justificación

Actualmente existe un gran avance en el desarrollo de nuevas tecnologías, en especial, existen muchos centros de investigación que realizan proyectos sobre temas relacionados con los quadrotores, tanto nacional como internacionalmente en diferentes ámbitos de investigación.

En la Universidad Tecnológica de la Mixteca (UTM) no se encontraron trabajos previos a esta investigación que hayan utilizado drones o quadrotores. Por lo cual se vio la oportunidad para realizar un trabajo de tesis con este tipo de dispositivos. Además permite sentar las bases para que en la UTM se desarrolle investigación usando quadrotores y sitúe a la Universidad dentro de los muchos centros de estudios e investigación que exploran esta área.

Por otro lado, el desarrollo de este proyecto mediante diversos módulos permitirá a estudiantes y profesores de la universidad, llevar a la práctica algunos conceptos de Ingeniería en Mecatrónica.

Pudiendo utilizar este módulo en materias como: Teoría de control, Diseño de sistemas de control, Sistemas digitales, Procesamiento de señales e imágenes, Sistemas de comunicación I y II, Robótica II. Considerando que en dichas materias el presente trabajo se podría mostrar como un ejemplo práctico de la aplicación de los conocimientos teóricos.

1.3. Hipótesis

A partir de la descripción del problema planteado y de las características y potencial que posee el AR.Drone, para la elaboración de esta tesis se comenzó formulando la siguiente hipótesis:

Es posible realizar el seguimiento autónomo de trayectorias mediante el Quadrotor Ar. Drone 2.0 utilizando un sistema de control implementado con lógica difusa.

1.4. Objetivos

Para la realización de este trabajo de tesis, se plantearon los siguientes objetivos:

Objetivo General

Desarrollar un módulo de control para el quadrotor *Ar.Drone 2.0* mediante lógica difusa de tal manera que sea capaz de realizar el seguimiento autónomo de trayectorias.

Objetivos Particulares

1. Diseñar una interfaz que permita al usuario la manipulación y control del quadrotor.
2. Desarrollar un control por lógica difusa para el seguimiento autónomo de trayectorias.
3. Generar escenarios para validar la implementación del control en el quadrotor, tanto virtual como real.

1.5. Robots Móviles

Dentro del grupo de robots móviles, los helicópteros reciben una diferente nomenclatura dependiendo del número de motores con los que cuente, desde 1 hasta n, siempre y cuando el tamaño lo permita. En la figura 1.8 se pueden apreciar diversos tipos de helicópteros y multicópteros. Como un ejemplo singular se tiene al multicóptero que ha revolucionado la aviación es el E-Volo's Volocoptero VC200, el cual es un helicóptero con 18 hélices y permite transportar a 2 personas (figura 1.9).

Dentro de los helicópteros, aquellos que poseen 4 hélices han tenido una gran aceptación como elementos de entretenimiento [11] o de investigación [12]. Este tipo de helicópteros son llamados quadrirotor, quadricóptero o quadrotor (en esta tesis se utilizará el término quadrotor), éstas son aeronaves con cuatro motores usados para su sostén y propulsión. A fin de lograr que el aparato se mantenga estable de manera horizontal respecto a su eje de orientación, es necesario que dos hélices giren en un sentido y las otras en sentido contrario.

a

b

c

d

Figura 1.8: Variaciones de Helicópteros: *(a) tricóptero*; *(b) Pentacóptero*; *(c) Hexacóptero*; *(d) Octacóptero*

Figura 1.9: Volocoptero VC200

El concepto de quadrotor no es nuevo. El primer quadrotor fue un helicóptero llamado Gyroplane No.1 [13], el cual voló en 1907 (figura 1.10).

En el aeromodelismo los quadrotores son vehículos a escala no tripulados (UAVs). Los cuales, además de servir de prueba para sistemas reales a menor costo, tienen muchas aplicaciones dependiendo del área en donde se utilicen.

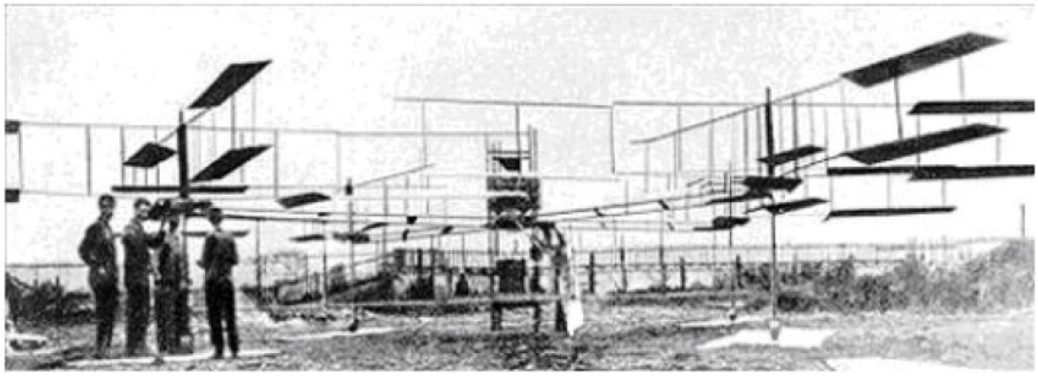


Figura 1.10: Bréguet Richet Gyroplane No.1

Los UAVs son actualmente una parte importante de estudios científicos ya que tienen la ventaja de proteger a vidas humanas de ambientes peligrosos, consiguiendo un sustituto a los vehículos necesariamente pilotados por humanos.

a b c

Figura 1.11: Vehículos Aéreos no tripulados *(a) Robot dirigible*; *(b) Robot aerodeslizador*; *(c) Robot helicóptero*

Dentro de los UAVs se puede mencionar diferentes tipos, por ejemplo: dirigibles, aeroplanos, helicópteros y en una subcategoría de estos últimos los quadrotores (figura 1.11).

Al elegir entre uno de estos vehículos para utilizarlos en una aplicación específica, o realizar investigación, se deben valorar los puntos a favor que cada uno tenga dependiendo de la necesidad deseada. Los aviones o aeroplanos, permiten una velocidad de desplazamiento alta y una cierta resistencia a perturbaciones externas; pero tienen como desventaja que no pueden realizar despegues o vuelos verticales y son para usos en exteriores unicamente. A diferencia de los aerodeslizadores y otros UAVs, los quadrotores son vehículos que permiten una mayor maniobrabilidad ya que su dinámica y capacidad de vuelo se lo permiten, siendo también óptimos para vuelos tanto en interiores como exteriores.

Tomando en consideración las características antes mencionadas, este proyecto se va a enfocar en un quadrotor, el cual es un helicóptero propulsado por cuatro motores eléctricos que consisten de 2 pares de rotores contrarios ubicados en las cuatro esquinas de la estructura [14]. Se pretende en este proyecto controlar el quadrotor para que sea capaz de seguir trayectorias específicas de manera autónoma. Por lo cual, para el seguimiento de trayectorias es necesario la

implementación del control adecuado.

1.6. Estado del Arte

Los UAV tienen una gran variedad de aplicaciones tanto en interiores como en exteriores. Dentro de las aplicaciones de los quadrotores se pueden mencionar las siguientes: La vigilancia aérea [15] de la frontera entre países, seguridad y vigilancia por las agencias de seguridad [16], sistemas de información agroalimentaria y pesquera [17], plataformas educacionales de aprendizaje [12], así como para servicios de paquetería utilizados por DHL [18], Amazon [19] e inclusive en el envío de pizza por Domino's (figura 1.12).

a b c

Figura 1.12: Servicios de Paquetería: *(a) DHL*; *(b) Amazon*; *(c) Domino's Pizza*

Actualmente existen varias plataformas de quadrotores de acceso al público como lo son el DJI Phantom [20], el Foxtech D130 [21], el Traxxas QR1 [11], el AeroDrone MR4 [22], el Ar.Drone [23], entre otros (figura 1.13). Además de las plataformas que son de acceso al público, existen muchas plataformas que han sido desarrolladas como parte de proyectos o incluso con fines específicos como los apreciados en la figura 1.12.

Muchas Universidades y centros de investigación se encuentran actualmente realizando investigaciones con quadrotores. Ejemplos notables son: el Computer Vision Group de la Universidad de Münich (TUM) [24], el laboratorio GRASP en la Universidad de Pennsylvania [25] y el Instituto Politécnico Nacional [26].

1.6.1. Control de Vehículos Aéreos

Para el desarrollo de todas las aplicaciones mencionadas anteriormente se requieren de diferentes niveles de control, algunas de las cuales pueden necesitar una precisión mayor y otras tener una tolerancia más grande. Un ejemplo de esto es la tesis de Dijkshoorn [27] en la cual realiza una investigación con el Ar.Drone enfocada en la localización y mapeo de forma simultanea (SLAM por sus siglas en inglés) para que un sistema pueda ubicarse en su entorno. Para lograr dicha tarea se necesita del uso de sus sensores y de un adecuado control, obteniendo información de su posición y orientación. Esta investigación es importante para que los robots de este tipo operen de manera autónoma. Para llegar a operar de esta forma, hoy en día se tienen diferentes técnicas

a

b

c

d

Figura 1.13: Plataformas de Quadrotores públicas: *(a) DJI Phantom*; *(b) Foxtech D130*; *(c) Traxxas QR1*; *(d) AeroDrone MR4*

de control y se necesita el conocimiento del modelo matemático del sistema para poder realizar un control correcto.

Hoffmann *et al.* [14], trabajaron en la dinámica y control de un quadrotor con PID (Proporcional Integral Derivativo), enfocando su investigación en el análisis de los efectos aerodinámicos que ocurren en la realización de sus movimientos. Logrando mejorar el control del quadrotor a altas velocidades.

Existen muchas otras investigaciones en las cuales se realiza el control de quadrotores, donde utilizan el modelo dinámico del sistema para el diseño del control. Esto se puede observar en los trabajos que en los párrafos siguientes se comentan.

En el reporte presentado por un grupo del Instituto de Sistemas Electrónicos de la Universidad de Aalborg [28]. Se realiza el control de un Quad-Rotor X3D y para ello, se obtuvo el diseño e implementación de un sistema de control que permite que la plataforma vuele de manera autónoma utilizando un sistema de ubicación Vicon. Este sistema consiste en un conjunto de cámaras, un módulo de control y una computadora con software para analizar los datos. Mediante este sistema se obtiene la posición y orientación del objeto deseado. El sistema de control está basado en un controlador lineal de retroalimentación de los estados, el cual fué probado tanto en el sistema linealizado como en el no lineal.

En la tesis de Bresciani [29], se realiza el control de un quadrotor enfocado en el despegue vertical y aterrizaje (VTOL, por sus siglas en Inglés). Para llevar a cabo esto, se calculó el modelo dinámico del sistema y el algoritmo de control. Se implementó tanto en simulación como en la plataforma real. El control fue realizado mediante el método PID.

Pounds *et al.* [30], realizaron el modelado, construcción y control del Quadrotor X-4 Flyer. Para el desarrollo de este trabajo se realizaron controles de velocidad, resolviendo problemas de par y de la dinámica del desempeño de los motores. Se incluyeron en el modelado, efectos aerodinámicos tanto de los rotores como de las hélices. Se comparan dos métodos de control, especificando que el PID (el elegido), es considerado óptimo y se implementó un controlador puramente integral.

En el artículo de Naidoo *et al.* [31], se realiza el modelado y control de un Quadrotor. Para su modelado se utilizó el método de Euler-Lagrange y se consideró la teoría de los momentos (momentum theory) y la teoría de los elementos de pala (blade element theory). Con el uso de estas teorías se permitió el cálculo de la capacidad de carga y el desempeño para realizar la elevación del sistema. El controlador fue un PD y para su implementación se uso Matlab-Simulink. Los resultados obtenidos tanto en simulación como en la plataforma real son muy similares y cumplen con las espectativas del proyecto, pero las diferencias mostradas en la implementación con la plataforma real son principalmente por los efectos producidos por el aire, ya que este parámetro no fue considerado.

Rodić y Mester [32], modelaron y simularon un quadrotor en un escenario virtual. Se realizó el simulador de vuelo específico para la investigación y desarrollo, con la finalidad de evaluar el diseño, modelado y control de sistemas quadrotores. Se consideran los principales aspectos de modelado, cinemática y dinámica de cuerpos rígidos, efectos espaciales de localización y navegación.

Para obtener el modelo dinámico del quadrotor existen diferentes métodos analíticos o numéricos. En el trabajo de Rubio *et al.* [33] se muestra una comparación de estos dos modelos para describir el comportamiento del quadrotor.

En la tesis de Özgür [34] se puede apreciar el diseño de un sistema de control para un vehículo aéreo de tipo quadrotor y también el modelado matemático necesario para el desarrollo del controlador para la estabilización, el cual se realizó el control utilizando el método LQR (del ingles linear Quadratic Regulator).

Besnard *et al.* [35], realizan el control de un quadrotor utilizando un control de modos deslizantes con un observador de perturbaciones también de modos deslizantes. La implementación de este método fue probada en simulación mediante Simulink.

Neff en su tesis [36], describe dos controladores diseñados específicamente para un vehículo aéreo no tripulado, quadrotor. Un controlador lineal y un controlador no lineal se desarrollan para su uso en el quadrotor. El controlador lineal es un controlador de orientación PID, basado en controlar los ángulos de la UAV. El controlador no lineal se desarrolla utilizando métodos de

estabilidad de Lyapunov. El objetivo de diseño para este controlador es añadir un posicionador de cámara de dos grados de libertad a la quadrotor para un total de seis grados de libertad. El UAV hará un seguimiento de tres velocidades de traslación deseados y tres velocidades angulares utilizando sólo las velocidades de traslación y rotación para la retroalimentación. Se realizaron simulaciones para verificar ambos controladores y los experimentos se llevaron a cabo utilizando este controlador en un quadrotor DraganFlyer X-Pro [37].

1.6.2. Aplicaciones de Lógica Difusa en la Robótica

Por otro lado, el control por lógica difusa está diseñado principalmente para sistemas no-lineales o complejos en los cuales es difícil obtener o construir un modelo matemático exacto o para cuando no se tiene un modelo, de tal manera que conociendo el funcionamiento del sistema es suficiente para realizar su control. Por ejemplo, el control de un péndulo invertido que podemos apreciar en [38]. En este trabajo se muestra cómo el sistema dinámico del péndulo, puede ser controlado sin la necesidad del modelo. Sin importar la posición deseada, el control difuso es capaz de llevar el péndulo a esa posición. Así como éste, podemos observar diferentes ejemplos de la aplicación de la lógica difusa como los descritos en los párrafos siguientes.

González y Bravo [39], desarrollaron un controlador fuzzy enfocado en el cambio del centro de gravedad de un sistema para evitar que éste, se caiga en terrenos irregulares. Para el desarrollo de este controlador se consideró el concepto de Vector de Distancia Mínima (VDM). Con el control difuso se regula o ajusta la posición del robot sobre la plataforma móvil, produciendo una estabilidad cuasi-estática. Para el controlador se desarrollaron dos tipos de control difuso, realizando comparaciones entre ellos para validar cual es mejor, concluyendo que la implementación de un control difuso es eficiente y además no necesita de las matemáticas inherentes al proceso, permite un ajuste y entonación simple del controlador permitiendo un control robusto.

En la tesis de Jiménez [40], se presenta la forma en la cual se diseñó y construyó un sistema para controlar la temperatura de un horno eléctrico para la cocción de piezas cerámicas. Se empleó un controlador difuso de tipo Mamdani. El control permite un encendido y apagado mediante ciclos completos de la línea de alimentación de voltaje que nos proporciona CFE.

En el trabajo de Vargas *et al.* [41], se presenta los resultados del diseño y desarrollo de un prototipo basado en visión artificial y lógica difusa para identificar procesos de representación espacial en discapacitados visuales que utilizan este dispositivo como ayuda aumentativa.

También se ha usado la lógica difusa para mejorar la activación del sistema de ABS (Anti-lock Brake System) a partir de reglas de control difuso y visión por computadora. Este sistema se ha implementado tanto en simulación como en laboratorio [42].

Finalmente, está el uso de un controlador híbrido mediante una combinación de un control Difuso y un control PID, para controlar los ángulos *Pitch* y *Yaw* de un cohete [43].

1.6.3. Control de Quadrotores mediante Lógica Difusa

En la tesis de maestría de Morata [44], se realiza un control del movimiento de un quadricóptero empleando una combinación de dos métodos de control, Lógica difusa y PID. Usando un control de PID difuso para cada variable de control (la altura y los 3 ángulos de inclinación) y uniéndolos después en un módulo de agregación.

Martin [45] en su investigación realiza el control de un quadrotor Ar.Drone. Para poder llevar a cabo el desarrollo de su trabajo hace uso del SDK que provee el fabricante del drone (Parrot). Para la implementación considera al drone como una caja negra y para realizar el control en simulación obtiene esa caja negra mediante el toolbox System Identification con el cual se obtiene el modelo del sistema sólo con los valores de entrada y salida previamente medidos de la plataforma real.

Raza y Gueaieb [46], describen los diferentes pasos para diseñar, construir, simular y probar un módulo de control inteligente para un quadrotor. Para llevar a cabo el control, utilizan lógica difusa y con esto muestran que este tipo de control ofrece ventajas en comparación con ciertos métodos de control convencional. En especial al tratar con un sistema altamente no lineal y no conocer el modelo exacto.

En la tesis de Bhatkhande [47] se observa el uso de un control difuso en la estabilidad de un quadrotor, probando sus resultados en diversas plataformas para mostrar la ventaja de este método. Para el control, se combinan redes neuronales con lógica difusa para conseguir de esta manera un mecanismo de aprendizaje de las reglas difusas.

En la tesis de Martínez [48] se pueden apreciar las características particulares de las dos versiones del ArDrone (Ar.Drone1.0 y Ar.Drone2.0). Además de esto, se describen detalladamente las funciones del kit de desarrollo de software y muestra como se puede utilizar para desarrollar aplicaciones utilizando esta herramienta, principalmente aplicaciones enfocadas para juegos.

En [49] se presenta un control difuso mediante Takagi-Sugeno de un modelo de seis grados de libertad, un helicóptero de cuatro rotores. Para este modelo, una retroalimentación de los estados con un compensador paralelo distribuido se ha diseñado utilizando un algoritmo de optimización de identificación de modelos lineales (LMI, por sus siglas en inglés) para garantizar condiciones de estabilidad. Se consigue la estabilización sobre las coordenadas de traslación y angulares. Las simulaciones se han realizado sin tener en cuenta el efecto del viento y las turbulencias.

El control de la estabilización de un quadrotor mediante un método de control robusto adaptativo-difuso se realizó en [50], donde mediante la simulación demuestran el funcionamiento de este método para controlar un quadrotor.

En situaciones en las cuales el desempeño del sistema implique un reajuste continuo del controlador, es posible realizarlo mediante sistemas de inferencia difusa auto ajustables. Un ejemplo de esto, se puede apreciar en [51], en donde se presenta una comparación de este método con un sistema de inferencia difusa basado en una red adaptativa.

Capítulo 2

Quadrotor Ar.Drone 2.0

En este capítulo se proporcionan las bases para entender el funcionamiento de un quadrotor. Además se explican brevemente las diferentes técnicas de control consideradas para controlar al quadrotor Ar.Drone. También a lo largo del capítulo se describe el hardware necesario para el desarrollo de esta tesis, así como el software usado para la implementación tanto para la simulación como para la plataforma real. Finalmente, se describe el kit de desarrollo de software necesario para la implementación de aplicaciones con el quadrotor. Este kit de desarrollo es proporcionado por la empresa fabricante del Drone para la programación de aplicaciones.

2.1. Características del Ar.Drone 2.0

La plataforma a utilizar para este trabajo de tesis es el Ar.Drone2.0® [23] de la empresa Parrot (figura 2.1). Esta plataforma es un quadrotor controlado mediante radiofrecuencia y es usado para actividades lúdicas.

Las características que posee el Ar.Drone 2.0 son las siguientes:

✓ Sensores:

- Acelerómetro de 3 ejes +/-50mg
- Giroscópio de 3 ejes 2.000/s
- Magnetómetro de 3 ejes, precisión de 6 grados
- Cámaras QVGA a 60fps y HD(720ppx) a 15fps
- Presión +/-10 Pa
- Ultrasónico

✓ Actuadores:

- 4 Motores sin escobillas. 14.5W, 28500RPM

✓ Comunicación:

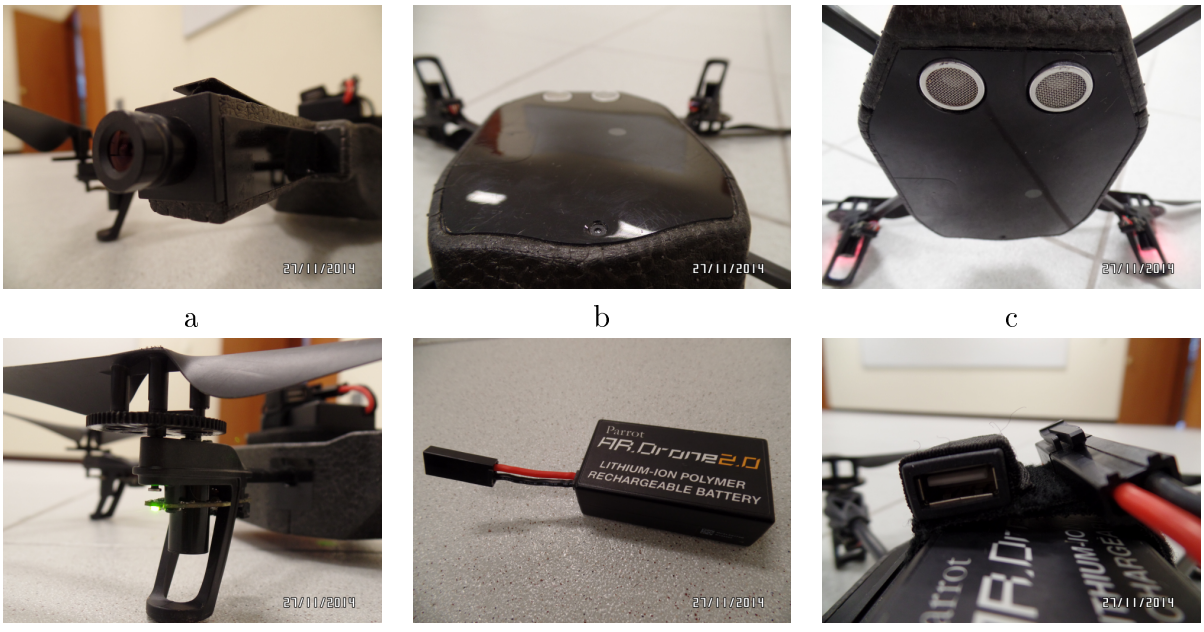
- Módem Wifi(b/g/n)

Figura 2.1: Quadrotor Ar.drone2.0

- Conector USB 2.0

✓ Batería:

- Li-Po recargable de 1000mAh a 11.1V.



a b c

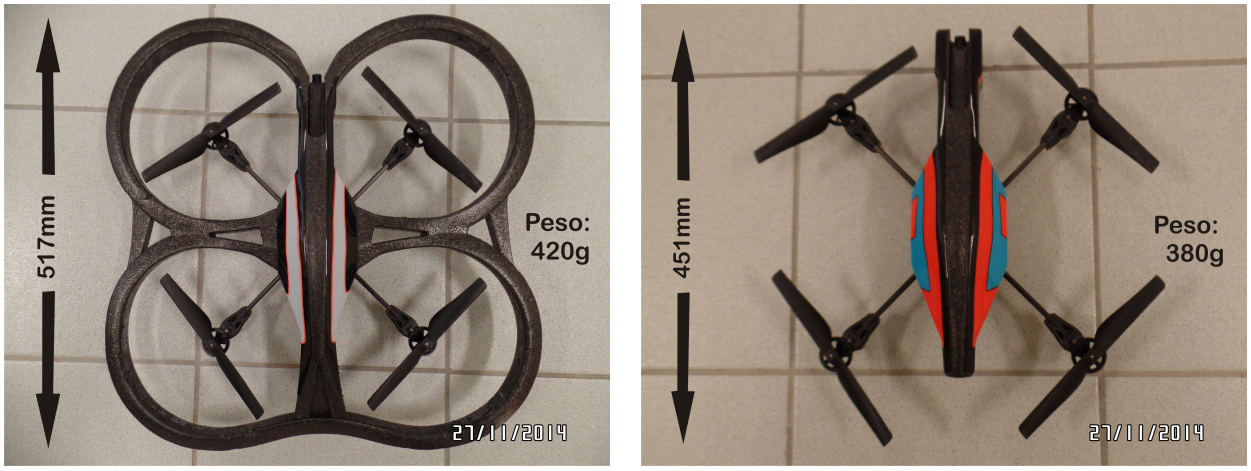
d e f

Figura 2.2: Componentes del Drone: *(a) Cámara Frontal*; *(b) Cámara Inferior*; *(c) Sensor Ultrasónico*; *(d) Rotor*; *(e) Batería*; *(f) Puerto USB*

Algunos de los componentes visibles del drone mencionados en las características son mostrados en la figura 2.2.

La estructura mecánica del quadrotor AR.Drone 2.0 comprende de cuatro rotores unidos a los extremos de una cruz, en cuyo centro se encuentran tanto los sensores como la batería además de toda la electrónica de control y comunicaciones del sistema. El sistema electrónico consiste en un microprocesador ARM Cortex A8 de 32 bits a 1GHz, una memoria RAM DDR2 de 1GB a 200MHz, un microcontrolador AVR de 8 MIPS por cada motor y un sistema operativo con núcleo Linux 2.6.32.

Este quadrotor tiene dos carcasas intercambiables mostradas en la figura 2.3: a) la carcasa de interiores que protege las aspas del contacto con otras superficies y b) la carcasa para el uso de exteriores. Las medidas de los lados son de 517mm y 451mm respectivamente. Los pesos del sistema en ambos casos es de 420g para interiores y 380g para exteriores.



a b

Figura 2.3: Carcasas del Ar.Drone2.0: *(a) Interiores*; *(b) Exteriores*

El primer paso antes de la etapa de control es entender el funcionamiento del sistema y realizar el modelado adecuado de la dinámica del sistema. Esta fase, es de esperar, facilitará el control de la aeronave, ya que proporcionará una mejor comprensión de las capacidades y limitaciones del sistema en general. Por lo cual en las siguientes secciones se analizará el funcionamiento del quadrotor así como el modelado matemático del mismo.

2.2. Funcionamiento del Quadrotores

El quadrotor, según Martin[45], es un sistema sub-actuado con 6 GDL (grados de libertad) pero solo 4 entradas, las cuales pueden ser definidas en términos de sumas y diferencias de las velocidades de los motores. Esto permite que las ecuaciones de control, sean cuatro ecuaciones independientes.

El quadrotor se considera un sistema inestable y requiere de un control adecuado para lograr

su movimiento. Sin embargo, es esta inestabilidad del sistema lo que hace posible la facilidad de movimiento y aprovechar la gran maniobrabilidad del sistema en el entorno.

Para poder dirigir el dispositivo cada uno de los pares de hélices opuestas deben girar en el mismo sentido, donde un par está girando en sentido horario y el otro sentido antihorario. Los rotores deben girar a una misma velocidad para que el sistema se mantenga suspendido en el aire a una altura constante; si se necesita que el dispositivo se eleve o se baje de su posición sobre el eje Z, se logra aumentando o disminuyendo la velocidad de los 4 rotores equitativamente.

El control del movimiento del vehículo se consigue variando la velocidad relativa de cada rotor para cambiar el empuje y el torque producido por cada uno de ellos [52]. La figura 2.4 permite apreciar las variaciones necesarias de los rotores para realizar algún movimiento en específico y esto se explica en los puntos siguientes:

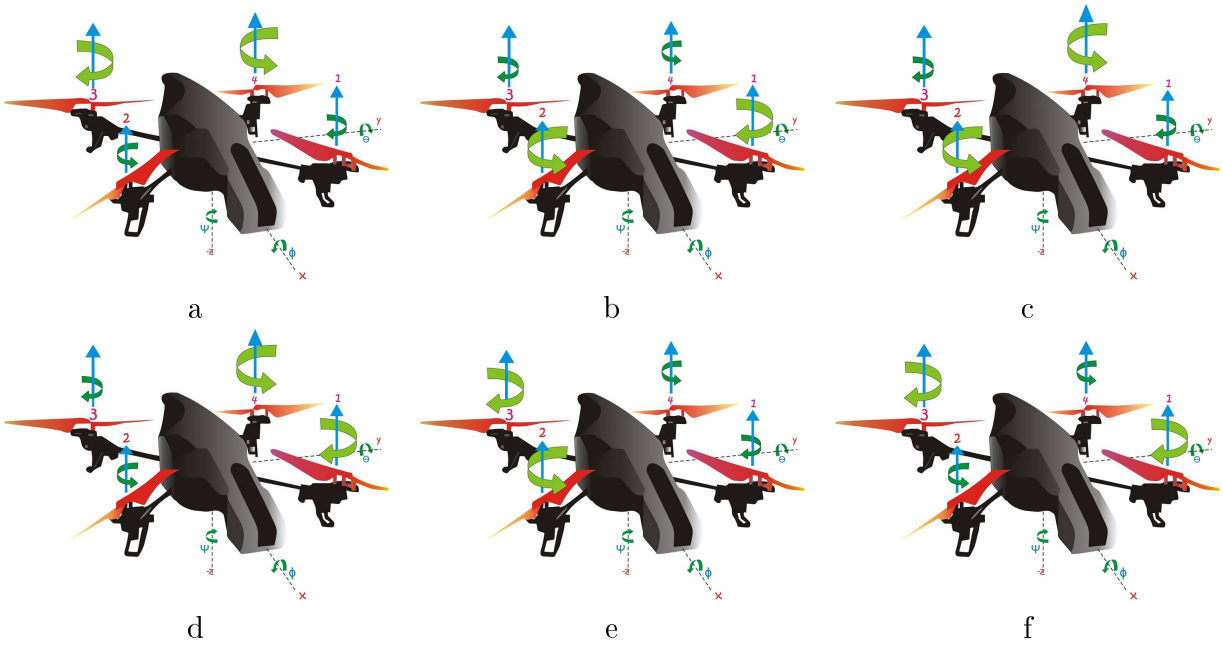


Figura 2.4: Movimientos Basicos del Drone: *(a) Adelante*; *(b) Atrás*; *(c) Giro Derecha*; *(d) Izquierda*; *(e) Derecha*; *(f) Giro Izquierda*

- ✓ Para un movimiento hacia adelante es necesario que los rotores 3 y 4 tengan una velocidad mayor a los demás rotores.
- ✓ Para un movimiento hacia atrás es necesario que los rotores 1 y 2 tengan una velocidad mayor a los demás rotores.
- ✓ Para un movimiento hacia la derecha los rotores 1 y 4 deben tener una velocidad mayor a los demás rotores.
- ✓ Para un movimiento hacia la izquierda los rotores 2 y 3 deben tener una velocidad mayor a los demás rotores.

✓ Para un giro hacia la derecha los rotores 1 y 3 deben tener una velocidad mayor a los demás rotores.

✓ Para un giro hacia la izquierda los rotores 2 y 4 deben tener una velocidad mayor a los demás rotores.

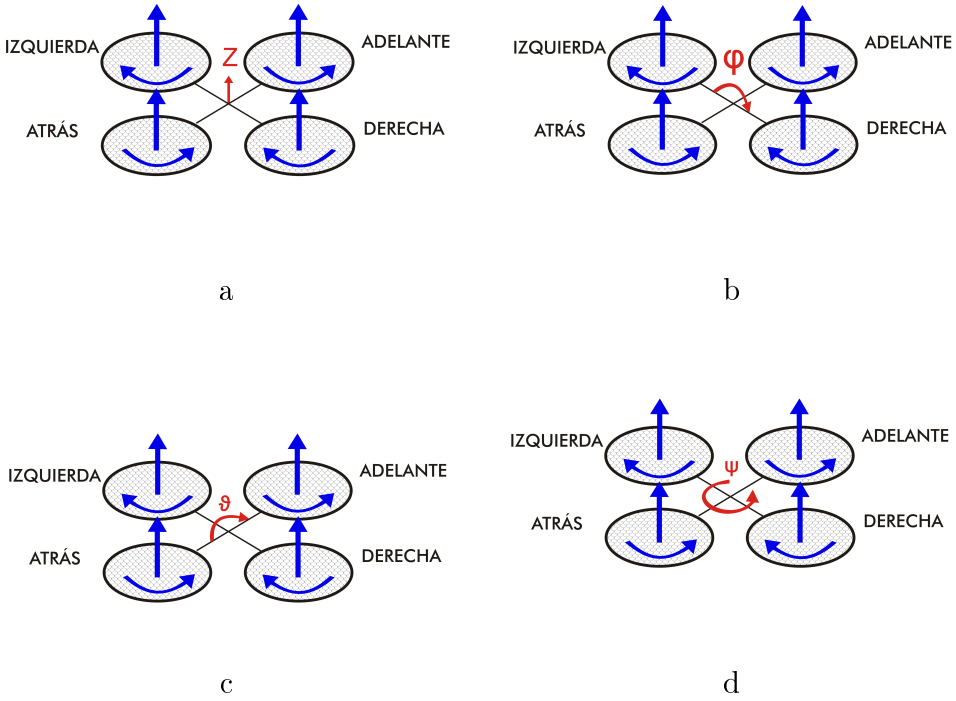


Figura 2.5: Variables de control: *(a) Elevación*; *(b) Roll*; *(c) Pitch*; *(d) Yaw*

Según la guía de desarrollo de Parrot [53] las maniobras se obtienen cambiando los ángulos *roll, pitch y yaw* del quadrotor, así como se muestra en la figura 2.5. Variando los motores izquierdos y derechos en direcciones opuestas se produce el *pitch*. Esto permite ir adelante o hacia atrás. La variación de los motores delanteros y traseros en direcciones opuestas se produce el *roll* con el cual el sistema se desplaza de izquierda a derecha. La variación de la velocidad de cada par del rotor de manera opuesta produce el *yaw*. Esto permite girar a la izquierda o derecha.

2.3. Modelado del Quadrotor

Una vez entendido el funcionamiento de los quadrotores y considerando el sentido de giro de cada uno de los rotores se establece el modelo del Quadrotor tanto de forma matemática como mediante el uso de un método de Identificación del Sistema.

2.3.1. Modelo Matemático Teórico

En el diagrama de cuerpo libre de la figura 2.6 se pueden observar las fuerzas y momentos que se producen en el sistema.

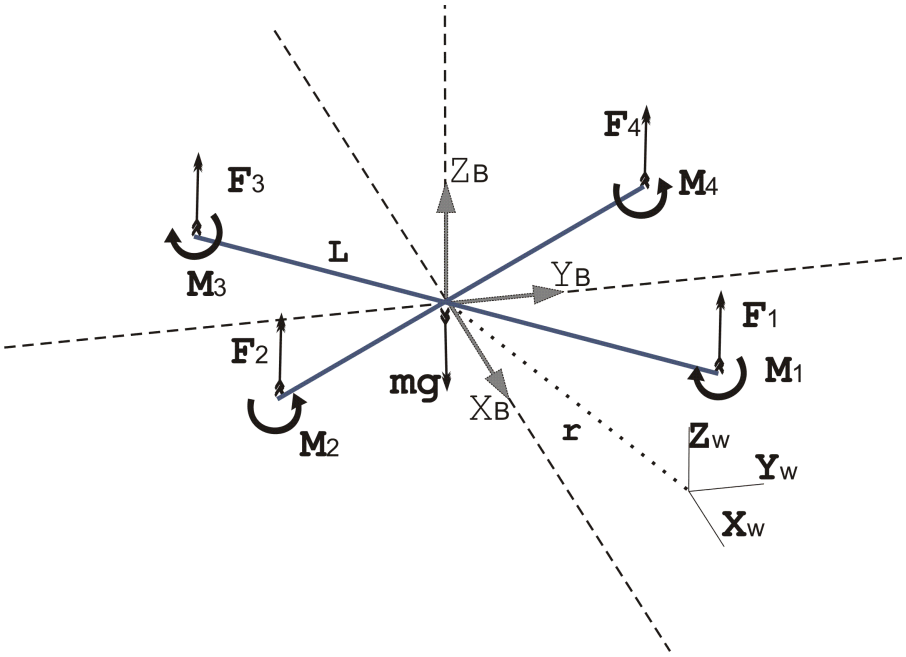


Figura 2.6: Diagrama de Cuerpo libre del Quadrotor

El origen del sistema es $(X, Y, Z)_B$, colocado en su centro de gravedad. La orientación del eje de referencia del Quadrotor respecto del eje de referencia en la Tierra es $(X, Y, Z)_W$.

La matriz de rotación del sistema es descrita en la ecuación, en la cual se considera a $C = cos, S = sen, \psi = Yaw, \theta = Pitch, \phi = Roll$.

$$R = \begin{pmatrix} C_\psi C_\theta - S_\phi S_\psi S_\theta & -S_\psi C_\phi & C_\psi S_\theta + C_\theta S_\phi S_\psi \\ S_\psi C_\theta + S_\phi C_\psi S_\theta & C_\psi C\phi & S_\psi S_\theta - C_\theta S_\phi C_\psi \\ -C_\phi S_\theta & S_\phi & C_\theta C_\phi \end{pmatrix}$$

Considerando que las fuerzas que actúan sobre el sistema son: la gravedad y las fuerzas de los rotores. Se tiene la siguiente Dinámica Traslacional.

$$m\ddot{r} = \begin{pmatrix} \sum F_i \left(C_\psi S_\theta + C_\theta S_\phi S_\psi\right) \\ \sum F_i \left(S_\psi S_\theta - C_\theta S_\phi C_\psi\right) \\ \sum F_i \left(C_\theta C_\phi\right) - mg \end{pmatrix}$$

El modelo linealizado del sistema realizado en [45], fue obtenido mediante series de Taylor de primer orden y es descrito a continuación. Para este modelo se considera que, para un estado de operación de un vuelo que permanezca estacionario se tiene lo siguiente:

$$\bar{W}_1 = \bar{W}_2 = \bar{W}_3 = \bar{W}_4 = W_{h'}$$

$$(W_h)^2 = \frac{mg}{4K_F}$$

$$\bar{\phi} = \bar{\theta} = 0$$

$$\psi = \psi_0$$

La Dinámica Traslacional linealizada es la siguiente:

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} g\left(\theta\cos\psi_0 + \phi\sin\psi_0\right) \\ g\left(\theta\sin\psi_0 - \phi\cos\psi_0\right) \\ \frac{2K_F W_h}{m}\left(W_1 + W_2 + W_3 + W_4 - 4W_h\right) \end{pmatrix}$$

La Dinámica Rotacional linealizada donde (p,q,r) son las velocidades angulares respecto de los 3 ejes (X,Y,Z), es la que a continuación se muestra:

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \begin{pmatrix} \frac{2LK_F W_h}{I_{xx}}\left(W_2 - W_4\right) \\ \frac{2LK_F W_h}{I_{yy}}\left(W_3 - W_1\right) \\ \frac{2K_M W_h}{I_{zz}}\left(W_1 - W_2 + W_3 - W_4\right) \end{pmatrix}$$

Considerando que las ecuaciones descritas anteriormente son las ecuaciones básicas para describir la dinámica del sistema. Éstas no se desarrollan de una manera más detallada, debido a que el modelo matemático no se utilizará en este trabajo para realizar el control del sistema.

2.3.2. Identificación del Sistema

Existen diferentes maneras de obtener el modelo de un sistema, incluso es posible mediante el uso de software especializado. Como lo es el toolbox de Matlab "*System Identification Toolbox*" [54], es una herramienta que permite la construcción de modelos de sistemas dinámicos a partir de datos de Entrada-Salida medidos. En [55], Ljung describe de una manera subjetiva el uso de la técnica de Identificación del Sistema.

Como ya se ha comentado en los párrafos anteriores el uso del modelo dinámico del sistema es importante para el control del mismo, y existen diferentes metodologías para obtenerlo. Considerando lo anterior, se tomó en cuenta que el uso de este toolbox es adecuado para la realización de este trabajo de investigación.

En este método se considera al sistema como una caja negra en la cual las variables de entrada a ésta, sean las entradas de control. A la salida de la caja se obtiene el estado del sistema, de donde se tomaran las variables para poder interpretar la posición y orientación del Drone. En la figura 2.7 se puede observar el ArDrone como una caja negra en la cual se muestran las

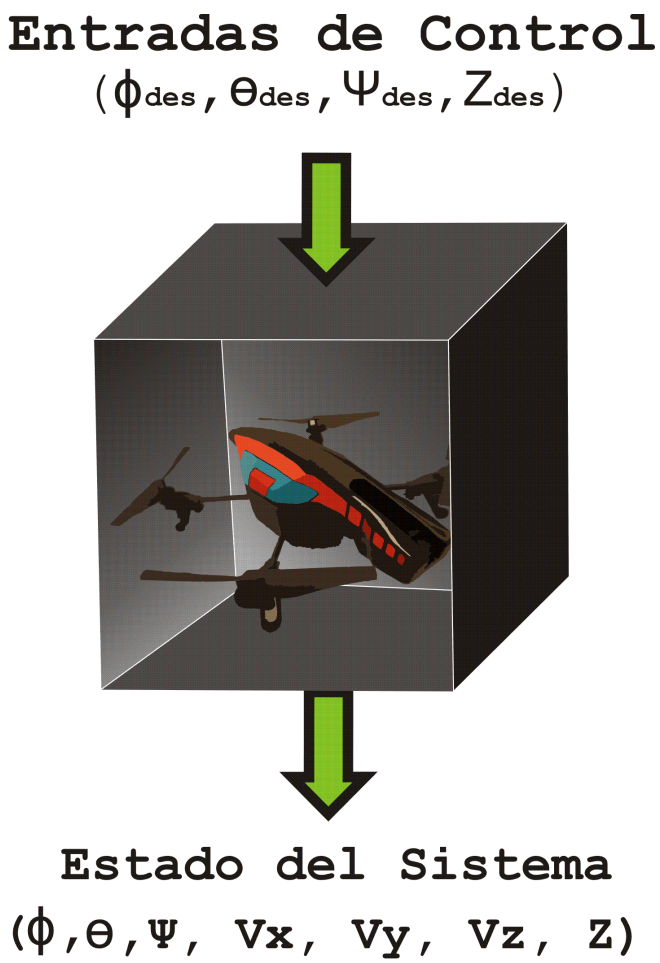


Figura 2.7: Drone visto como Caja Negra

variables de entrada y salida de la misma.

Ejemplos de la caracterización de las entradas y salidas para la obtención del modelo del Quadrotor Ar.Drone se pueden observar en [45] y [56], los cuales realizan pruebas con señales tipo rampa con variaciones de -1 a 1. Considerando que el cálculo del modelo no es un requerimiento fundamental para el desarrollo de esta tesis, se hace uso del modelo existente en el ARDrone Simulink Development Kit [57]. Dicho kit se explica en la sección 2.6.

Cabe mencionar que este modelo solo se utilizó en la simulación, ya que en la implementación en la plataforma real para el control mediante lógica difusa, no se hace uso del modelo matemático.

2.4. Métodos de control

En términos de la teoría de control, una planta puede ser desde una parte de un equipo o hasta un conjunto de los elementos de una máquina que funcionan juntos, y cuyo objetivo

es efectuar una operación particular [58]. Para controlar una planta o sistema existen muchos métodos de control, los cuales han ido evolucionando y las técnicas de control han mejorado a través de los años; sin embargo, a pesar de que es importante conocer la teoría básica de control y los métodos clásicos, existen métodos de control moderno que permiten controlar a los sistemas de manera que sean adaptables y robustos.

En las siguientes secciones se mencionan algunos métodos de control existentes que han sido usados para controlar, principalmente, sistemas robóticos como el que se utiliza en este trabajo.

2.4.1. Controlador PID

Es un mecanismo de control en el cual mediante la retro-alimentación se calcula la desviación o el error entre un valor medido y un valor deseado, para aplicar una acción correctora que ajuste el proceso. El algoritmo de cálculo del control se da en tres parámetros distintos: el proporcional, el integral, y el derivativo. La ganancia *Proporcional* determina la reacción del error actual. La *Integral* genera una corrección proporcional a la integral del error, esto asegura que, aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero. La *Derivativa* determina la reacción del tiempo en el que el error se produce. La suma de estas tres variables en el algoritmo de control del PID produce un controlador cuya respuesta es mejor a la de cualquiera de las tres variables de forma independiente.

En el artículo de Curi *et al.* [59], se puede observar la aplicación de este tipo de controladores donde se realizó el control de un quadrotor mediante un controlador PD y la implementación de este desarrollo se llevo a cabo en ROS (Sistema Operativo para Robots).

Otro ejemplo lo vemos en [60], en el cual Aguilera *et al.* trabajaron con un robot con ruedas de forma diferencial, el cual para seguir una trayectoria, partiendo de su origen, hasta el punto objetivo, utiliza el control PID.

Para el seguimiento de trayectorias de un manipulador robótico, Merchán en su tesis [61], teniendo la cinemática y dinámica de un brazo manipulador, utiliza el PID para el seguimiento de trayectorias.

En el proyecto desarrollado por Verdeja *et al.* [62], realizan el control de un vehículo aéreo no tripulado (un helicóptero) aplicando el PID, ya que, según su investigación, deducen que es mucho más eficiente y exacto en esta situación que la lógica difusa. Esto debido a que la complejidad del sistema no es muy alta y se conoce de manera precisa el modelo matemático.

El control de un quadrotor mediante el PID se puede observar en el trabajo de Bresciani [29]. En dicha investigación se calculó el modelado dinámico del sistema utilizando Newton-Euler. Se implementó tanto en simulación como en la plataforma real, realizando el simulador en Matlab-Simulink. Con el uso de un Microcontrolador (MCU) fue posible la realización de vuelos guiados y/o vuelos de manera autónoma. Se utilizó un sonar y un módulo infrarrojo para la detección de obstáculos.

2.4.2. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) son un paradigma de aprendizaje y procesamiento automático inspirado en la forma en que funciona el sistema nervioso de los animales y los humanos. Se trata de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. En inteligencia artificial es frecuente referirse a ellas como redes de neuronas o redes neuronales. Estas redes de neuronas, son conocidas por la aplicación en técnicas de control adaptativo para sistemas no lineales

Las redes neuronales consisten en unidades de procesamiento que intercambian datos de información y se utilizan para reconocer patrones, además estos sistemas son capaces de aprender y mejorar su funcionamiento. Éstas se pueden clasificar como de tipo biológicas o dirigidas a su aplicación.

De acuerdo a Rao [63] una red neuronal es una estructura computacional inspirada en el estudio del proceso biológico neuronal. La aplicación de este método se puede observar en el trabajo realizado por Pacheco [64], donde se realizó el monitoreo en línea de los cambios de color para muestras de Jamaica en procesos industriales. Arroyo [65], desarrolló un mecanismo de detección de fallas del sistema de Pilotaje automático del STC Metro de la Cd. de México. En la investigación se compararon dos métodos de reconocimiento de patrones, de los cuales uno de ellos era RNA's(entrenadas mediante retropropagación).

Un ejemplo de uso de las redes neuronales artificiales aplicadas a los robots móviles se pueden apreciar en el artículo realizado por Rios *et al.*[66], en el cual, con un robot con ruedas, se muestra la potencialidad de este método para la solución de problemas comprobando la efectividad en la clasificación de superficies.

2.4.3. Lógica Difusa

Las lógicas multivaluadas son aquellas en las cuales hay un mayor número de valores de verdad, por lo cual permiten agregar más valores aparte de falso y verdadero [67]. La lógica pertenece al grupo de las lógicas multivaluadas. Ésta pretende producir resultados exactos a partir de datos imprecisos, donde los valores de verdad tienen cierta incertidumbre.

Con estos valores de verdad se realiza un algoritmo en el cual se especifican varias reglas, con las cuales el sistema es capaz de tomar una decisión de qué proceso realizar dependiendo de las entradas.

La aplicación de la lógica difusa es útil en situaciones donde se tienen procesos complejos, no existe modelo de solución simple, se tienen conceptos ambiguos o imprecisos, situaciones donde las partes de un proceso o sistema se desconocen o su medición no es fiable.

Ross [68], afirma que el principal beneficio de los sistemas difusos es la aproximación al comportamiento del sistema donde las relaciones numéricas o analíticas no existen. Además, los sistemas difusos son útiles en dos contextos principales: (1) Situaciones en las cuales se trabaja con sistemas complejos en donde su comportamiento no esta del todo entendido, y (2) Situaciones en las que una solución aproximada pero rápida es garantizada.

Un ejemplo de aplicación de la lógica difusa es el trabajo realizado por Vargas *et al* [41], el cual tiene como propósito evaluar la eficacia de un dispositivo tecnológico como ayuda para discapacitados visuales, en la representación que hacen del entorno que les rodea y su incorporación en el sistema de representación espacial de ellos mismos.

En el trabajo de Raza y Gueiaeb [46] se proponen dos métodos de control por lógica difusa, los cuales fueron diseñados y puestos en práctica en un quadrotor desarrollado por ellos mismos. Ambos métodos fueron evaluados en simulación y después llevados al prototipo, obteniendo resultados que muestran un comportamiento de control exitoso.

Considerando las posibles aplicaciones y situaciones donde es útil utilizar este método, además de considerar que en la literatura se han obtenido resultados aceptables, se decidió trabajar con la Lógica Difusa como método de Control.

2.5. Sistema de Comunicación

La Comunicación es la transferencia de información con sentido desde un lugar (origen, transmisor) a otro lugar (destino, receptor). Por otra parte, la información es un patrón físico al cual se le ha asignado un significado comúnmente acordado. El patrón debe ser único, capaz de ser enviado por el transmisor y capaz de ser detectado y entendido por el receptor.

En toda comunicación existen tres elementos básicos (imprescindibles uno del otro) de un sistema de comunicación: el transmisor, el canal de transmisión y el receptor.

Las comunicaciones móviles se dan cuando tanto el emisor como el receptor están, o pueden estar, en movimiento. La movilidad de estos dos elementos que se encuentran en los extremos de la comunicación hace que no sea factible la utilización de hilos (cables) para realizar la comunicación entre dichos extremos. Por lo tanto utilizan básicamente la comunicación vía radio.

El Ar.drone2.0 puede ser controlado por cualquier dispositivo que soporte WiFi, ya que esta plataforma crea una red y asigna una dirección IP(típicamente la 192:168:1:1). El alcance de la conexión WiFi le permite llegar a altitudes de hasta 50-120 metros (dependiendo de las condiciones climáticas).

La frecuencia en la cual se realiza el envío de datos del drone y hacia él es de 15 veces por segundo en modo Demo y 200 veces por segundo en modo Debug.

Para su control se necesita tener acceso a los puertos asignados por el fabricante, los cuales son explicados en el kit de desarrollo de software.

Según los datos transmitidos en la comunicación con el drone, la emisión desde el dispositivo de control se realiza de una u otra forma. Para esto existen 4 vías principales: *AT Commands, Navdata, Video Stream y Datos críticos.*

Figura 2.8: AR.FreeFlight (App Oficial)

En el mercado existen numerosas aplicaciones de dispositivos móviles para el AR.Drone, desde la aplicación oficial que permite controlar el vuelo del dispositivo, hasta aplicaciones para realizar luchas entre drones o de Realidad Aumentada. Por nombrar algunas: *AR.FreeFlight* (Aplicación oficial de Parrot, figura 2.8), *Flight Record, Drone Ace, AR.PowerFlight, Drone Master, Flying Ace*, etc.

2.6. Kit de Desarrollo de Software ArDrone

Un kit de desarrollo de software o SDK, es un conjunto de herramientas que le permite al programador crear aplicaciones para un sistema concreto. Frecuentemente, incluyen también códigos de ejemplo y notas técnicas de soporte u otra documentación que pueda ayudar a clarificar ciertos puntos.

El SDK que ofrece Parrot para el Ar.Drone2.0 permite que desarrolladores externos a la empresa puedan tener acceso a las funciones principales, para poder realizar aplicaciones propias, enfocadas principalmente en la creación de nuevos juegos [48]. El lenguaje de programación depende principalmente del sistema operativo, aunque preferentemente es utilizado el lenguaje C. Los posibles sistemas operativos son: Linux, iOS, Android y Windows. Este kit permite incluso, desarrollar aplicaciones en cualquier dispositivo programable con una tarjeta de red Wifi

y protocolos de comunicación TCP/UDP/IP.

En la guía de desarrollo del SDK [52] se explica como utilizar el kit, se describen los protocolos de comunicación del drone y también se especifica que no es posible programar código directamente en la plataforma, ni permite tener acceso directo al hardware (sensores y actuadores). Esta guía contiene las siguientes librerías:

- ✓ **AR.Drone2.0 Library (ARDroneLIB)**, la cual provee al usuario las API's (del inglés application programing interface) necesarias para comunicarse y configurar fácilmente un Ar.Drone2.0
- ✓ **AR.Drone 2.0 Tool library (ARDroneTool)**, con la cual los desarrolladores solo necesitan insertar el código de su aplicación específica.
- ✓ **AR.Drone 2.0 Control Engine library**, que provee una interfaz de control intuitivo desarrollado por Parrot para un control remoto del Ar.Drone2.0 desde un dispositivo iOS.

El SDK comprende las siguientes carpetas:

- ✓ **ARDroneLib**: contiene el API para configurar el AR.Drone.
- ✓ **Control Engine**: contiene ficheros específicos para iOS.
- ✓ **Examples**: contiene ejemplos compilados realizados con ARDroneLib.

La carpeta **ARDroneLib** es la más importante y está estructurada de la siguiente forma:

- ✓ ARDroneLib.xcodeproj: proyecto de xcode para poder desarrollar aplicaciones en el entorno de desarrollo integrado de Apple para iOS.
- ✓ FFMPEG: librería encargada de procesar el vídeo del ARDrone.
- ✓ ITTIAM: librería preconstruida (ARM v7 + Neon) y altamente optimizada de decodificación de vídeo para aplicaciones iOS y Android.
- ✓ Soft: contiene el código específico del drone.
- ✓ VLIB: librería de procesamiento de vídeo del AR.Drone 1.0 que contiene las funciones para recibir y decodificar el vídeo.
- ✓ VP_SDK: librerías de propósito general.

Mediante la librería **ARDroneLIB** es posible tener acceso a funciones básicas de control, realizando movimientos predefinidos de manera automática. Ejemplos de éstos son: despegue, aterrizaje, giro en alguna dirección, subir, bajar, entre otras. Además, es posible configurar ciertos límites de velocidad, altitud o incluso el ángulo de inclinación. Con esta librería es posible tener acceso a cada uno de los sensores o variables de control, utilizando las funciones predefinidas por el fabricante.

Tabla 2.1: Puertos de comunicación del sistema

Puerto	Acción	Descripción
5554	Navdata	Permite tener acceso al estado del sistema por medio de los sensores.
5555	Video Stream	Obtiene información de las cámaras, imágenes codificadas.
5556	AT Commands	Con este puerto se accede a la inicialización de comunicación, limites máximos y control de los movimientos.
5559	Datos críticos	Se utiliza para enviar información de control crítico.

Para tener una comunicación entre el quadrotor y el dispositivo de control es necesario el uso de los puertos que establece el fabricante, los cuales se muestran el la Tabla 2.1.

Después de revisar los conceptos básicos relacionados con la plataforma a utilizar en este trabajo de tesis, en el siguiente capítulo se explican los conceptos básicos de la lógica difusa y su importancia como un método de toma de decisiones.

Capítulo 3

Lógica Difusa

En los últimos años, el número y la variedad de aplicaciones de la lógica difusa se han incrementado significativamente. Las aplicaciones van desde productos como cámaras fotográficas, cámaras de vídeo, hornos, hasta en el control de procesos industriales y en sistemas de soporte de toma de decisiones. Para comprender por qué este método es utilizado hay que analizar primero lo que se entiende por lógica difusa, lo cual es explicado en este capítulo.

La lógica clásica es un conjunto de cálculos lógicos debidos a la necesidad del ser humano de clasificar objetos y conceptos, esta lógica es también conocida como lógica bivalente; esto es, solo operan con dos valores de verdad: verdadero y falso.

La lógica difusa pertenece al grupo de las llamadas lógicas multivaluadas. Estas son aquellas en las cuales hay un mayor número de *valores de verdad*. Este tipo de lógicas permiten agregar más valores aparte de falso y verdadero [67] (figura 3.1). A los *valores de verdad* se les considera grados de veracidad o falsedad. Estos *valores de verdad* son considerados no deterministas ya que esta lógica pretende producir resultados exactos a partir de datos imprecisos, donde los *valores de verdad* tienen cierta incertidumbre.

Figura 3.1: Valores de Verdad Difusos

La lógica difusa se dio a conocer por primera vez en el artículo ''*Fuzzy Sets*''publicado por Lofti Zadeh [69]. La lógica difusa es considerada como la teoría de conjuntos difusos (Fuzzy Sets), esta teoría se refiere a objetos con límites poco definidos en los que la pertenencia es una cuestión de grado.

Un conjunto es una colección definida de elementos en la cual se pueden identificar sus in-

dividuos dependiendo de su pertenencia. Un conjunto difuso es una función cuyo dominio es el universo y contradominio es el intervalo [0,1]. Considerando lo anterior, un conjunto difuso tiene un grado de pertenencia entre 0 y 1. Este tipo de conjuntos no tienen limites claros y tienen asociados un valor lingüístico con una función de pertenencia con valor entre 0 y 1 con una transición gradual. Por ejemplo: al considerar la temperatura de una taza de café se establecen los conjuntos que serían frío (Temp <18 grados) y caliente (Temp >30 grados). En el momento que la temperatura se encuentra intermedia no pertenecería a ninguno de los dos conjuntos. Considerando conjuntos difusos, se establecerían los siguientes conjuntos: el frío (Temp <22 grados), el tibio (18 grados <Temp <30 grados) y el caliente (Temp >26 grados); el grado de pertenencia de uno u otro conjunto seria mediante una transición gradual como lo mostrado en la figura 3.2.

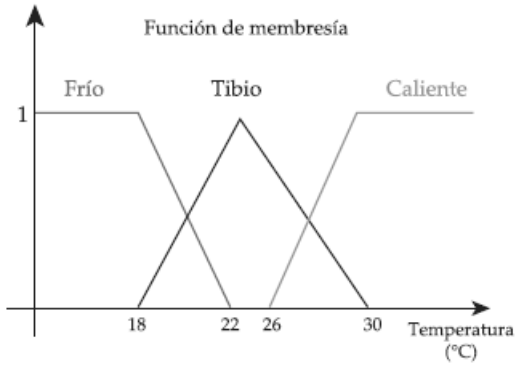


Figura 3.2: Grados de Pertenencia para la temperatura del café

Una variable lingüística es una variable cuyos valores se representan mediante términos lingüísticos o etiquetas (palabras) en lugar de números. En efecto, la lógica difusa puede ser mirada como una metodología para calcular con palabras en lugar de números. Aunque las palabras son inherentemente menos precisas que los números, su uso está más cerca de la intuición humana. Por otra parte, la computación con palabras explota la tolerancia en la imprecisión y por lo tanto reduce el costo de la solución.

El significado de estas etiquetas se determina mediante conjuntos difusos. A dichas etiquetas se les conoce como **Funciones de membresía** y representan un rango de valores de pertenencia dentro del rango total de cada una de las variables de entrada o de salida. Estos valores de pertenencia pueden ser variables dependiendo de las necesidades.

Considerando lo anterior, los sistemas difusos proveen una representación natural y lingüística del conocimiento y razonamiento humano (experto) de reglas vagas que describen de manera imprecisa y cuantitativa las relaciones que existen entre las entradas y salidas [70]. Es por eso que en la mayoría de las aplicaciones de la lógica difusa, una solución lógica difusa es, en realidad, una traducción de una solución humana.

El razonamiento humano utiliza valores de verdad que no necesariamente son tan deterministas. Por ejemplo, al calificar que tan rápido se mueve un automóvil, que tan alta es una persona o incluso que tan lejos esta Huajuapan de Oaxaca. Al considerar estos ejemplos no es

necesario cuantificar las variables con toda precisión. La lógica difusa procura crear aproximaciones matemáticas en la resolución de estos tipos de problemas. Razón por la cual se pretende producir resultados exactos a partir de datos imprecisos.

Al referirnos a valores de verdad difusos, se consideran como valores no-deterministas ya que tienen una cierta incertidumbre. Por ejemplo, cuando hablamos la temperatura del ambiente, esta puede ser fría o caliente; dependiendo de la ubicación geográfica y/o de la persona, la temperatura en la cual se considera que hace frío, es diferente; incluso para una temperatura en la cual no hace ni frío ni calor, se tiene una incertidumbre o difusidad, lo cual es una propiedad de indeterminismo.

Por lo tanto, se puede definir un sistema difuso como aquel que tiene enunciados de entrada y enunciados de salida. El objetivo de este sistema es describir los grados de veracidad de los enunciados de salida en función de los grados de las entradas. Para entender el funcionamiento de un sistema difuso a continuación se describen cada una de sus partes (figura 3.3).

La Fusificación es el término que describe el proceso que se lleva acabo para la transformación de los valores reales a etiquetas (funciones de membresía), con esto asigna valores de pertenencia a cada una de las variables de entrada con relación a los conjuntos difusos previamente definidos y así sean entendidos por el sistema de inferencia.

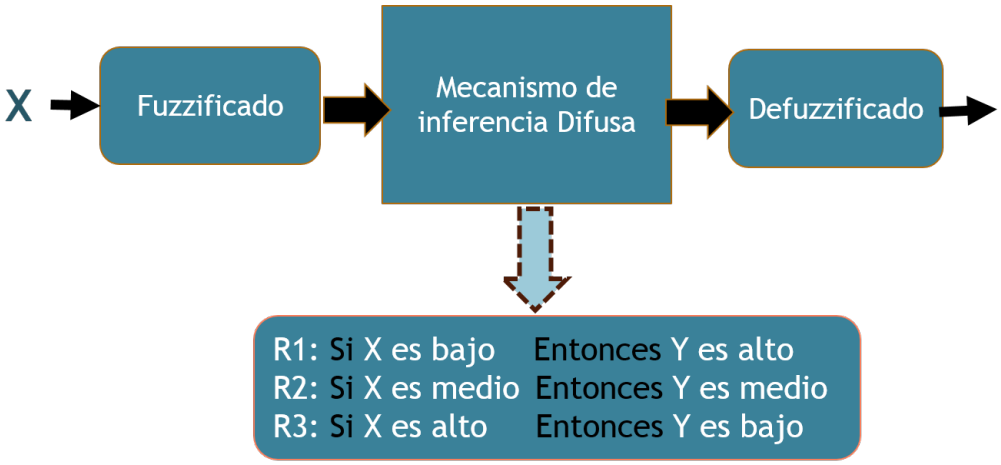


Figura 3.3: Sistema Difuso

Por lo cual, se puede ver la **inferencia difusa** como un método que interpreta los valores en el vector de entrada, con base en reglas del sistema generadas previamente a partir de condiciones y con esto asigna valores al vector de salida. Este proceso se puede observar en la figura 3.3.

Existen diversos métodos de inferencia difusa dentro de los cuales los desarrollados por Mamdani y Sugeno son los más utilizados.

El método de inferencia difusa Mamdani es la metodología que se observa con mayor frecuencia en la literatura. Este método fue uno de los primeros sistemas de control construidos

utilizando la teoría de conjuntos difusos, elaborado para la regulación de un motor de vapor por Ebrahim Mamdani en 1974 [71] sobre la lógica difusa desarrollada por Zadeh. La arquitectura Mamdani consiste en una serie de reglas *ii-Entonces* de la forma:

Si X es **Alto** entonces Y es **Bajo**

El método Takagi-Sugeno se desarrolló como alternativa al método Mamdani, elaborado por Takagi y Sugeno en 1985 [72]. La arquitectura consiste en una serie de reglas Si-Entonces de la forma:

$$\text{Si X(t) es } \mathbf{A} \text{ entonces Y es } Z = k_{i0} + k_{i1}X(t)$$

A partir de los valores de verdad se puede realizar un algoritmo en el cual se especifican las reglas de inferencia. Usando estas reglas, el sistema es capaz de tomar una decisión sobre la acción a realizar dependiendo de las entradas. Este conjunto de reglas se describen mediante el razonamiento deductivo de una persona para controlar el proceso. Es decir, una persona que entienda el funcionamiento del sistema y a partir de estas reglas generar acciones de control.

La Defusificación es el proceso inverso a la fusificación, donde adecua los valores difusos generados en la inferencia en valores reales. Para este proceso de defusificación se utilizan métodos matemáticos simples como: el centroide, el promedio ponderado o la membresía del medio máximo.

3.1. Ventajas y Desventajas

Antes de tomar la decisión de utilizar la lógica difusa se analiza el porqué utilizarla, sus ventajas y desventajas.

3.1.1. Ventajas

- ✓ Para empezar, la lógica difusa es fácil de entender ya que es una aproximación intuitiva del funcionamiento. Ésta se puede construir con base en la experiencia de expertos o personas que entiendan el funcionamiento de un sistema.

- ✓ Permite modificar cualquier sistema dado sin necesidad de empezar de cero, es un método flexible y además permite trabajar con datos imprecisos como se mencionó anteriormente.

- ✓ Es una forma rápida de resolver un problema y no necesita conocer el modelo matemático que rige su funcionamiento.

- ✓ Es posible controlar sistemas no lineales de cualquier tipo de complejidad, esto combinando cualquier conjunto de entradas y salidas. La base de la lógica difusa es la base para la comunicación humana. Esto último debido a que se basa en el lenguaje habitual de la vida diaria de los humanos.

3.1.2. Desventajas

- ✓ Ante un problema en el cual se tiene su solución exacta mediante el modelo matemático la lógica difusa puede no ser tan adecuada.
- ✓ Existe una dependencia de la experiencia del experto en el área, para lograr un funcionamiento correcto.

Después de hacer una revisión general de los elementos que constituyen un sistema difuso, este se puede ver como un sistema de razonamiento difuso o un sistema de toma de decisiones. Esto se observa a manera detallada en el ejemplo de la siguiente sección.

3.2. Control de un móvil con dirección diferencial

El problema que se quiere resolver es el de un móvil con dirección diferencial de 3 ruedas, el cual debe transitar dentro de un laberinto. En este móvil, la rueda delantera es una rueda de apoyo y las ruedas traseras funcionan por separado, cada una con su propio motor (ver figura 3.4).

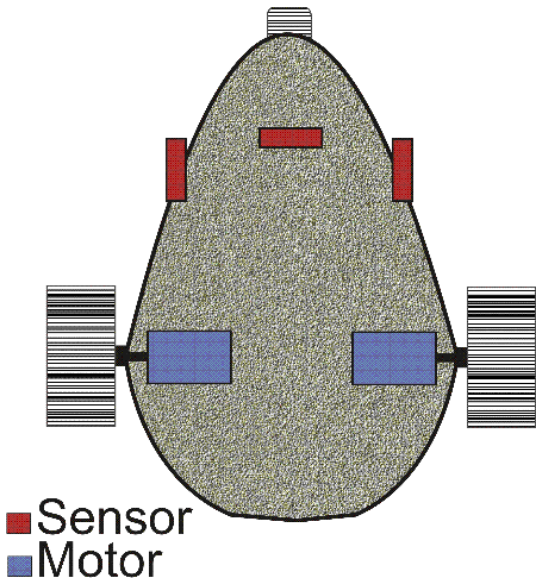


Figura 3.4: Móvil con dirección diferencial

El funcionamiento básico del móvil es el siguiente:

- ✓ Para que el Automóvil avance en linea recta es necesario que ambas ruedas traseras (*Rder* y *Rizq*) giren a la misma velocidad.
- ✓ Para que se realice un giro a la derecha es necesario que Rizq gire a mayor velocidad que Rder.
- ✓ Para que se realice un giro a la izquierda es necesario que Rder gire a mayor velocidad que Rizq.

- ✓ Es posible detener una o las dos ruedas de ser necesario (no hay reversa).
- ✓ Se tienen 3 Sensores Ultrasónicos para la detección de las paredes (Frontal=*SFront*, Lateral Izquierdo=*SIzq* y Lateral Derecho=*SDer*).
- ✓ El móvil no puede ir en Reversa.

Para el problema planteado se consideran las siguientes especificaciones:

- ✓ Las dimensiones del auto son: ancho de 1 metro y largo de 1 metro.
- ✓ El laberinto en los espacios más angostos tiene una distancia de pared a pared 3 metros.
- ✓ Las velocidades de las ruedas van desde 0 hasta 15 Km/h.
- ✓ Los sensores tienen un rango de medición de 0 a 2 metros.

3.2.1. Funcionamiento del Sistema

Considerando que se tiene una persona (conductor) con conocimientos previos del funcionamiento del móvil, se supone una reacción para cada posible situación que se pueda encontrar dentro del laberinto (figura 3.5).

- ✓ **Caso 1**: El móvil se encuentra en un lugar en donde hay paredes cerca en las 3 direcciones.
  - *Reacción*: Se tendrá que dar una vuelta en **U** y lo hace deteniendo una rueda y girando la otra a una velocidad lenta.
- ✓ **Caso 2**: La pared izquierda esta cerca pero adelante y a la derecha no hay nada cerca.
  - *Reacción*: Se acciona la rueda *Rizq* a mayor velocidad que la rueda *Rder*.
- ✓ **Caso 3**: La pared derecha esta cerca pero adelante y a la izquierda no hay nada cerca.
  - *Reacción*: Se acciona la rueda *Rder* a mayor velocidad que la rueda *Rizq*.
- ✓ **Caso 4**: La pared derecha y la pared izquierda están cerca, pero adelante no hay obstáculo.
  - *Reacción*: Se accionan ambas ruedas a la misma velocidad.
- ✓ **Caso 5**: Adelante hay un obstáculo, pero en la derecha o izquierda está libre.
  - *Reacción*: Se acciona una de las dos ruedas y la otra se mantiene sin accionar.
- ✓ **Caso 6**: Adelante y a la derecha está la pared, pero a la izquierda está libre.
  - *Reacción*: Se acciona la rueda *Rder* pero la rueda *Rizq* se mantiene sin accionar.
- ✓ **Caso 7**: Adelante y a la izquierda está la pared, pero a la derecha está libre.
  - *Reacción*: Se acciona la rueda *Rizq* pero la rueda *Rder* se mantiene sin accionar.

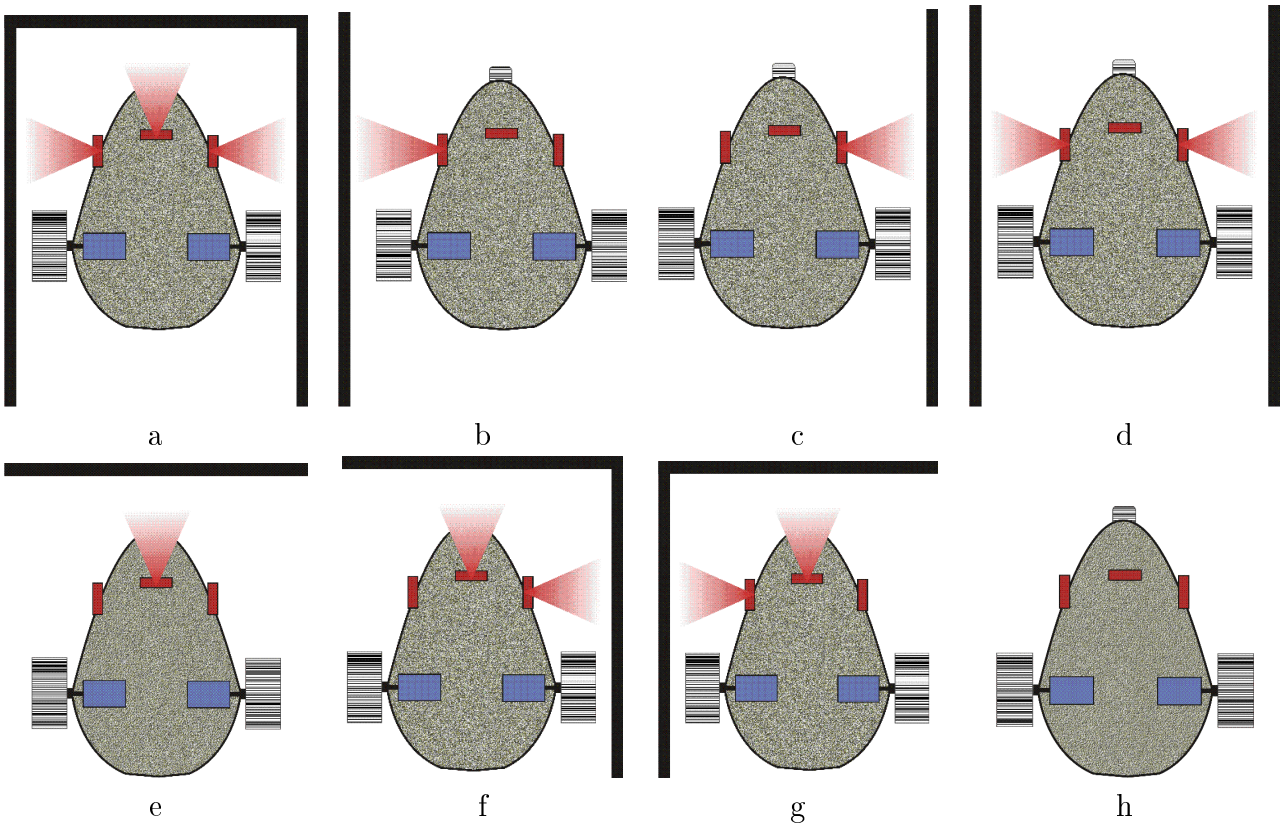


Figura 3.5: Situaciones del móvil en el laberinto: *(a) Caso 1*; *(b) Caso 2*; *(c) Caso 3*; *(d) Caso 4*; *(e) Caso 5*; *(f) Caso 6*; *(g) Caso 7*; *(h) Caso 8*;

- ✓ **Caso 8**: El móvil se encuentra en un lugar donde no tiene paredes demasiado cerca ni adelante, ni a la izquierda, ni a la derecha.
  - *Reacción*: Se accionan ambas ruedas a la misma velocidad y ya encontró la salida del laberinto.

Solución mediante Lógica Difusa

Para la solución de esta problemática, se hace uso de un controlador Difuso con el método de inferencia Mamdani, desarrollado utilizando el Toolbox de lógica difusa de Matlab, y se establece lo siguiente:

Las reglas de control se realizan del tipo *Si-Entonces* y se tienen 3 variables de entrada y 2 variables de salida (figura 3.6).

Las variables de entrada (sensores), tienen 3 funciones de membresía triangulares (Cerca, Medio y Lejos). Las variables de Salida (motores), tienen 4 funciones de membresía triangulares (Cero, Lento, Normal, Rápido). En la figura 3.7 se puede observar un ejemplo de una variable de entrada (3.7a) y un ejemplo de una de las salidas (3.7b).

Con base en los conocimientos del experto y considerando las posibles reacciones para cada

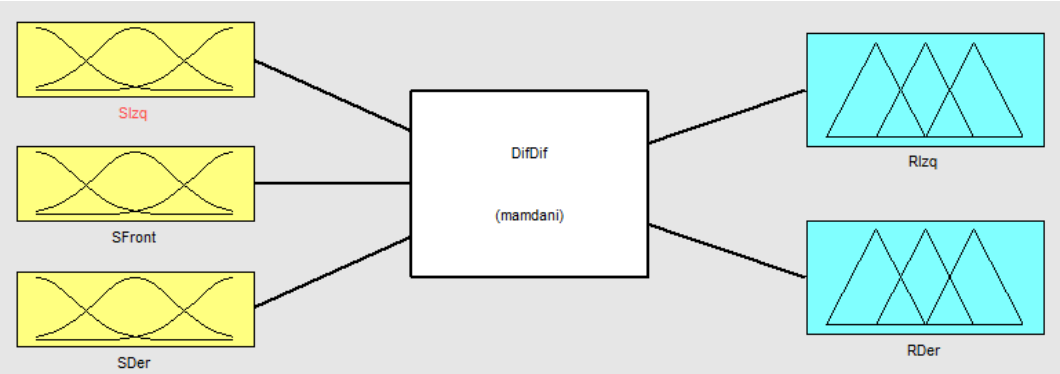
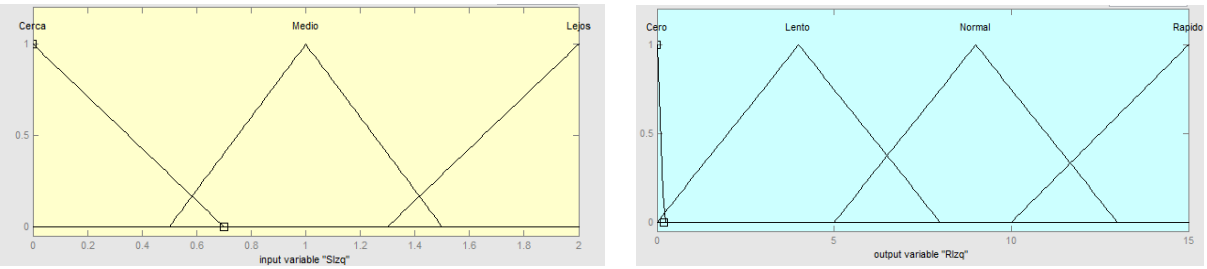


Figura 3.6: Controlador Difuso del Automóvil Diferencial



a b

Figura 3.7: Variables Difusas: *(a) Entrada Sensor Izquierdo (SIzq)*; *(b) Salida Motor Izquierdo (RIzq)*

caso de los mencionados anteriormente, se establecen las reglas de inferencia difusas. Para este ejemplo se tienen 8 casos principales para los cuales se establecen 27 reglas. El número de reglas está en base a las funciones de membresía de las variables de entrada.

Para los 8 casos se tienen las siguientes reglas:

- ✓ **Si** (*SIzq* es Cerca) y (*SFront* es Cerca) y (*SDer* es Cerca) **Entonces** (*Rizq* es Cero) y (*Rder* es Lento)
- ✓ **Si** (*SIzq* es Cerca) y (*SFront* es Lejos) y (*SDer* es Lejos) **Entonces** (*Rizq* es Rápido) y (*Rder* es Normal)
- ✓ **Si** (*SIzq* es Lejos) y (*SFront* es Lejos) y (*SDer* es Cerca) **Entonces** (*Rizq* es Normal) y (*Rder* es Rápido)
- ✓ **Si** (*SIzq* es Cerca) y (*SFront* es Lejos) y (*SDer* es Cerca) **Entonces** (*Rizq* es Rápido) y (*Rder* es Rápido)
- ✓ **Si** (*SIzq* es Lejos) y (*SFront* es Cerca) y (*SDer* es Lejos) **Entonces** (*Rizq* es Lento) y (*Rder* es Lento)
- ✓ **Si** (*SIzq* es Lejos) y (*SFront* es Cerca) y (*SDer* es Cerca) **Entonces** (*Rizq* es Lento) y (*Rder* es Normal)

Tabla 3.1: Entradas y salidas del Sistema Difuso

#Caso/Valor	SIzq(m)	SFront(m)	SDer(m)	Rizq(Km/h)	Rder(Km/h)
Caso1	0.5	0.5	0.5	0.07	4
Caso2	0.5	2	2	12.9	9
Caso3	2	2	0.5	9	12.9
Caso4	1	2	1	13.4	13.4
Caso5	2	1	2	4	4
Caso6	2	1	1	4	9
Caso7	1	1	2	9	4
Caso8	2	2	2	13.4	13.4

- ✓ **Si** (*SIzq* es Cerca) y (*SFront* es Cerca) y (*SDer* es Lejos) **Entonces** (*Rizq* es Normal) y (*Rder* es Lento)
- ✓ **Si** (*SIzq* es Lejos) y (*SFront* es Lejos) y (*SDer* es Lejos) **Entonces** (*Rizq* es Rápido) y (*Rder* es Rápido)

A partir de las reglas y considerando las características del sistema difuso implementado se obtienen los valores para los casos propuestos, en la Tabla 3.1 se pueden observar los valores de las entradas y salidas del sitema. Dichos valores son obtenidos después del sistema de inferencia y defusificados utilizando el método del centroide.

Para entender el método de defusificación, se tiene la figura 3.8 en la cual se pueden apreciar 5 columnas, las 3 primeras (relleno color amarillo) representan a las variables de entrada y las ultimas 2 (relleno color azul) representan las variables de salida. En esta figura se puede observar que de las 27 reglas posibles, algunas de ellas son activadas mostrando sus respectivas salidas.

Para el cálculo del valor de salida del sistema, se utiliza el método del centroide que se puede apreciar en la parte inferior de la misma imagen, en donde todas las salidas activadas se muestran, pero se utiliza solo el valor del centroide calculado, marcado con una linea roja.

En la parte superior de la figura 3.8 se observan los valores equivalentes a cada una de las variables, tanto de entrada como de salida. Éstas últimas producidas como resultado del método de defusificación.

Por lo anterior, se escogió a la **Lógica Difusa** como método de control para este proyecto. Esto considerando que el control de la plataforma puede prescindir del modelo matemático y que es posible controlar al sistema mediante el conocimiento de su funcionamiento.

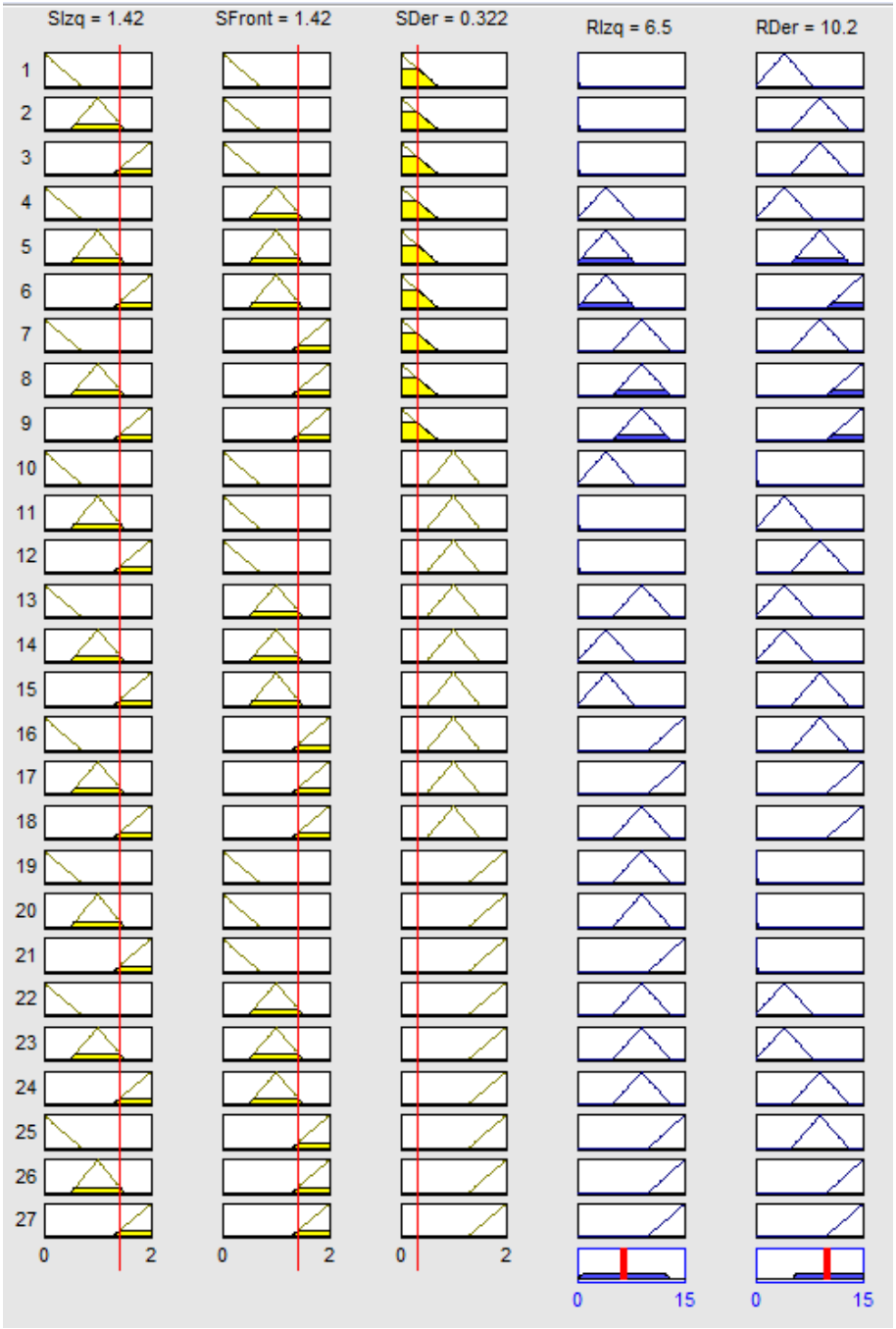


Figura 3.8: Método de Defusificación del Centroide

Capítulo 4

Implementación

En este capítulo se muestra el procedimiento llevado a cabo para la realización de los módulos de trayectorias, comunicación y control, así como también los elementos de software para la implementación en simulación y en la plataforma real. Las simulaciones se han realizado sin tener en cuenta el efecto de las incertidumbres asociadas a efectos de viento, luminosidad o de medición.

4.1. Funcionamiento

En la figura 4.1 se puede apreciar el funcionamiento del sistema, donde se muestra la forma en la cual se ingresa una trayectoria y por medio del control se realizan las correcciones necesarias para obtener la trayectoria deseada.

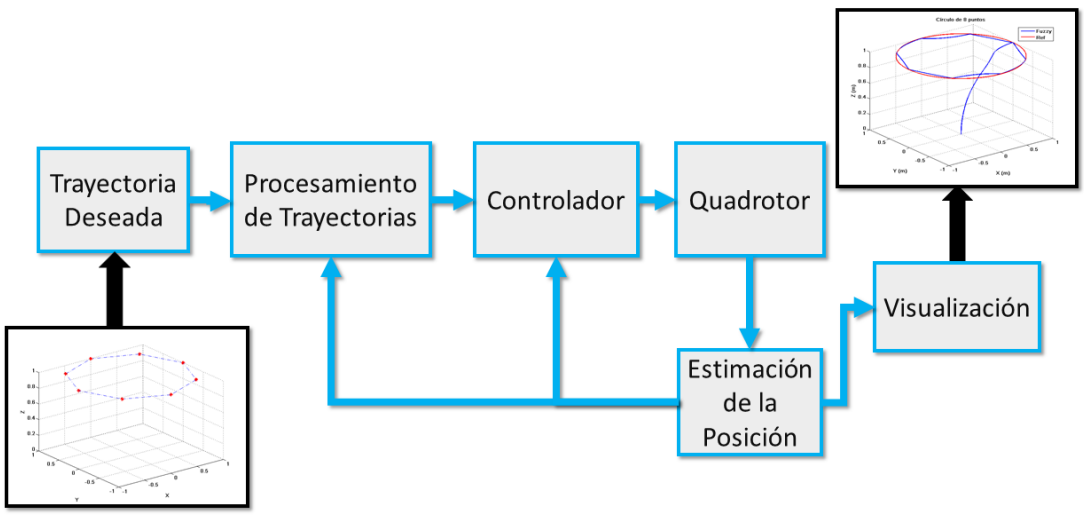


Figura 4.1: Esquema de funcionamiento del sistema

La trayectoria deseada es agregada al sistema mediante una serie de puntos previamente establecidos, cada uno de estos puntos es validado en una etapa de procesamiento de trayectoria para determinar si ya se alcanzó el punto y también se valida un tiempo específico de espera por cada punto. El controlador recibe cada uno de los puntos deseados y envía las señales necesarias para que el quadrotor realice el desplazamiento. La posición del sistema se estima mediante el uso de los sensores para determinar su ubicación y con esto retroalimentar al sistema para saber si es necesario corregir el movimiento o si ya se encuentra en el punto y se procede al siguiente. Para vizualizar el desempeño del sistema es posible emplear los "axes" (gráficas dentro de la interfaz) de la interfaz o en el programa de Simulink mediante los "scopes" (gráficas dentro de Simulink). Esto considerando las gráficas del movimiento realizado como parte de la evolución del sistema en la trayectoria deseada.

4.2. Diseño y desarrollo de los módulos del sistema

El sistema consiste en un software desarrollado en Matlab, que permite el control del quadrotor Ar.Drone2.0 tanto en simulación como en la plataforma real. Para ello se realizó una interfaz gráfica, la cual sirvió como medio de comunicación entre el algoritmo de control implementado y el Ar.Drone mediante una red Wifi. Esta interfaz hace uso de las ventajas de diseño Simulink, ya que proporciona elementos gráficos configurables para la implementación de algoritmos de control. Además, en el desarrollo de este trabajo se hizo uso del ARDrone Simulink Development Kit [57]. Este Toolkit contiene una librería de Simulink y cuenta con un conjunto de 5 bloques. Estos bloques se muestran en la figura 4.2 y se describen en los párrafos siguientes.

De estos bloques, el de Wi-Fi y el Simulation blocks, bloques 1 y 2 respectivamente, son los principales componentes del kit de desarrollo. Estos bloques tienen los mismos datos de entrada y salida, lo cual permite la rápida transición entre los bloques del sistema para la ejecución del algoritmo tanto en la simulación como en el real con un tiempo de muestreo de 0.065s. Además, el kit provee un framework para el control y la guía del vehículo. Por ejemplo, para el control posee un controlador proporcional de posición y de estimación de velocidad.

La función de cada uno de los cinco bloques de esta librería se describe a continuación:

1. **ARDrone Wi-Fi Block**: es la interfaz con el Parrot AR.Drone que permite al usuario enviar comandos al vehículo y al mismo tiempo leer los estados del vehículo. Este bloque fue construido con bloques de *Real-Time Windows Target*. El bloque ARDrone se ejecuta a través del Real-Time Windows Target para proporcionar periodicidad de las tareas de navegación, control y orientación.

2. **ARDrone Simulation Block**: contiene un modelo de la dinámica del vehículo obtenida a través del "System Identification Toolbox" en Matlab. Este bloque permite obtener el estado del sistema a partir del modelo establecido.

3. **Baseline Controller**: es donde se ingresa el controlador deseado para el control del Ar.Drone 2.0. El cual contiene un controlador proporcional que se modificó para tener

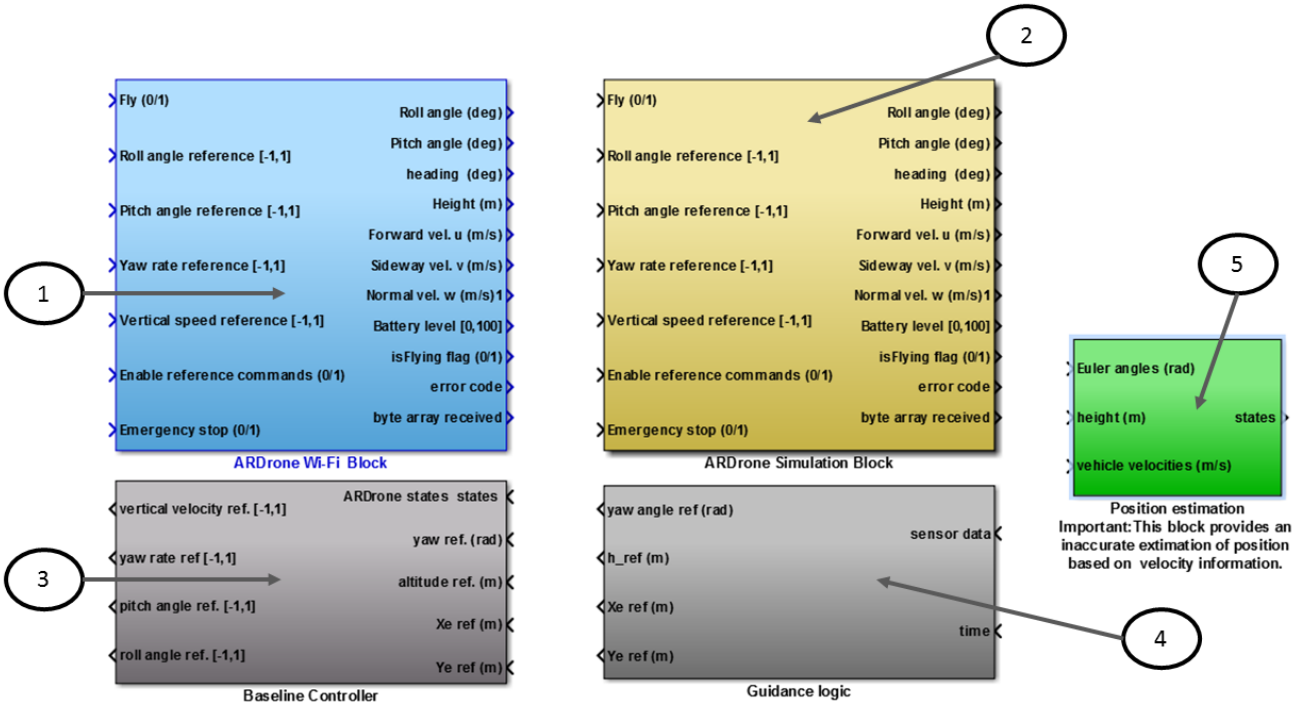


Figura 4.2: Bloques en Simulink de la Librería ARBlocks

un controlador mediante lógica difusa.

4. **Guidance logic**: la referencia deseada se agrega en este bloque, la cual es la trayectoria o los puntos por donde se desea que el sistema pase. En este bloque se toman las decisiones de validar el punto en el que se encuentra el sistema y cuanto tiempo debe mantenerse en ese punto.

5. **Position Estimation**: consiste en estimar el estado en el cual se encuentra el sistema, es decir: la altura, los ángulos de orientación y las velocidades.

Usando como elementos básicos los bloques arriba mencionados, se llevó acabo un desarrollo modular con lo cual se facilita hacer cualquier modificación o adaptación al sistema. Una estructura modular es útil sobretodo para la realización de trabajos futuros en donde se necesitará probar otros algoritmos de control. Permitiendo así un rápido acceso a la estructura de los módulos para su modificación.

Después de haber comentado los bloques funcionales que forman al Toolkit ARDrone se procederá a explicar el diseño y el funcionamiento de los módulos desarrollados para este proyecto.

4.2.1. Módulo de comunicación

El sistema de comunicación provisto por el fabricante permite tener acceso mediante el uso de la red Wi-Fi a las funciones principales del SDK, el módulo de comunicación entre el quadrotor y el dispositivo de control se implementa mediante el acceso a los puertos establecidos

por el mismo. Los puertos asignados para la tarea de comunicación son: 5554 y 5556. Para el uso de estos puertos se emplearon los protocolos UDP (Protocolo de Datagrama de Usuario) de comunicación respectivos: el 15B2h y el 15B4h. La figura 4.3 muestra estos dos bloques en el entorno de Simulink.

El protocolo UDP permite el envío de datagramas (paquetes de datos) a través de la red sin que se haya establecido previamente una conexión, es decir, la transferencia de los datos entre el emisor y el receptor, aunque no estén directamente conectados. Esto es posible ya que el emisor no necesita de una confirmación del receptor por parte del destinatario.

Este protocolo es utilizado en este trabajo debido a que la plataforma de quadrotor elegida, utiliza el Protocolo de Datagrama de Usuario para el envío y recepción de datos en sus puertos.

a b

Figura 4.3: Puertos de comunicación: *(a) Puerto 5554*; *(b) Puerto 5556*

El puerto 5554 permite la lectura del sistema, para obtener información del estado en el que se encuentra el quadrotor. A partir de la lectura se puede conocer la altura, ángulos de orientación y velocidades; así como también el estado de la batería. Mediante el puerto 5556 es posible el envío de los valores de control, así como también comandos de funciones predefinidas como el despegue o el STOP.

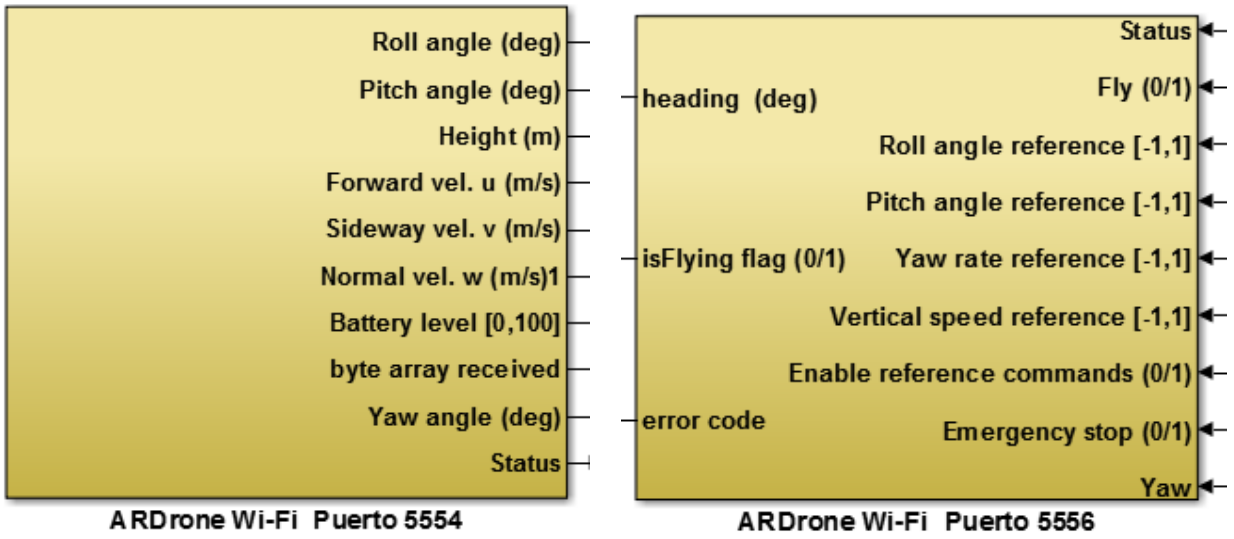
Para la implementación de este módulo, de la plataforma real, fue necesario que el bloque 1 de la figura 4.2 se modificara. Estableciendo un bloque para la entrada de información del estado del sistema y otro para la salida de las variables de control al quadrotor (ver figura 4.4).

Una vez establecido el enlace de comunicación entre el sistema de control y el Ardrone mediante la red Wi-Fi por medio de los puertos, se puede empezar con el envío y recepción de información.

4.2.2. Módulo de Trayectorias

La planificación de trayectorias es básicamente un problema que consiste en la relación que existe entre el tiempo y el espacio. Esto provee una posición deseada correspondiente a cada instante de tiempo.

En el proceso de planificación de trayectorias es necesario realizar un cálculo de toda la trayectoria para desplazarse de un punto inicial a uno final y validar constantemente la posición



a b

Figura 4.4: Bloques del ARDrone Wifi: *(a) Bloque de Simulink de la Entrada al sistema*; *(b) Bloque de Simulink de la Salida al quadrotor*

actual para con esto realizar la corrección necesaria respecto a la posición adecuada de la trayectoria establecida y el tiempo actual.

A diferencia de la planificación de trayectorias se encuentra el seguimiento de trayectorias en donde no es necesario conocer todo el camino por el cual va a recorrer el sistema. En el seguimiento de trayectorias obtiene mas importancia el "*llegar*" de un punto a otro, de manera distinta en la planificación de trayectorias importa más el "*como*" y "*cuando*" llegar de un punto a otro.

Una vez realizada la planificación de la trayectoria o como en el caso de esta tesis que se tienen trayectorias predefinidas y se realiza un seguimiento de trayectorias; se necesita controlar el vehículo para mantenerse en la trayectoria deseada pasando por todos los puntos de referencia. Así, el problema de *Seguimiento de Trayectorias* se concreta en determinar la posición y orientación actual con respecto a la trayectoria que debe seguir y mediante el control realizar las correcciones adecuadas.

Para ello se diseñó un bloque de Simulink para el seguimiento de las trayectorias. El bloque 4, Guidance Logic, tiene como entradas el estado del sistema y las trayectorias de referencia deseadas. En este bloque se realiza una comparación de ambas entradas y se determina el error que será enviado al sistema de control para la toma de decisión.

La trayectoria deseada de referencia es creada mediante un conjunto de puntos por los que el sistema debe pasar. Por lo cual, en este bloque se encarga de enviar punto por punto al sistema y validar mediante el error, cuando el sistema se encuentra en un punto de la trayectoria, para con esto cambiar al punto siguiente.

Cuando el error es pequeño se entiende que el sistema se encuentra en uno de los puntos

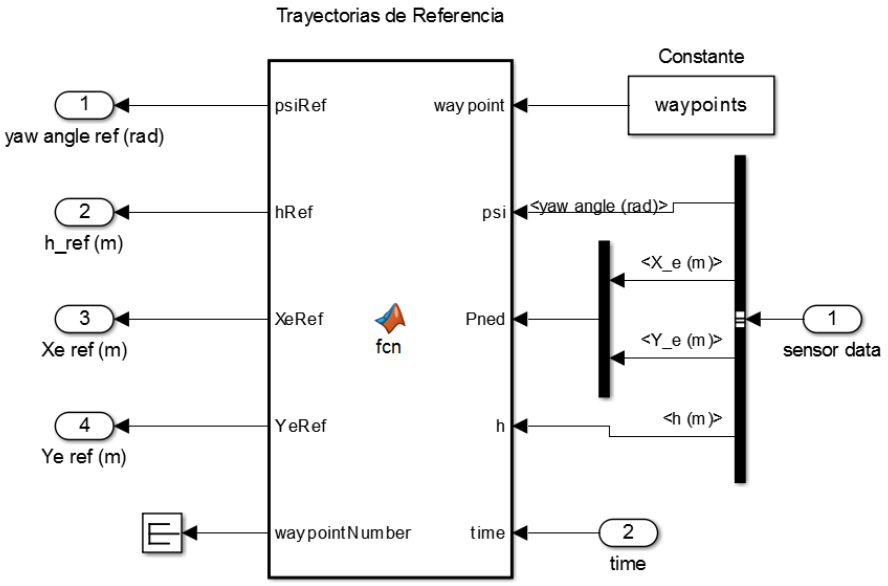


Figura 4.5: Bloques de Trayectorias de referencia (waypoints)

deseados, entonces el sistema debe pertenecer en el punto el tiempo asignado para cada punto respectivamente y así se valida cuando termina el tiempo de un punto, se asigne el siguiente punto de la trayectoria de referencia.

Para que las trayectorias sean ingresadas al sistema, se agregan mediante un bloque constante llamado *waypoints*, el cual se encuentra dentro del bloque "*Referencia*" como se puede apreciar en la figura 4.5.

Para añadir una trayectoria que el usuario desee, será necesario modificar el archivo de Matlab con el nombre de "*getWaypoints.m*" y seleccionar la opción de trayectoria nueva en la Lista de Añadir Trayectorias de la interfaz gráfica. En este archivo están las instrucciones necesarias para ingresar una trayectoria específica o marcar los puntos por los cuales se requiere que el sistema pase. Sin embargo, en el Listado 4.1 se describen las instrucciones del código en Matlab.

Listado 4.1: Código en Matlab para asignar una trayectoria de referencia

```
1  function [waypoint] = getWaypoints( )
2      nPoints = 5;
3      waypointsListARDrone = zeros(5,nPoints);
4      %                          [X(m);Y(m);Z(m);Yaw(deg);T(seg)] ;
5      waypointsListARDrone(:,1) = [ 0  ; 0  ;  1  ;     0      ;  2   ] ;
6      waypointsListARDrone(:,2) = [-1  ; 0  ;  1  ; 45*pi/180  ; 0.1  ] ;
7      waypointsListARDrone(:,3) = [ 1  ; 0  ;  1  ;     0      ; 0.1  ] ;
8      waypointsListARDrone(:,4) = [ 0  ; 0  ;  1  ;     0      ;  2   ] ;
9      waypointsListARDrone(:,5) = [ 0  ; 0  ; 0.5 ;     0      ;  2   ] ;
10     waypoint = waypointsListARDrone ;
11 end
```

En la línea 1 del código se establece el nombre de la función y la variable de salida. Este nombre debe de ser el mismo que el del archivo donde se guarde.

El número de puntos por los cuales se desea que pase la trayectoria se establece en la línea 2 en la variable *nPoints*.

La variable *waypointsListARDrone* es una matriz donde se guardan todos los puntos de la trayectoria y se debe inicializar en ceros con las medidas específicas de 5 x nPoints, esto debido a que son 5 los parámetros que debe tener cada punto y *nPoints* será el número de puntos de la trayectoria.

De la línea 5 a la 9 se agregan cinco diferentes puntos a la trayectoria, estableciendo en cada uno de ellos los parámetros adecuados según la descripción de la línea 4.

Para finalizar el programa es necesario asignar la variable *waypointsListARDrone* a la variable de salida de la función, que en este caso es "*waypoint*".

Si se desea asignar una función o ingresar una trayectoria de alguna otra forma, ya sea iterativa o recursiva, es necesario realizar la asignación de cada punto de la trayectoria con sus 5 parámetros correspondientes. Al final asignar la variable donde estén guardados todos los puntos a la variable de salida correspondiente.

4.2.3. Módulo de Control

Como ya se ha mencionado, el sistema de control desarrollado para esta tesis es un control difuso. Para llevar a cabo este controlador se utilizó el *Fuzzy Logic* toolbox de Matlab/Simulink.

Matlab Fuzzy Logic Toolbox

Este toolbox permite al usuario tener acceso a una aplicación para el diseño de la Lógica Difusa para la construcción de Sistemas de Inferencia Difusos(FIS por sus siglas en Inglés), y la visualización y el análisis de resultados.

Con el uso de los editores del Fuzzy Logic Toolbox (figura 4.6), es posible construir las reglas, definir las funciones de pertenencia y analizar el comportamiento de un sistema de inferencia difuso. Los siguientes editores y visores se proporcionan en el *fuzzy logic* toolbox de Matlab:

- ✓ **Editor FIS**: muestra información general sobre un sistema de inferencia difuso.
- ✓ **Editor de Funciónes de Membresía**: le permite ver y editar las funciones de pertenencia asociadas a las variables de entrada y salida del FIS.
- ✓ **Editor de Reglas**: le permite ver y editar reglas difusas utilizando uno de los tres formatos: sintaxis en Inglés, notación simbólica o notación indexada.

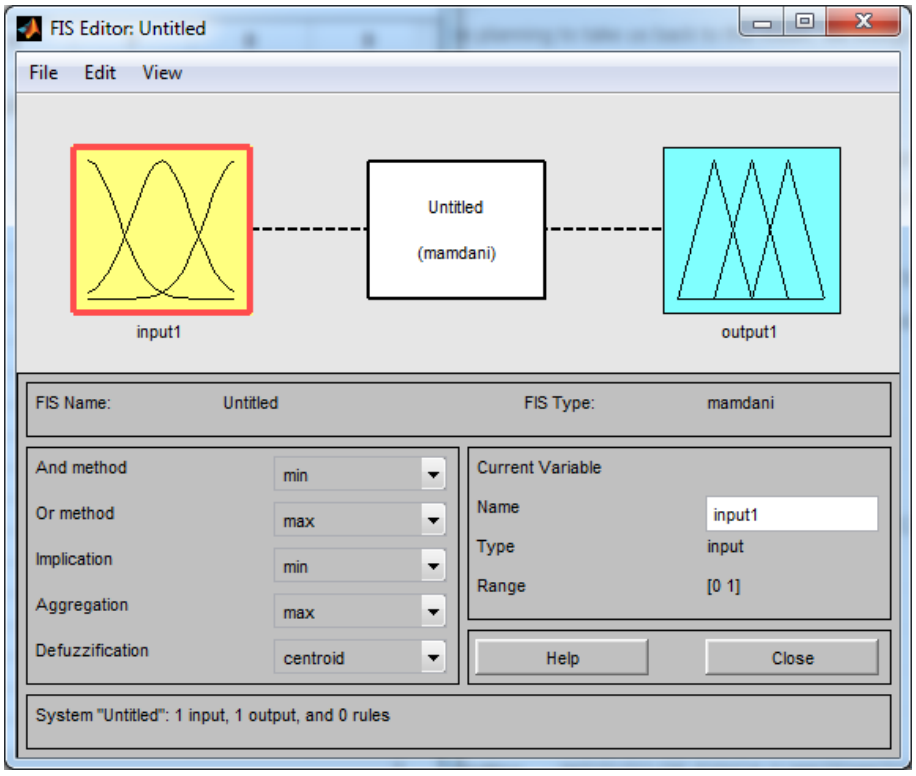


Figura 4.6: Editor del Fuzzy logic Toolbox

- ✓ **Visor de Reglas**: permite ver el comportamiento detallado de una FIS para ayudar a diagnosticar el comportamiento de normas específicas o estudiar el efecto de la evolución de las variables de entrada.

- ✓ **Visor de Superficie**: genera una superficie 3-D a partir de dos variables de entrada y la salida de un FIS.

Para el desarrollo del control, los sistemas de inferencia, que incluye el Toolbox, que se pueden utilizar son el Mamdani y el Sugeno. Considerando estos dos sistemas se decidió trabajar sólo con el método Mamdani ya que es el que se considera el más documentado y aceptado para el uso como método de inferencia para sistemas de control. En cuanto a su implementación, este método de inferencia permite un desempeño aproximado al razonamiento humano [69].

Control del Quadrotor

El control realizado con este toolbox es posible utilizarlo y agregarlo a un modelo de Simulink para controlar un sistema, el cual en este caso es el quadrotor.

El control del quadrotor se realizó por medio de 4 diferentes controladores difusos (figura 4.7), los cuales se ejecutan simultáneamente para el seguimiento de trayectorias. Estos controladores son el de elevación y el de cada uno de los 3 ángulos de orientación del sistema (*roll, pitch y yaw*).

En la entrada de los controladores se tienen los valores medidos y los valores de referencia

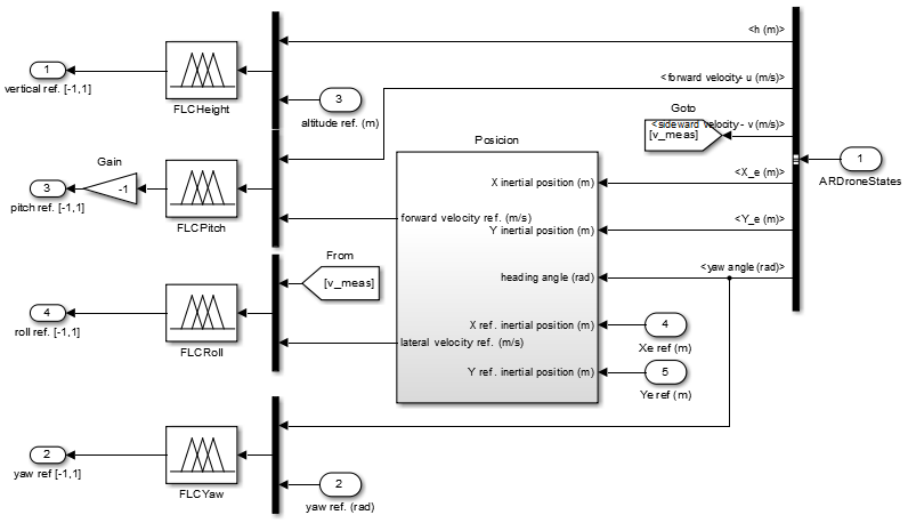


Figura 4.7: Bloque de Simulink del Controlador Difuso

deseados de cada variable, respectivamente. El método de inferencia procesa los valores de entrada después de haber sido fusificados. Estos valores entran en el sistema de inferencia difuso y éste realiza el proceso de control necesario para cada controlador.

Como método defusificador se tiene el de centroide, el cual calcula el valor del centroide promedio de todas las reglas existentes.

A la salida de cada uno de estos controladores se obtiene la tasa de cambio necesaria de cada una de las 4 variables en consideración, para llegar a los puntos establecidos en la trayectoria de referencia.

Con estos controladores se modificará el estado del sistema mediante el uso de la librería **ARDroneLIB**, infiriendo en las funciones de control establecidas en el SDK, logrando así operar al sistema de forma deseada.

Cada uno de los controladores fuzzy utilizados contiene las siguientes especificaciones:

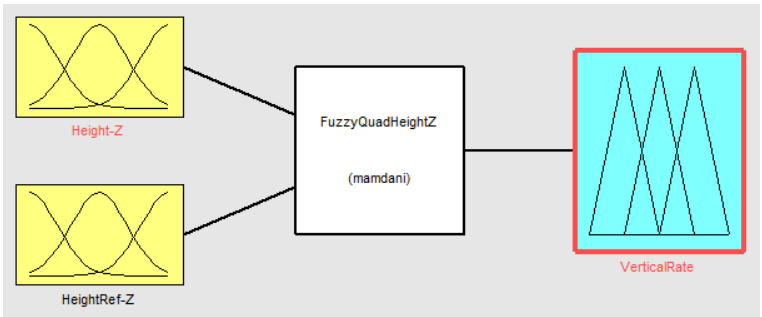


Figura 4.8: FIS del controlador de la Elevación

✓ **Elevación_FLC**: tiene 11 funciones de membresía para cada entrada y 11 funciones para la salida. Rango de entrada [0 3] y rango de salida [-1 1]. Cada función de membresía es de tipo triangular y posee 121 Reglas (figura 4.8).

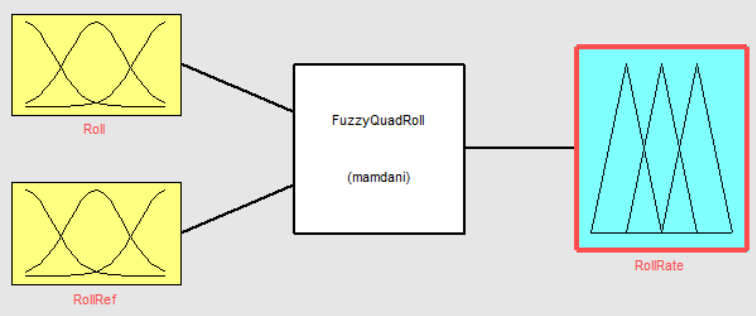


Figura 4.9: FIS del controlador del ángulo Roll

✓ **Roll_FLC**: tiene 11 funciones de membresía para cada entrada y la salida también 11. Rango de entrada [-30 30] y rango de salida [-1 1]. Cada función de membresía es de tipo triangular y posee 121 Reglas (figura 4.9).

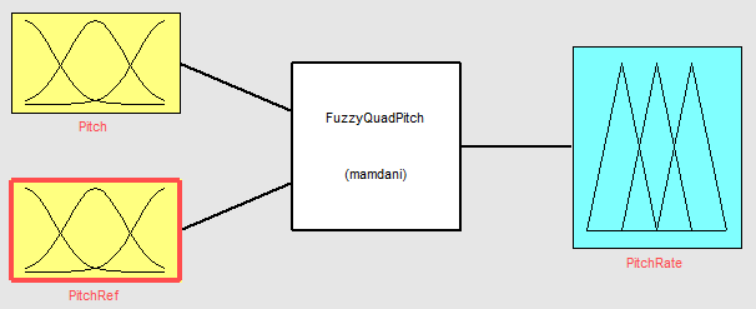


Figura 4.10: FIS del controlador del ángulo Pitch

✓ **Pitch_FLC**: tiene 11 funciones de membresía para cada entrada y la salida también 11. Rango de entrada [-30 30] y rango de salida [-1 1]. Cada función de membresía es de tipo triangular y posee 121 Reglas (figura 4.10).

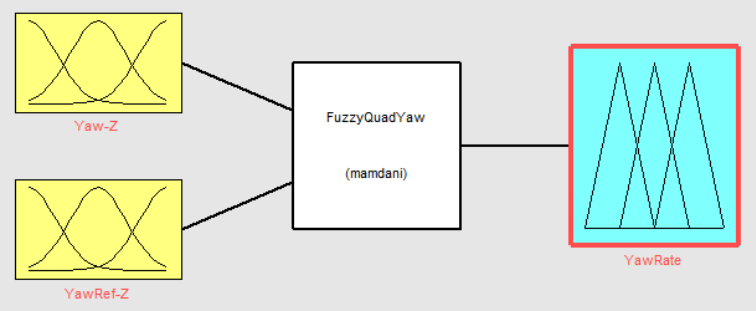


Figura 4.11: FIS del controlador del ángulo Yaw

✓ **Yaw_FLC**: tiene 11 funciones de membresía para cada entrada y la salida también 11. Rango de entrada [0 π] y rango de salida [-1 1]. Cada función de membresía es de tipo triangular y posee 121 Reglas (figura 4.11).

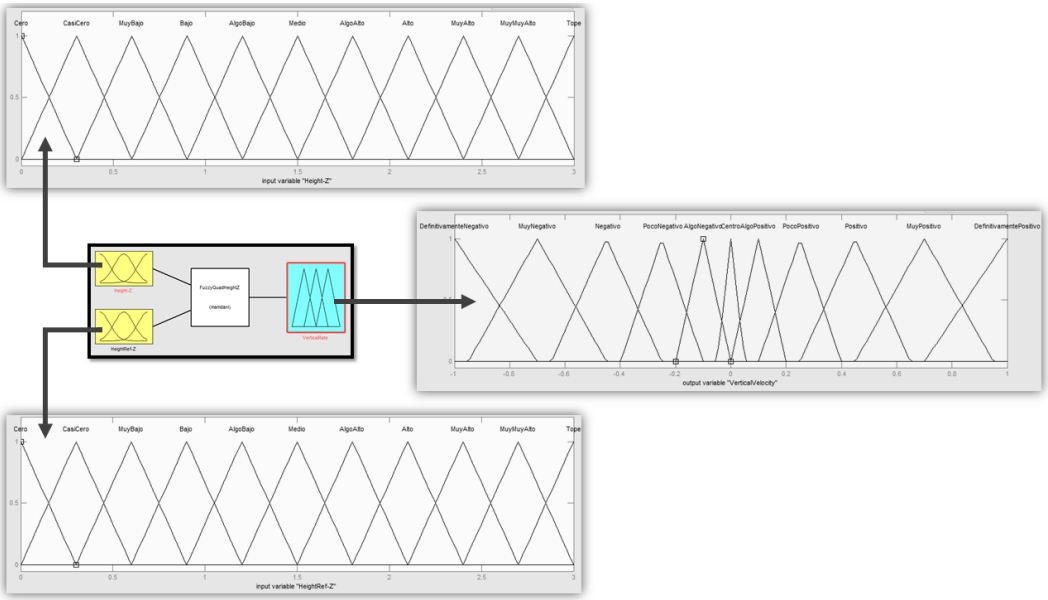


Figura 4.12: Partes del controlador de Elevación

Para entender el funcionamiento de estos controladores se explica de una manera mas detallada el controlador de la Elevación, mostrado en la figura 4.8.

En la figura 4.12 se desglosan las dos entradas y la salida del controlador. De las dos entradas, la superior es la que proviene del estado del sistema, y en la inferior se recibe el valor de referencia deseado.
En la Entrada 1 del estado del sistema *Height-Z* (figura 4.13), se utilizaron 11 Funciones de

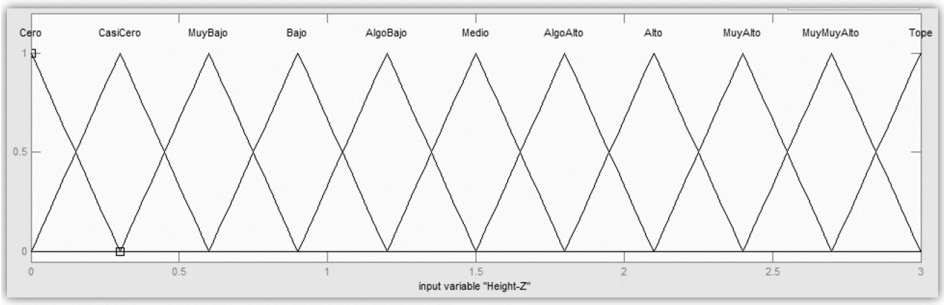


Figura 4.13: Funciones de membresía de la entrada 1 del controlador de Elevación

membresía de tipo Triangular en un rango de 0 a 3m. El rango es establecido asumiendo que se va a usar en interiores y que se podría alcanzar una altura máxima de 3m.
Para la Entrada 2 de la referencia deseada *HeightRef-Z* (figura 4.14), también se hizo uso de 11 Funciones de membresía de tipo Triangular distribuidas uniformemente en un rango de 0 a 3m.

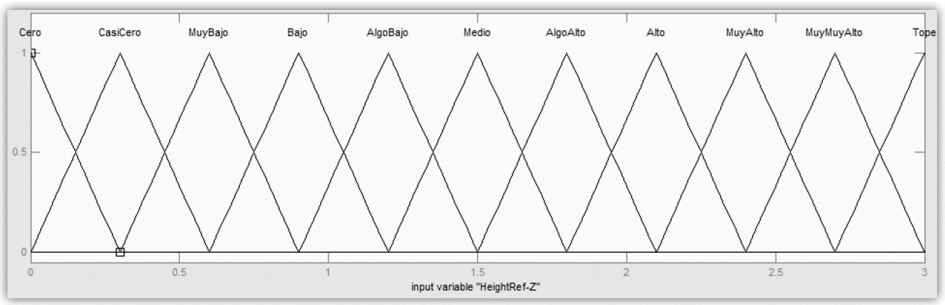


Figura 4.14: Funciones de membresía de la entrada 2 del controlador de Elevación

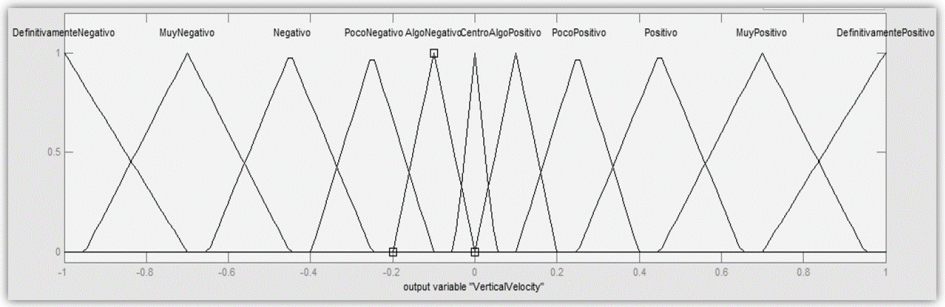


Figura 4.15: Funciones de membresía de la Salida del controlador de Elevación

En la salida del controlador *VerticalVelocity* (figura 4.15), se utilizaron 11 funciones de membresía Triangulares, pero distribuidas 5 a la derecha y 5 a la izquierda del cero de manera creciente, es decir, que con forme se alejaban del cero el rango de cada función es mayor. Esto considerando que para cambios pequeños las variaciones deben ser menos bruscas.

En la elaboración de los cuatro controladores, las 3 variables lingüísticas fueron hechas considerando el funcionamiento de los quadrotores. La distribución y el número de funciones de membresía se realizaron mediante el método heurístico, tomando en cuenta que a mayor número de funciones de membresía entonces una mayor cantidad de reglas son necesarias.

Considerando el controlador de elevación mencionado anteriormente, las reglas de inferencia que se elaboraron para su control fueron hechas en base al funcionamiento de los quadrotores descrito en el Capítulo 2. Para entender el funcionamiento de estas reglas, en la tabla 4.1 se mencionan algunas de ellas.

Tabla 4.1: Entradas y salidas del Sistema Difuso del Quadrotor

#Caso/Valor	Entrada 1(m)	Entrada 2(m)	Salida(m)
Elevarse de 0 a 1m	0	1	0.56
Elevarse de 1m a 2m	1	2	0.56
Mantenerse en 2m	2	2	0
Descender de 2m a 0.5m	2	0.5	-0.79

4.3. Simulación

Para realizar la simulación del quadrotor se implementó el diagrama de bloques de Simulink mostrado en la figura 4.16, en este diagrama se puede observar el bloque de control Fuzzy de color azul obscuro. Este bloque es el más importante, ya que es el que contiene el núcleo del funcionamiento de este trabajo.

Con la finalidad de hacer la simulación lo más parecida al funcionamiento del Drone real, se agregaron tiempos de retardo. El tiempo de muestreo utilizado en simulación es de 0.065s para obtener el estado del sistema, se agrega un retardo de 0.26s considerando el tiempo que tarda la comunicación entre el Drone y la computadora. Para la ejecución de la simulación se utiliza un tiempo de simulación de 0.005s.

Es importante señalar que, se considera al sistema como ideal, de tal manera que para esta implementación se desprecian los efectos del aire, la inercia, y cambios de intensidad luminosa en el ambiente.

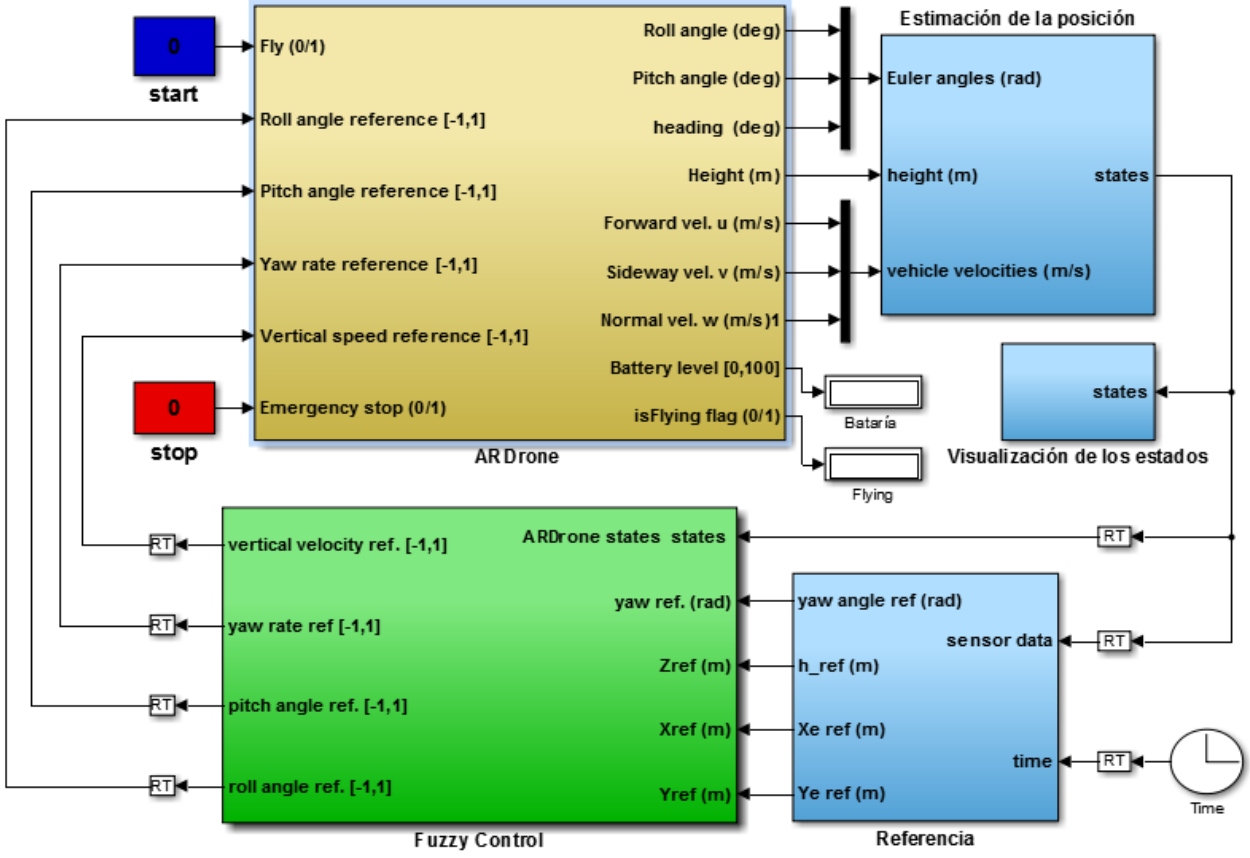


Figura 4.16: Diagrama de bloques del control en simulación

4.4. Plataforma Real

Para la implementación en la plataforma real, se utilizó el diagrama de bloques de Simulink mostrado en la figura 4.17, en el cual se puede observar como bloque importante el de Control Fuzzy y, a diferencia de la implementación en simulación, en este proceso son importantes los bloques de comunicación mencionados en la figura 4.4.

En el caso de la implementación en la plataforma real, se tiene un tiempo de muestreo general de 0.065s.

En el control de la plataforma real, fue necesario considerar los cambios de intensidad luminosa en el ambiente y las características del piso en el área donde vuele el sistema (ver figura 4.18). Esto, es debido a que el quadrotor Ar.Drone dentro de las funciones establecidas en la API para su ubicación, utiliza un algoritmo para detectar, mediante el uso de la cámara inferior, si se desplaza en alguna dirección.

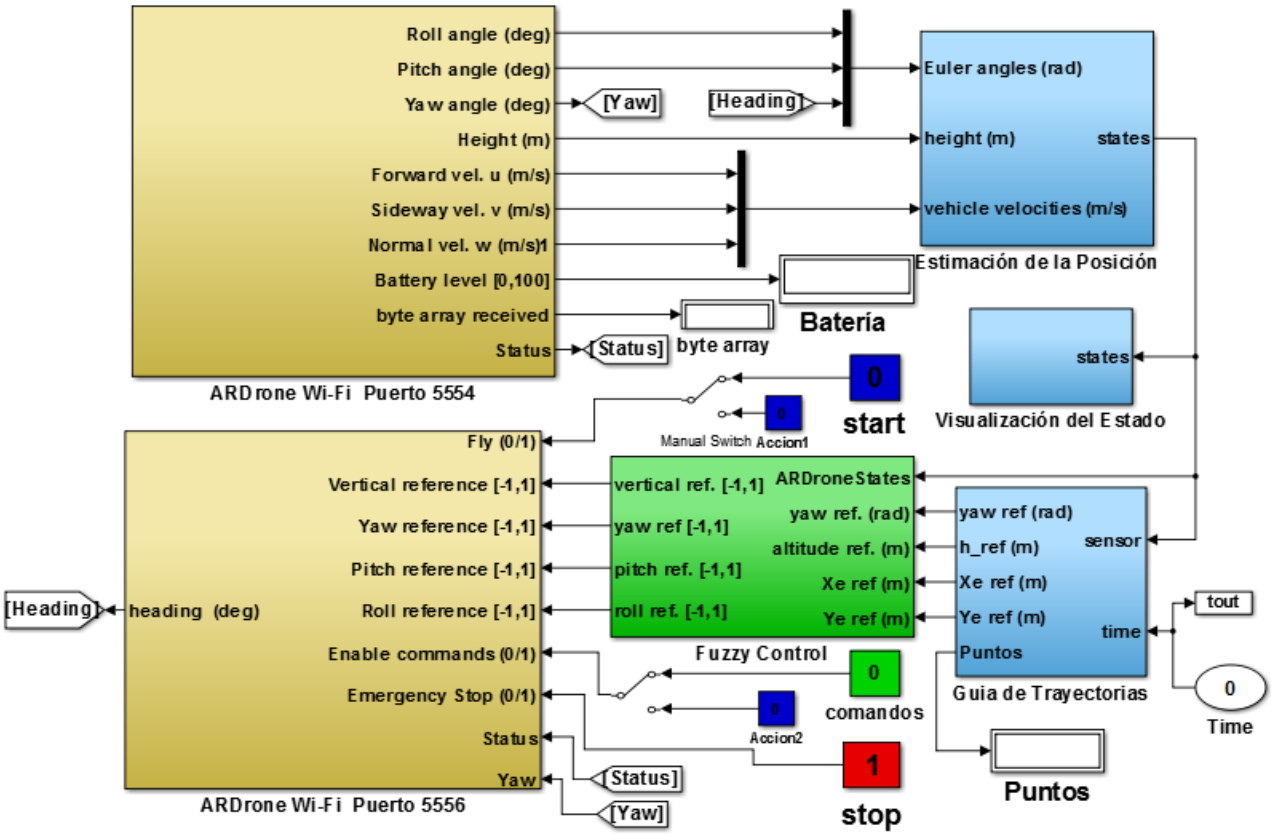


Figura 4.17: Diagrama de bloques del control de la plataforma real

Considerando lo anterior, si la superficie no tiene variaciones o si debido a la intensidad de luz no logra detectar variaciones; el sistema no va a detectar algunos desplazamientos en los

a b

Figura 4.18: Caracteristicas agregadas al piso

cuales las variaciones de los ángulos sean muy pequeñas y que la cámara no detecte variación haciendo que la plataforma continúe desplazándose, y siendo éste un problema de inestabilidad en el sistema.

En esta implementación no se consideraron características como la aerodinámica del sistema, o efectos del aire. Además existen variaciones o pequeñas oscilaciones en los movimientos desempeñados. Éstas son debidas a que el error utilizado en la implementación es de ±8cm. Esto debido a que el sistema pierde noción de donde se encuentra, debido a perturbaciones muy pequeñas como las mencionadas en el párrafo anterior. Produciendo que el espacio de trabajo en el que el sistema considera estar en un punto indicado sea un cubo de 16cm por lado, aproximadamente.

4.5. Interfaz Gráfica

La interfaz gráfica utilizada fue creada mediante un GUI de Matlab y la interfaz final es mostrada en la figura 4.19.
Mediante esta interfaz se realiza el control de todo el proceso de implementación. Para cualquiera de las dos implementaciones posibles, esta interfaz abre el programa de Simulink correspondiente y mediante comandos se hace uso de sus menús o de los bloques necesarios.

Es posible realizar un intercambio de información entre la Interfaz, Simulink, y el Workspace. Esto para que el programa tenga un funcionamiento adecuado. Es necesario considerar que Simulink hace el envío de datos al Workspace una vez terminada su ejecución. Dicho envío de información es necesario por que se tienen los datos disponibles para las gráficas de las diferentes variables y con esto también poder calcular el error entre las trayectorias.

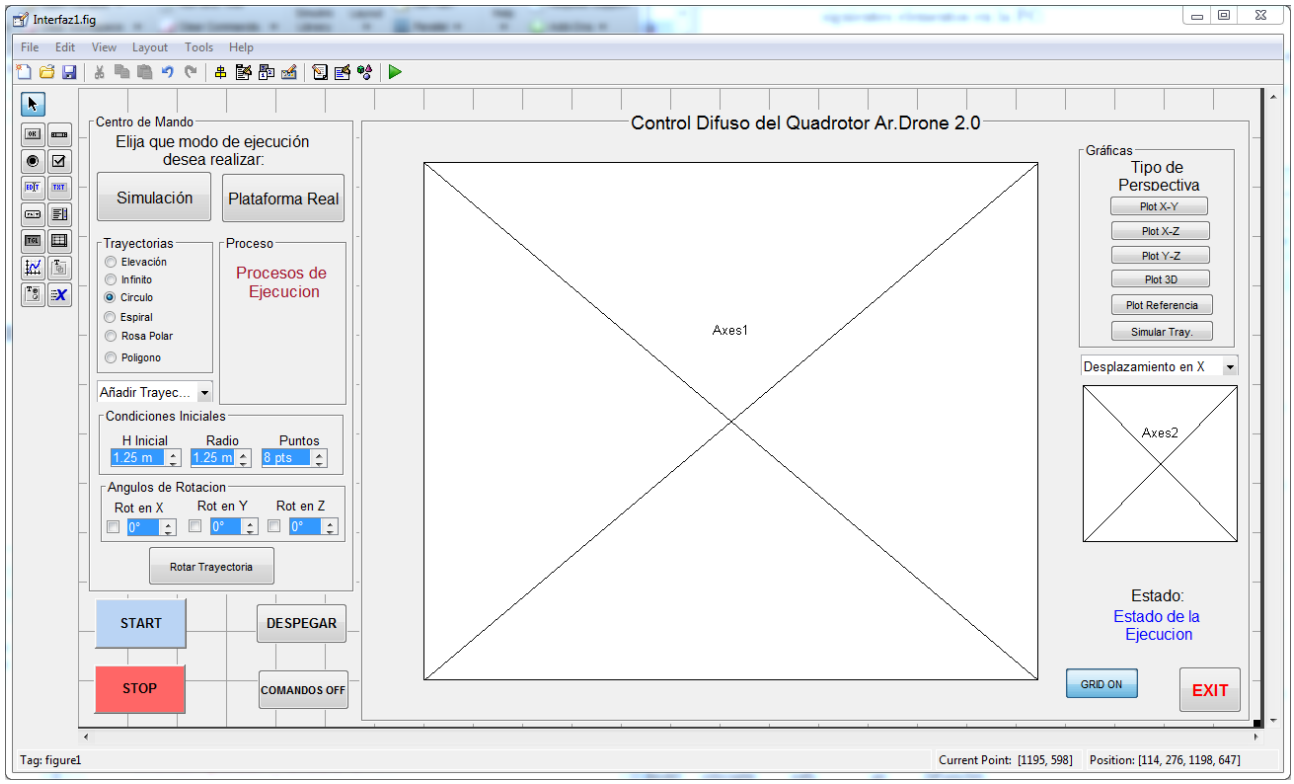


Figura 4.19: Interfaz implementada en el GUI de Matlab

4.5.1. Descripción de la Interfaz

El funcionamiento de la interfaz es descrito a continuación, describiendo cada una de sus partes (ver figura 4.20).

1. Botón de Simulación: cierra todos las ventanas de matlab abiertas, limpia el workspace y la línea de comandos; después de esto abre el programa de Simulink de la simulación.

2. Botón de Plataforma Real: cierra todos las ventanas de matlab abiertas, limpia el workspace y la línea de comandos; después de esto abre el programa de Simulink del control de la plataforma real.

3. Panel de Trayectorias: se encuentran las diferentes trayectorias predefinidas para realizar el seguimiento.

4. Lista Añadir Trayectorias: se encuentran más trayectorias posibles para el seguimiento y mediante la opción Añadir Trayectoria, es posible ingresar una trayectoria nueva a la base de datos.

5. Panel de Condiciones Iniciales: permite ingresar los valores deseados de altura inicial (H Inicial), Radio (o fóco) y el número de Puntos.

6. Panel Proceso: es donde se da al usuario las indicaciones del siguiente paso a realizar para ejecutar el programa de forma correcta.

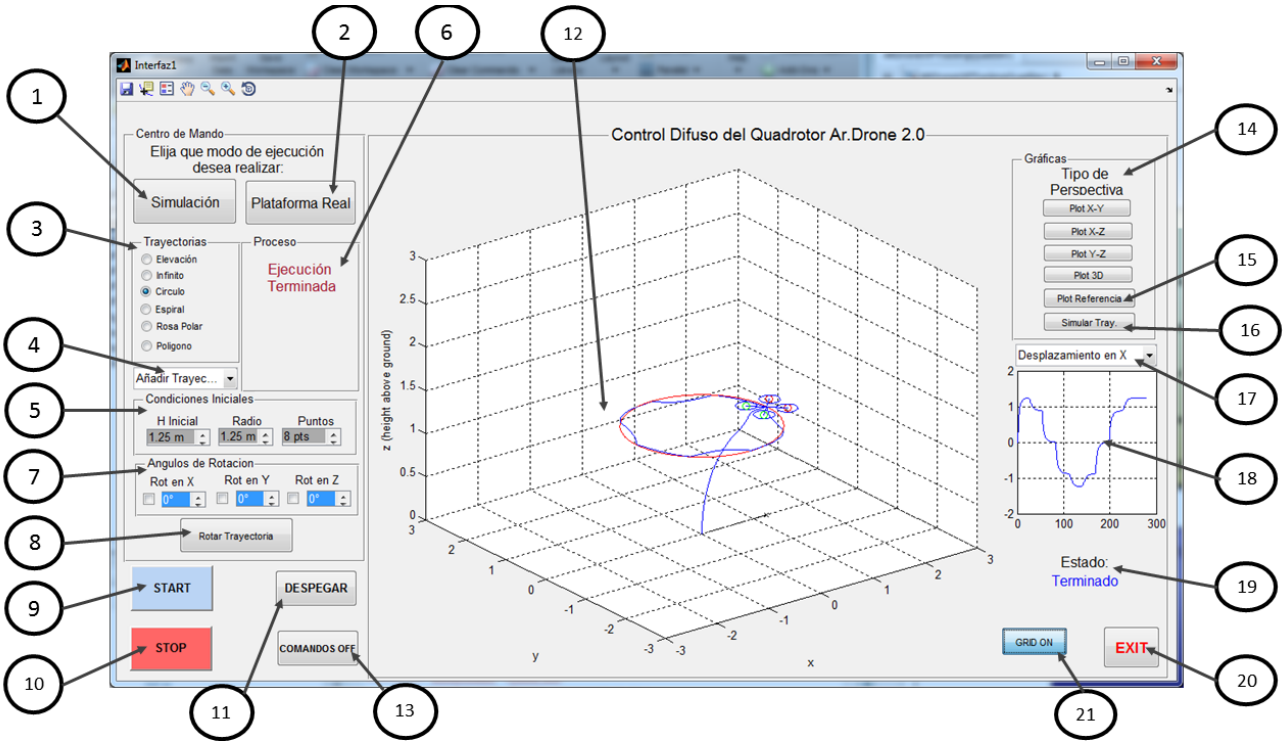


Figura 4.20: Interfaz Gráfica en Matlab

7. Panel Ángulos de Rotación: permite modificar los ángulos roll, pitch y yaw (inicialmente [0,0,0]).

8. Botón Rotar Trayectoria: realiza una rotación de la trayectoria deseada según los valores de los ángulos especificados en el panel de Ángulos de Rotación.

9. Botón START: compila y ejecuta el programa de Simulink seleccionado.

10. Botón STOP: detiene la ejecución del programa de Simulink seleccionado.

11. Botón Despegar: esta opción solo afecta al programa de control de la plataforma real, produciendo el despegue del sistema. Al estar en el aire, y ser oprimido nuevamente produce que la plataforma aterrice.

12. Axes1: es donde se muestran las diferentes gráficas obtenidas al terminar el proceso, dependiendo la escala de los ejes del tipo de gráfica seleccionada en el panel de Gráficas.

13. Botón Commandos ON/OFF: es un switch que activa o desactiva el envío de las señales de control de la PC al quadrotor.

14. Panel Gráficas: muestra una serie de botones que permiten elegir entre diferentes tipos de gráficas posibles para observar los resultados, realizando combinaciones de 2 ejes o mostrar el resultado en 3 dimensiones.

15. Botón Plot Referencia: permite activar o desactivar la gráfica de la trayectoria de referencia, de la cual el sistema esta realizando el seguimiento.

16. Botón Animación: permite observar una animación de un quadrotor siguiendo la trayectoria deseada (basado en el Robotic Toolbox).

17. Lista de gráficas del desempeño: con las cuales se puede observar cada uno de los 6 grados de libertad del sistema, es decir los 3 desplazamientos y los 3 ángulos de movimiento, todos graficados respecto del tiempo.

18. Axes2: es donde se muestran las gráficas obtenidas al terminar el proceso, de los desplazamientos en X, Y, Z o los ángulos Roll, Pitch, Yaw. Cada uno respecto del tiempo.

19. Cuadro de Texto: se muestra el estado en el cual se encuentra el programa, ya sea ingresando valores, compilando, simulando, etc.

20. Botón EXIT: mediante el cual se cierran todas las ventanas abiertas durante el proceso, y se cierra también la interfaz gráfica.

21. Botón GRID ON/OFF: es un switch que permite activar o desactivar el grid de las gráficas.

Capítulo 5

Resultados

En esta sección se presentan algunos de los resultados que se obtuvieron con las diferentes trayectorias que se usaron para validar nuestro enfoque de control difuso. En primer lugar se muestran los resultados de la simulación y después los resultados obtenidos de la plataforma real. En ambos casos, los resultados se generan a partir de trayectorias simples donde se tienen varios movimientos de manera simultánea ya que afecta a más de una de las variables de control.

Se realizó un seguimiento de trayectorias simples como desplazamientos en una o más direcciones, curvas específicas, polígonos regulares, círculos, lemniscatas, espirales, entre otras. Como base para estas trayectorias se realizaron pruebas con movimientos básicos en los 3 ejes cartesianos, y con giros de orientación.

Para cada trayectoria se tiene un script específico, en el cual se establecen los puntos por los cuales el sistema, mediante el control inteligente, es capaz de desplazarse para aproximarse a la trayectoria seleccionada. Considerando que a mayor cantidad de puntos existentes en una misma trayectoria, más parecido será el seguimiento y la curva de resultado será más cercana a la referencia.

Analizando como ejemplo la trayectoria de un círculo, con un mayor número de puntos el resultado del seguimiento se parecerá más al círculo de referencia, mientras que si la cantidad de puntos es pequeña, la trayectoria tenderá a parecerse a un polígono. Por ejemplo si son 6 puntos será parecida a un hexágono o, con 8 puntos, a un octágono y así dependiendo del número de los mismos (figura 5.1).

Además, con la finalidad de obtener un parámetro de verificación y ver el error que presenta la trayectoria seguida, se compararon con la curva de la trayectoria deseada.

Se consideraron dos conjuntos de desplazamiento como trayectorias específicas: movimientos básicos y trayectorias simples.

Los movimientos básicos fueron las pruebas iniciales realizadas del sistema, se realizaron dos pruebas por cada uno, considerando que para todas las trayectorias se estableció como punto inicial elevarse a una altura de 1 metro y a partir de este punto realizar cualquiera de los siguientes

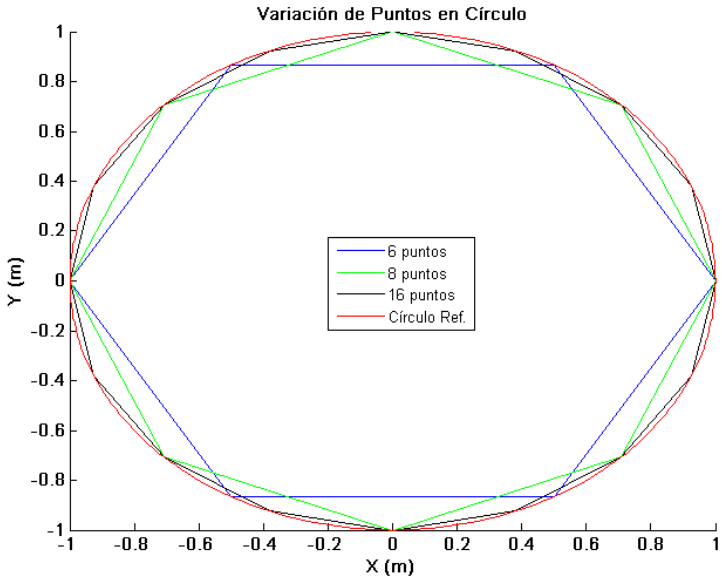


Figura 5.1: Círculos con variación de puntos

movimientos:

- ✓ Para la *altura* se verán los resultados de bajar y subir.
- ✓ Para el *roll* los resultados de desplazarse a la izquierda y derecha.
- ✓ Para el *pitch* los resultados de ir para delante y para atrás.
- ✓ Para el *yaw* los resultados de girar a la izquierda y a la derecha.

En las trayectorias simples, las cuales son combinaciones de los movimientos básicos, se realizaron pruebas con desplazamientos previamente definidos en la base de datos. Las trayectorias consideradas fueron las siguientes:

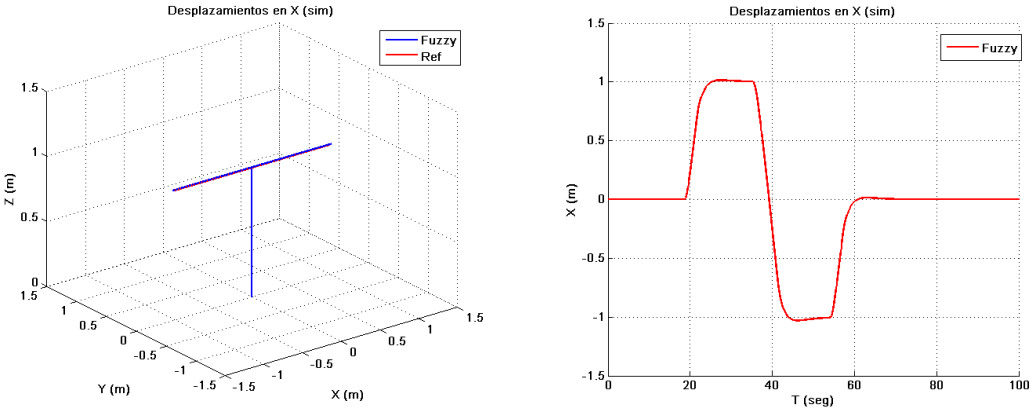
- ✓ Círculo de 8, 16 y 32 puntos.
- ✓ Infinito de 8, 16 y 32 puntos.
- ✓ Espiral de 8, 16, 32 y 64 puntos.
- ✓ Polígonos regulares desde 3, 4, 5, 6, 7 y 8 lados.
- ✓ Rosa Polar de 3 hojas con 32 puntos.
- ✓ Cubo de 1 metro de lado.

Todos los desplazamientos realizados son utilizando medidas absolutas respecto al punto inicial. En las gráficas mostradas de la implementación en la plataforma real, es necesario considerar que, los desplazamientos inicial y final del sistema, no forman parte de la trayectoria o movimiento. Y en la simulación solo existen desplazamientos iniciales, pero tampoco forman parte de la trayectoria considerada.

5.1. Movimientos Básicos

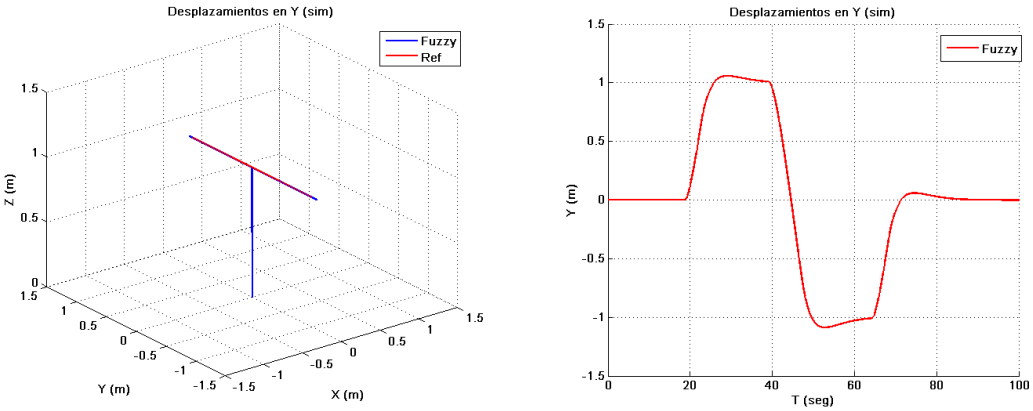
5.1.1. Simulación

A continuación se presentan los resultados obtenidos en la simulación para los movimientos básicos son los siguientes.



a b

Figura 5.2: Desplazamientos en X: *(a) Vista en 3D*; *(b) X Vs Time*



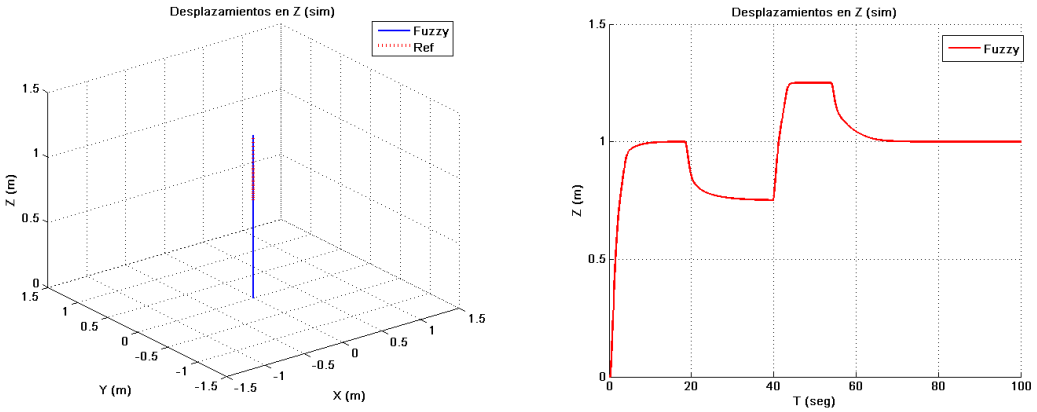
a b

Figura 5.3: Desplazamientos en Y: *(a) Vista en 3D*; *(b) Y Vs Time*

En cada una de las figuras de resultados se observan dos gráficas, de las cuales la primera (inciso a) es la gráfica del desplazamiento del sistema en 3D y la segunda (inciso b) es la gráfica del eje correspondiente al movimiento respecto del tiempo.

Para los desplazamientos en X se observa en la figura 5.2 que el sistema se eleva a 1 metro como punto inicial; una vez llegando a la altura de 1 metro se desplaza 1 metro en X y después retrocede 1 metro en X, regresando al final al punto inicial.

Para los desplazamientos en Y se realizaron pruebas similares a las de los desplazamientos en X. Elevándose a 1 metro de elevacion como punto inicial; una vez en 1 metro de altura se avanza 1 metro en Y y después retrocede 1 metro en Y, regresando al final al punto inicial.



a b

Figura 5.4: Desplazamientos en Z: *(a) Vista en 3D*; *(b) Z Vs Time*

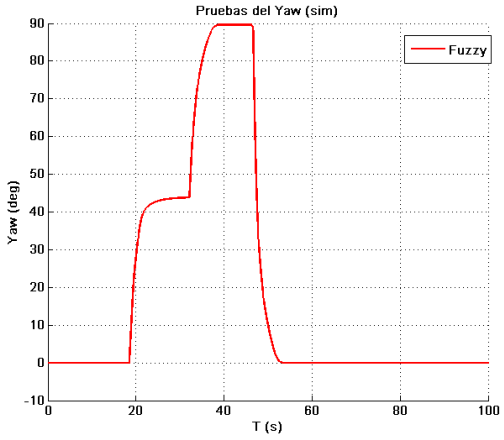


Figura 5.5: Pruebas del ángulo Yaw

Las pruebas de elevación se pueden observar en la figura 5.4 donde se puede apreciar que el sistema sube y baja, modificando su altura para lograr pasar por los puntos indicados en el eje

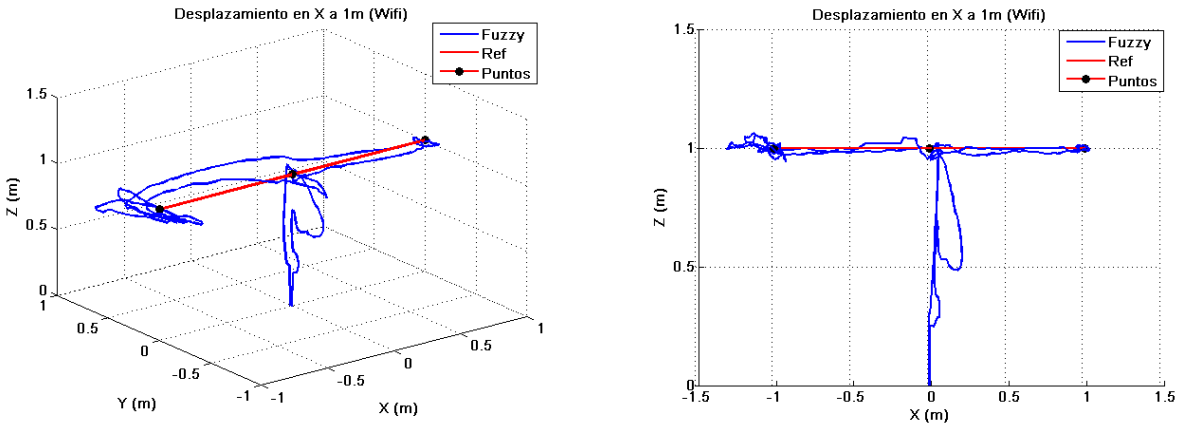
Z. El sistema inició a 1m, descendió a 0.75m, después se elevó a 1.25m y al final regresó al punto inicial de 1m.

Para validar el funcionamiento del control del ángulo *Yaw*, se elevó al sistema a 1 metro de altura y ahí se varió la referencia del *Yaw* a diferentes puntos, permitiendo el cambio de orientación en el sistema. En la figura 5.5 se pueden observar cambios a 45 grados, después a 90 grados y para terminar se regresó al punto inicial.

5.1.2. Plataforma Real

Como resultado de las trayectorias, en las cuales se realizaron movimientos básicos pero ahora implementados en la plataforma real se obtuvieron las siguientes gráficas.

Para observar la respuesta de los desplazamientos en X de la plataforma real, se puede apreciar en la figura 5.6 que el sistema se eleva a 1 metro de altura como punto inicial; una vez en 1 metro de altura se desplaza hacia adelante 1 metro en X y después 1 metro hacia atrás en X, regresando al final al punto inicial.



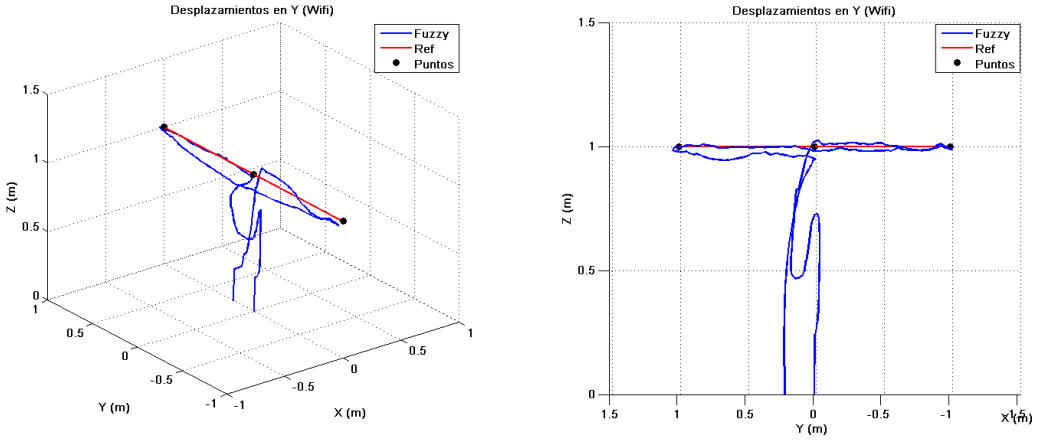
a b

Figura 5.6: Desplazamientos en X (plataforma real): *(a) Vista en 3D*; *(b) Vista en el plano XZ*

Para los desplazamientos en Y se elevó nuevamente a 1 metro de altura como punto inicial y se desplazó 1 metro en Y y después retrocede 1 metro en Y, regresando al final al punto inicial (ver figura 5.7). En ambos desplazamientos, en el eje X y Y, se puede observar que el sistema transita de un punto a otro de los establecidos, manteniendo su altura aproximada a un metro.

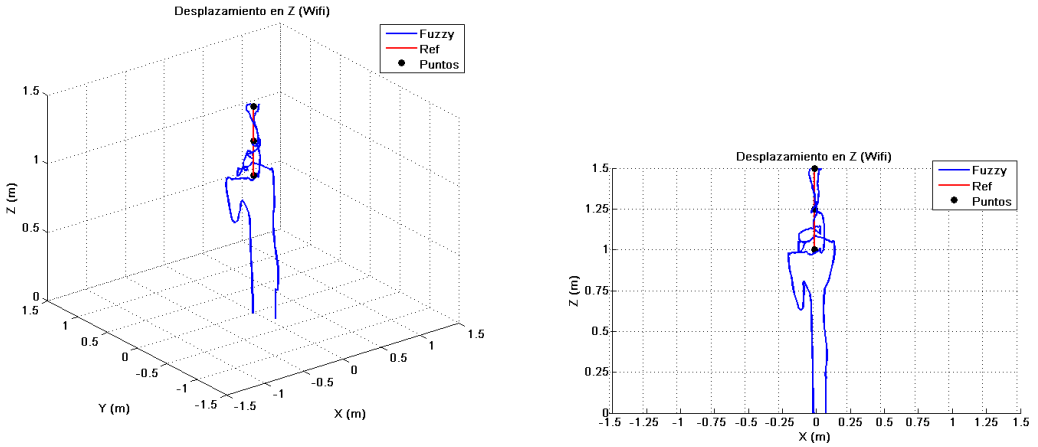
En la plataforma Real se realizó una prueba similar de elevación, en la cual el sistema tenía que pasar por 3 puntos de altura: a 1m, a 1.25m y a 1.5m manteniéndose 20 seg en cada uno (ver figura 5.8).

Para el validar el control del ángulo *Yaw*, se elevó al sistema a 1 metro, se varió el *Yaw* a 45 grados, después a 90 grados y para terminar se regresó al punto inicial (figura 5.9).



a b

Figura 5.7: Desplazamientos en Y (Plataforma Real): *(a) Vista en 3D*; *(b) Vista en el plano YZ*



a b

Figura 5.8: Desplazamientos en Z (Plataforma Real): *(a) Vista en 3D*; *(b) Vista en el plano XZ*

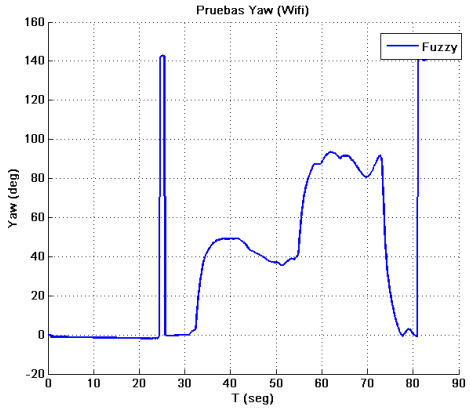


Figura 5.9: Pruebas del ángulo *Yaw* (Plataforma Real)

En las gráficas de los movimientos básicos se puede apreciar que el sistema logra llegar a los puntos específicos, a pesar de que en el transcurso de un punto a otro tenga movimientos sacádicos y pierda su posición momentaneamente. Estos movimientos sacádicos se consideran que son debidos a agentes externos como los mencionados anteriormente que a diferencia de la simulación estos se omiten y el desempeño del sistema produce una trayectoria cercana a la referencia.

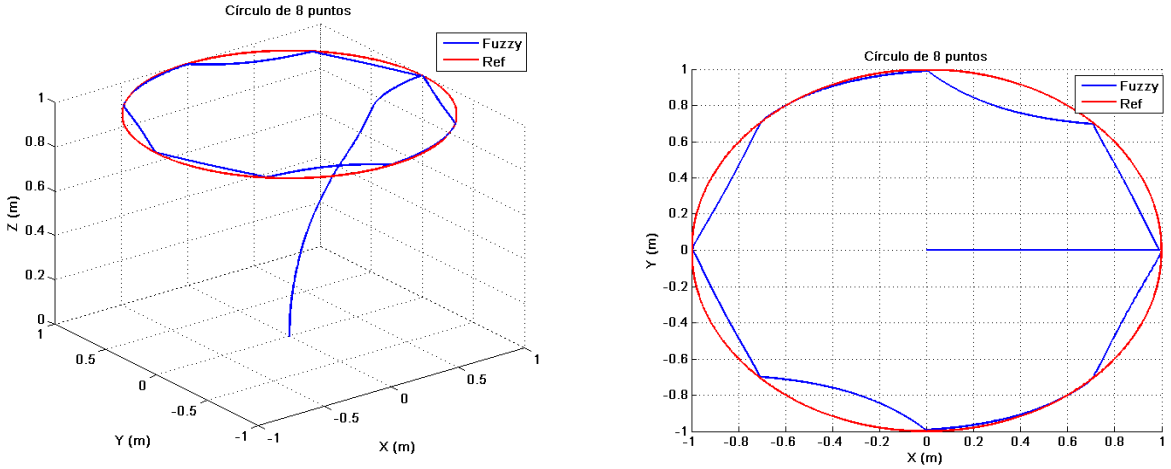
5.2. Trayectorias Simples

Se realizó el seguimiento de las trayectorias simples previamente establecidas en la sección de trayectorias específicas. Este seguimiento se realizó de la misma manera que los movimientos básicos, es decir, tanto en simulación como en la plataforma real.

5.2.1. Simulación

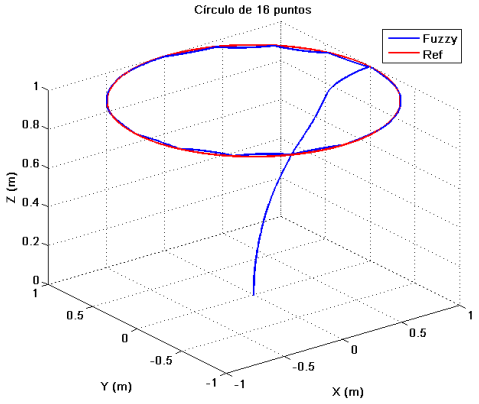
Los resultados de simulación se pueden considerar como ideales. Ya que se omiten aspectos externos como el aire o la intensidad luminosa, o incluso aspectos internos como la lectura de datos en la comunicación o el sensado correcto de su posición, etc. Sin embargo se consideran los retrasos de tiempo debidos a la comunicación y el modelo obtenido mediante el toolbox de Identificación del Sistema con Matlab.

Una vez considerado lo anterior, se utilizó el modelo obtenido mediante la identificación del sistema, para realizar el seguimiento con el quadrotor de trayectorias en forma de círculos a 1 metro de altura, con diferente número de puntos permitiendo observar los diferentes comportamientos de cada uno de los círculos. En la figura 5.10 se puede observar el seguimiento de la trayectoria con 8 puntos. Esta misma trayectoria realizada con 16 puntos se observa en la figura 5.11 y en la figura 5.12 se puede ver el seguimiento de la trayectoria con 32 puntos.

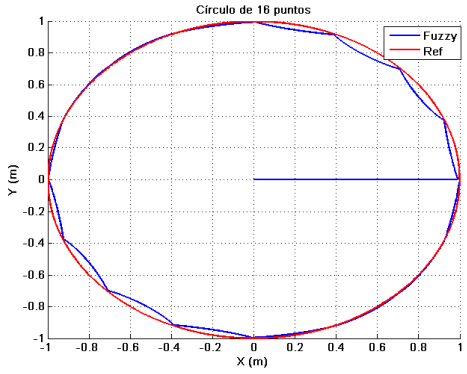


a b

Figura 5.10: Seguimiento de un Círculo de 8pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*

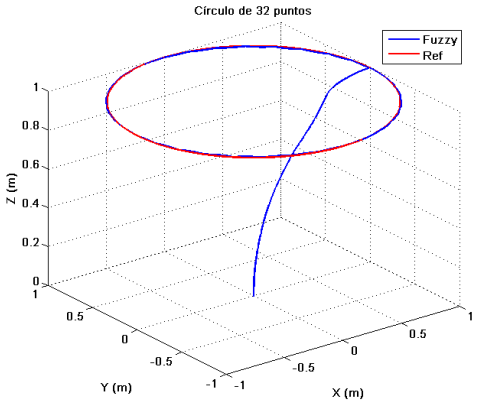


c

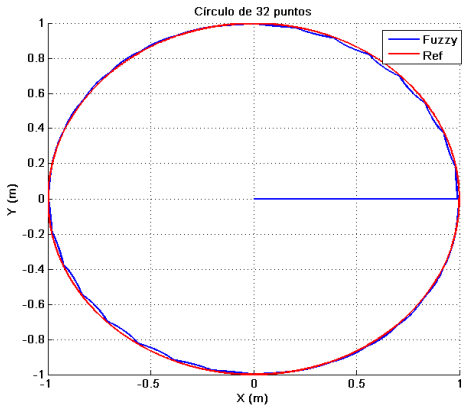


d

Figura 5.11: Seguimiento de un Círculo de 16pts: *(c) Vista en 3D*; *(d) Vista en el plano XY*

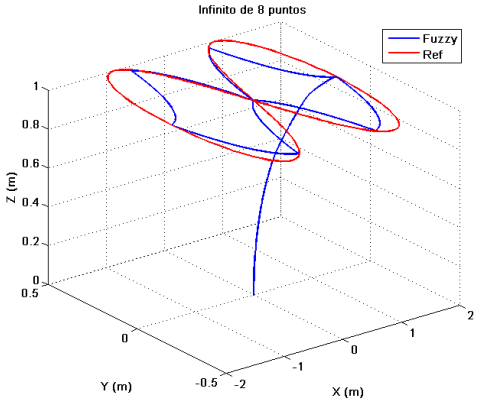


e

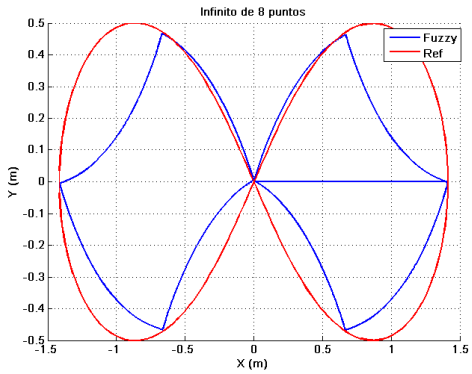


f

Figura 5.12: Seguimiento de un Círculo de 32pt: *(e) Vista en 3D*; *(f) Vista en el plano XY*



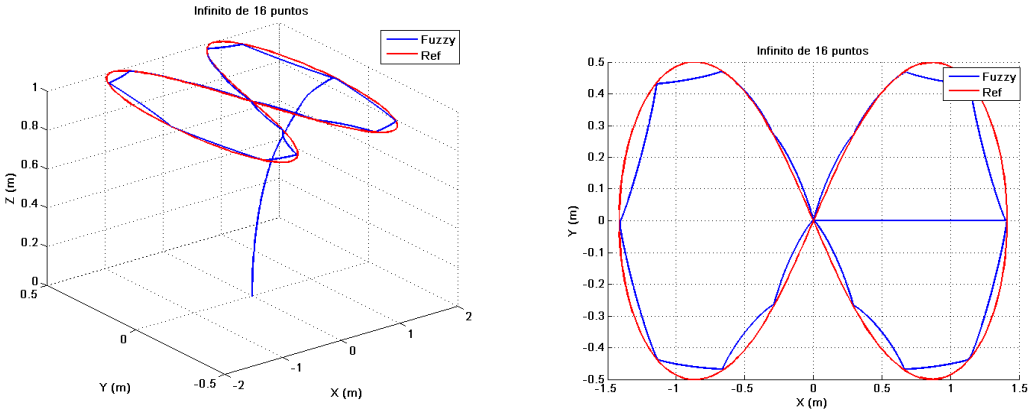
a



b

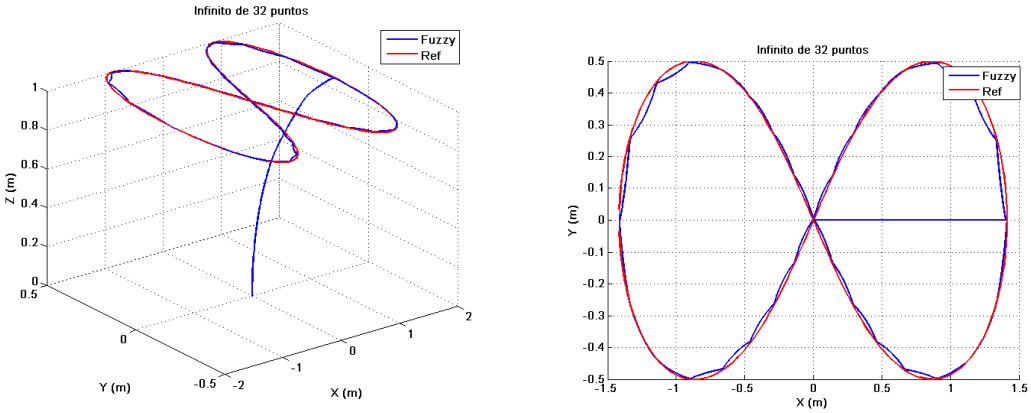
Figura 5.13: Seguimiento de un Infinito de 8pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*

La respuesta del controlador difuso en el sistema ante una trayectoria en forma de infinito se pueden apreciar en la figura 5.13, la cual se realizó con 8 puntos. Al igual que la trayectoria anterior, también se realizó el seguimiento con 16 puntos como se puede ver en la figura 5.14 y con 32 puntos en la figura 5.15. Estas gráficas se realizaron a una altura de 1 metro y se obtuvieron los resultados de 3 diferentes números de puntos.



c d

Figura 5.14: Seguimiento de un Infinito de 16pts: *(c) Vista en 3D*; *(d) Vista en el plano XY*



e f

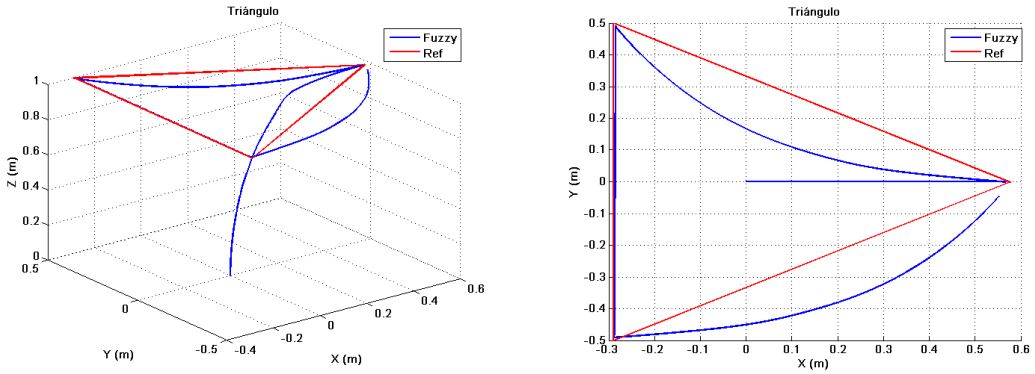
Figura 5.15: Seguimiento de un Infinito de 32pts: *(e) Vista en 3D*; *(f) Vista en el plano XY*

En las gráficas de los círculos y los infinitos se puede observar el comportamiento antes mencionado al inicio de este capítulo, en donde a mayor número de puntos que tenga la trayectoria de referencia, la respuesta del sistema será una trayectoria más cercana a la deseada. Por ejemplo, en las figuras 5.12 y 5.15 se puede observar que la respuesta del sistema fue muy similar a la trayectoria de referencia, a diferencia de las figuras 5.10 y 5.13 en las cuales solo se aprecia que llega solo a los puntos específicos. Sin embargo cabe resaltar que el comportamiento que se espera del sistema es que pase por los puntos de referencia, lo cual efectivamente lo hace.

Para validar el funcionamiento del controlador ante otro tipo de trayectorias, se realizó también el seguimiento de trayectorias de polígonos regulares, de los mencionados al inicio de esta sección.

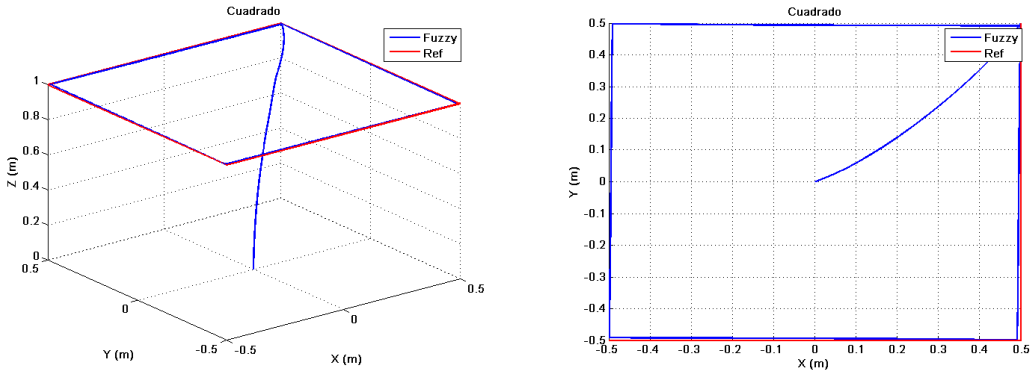
Analizando el resultado obtenido de los polígonos, se pueden observar ciertas diferencias entre ellos. Por ejemplo, en la trayectoria en forma de triángulo, el sistema llega a los puntos de referencia, pero el desplazamiento no es en línea recta, a diferencia del cuadrado en el cual su movimiento es aproximadamente mediante líneas rectas. El seguimiento de triángulo se puede observar en la figura 5.16, Mientras que en la figura 5.17 se presenta el seguimiento de una trayectoria de un cuadrado.

Éste comportamiento es debido a que en los desplazamientos del cuadrado, de un punto a otro, solo se varia uno de sus ejes. Y en trayectorias como el triángulo o el hexágono (ver figura 5.18) solo dos de ellos lo son.



a b

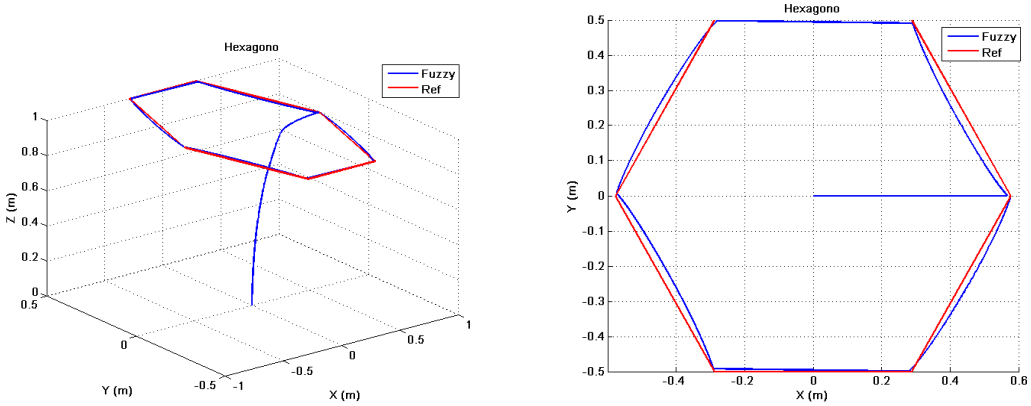
Figura 5.16: Seguimiento de un Triángulo:*(a) Vista en 3D*; *(b) Vista en el plano XY*



a b

Figura 5.17: Seguimiento de un Cuadrado:*(a) Vista en 3D*; *(b) Vista en el plano XY*

Además de usar trayectorias con puntos en un solo plano, se realizaron otras considerando puntos de seguimiento con componentes en X, Y y Z. Como en el caso del seguimiento sobre las



a b

Figura 5.18: Seguimiento de un Hexágono:*(a) Vista en 3D*; *(b) Vista en el plano XY*

aristas de un cubo y la trayectoria de una espiral.

Para el cubo de 1m de lado mostrado en la figura 5.19, se realizó mediante dos cuadrados, uno en la cara inferior y otro en la cara superior del cubo, omitiendo algunas aristas debido a que se tendrían que repetir otras para lograr pasar por todas las caras del mismo.

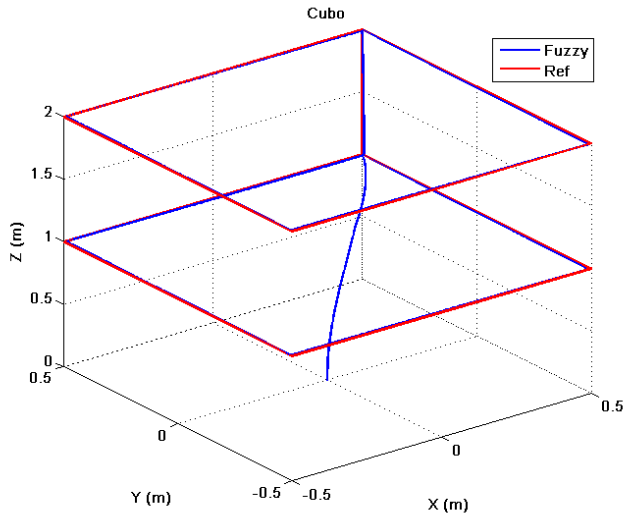
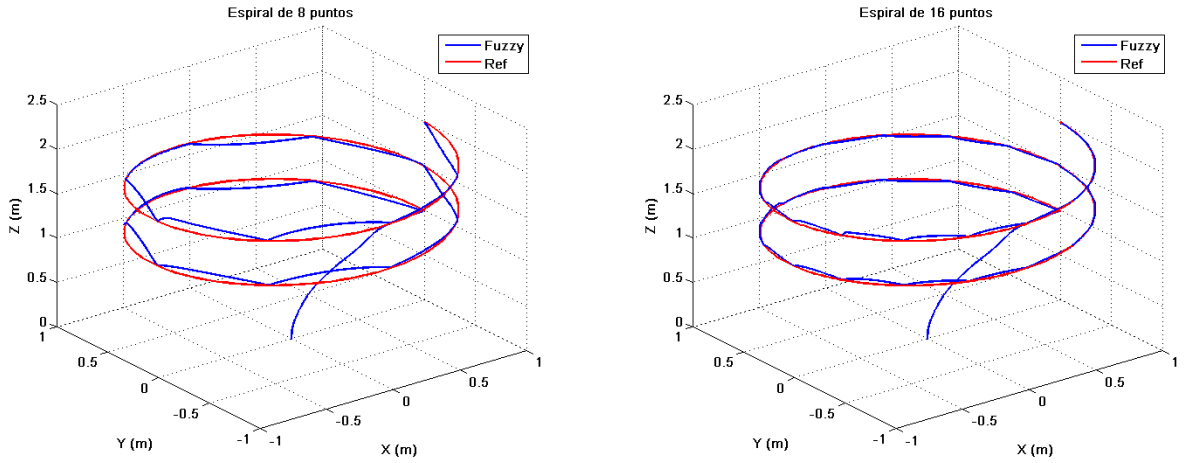


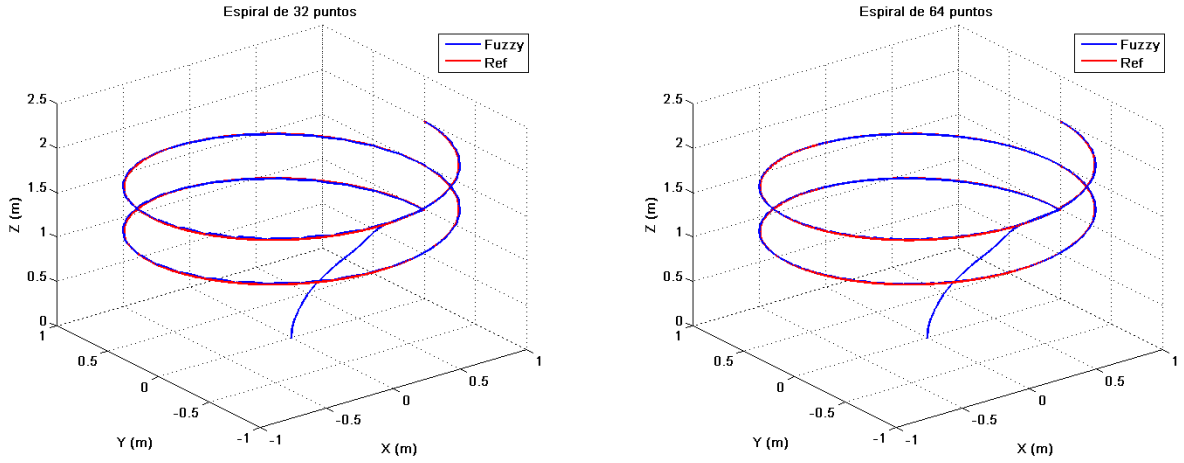
Figura 5.19: Seguimiento de un Cubo de 1m de lado.

La complejidad aumentó al realizar una espiral (figura 5.20 y 5.21), ya que en esta trayectoria se modifican los tres ejes constantemente, pero aun así el sistema realizó el seguimiento adecuadamente.



a b

Figura 5.20: Seguimiento de un Espiral: *(a) Espiral de 8pts/vuelta en 3D*; *(b) Espiral de 16pts/-vuelta en 3D*



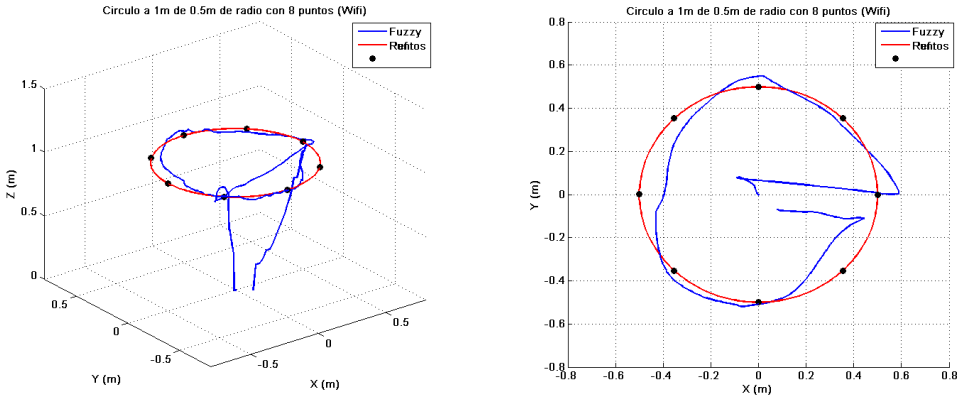
a b

Figura 5.21: Seguimiento de un Espiral: *(a) Espiral de 32pts/vuelta en 3D*; *(b) Espiral de 64pt-s/vuelta en 3D*

5.2.2. Plataforma Real

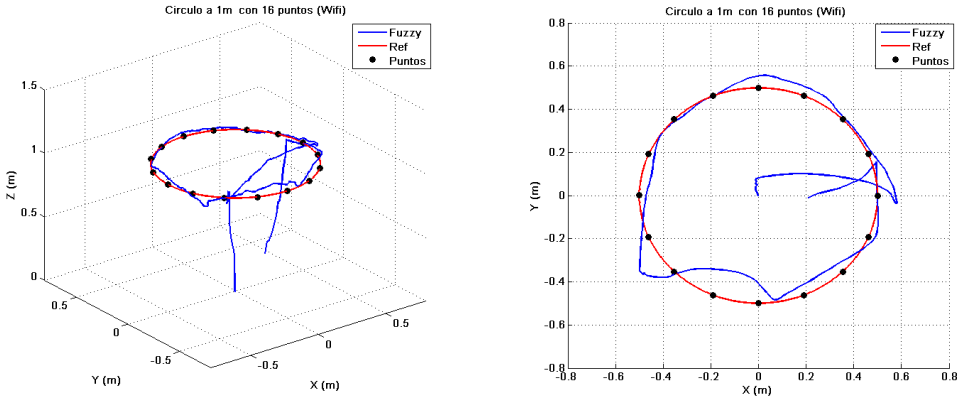
Como resultados de trayectorias en las cuales se realizaban desplazamientos combinados implementados en la plataforma real, se obtuvieron las siguientes gráficas:

Para validar el comportamiento del sistema ante las mismas trayectorias pero con diferente número de puntos, se realizaron nuevamente pruebas en trayectorias circulares y de un infinito, así como también en algunos polígonos y otras trayectorias de las utilizadas para validar el comportamiento en la simulación.



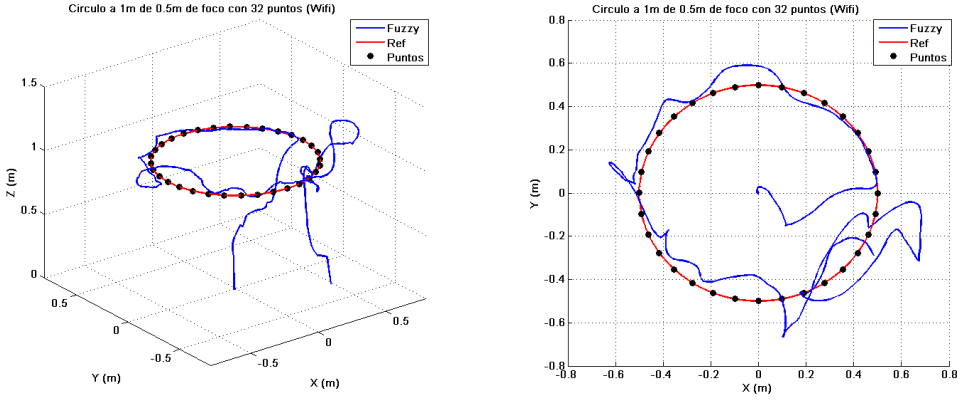
a b

Figura 5.22: Seguimiento de un Círculo de 8pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*



a b

Figura 5.23: Seguimiento de un Círculo de 16pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*

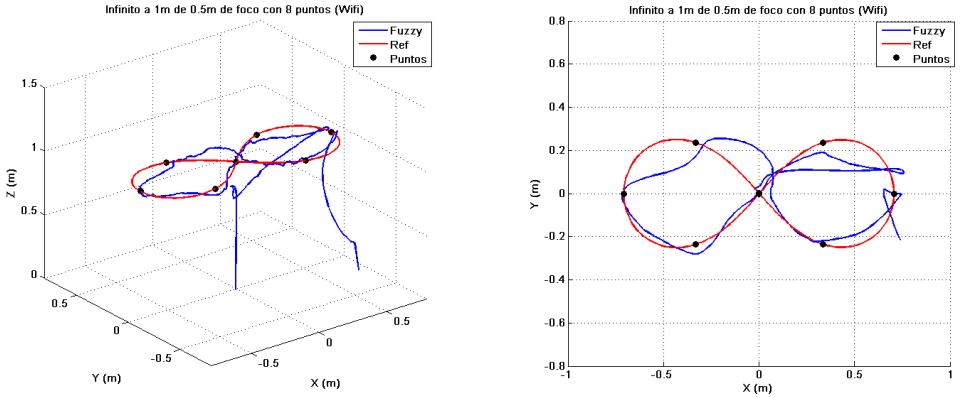


a b

Figura 5.24: Seguimiento de un Círculo de 32pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*

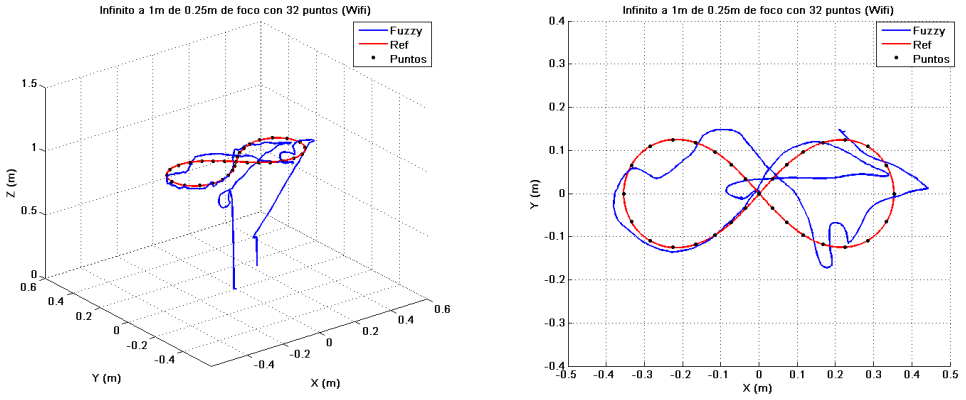
En las tres figuras de las trayectorias en forma de círculos se puede apreciar que el sistema se desplazó por toda la trayectoria pasando por los puntos de referencia. Para validar el momento en el cual el sistema se encuentra en un punto deseado de la trayectoria, se estableció un margen de error para cada una de las variables. Es por eso que en ocasiones el sistema no pasa exactamente sobre el punto, pero considerando errores en la medición mediante el sensado, el sistema pasa cerca de los puntos, dentro del margen establecido.

El comportamiento del sistema en el seguimiento de trayectorias en forma de infinito con 8 y 32 puntos mediante el control difuso, se puede apreciar en las figuras 5.25 y 5.26 respectivamente.



a b

Figura 5.25: Seguimiento de un Infinito de 8pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*



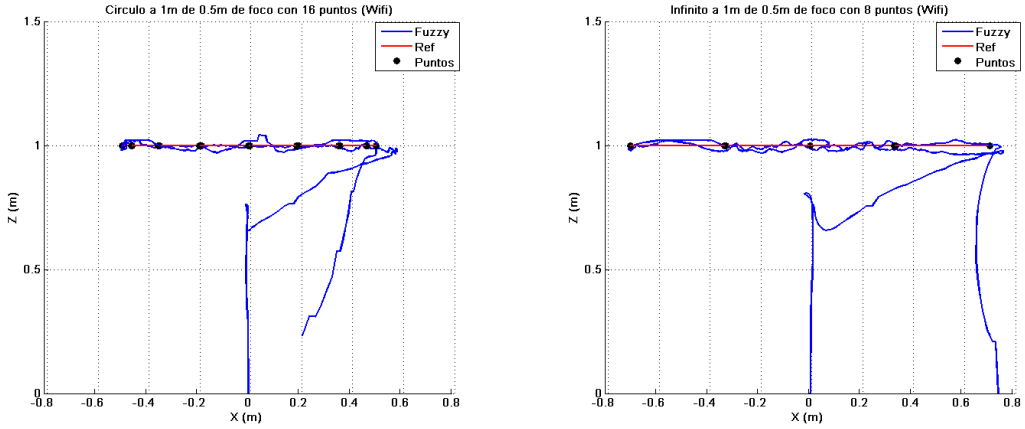
a b

Figura 5.26: Seguimiento de un Infinito de 32pts: *(a) Vista en 3D*; *(b) Vista en el plano XY*

En las trayectorias de infinitos, al igual que en los círculos seguidos por el sistema, se puede apreciar que en ambos casos, la trayectoria (considerando los movimientos sacádicos) tiene una forma aproximada a la deseada. Corroborando con ésto lo antes mencionado en la simulación, que a mayor cantidad de puntos más cercano es el seguimiento a la trayectoria de referencia.

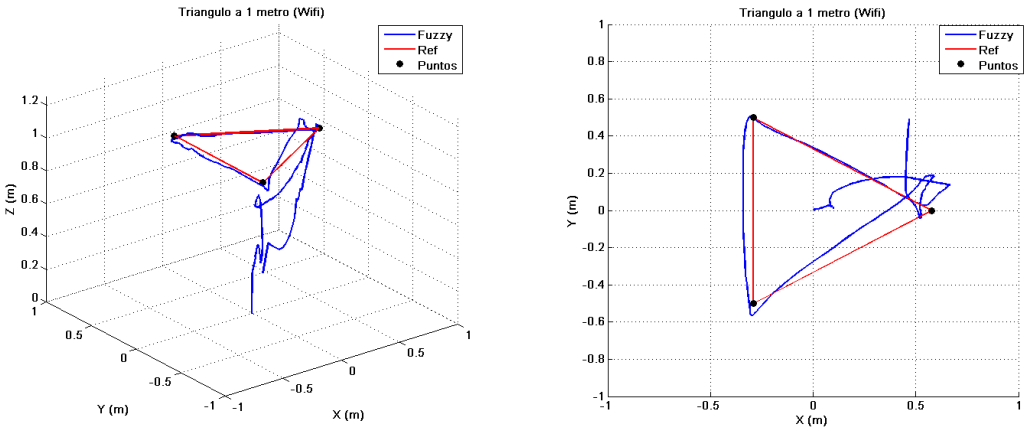
Aunque en las gráficas mostradas del seguimiento de las trayectorias por la plataforma real pareciera que se mueve de manera aleatoria debido a las oscilaciones. Tales oscilaciones se producen, principalmente, en el plano XY, ya que como se puede observar en la figura 5.27 el sistema es capaz de mantener su altura y al mismo tiempo realizar desplazamientos en el plano.

Estos movimientos sacádicos antes mencionados son más notables si la batería se encuentra en un nivel bajo y para el sistema es más difícil mantener su posición o desplazarse a un punto nuevo.



a b

Figura 5.27: *(a) Seguimiento de un Círculo, Vista en XZ*; *(b) Seguimiento de un Infinito, Vista en XZ*

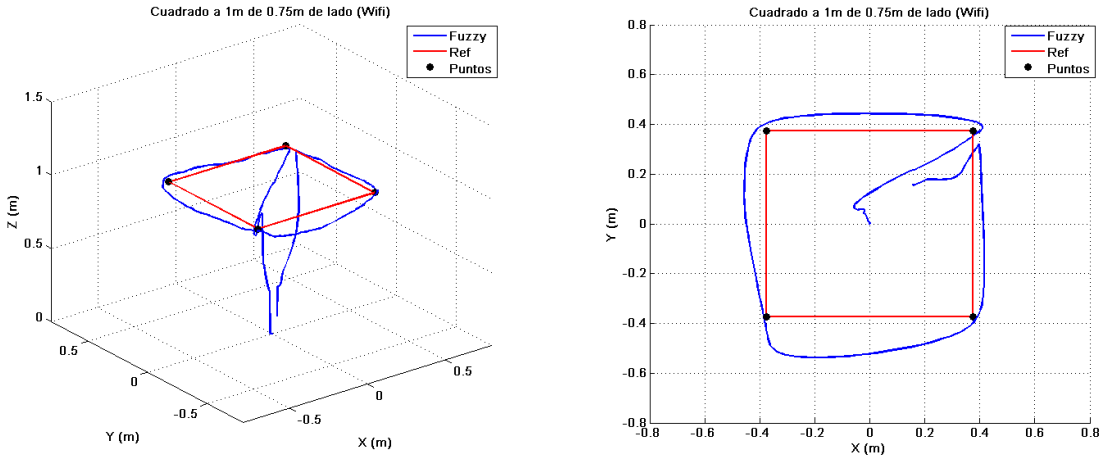


a b

Figura 5.28: Seguimiento de un Triángulo:*(a) Vista en 3D*; *(b) Vista en el plano XY*

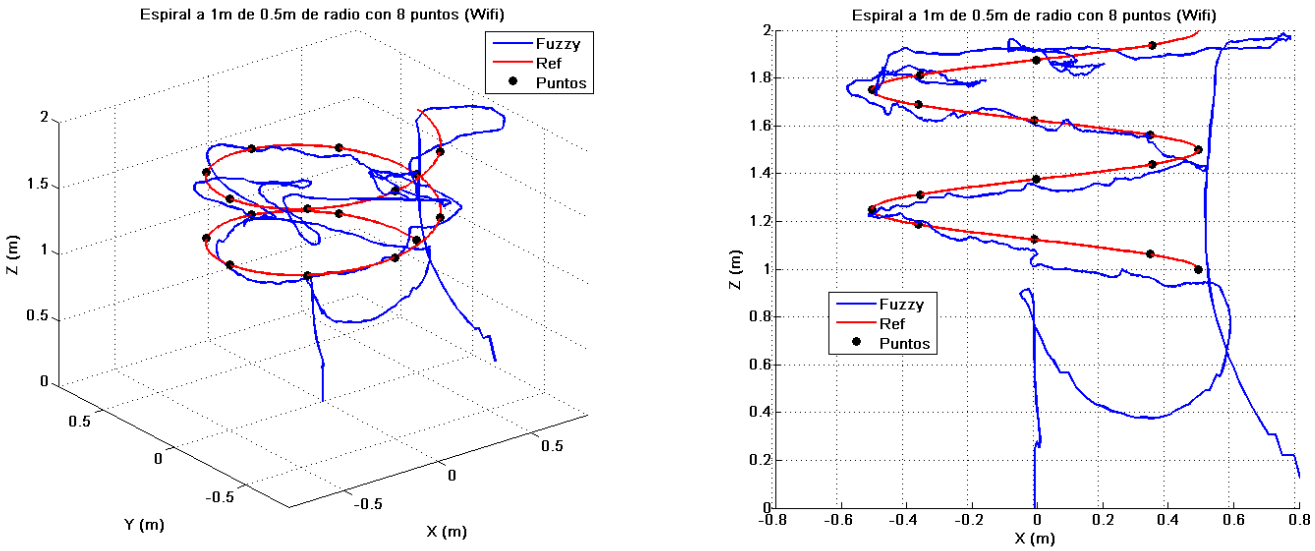
Si bien es cierto que se observan algunos movimientos sacádicos el sistema se mantiene robusto, pero en el seguimiento de trayectorias poligonales con la plataforma real se obtuvo un desempleño muy similar a la trayectoria de referencia, siendo más estable que las trayectorias

anteriores. Ésto se puede apreciar en la figura 5.28 en donde se realiza el seguimiento de un triángulo, y en la figura 5.29 el seguimiento de un cuadrado.



a b

Figura 5.29: Seguimiento de un Cuadrado: *(a) Vista en 3D*; *(b) Vista en el plano XY*



a b

Figura 5.30: Seguimiento de un Espiral: *(a) Vista en 3D*; *(b) Vista en el plano XZ*

En la figura 5.30 se puede observar que el sistema es capaz de realizar el seguimiento de la trayectoria más compleja considerada en este trabajo, un espiral. En dicha figura se aprecia que mientras más alto se encuentra el sistema, más difícil es el movimiento. Esto debido a que las hélices por el uso proveen de menos empuje al sistema, este aspecto también influye en la estabilidad y en las oscilaciones de todas las trayectorias desempeñadas.

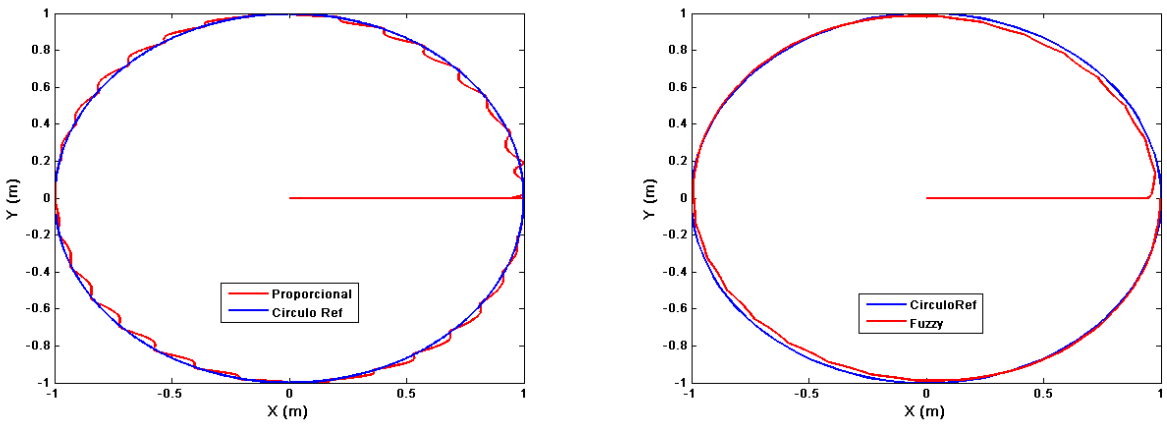
5.3. Comparación de Resultados

Los resultados del desempeño del sistema con el controlador difuso que se implementó, se compararon con los resultados obtenidos mediante el control proporcional incluido en el Kit ARDrone. Estas comparaciones se realizaron tanto en simulación como en la plataforma real, en ambos caso se tiene un margen de error de 0.08m para la detección de cada punto de la trayectoria.

Este margen de error, influye en la medición para saber si se encuentra en el punto deseado. Al ser mayor, es más fácil para el sistema realizar una trayectoria, ya que aunque tenga oscilaciones permanece dentro del rango de error establecido.

En las figuras 5.31 y 5.32, se puede observar la comparación realizada del seguimiento de las trayectorias utilizando el controlador proporcional incluido en el kit de desarrollo, utilizado para la implementación en Matlab, en contra del seguimiento, utilizando el controlador difuso desarrollado en este trabajo.

En el seguimiento de un círculo de 32 puntos mostrado en la figura 5.31, se puede observar que ambas trayectorias pasan por los puntos deseados, pero se puede apreciar que el seguimiento con el control proporcional tiene más oscilaciones que el control difuso.

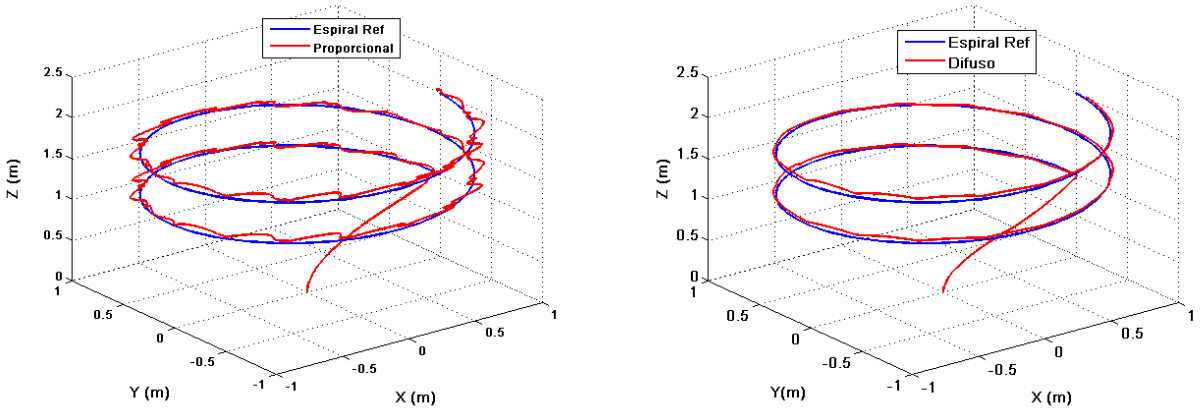


a b

Figura 5.31: Comparación del seguimiento de un Círculo de 32pts: *(a) Control Proporcional*; *(b) Control Difuso*

De manera similar al seguimiento de un círculo, se realizó la comparación del desempeño del sistema en el seguimiento de un espiral de 32 puntos por vuelta, utilizando ambos métodos de control.

En el seguimiento del espiral se observa que al igual que en el caso de los círculos, el control proporcional tiene una mayor cantidad de oscilaciones y el control difuso lleva a cabo un seguimiento más fiel a la trayectoria de referencia.

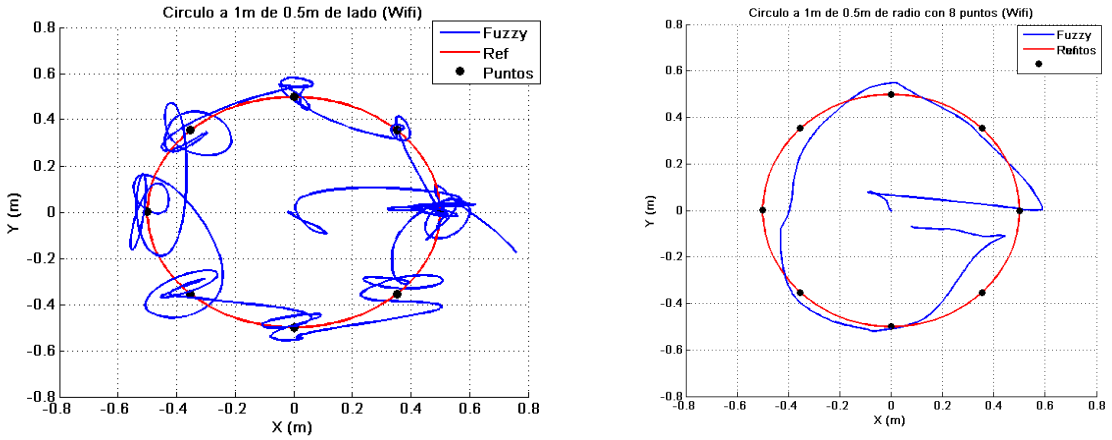


a b

Figura 5.32: Comparación del seguimiento de un Espiral de 32pts: *(a) Control Proporcional*; *(b) Control Difuso*

La comparación también se realizo implementando ambos métodos de control en la plataforma real, en las figuras 5.33 y 5.34 se observa la comparación de una trayectoria en forma de círculo.

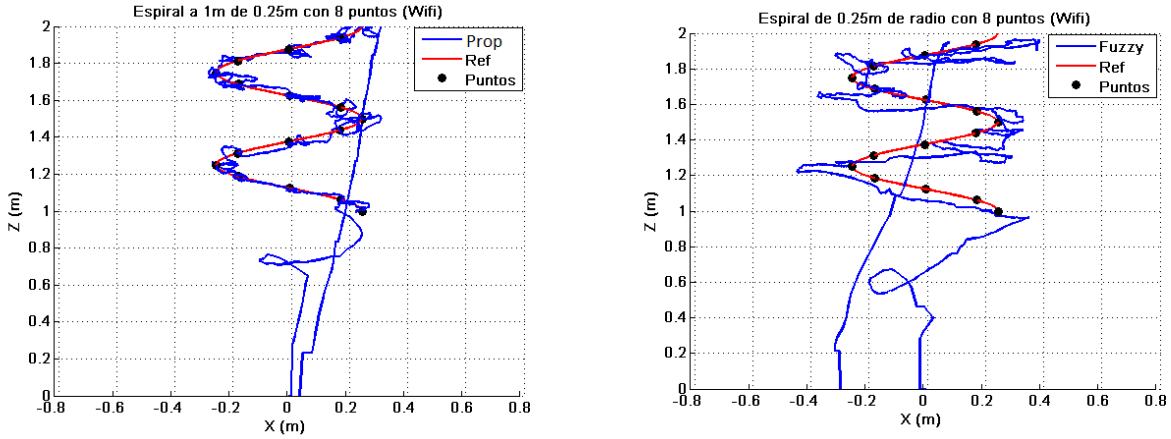
Al igual que en la simulación, ambas trayectorias pasan por los puntos deseados, aunque en el transcurso de punto a punto oscilen un poco (ver figura 5.33).



a b

Figura 5.33: Comparación del seguimiento de un Círculo de 8pts: *(a) Control Proporcional*; *(b) Control Difuso*

En el seguimiento del espiral ambos controladores realizaron el seguimiento de la trayectoria de manera similar a los círculos anteriores, solo que en esta ocasión se aumentaba la complejidad al tener que modificar constantemente la altura de cada punto. Esto se puede apreciar en la figura 5.34.



a b

Figura 5.34: Comparación del seguimiento de un Espiral de 32pts: *(a) Control Proporcional*; *(b) Control Difuso*

En el controlador difuso se implemento un valor de error menor, con la finalidad de obtener un control más preciso. Al observar las gráficas de comparación de un circulo (figura 5.33) y un espiral (figura 5.34), el comportamiento del quadrotor usando el controlador difuso da la impresión de un movimiento más suave, lo cual es debido a que como detecta estar en su posición no necesita hacer una corrección de la trayectoria.

5.3.1. Gráficas del error

Para observar el error de cada una de las gráficas en simulación de ambos controladores comparados se obtuvieron las siguientes:

Las figuras 5.35 y 5.36 muestran una comparación del desempeño de ambos metodos de control mencionados en la sección anterior, el proporcional (inciso a) y el difuso (inciso b).

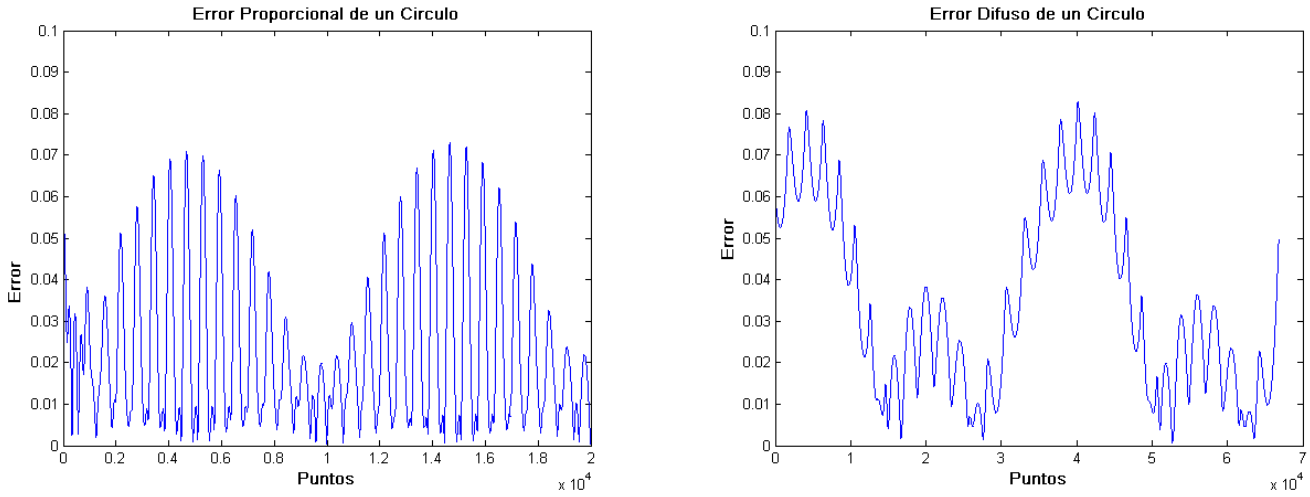
En el desempeño de la trayectoria de un círculo se compararon solo el error en el plano XY para observar que tan cercano es su posición respecto a la trayectoria deseada (vease figura 5.35).

En la figura 5.35 se observa la distancia del error existente por cada punto del sistema respecto a la referencia deseada. Ambos casos controladores tuvieron un error pequeño, pero en el control proporcional se aprecian mas oscilaciones que en el difuso.

De manera similar se realizo una gráfica del error existente en el seguimiento de una trayectoria en 3D, un espiral, lo cual se puede observar en la figura 5.36. En estas gráficas se aprecia que el controlador difuso tiene un error aunque éste tiene menos oscilaciones y es de una magnitud menor que en el controlador proporcional.

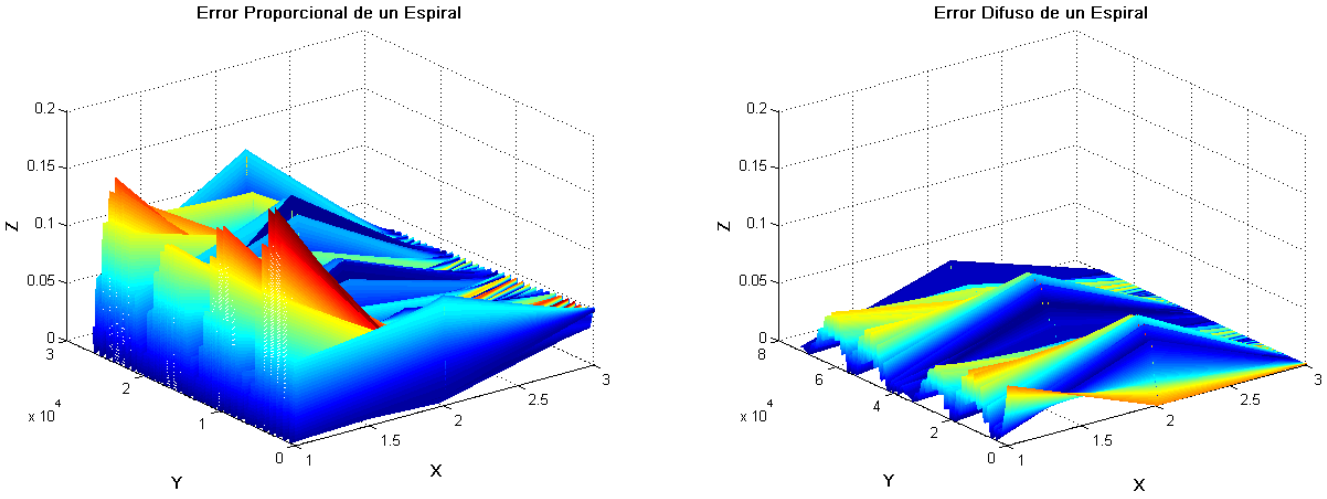
La comparación del error con respecto a la trayectoria de referencia solo se realizó en si-

mulación, ésto debido a que en la implementación real, las variaciones del sistema respecto al tiempo, producidas por las oscilaciones hace que el comparar punto a punto con la trayectoria deseada se torne complicado.



a b

Figura 5.35: Comparación del Error en el seguimiento de un Círculo de 8pts: *(a) Control Proporcional*; *(b) Control Difuso*



a b

Figura 5.36: Comparación del Error en el seguimiento de un Espiral de 32pts: *(a) Control Proporcional*; *(b) Control Difuso*

Capítulo 6

Conclusiones y Perspectivas

El objetivo principal de esta tesis fue desarrollar un controlador difuso para un quadrotor, el cual cuenta con un acelerómetro y un giroscopio de tres ejes, asi como un sensor ultrasónico.

Se desarrolló un controlador difuso mediante Matlab/Simulink y se realizó el módulo de trayectorias. Para el software desarrollado se implementó una interfaz gráfica, hecha mediante un GUI de Matlab.

Se diseño el módulo de comunicaciones, incluido en la interfaz, con el cual fue posible realizar el control de la plataforma real. Este módulo está basado en los bloques de comunicación incluidos en el kit de desarrollo en Simulink V1.1.

En la implementación en la plataforma real se tuvieron algunos problemas los cuales se consideran razonables ya que el sistema no es ideal como el considerado en la simulación. Utilizando el software de *Real-Time Windows* de Matlab para la comunicación de la PC con la plataforma, fue algo problemático ya que en ocasiones al ejecutar el programa, indicaba que el sistema se encontraba conectado a otro dispositivo aunque en la red Wi-Fi no existiera otro dispositivo conectado.

En la plataforma real se observaron oscilaciones con respecto al plano XY. Estas oscilaciones son debidas a que la medición de la posición es estimada con base a los sensores con los que cuenta el sistema, ya que ese valor no es exacto, existen factores como el uso de la cámara inferior que para detectar algún desplazamiento que necesitan de marcas en el piso para identificar algún cambio. El uso de la cámara en lugares donde tiene una superficie uniforme perjudica la medición de los desplazamientos. Para corregir esto, se podría lograr de diversas maneras, como utilizando una cámara extra en el techo del cuarto para medir esos desplazamientos del drone, pero esto limitaría el uso de este sistema en un escenario donde se tenga un techo.

Considerando los errores producidos por el uso de la cámara inferior para detectar los desplazamientos, el problema influye en la estimación de la posición; los sensores de la plataforma no permiten medir directamente los desplazamientos en le plano XY, pero si miden los ángulos de giro mediante el giroscopio. Por lo cual es necesario realizar un cálculo de donde se encuentra el sistema, utilizando las variaciones en los ángulos.

Además de los ángulos, el quadrotor utiliza la cámara inferior para determinar mediante un algoritmo interno, si existe algún desplazamiento en el plano XY. Dicha detección de movimientos utilizando la cámara inferior fue uno de los principales contratiempos ya que en ocasiones no se detectaba ningún movimiento, esto debido a la iluminación y a que la superficie de el piso debe de tener diferentes formas o marcas para diferenciar su ubicación. Éste problema afectaba directamente en el control de la plataforma ya que si el sistema no detecta un desplazamiento el control no lo va a corregir, produciendo que el quadrotor se mueva de el punto deseado.

El uso de este protocolo puede ser causa de comportamientos erroneos en el sistema debido a que el protocolo UDP realiza el envío de paquetes de datos sin una retroalimentación, es posible que no lleguen al destino o que lleguen por alguna razón desconocida en orden equivocado o incluso retrasados.

Además de los problemas enunciados antes, el desgaste producido por el uso del sistema, debido a los choques de las diferentes pruebas o caídas de la plataforma, producía que el sistema disminuyera su desempleño y también fuera más difícil realizar las trayectorias. Con el paso del tiempo las hélices tienden a pandearse y es necesario realizar el mantenimiento adecuado.

La carga de las pilas también es un aspecto importante que se debe considerar, ya que cuando la batería es muy baja (abajo del 20 %) el sistema ya no responde adecuadamente.

De manera general, se concluye que a pesar de las problemáticas tenidas en el desarrollo de esta tesis, el control de un quadrotor es posible utilizando un controlador con Lógica Difusa.

Considerando el desarrollo de este proyecto y en base a los experimentos realizados y a los resultados obtenidos se consideran algunas posibles tareas para continuar el desarrollo de este trabajo. Algunas de ellas se enlistan a continuación:

- ✓ Agregar otro tipo de sensores para medir su posición permitirían lograr un control más acertado, ya que la ubicación sería precisa.
- ✓ Mejorar la estimación de la posición del sistema para lograr tener un control con un error mínimo.
- ✓ Aproximar las trayectorias mediante algún algoritmo que permita que los movimientos sean más fieles a la trayectoria de referencia deseada; como por ejemplo el uso de mínimos cuadrados.
- ✓ Extender esta investigación hacia el estudio de la planificación de trayectorias permitiendo que el sistema planifique y siga una trayectoria de manera autónoma.
- ✓ Generar un conjunto de prácticas donde se permita beneficiar a los posibles estudiantes que en un futuro hagan uso de esta investigación o de la plataforma, para aprender el funcionamiento y también crear nuevos desarrollos a partir de ellas.

Bibliografía

[1] IFR. Internacional Federation of Robotics, website, http://www.ifr.org/service-robots/, [accesado agosto 2014].

[2] Aldebaran Robotics. Nao, website, http://www.aldebaran.com/en/humanoid-robot/nao-robot, [accesado agosto 2014].

[3] Honda. Asimo, website, http://www.honda.mx/asimo/, [accesado agosto 2014].

[4] Robot humanoide hrp-4c, website, http://www.fayerwayer.com/2009/03/hrp-4c-humanoide-fashionista/, [accesado noviembre 2014].

[5] TechJect. Dragonfly, website, http://techject.com/robot-dragonfly/ [accesado agosto 2014].

[6] Festo. Smartbird, website, http://www.festo.com/cms/encorp/11369.htm [accesado agosto 2014].

[7] Sony. Aibo, website, http://www.sony-aibo.co.uk/ [accesado agosto 2014].

[8] Automóvil sin conductor de google, website, http://www.nytimes.com/2011/05/11/science/11drive.html?_r=1&emc=eta1 [accesado noviembre 2014].

[9] Aníbal Ollero Baturone. *Robótica, Manipuladores y robots móviles.* Alfaomega, 2001.

[10] Autores Varios. Matlab&simulink, mathworks, website, http://www.mathworks.com/index.html?s_tid=gn_logo [accesado agosto 2014].

[11] Traxxas qr1 quadrotor, website, http://traxxas.com/products/models/heli/6208qr1 [accesado agosto 2014].

[12] RobotsLab. Robot box, website, www.robotslab.com [accesado agosto 2014].

[13] Louis Charles Bréguet. Gyroplane no.1, website, http://www.ctie.monash.edu.au/hargrave/breguet.html [accesado agosto 2014].

[14] Gabriel M. Hoffmann, Haomiao Huang, and Steven L. Waslander Claire J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. published by th american institute of aeronautics and astronautics. south carolina. 2007.

[15] Uso de uav's para monitoreo de fronteras, website, www.infodefensa.com/es/2011/11/17/noticia-el-gobierno-de-brasil-empleara-tres-uav-para-el-monitoreo-ininterrumpido-de-la-triple-frontera-del-sur-2.html [accesado agosto 2014].

[16] Seguridad y vigilancia, website, http://www.muyinteresante.com.mx/tecnologia/562466 /drones-vehiculos-aereos-no-tripulados-uso-industrial-cientifico-militar/ [accesado agosto 2014].

[17] Sagarpa. Sistema de información agroalimentaria y pesquera, website, http://www.horticultivos.com/component/content/article/49-front-page/763-sagarpa-incorpora-drones-para-generar-informacion-precisa [accesado agosto 2014].

[18] DHL. Envío de paquetería con drones, website, http://www.cnnexpansion.com/negocios /2013/12/09/dhl-entra-a-la-carrera-de-los-drones [accesado agosto 2014].

[19] Amazon. Prime air, website, http://www.amazon.com/b?node=8037720011 [accesado agosto 2014].

[20] Phantom dji quadrotor, website, http://www.dji.com/product/phantom [accesado agosto 2014].

[21] Foxtech d130 quadrotor, website, http://www.foxtechfpv.com/foxtech-products-foxtech-d130-quadcopter-c-240_234.html [accesado agosto 2014].

[22] Aerodrone mr4 quadrotor, website, http://diydrones.com/profiles/blogs/introducing-the-aerodrone-mr4 [accesado agosto 2014].

[23] Parrot. Ar.drone2.0, website, http://ardrone2.parrot.com/ [accesado septiembre 2014].

[24] Technische Universität München. Visual navigation for flying robots, website, http://vision.in.tum.de/teaching/ss2013/visnav2013#visual_navigation_for_flying_robots [accesado agosto 2014].

[25] University of Pensylvania. Grasp-lab, website, https://www.grasp.upenn.edu/ [accesado agosto 2014].

[26] R. Escamilla Núñez. Diseño, construcción, instrumentación y control de un vehículo aéreo no tripulado (UAV), Tesis de licenciatura, Instituto Politécnico Nacional, 2010.

[27] Nick Dijkshoorn. Simultaneous localization and mapping with the ar.drone, Tesis de licenciatura, Universiteit van Amterdam, 2012.

[28] Autores Varios. Modelling and control of autonomous quad-rotor Faculty of Engineering, science and medicine. University of Aalborg. Denmark, 2010.

[29] Tommaso Bresciani. Modelling, identification and control of a quadrotor helicopter, Fepartment of Automatic Control, Tesis de maestría, Lund University. Sweden, 2008.

[30] Paul Pounds, Robert Mahony, and Peter Corke. Modelling and control of a quad-rotor robot. in Proceedings Australasian Conference on robotics and automation. New Zealand. 2006.

[31] Yogianandh Naidoo, Riaan Stopforth, and Glen Bright. Quad-rotor unmanned aerial vehicle helicopter modelling & control. 2011.

[32] Aleksandar Rodić and Gyula Mester. The modeling and simulation of an autonomous quadrotor microcopter in a virtual outdoor scenario. Acta Polytechnica Hungarica. Vol. 8, 2011.

[33] J. de J. Rubio, J. H. P. Cruz, Z. Zamudio, and A. J. Salinas. Comparison of two quadrotor dynamic models. In IEEE Latin America Transactions, editor, *IEEE Latin America Transactions*, volume Vol.12, 2014.

[34] A. Özgür Kivrak. Design of control systems for a quadrotor flight vehicle equipped with inertial sensors, Tesis de maestría, Atilim University, 2006.

[35] L. Besnard, Y. B. Shtessel, and Brian. Landrum. Quadrotor vehicle control via sliding mode controller driven by sliding mode disturbance observer, journal of the Franklin Institute. 2011.

[36] A. Neff. Linear and non linear control of a quadrotor UAV, Tesis de licenciatura, Clemson University, 2007.

[37] Draganfly. Dragonflyer x-pro quadrotor, http://www.draganfly.com/uav-helicopter/draganflyer-x4es/why-draganfly/index.php, [accesado agosto 2014].

[38] G. Filo. Modelling of fuzzy logic control system using the matlab simulink program, Revista técnica de 8. de mecánica, Biblioteca de la Universidad de Tecnología de Cracovia. 2010.

[39] T. González and A. Bravo. Fuzzy controller to change gravity center on mobile manipulators, IEEE Latin America Transactions. Vol. 12, 2014.

[40] I. S. Jiménez Escamilla. Control de temperatura de un horno eléctrico mediante lógica difusa, Tesis de licenciatura, Universidad Tecnológica de la Mixteca, 2012.

[41] Omar López Vargas, Luis Bayardo Sanabria Rodríguez, and Jaime Ibáñez Ibáñez. Desarrollo de la cognición espacial en invidentes congénitos con apoyo de dispositivos tecnológicos. In *Cognicion*, 2013.

[42] G. G. Ponds, J. Villegas Cortez, C. Avilés Cruz, I.Vázquez Álvarez, and I. Osuna Galán. Sistema de freno ABS mejorado con modelo de control difuso y visión, avances de ingeniería electrónica. 2013.

[43] R. Sumathi and M. Usha. Pitch and yaw attitude control of a rocket engine using hybrid fuzzy-pid controller, the open automation and control systems journal. 2014.

[44] Francisco Morata Palacios. Controlador fuzzy de un quadrotor, Tesis de maestría, Universidad Complutense de Madrid, 2009.

[45] Gerard Martin. Modelling and control of the Parrot Ar.Drone,. Reporte técnico, School of Engineering and Information Technology, Canberra, 2012.

[46] Syed Ali Raza and Wail Gueaieb. *Motion Control*, Chapter Intelligent Flight Control of an Autonomous Quadrotor. InTech, 2010.

[47] P S. Bhatkhande. Real time fuzzy controller for quadrotor stability control, Tesis de maestría, Michigan Technological University, 2014.

[48] C. Martínez Poveda. Aplicaciones sobre ardrone, Tesis de licenciatura, Universitat Politécnica de Valéncia, 2013.

[49] F. Jurado, B. Castillo Toledo, and S. Di Gennaro. Stabilization of a quadrotor via takagi_sugeno fuzzy control. 2005.

[50] C. Coza and C. J. B. Macnab. A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization, department of electrical and computer engineering. 2005.

[51] K. M. Zemalache and H. Maaref. Intelligent control for a drone by self-tunable fuzzy inference system. 2009.

[52] Stephane Piskorski, Nicolas Brulez, Pierre Eline, and Frederic D'Haeyer. *Ar.Drone Developer Guide SDK2.0*. Parrot, Marzo 2014.

[53] Stephane Piskorski, Nicolas Brulez, Pierre Eline, and Frederic D'Haeyer. *Ar.Drone Developer Guide*. Parrot, Mayo 2012.

[54] L. Ljung. *System Identification Toolbox, http://www.mathworks.com/help/pdf_ doc/ident/ident.pdf*, 2014.

[55] L. Ljung. Perspectives on system identification. In *Division of Automatic Control, Linöpings Universitet, Sweden*.

[56] J. F. Almendariz Haro. Control del movimiento de un quadrotor mediante un sensor de profundidad kinect, Tesis de licenciatura, Escuela Politécnica Nacional, 2014.

[57] D. Escobar Sanabria and P. J. Mosterman. Ar drone simulink development-kit v1.1, website, http://www.mathworks.com/matlabcentral/fileexchange/43719-ar-drone-simulink-development-kit-v1-1, 2014.

[58] Katsuhiko Ogata. *Ingeniería de Control Moderna*. Pearson Educación S.A., 2010.

[59] S. Curi, I. Mas, and R. S. Pe na. Autonomous flight of a commercial quadrotor. 12(5), 2014.

[60] M. I. Aguilera Hernandez, M. A. Bautista, and I. Joaquin. Diseño y control de robots móviles. Segundo congreso nacional de mecatrónica. Ciudad de México, 2004.

[61] Emmanuel Alejandro Merchán Cruz. Metodología para generación de trayectorias de manipuladores robóticos, su cinemática y dinámica, Tesis de maestría, Instituto Politéecnico Nacional, 2000.

[62] Daniel Garijo Verdejo, Jesús Ismael López Pérez, and Isaac Perés Estrada. Control de un vehículo aéreo no tripulado. Reporte de fin de curso de sistemas informáticos Universidad Complutense de Madrid, 2009.

[63] V. B. Rao. *C++ Neural Networks and Fuzzy Logic*. MTBooks, 1995.

[64] G. Pacheco López. Diseño e implementación de un sistema de monitoreo de color in situ y en tiempo real en un equipo de extracción de jamaica utilizando una cámara ccd. Tesis de licenciatura. Universidad Tecnológica de la Mixteca, 2014.

[65] I. Arroyo Fernández. Evaluación de dos técnicas de reconocimiento de patrones para su implementación en el simulador de pilotaje automático (pa-135, nm-79 chopper) de taller del stc metro de la cd. de México. Tesis de maestría. Universidad Tecnológica de la Mixteca, 2013.

[66] L. H. Ríos and M. Bueno. Modelo matemático para un robot móvil. *Scientia Et Technica*, XIV:13–18, 2008.

[67] Guillermo Morales Luna. Introducción a la lógica difusa. Technical report, CINVESTAV-IPN, 2002.

[68] Timothy J. Ross. *Fuzzy Logic, with engineering aplications, University of New Mexico.* WILEY, 3erd edition, 2010.

[69] L. A. Zadeh. Fuzzy sets. In *Information and Control*, 1965.

[70] S. Sam Ge and F. L. Lewis. *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications.* Taylor & Francis, 2006.

[71] E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. In *Proceedings of the Institution of Electrical Engineers*, 1974.

[72] D. Guzmán and V. M. Casta no. La lógica difusa en ingeniería: Principios, aplicaciones y futuro. In *Ciencia y Tecnología*, 2006.