CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SECCIÓN DE MECATRÓNICA

"Evasión de colisiones para drones operando en un plano horizontal"

Tesis que presenta:
Ezequiel Huesca Hernández

Para obtener el grado de:
Maestro en Ciencias

En la especialidad de:
Ingeniería Eléctrica

Director de la Tesis:
Dr. Eduardo Aranda Bricaire

Ciudad de México Febrero, 2021

Agradecimientos

A Dios: *Por siempre estar en los momentos más difíciles, por brindarme salud y fortaleza, y por no dejarme vencer de las cosas que me atormentaron en algún momento, abriendo mi corazón a las cosas buenas que me ayudaron a salir adelante día con día.*

A mis padres: *Por siempre estar cuando los he necesitado, por brindarme su apoyo incondicional y por todo el esfuerzo que han realizado para ayudarme a cumplir una meta más en mi vida. Por hacerme una persona de bien y con principios, por siempre ayudarme a pesar de las consecuencias y nunca dejarme solo, por todo su amor y comprensión a lo largo de mi vida.*

A mi hermana: *Por su apoyo y comprensión a lo largo de este ciclo, por darme ánimo siempre que pensaba que las cosas iban mal, por su cariño y afecto, por sus buenos consejos y sus pláticas de motivación.*

A mi compañera: *Por comenzar a formar parte de mi vida y porque ha estado conmigo en las buenas y malas, en los ratos de enojo y estrés pero también en los de alegría y felicidad. Por ser un gran apoyo en lo sentimental y profesional, por siempre creer en mí y brindarme su amor y confianza.*

Al Dr. Eduardo Aranda: *Por compartirme parte de su conocimiento a lo largo de este ciclo y brindarme la oportunidad de formar parte de su equipo de trabajo. Por su apoyo y sus críticas constructivas para lograr la culminación de este proyecto.*

A los miembros del jurado: *Por sus comentarios y las aportaciones que hicieron a este trabajo de tesis.*

A mis amigos y compañeros del CINVESTAV: *Por brindarme su amistad en todo este tiempo y compartir momentos agradables, por su apoyo y compañía en esos momentos desesperantes. Por todas las locuras, aventuras y risas compartidas.*

A los doctores de la sección de mecatrónica: *Por su enseñanza, sus consejos y por compartir sus conocimientos y experiencias. Por confiar en mí y darme la oportunidad de realizar mis estudios de maestría en la sección de mecatrónica del departamento de ingeniería eléctrica del CINVESTAV.*

Al CONACYT: *Por el apoyo económico brindado para realizar mis estudios de maestría y poder realizar mi proyecto.*

Al CINVESTAV: *Por permitirme formar parte de su comunidad y darme acceso a sus instalaciones.*

Resumen

En este trabajo de tesis se proponen algunas estrategias de control de movimiento para sistemas multi-agente conformados por agentes tipo cuadrirrotores. Además, se considera la evasión de colisiones de obstáculos conocidos y también entre los agentes del sistema. La estrategia de control resuelve el problema de formación, marcha y asimismo la no colisión.

El control de cada agente se basa en la conexión en cascada de un control "virtual" que considera a los ángulos de alabeo y cabeceo como entradas virtuales y un control "real" que conduce a los ángulos a estos valores deseados.

Posteriormente, el control del sistema multi-agente se basa en el algoritmo fundamental de consenso para lograr una formación deseada en el sistema y para la evasión de colisiones se utilizan campos vectoriales repulsivos.

Abstract

In this thesis work some motion control strategies are proposed for multi-agent systems made up of quadcopter agents. Also, collision avoidance against known obstacles and between the agents of the system is considered.

The control strategy solves the problem of formation, march and the non-collision too. The control of each agent is based on the cascade connection of a "virtual" control that considers roll and pitch angles as virtual inputs and a "real" control that drives the angles to these desired values.

Subsequently, multi-agent control system is based on the fundamental consensus algorithm to achieve a desired formation in the system and repulsive vector fields are used for collision avoidance.

Contenido

Lista de figuras

Capítulo 1

Introducción

Este trabajo de tesis aborda el problema de control de movimiento con evasión de colisiones para drones operando en un plano horizontal. Por control de movimiento, nos referimos al control de formación y/o al control de marcha. La evasión de colisiones se entiende como que en ningún instante de tiempo exista contacto físico entre los drones u obstáculos conocidos.

Ambos problemas han sido ampliamente estudiados para agentes de primer orden, que esencialmente se modelan como integradores simples. La primera dificultad que surge al considerar un sistema multi agente conformado por drones es el modelo matemático, el cual es altamente no lineal y las coordenadas cartesianas en el plano horizontal tienen grado relativo cuatro. Esto conlleva el problema de que, aun prediciendo una colisión, los drones no pueden ser detenidos instantáneamente debido a la inercia.

En el caso que nos ocupa, el problema de control de movimiento se resuelve mediante el conocido algoritmo fundamental de consenso para agentes de segundo orden. El problema de evasión de colisiones se resuelve mediante una ley de control reactiva que se aplica cuando dos agentes están en peligro de colisión.

La ley de control reactiva consiste en un campo vectorial tipo foco repulsivo que se crea alrededor de cada uno de los agentes en peligro de colisión. Entonces, los agentes se evaden siguiendo trayectorias espirales en sentido antihorario.

Una de las motivaciones originales de este trabajo era la validación en tiempo real de los algoritmos propuestos en la plataforma experimental con que cuenta la Sección de Mecatrónica del CINVESTAV. Desafortunadamente, debido a la pandemia, fue imposible acceder a las instalaciones del CINVESTAV durante prácticamente un año. Por ello, los resultados obtenidos se validan mediante simulaciones numéricas en el software Simulink-Matlab.

1.1. Objetivo general

Diseñar un esquema de control que permita que un grupo de cuadrirrotores operando en el mismo plano horizontal resuleva el problema de formación y/o marcha, además de resolver el problema de evasión de colisiones. Se busca evaluar el esquema de control mediante simulaciones numéricas en el software Simulink-Matlab. Estos controles se pretenden probar con el modelo dinámico de un cuadrirrotor AR.Drone 2.0.

1.1.1. Objetivos específicos

- Revisión del estado del arte sobre drones, particularmente cuadrirrotores.
- Revisión de la obtención del modelo dinámico de un cuadrirrotor con controlador interno.
- Revisión de los conceptos del Sistema Operativo Robótico (en inglés Robot Operating System, **ROS**).
- Revisión de los conceptos de Gazebo.
- Validar el modelo dinámico mediante una simulacion del sistema en lazo abierto.
- Diseñar un control de seguimiento de trayectoria para la altura y guiñada.
- Diseñar un esquema de control de movimiento en el plano X-Y, que resuelva el problema de formación, marcha y no colisión.
- Realizar simulaciones numéricas para evaluar el desempeño del esquema de control.
- Discusión de los resultados obtenidos.

1.2. Contribuciones

La principal contribución de este trabajo de tesis es el diseño y validación mediante simulaciones numéricas de leyes de control que permiten:

1. Resolver el problema de control de movimiento (formación y/o marcha) para sistemas multi agente formados por grupos de cuadrirrotores.
2. Modificar las leyes de control mencionadas en el punto anterior para evitar colisiones entre los propios agentes y obstáculos conocidos.
3. Evaluar mediante simulaciones numéricas el desempeño de dichas leyes de control

1.3. Vehículos aéreos no tripulados

Un vehículo aéreo no tripulado, también conocido como UAV (del inglés Unmanned Aerial Vehicle), se define como una aeronave que vuela sin la necesidad de una tripulación. El vuelo de la aeronave puede ser controlado de forma remota por un operador en tierra o de forma autónoma mediante un controlador a bordo.

Los UAV's son comúnmente utilizados para realizar tareas que requieren un alto grado de precisión o que son de alto riesgo para el ser humano. Existen diversos tipos de vehículos aéreos, los cuales se pueden clasificar por la forma en que logran el vuelo.

- **Vehículos aerostáticos.** Se caracterizan por tener un contenedor con algún gas más ligero que el aire. Dicho gas al ser más ligero que el aire produce una fuerza de sustentación que causa que el vehículo flote en el aire. Vehículos de este estilo incluyen al globo de aire caliente, el cual no tiene un sistema de propulsión, y al zeppelin, el cual si lo tiene.
- **Vehículos aerodinámicos.** Se caracterizan por utilizar las fuerzas aerodinámicas generadas por sus alas. Este tipo de vehículos se pueden clasificar basados en movimiento relativo entre el fuselaje del vehículo y sus alas. La clasificación esta dada por vehículos de ala fija y vehículos de ala rotatoria.

  a) Vehículos de ala fija.- Este tipo de vehículos cuentan con uno o más pares de semialas fijadas de forma simétrica al fuselaje del vehículo. Para generar la fuerza de sustentación necesaria para lograr el vuelo, los vehículos de ala fija necesitan generar un movimiento relativo entre el viento y sus alas, dicho movimiento es generado al mover todo el vehículo hacia el frente. Como consecuencia el vehículo se debe de mantener moviéndose hacia el frente con un mínimo de velocidad para mantener un empuje adecuado en las alas. Ejemplos de este tipo de vehículos incluyen aviones y aeroplanos.

  b) Vehículos de ala rotativa.- Este tipo de vehículos cuentan con alas que giran alrededor de su propio eje, comúnmente conocidas como rotores, que generan una fuerza de sustentación suficiente para elevar el vehículo sin la necesidad de moverlo en el plano horizontal. Consecuentemente, los vehículos de ala rotativa son capaces de realizar despegue y aterrizaje vertical, así como vuelo estacionario. Ejemplos de este tipo de vehículos incluyen helicópteros y cuadrirrotores.

Para la realización de este trabajo de tesis se estudian vehículos de ala rotativa de tipo cuadrirrotor.

1.3.1. Cuadrirrotores

Un cuadrirrotor, también conocido como helicóptero de cuatro rotores, es un vehículo aéreo de ala rotativa propulsado por cuatro rotores. A diferencia de un helicóptero

convencional, las hélices de un cuadrirrotor tienen un ángulo de incidencia fijo, lo cual lo hace mecánicamente más simple. Sin embargo, el cuadrirrotor es inherentemente inestable, por lo que es difícil de controlar. Esta inestabilidad debe de ser compensada por el sistema de control del cuadrirrotor.

La estructura mecánica de un cuadrirrotor se compone de cuatro motores fijos en los extremos de una estructura en forma de cruz. Los circuitos electrónicos y la batería del cuadrirrotor comúnmente se encuentran protegidos en el centro de la estructura. El sentido de giro de los motores del cuadrirrotor se elige de forma que un par de motores opuestos giren en sentido horario y el otro par en sentido anti-horario, como se muestra en la Fig. 1.1. Esto es necesario para compensar los pares generados por los motores alrededor del eje vertical del cuadrirrotor y evitar que gire sobre su propio eje.

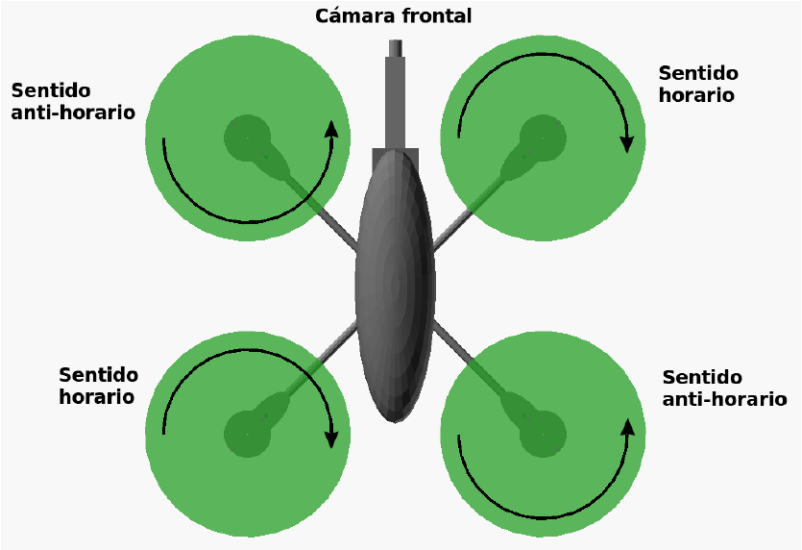


Figura 1.1: Sentido de giro de las hélices de un cuadrirrotor

Existen varias formas de representar la orientación de un cuerpo en el espacio, tales como son los cuaterniones unitarios, la matriz de rotación o los ángulos de Euler. Para este trabajo se ha elegido la representación por medio de los ángulos de Euler de cabeceo, alabeo y guiñada. Considere el marco de referencia fijo al cuerpo del cuadrirrotor mostrado en la Fig. 1.2, sus ángulos de Euler estan definidos como:

- Ángulo de alabeo (ϕ): Ángulo de rotación del cuadrirrotor alrededor del eje X.
- Ángulo de cabeceo (θ): Ángulo de rotación del cuadrirrotor alrededor del eje Y.
- Ángulo de guiñada (ψ): Ángulo de rotación del cuadrirrotor alrededor del eje Z.

Al modificar el empuje generado por los rotores de un cuadrirrotor se pueden modificar sus ángulos de Euler y producir un desplazamiento. Considere un cuadrirrotor como

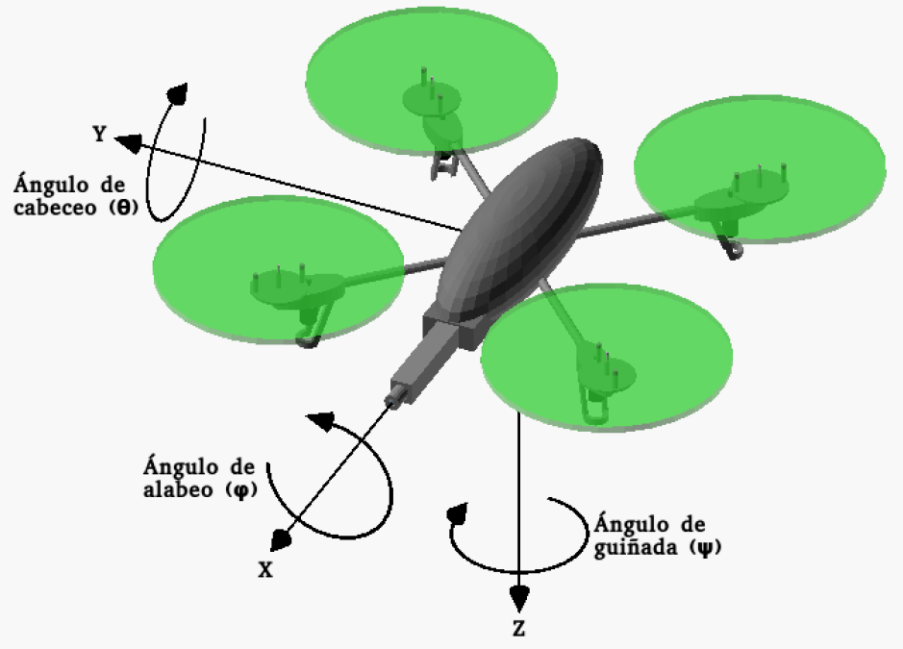
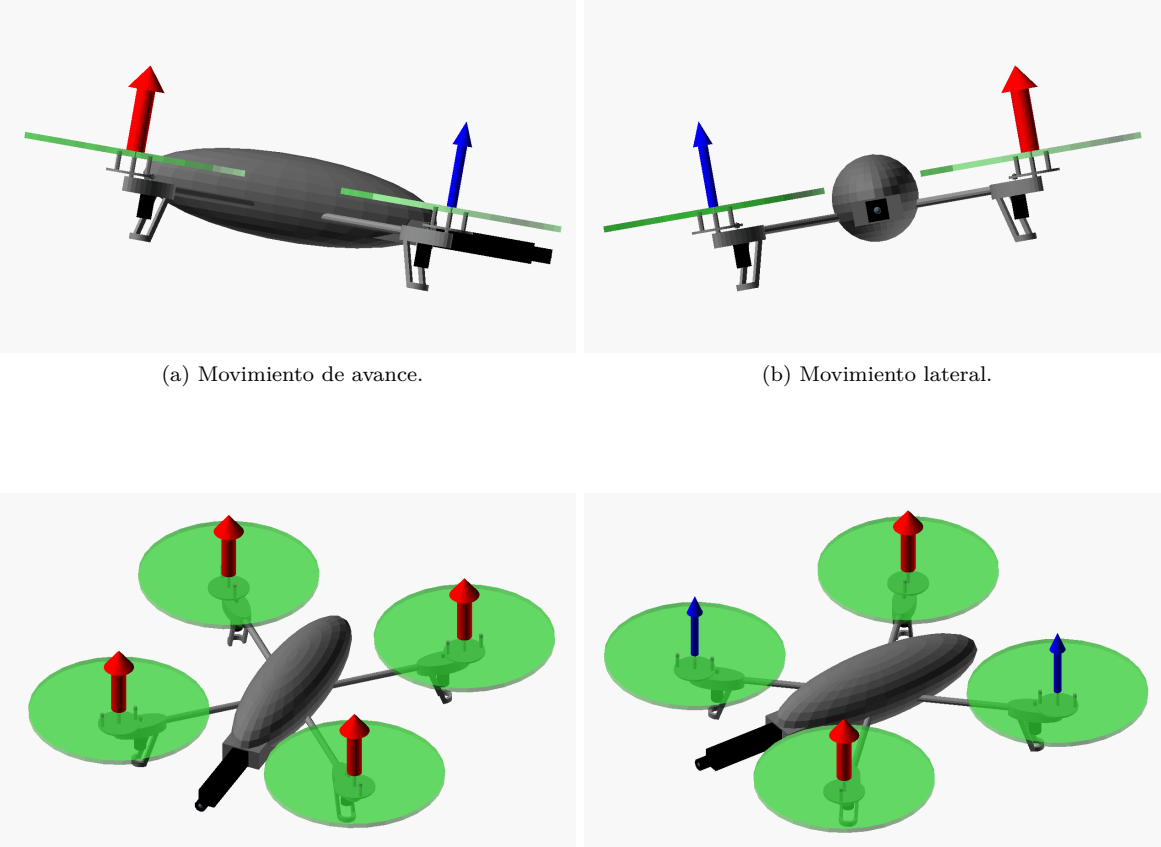


Figura 1.2: Ángulos de Euler de un cuadrirrotor

el que se muestra en la Fig. 1.2 en estado de vuelo estacionario. El cuadrirrotor se puede mover en cualquier dirección espacial al variar el empuje de sus rotores de la siguiente manera:

- Movimiento frontal (eje X): Aumentar/disminuir el empuje generado por los motores frontales o traseros y hacer lo contrario con los otros dos motores. Fig. 1.3a
- Movimiento lateral (eje Y): Aumentar/disminuir el empuje generado por los dos motores derechos o izquierdos y hacer lo contrario con los otros dos motores. Fig. 1.3b
- Movimiento de ascenso-descenso (eje Z): Aumentar/disminuir el empuje generado por los cuatro rotores en la misma proporción. Fig. 1.3c
- Rotación en guiñada: Aumentar/disminuir el empuje generado por uno de los pares de motores que tienen el mismo sentido de giro y hacer lo contrario con el otro par. Fig. 1.3d [1]



(a) Movimiento de avance.

(b) Movimiento lateral.

(c) Movimiento de ascenso.

(d) Rotación en guiñada

Figura 1.3: Movimientos de un cuadrirrotor.

Capítulo 2

Modelado de un cuadrirrotor con controlador interno

Para la obtención del modelo dinámico se considera al cuadrirrotor como un cuerpo rígido que se puede mover en un espacio tridimensional y esta sujeto a una fuerza de sustentación y a tres pares que se conocen comúnmente como los ángulos de Euler de cabeceo, alabeo y guiñada.

Se considera un cuadrirrotor con una configuración tipo-X, que consiste en dos motores frontales y dos motores traseros. Obtenemos el modelo dinámico no lineal del cuadrirrotor por el formalismo de Newton-Euler considerando un marco inercial F_e de tipo Norte-Este-Abajo (NED por sus siglas en inglés) fijado a la tierra y un marco de cuerpo F_b de tipo XYZ fijado al centro de masa del cuadrirrotor como se muestra en la Fig. 2.1. [1]

El vector de posición $\xi = [x, y, z]^T$ representa las coordenadas de la posición del centro de masa del cuadrirrotor con respecto al marco inercial F_e. El vector de orientación $\eta = [\theta, \phi, \psi]^T$ representa la orientación del cuadrirrotor con respecto al marco inercial F_e, donde θ, ϕ y ψ son, respectivamente, los ángulos de Euler de cabeceo, alabeo y guiñada. Las ecuaciones no lineales de movimiento del cuadrirrotor quedan expresadas por,

$$m\ddot{\xi} = -mgD + RF_s \tag{2.1a}$$

$$I\dot{\Omega} = -\Omega \times I_q\Omega + \tau \tag{2.1b}$$

donde m es la masa total del cuadrirrotor, g es la constante de aceleración gravitacional, $D = [0, 0, -1]^T$, $R \in SO(3)$ es una matriz de rotación que relaciona el marco inercial F_e con el marco de cuerpo F_b y esta dada por,

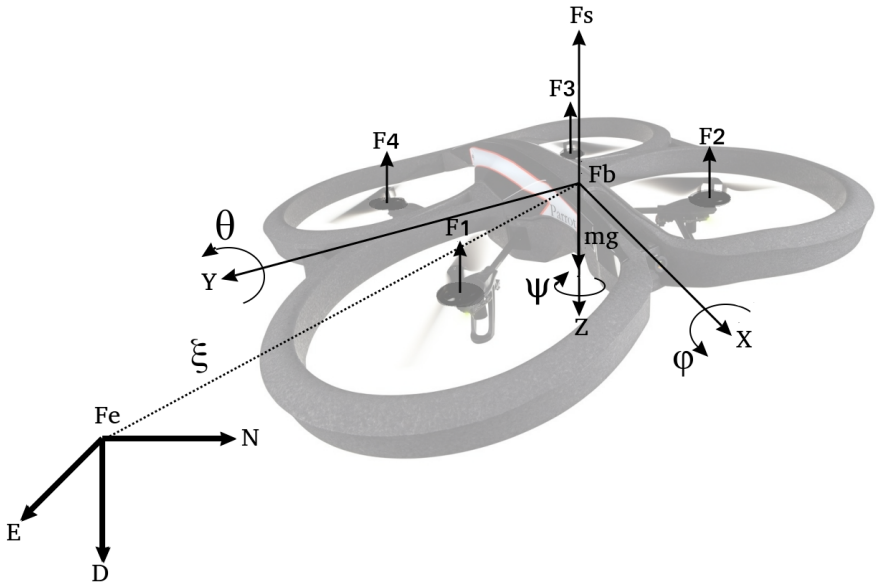


Figura 2.1: Diagrama NED de un cuadrirotor

$$R = \begin{bmatrix} \cos\theta\cos\psi & \sin\theta\sin\phi\cos\psi - \cos\phi\sin\psi & \sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\theta\sin\phi\sin\psi + \cos\phi\cos\psi & \sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \quad (2.2)$$

$F_s = [0, 0, u]^T$, con $u = \sum_{i=1}^{4} F_i$ es la fuerza de sustentación total aplicada al cuadrirrotor por los cuatro motores, I_q es la matriz de inercia del cuadrirrotor, $\Omega = [r, q, p]^T$ es la velocidad angular del cuadrirrotor respecto al marco de cuerpo F_b dada por,

$$\Omega = W\dot{\eta} \quad (2.3a)$$

$$W = \begin{bmatrix} 0 & 1 & -\sin\theta \\ \cos\phi & 0 & \cos\theta\sin\phi \\ -\sin\phi & 0 & \cos\theta\cos\phi \end{bmatrix}, \quad (2.3b)$$

y τ es el par total generalizado. En un cuadrirrotor con configuración tipo-X, los motores M_1 y M_3 giran en sentido horario, mientras que los motores M_2 y M_4 giran en sentido anti-horario. Considerando un cuadrirrotor en estado de vuelo estacionario podemos considerar que el par τ_i generado por cada motor es proporcional a la fuerza de sustentación. Es decir,

$$\tau_i = CF_i$$

donde C es una constante que depende de las características del motor y de las hélices.

Considerando lo anterior, obtenemos el par generalizado $\tau = [\tau_\theta, \tau_\phi, \tau_\psi]^T$ como

$$\begin{bmatrix} \tau_\theta \\ \tau_\phi \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} -l & -l & l & l \\ -l & l & l & -l \\ -C & C & -C & C \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{2.4}$$

Donde l es la distancia entre el centro de masa del cuadrirotor y el centro del motor. Definimos un vector auxiliar $\tilde{\tau} = [\tilde{\tau}_\theta, \tilde{\tau}_\phi, \tilde{\tau}_\psi]^T$ relacionado al par generalizado τ y basado en la ecuación (2.1b)

$$\tilde{\tau} = I^{-1}W^{-1}(-I\dot{W}\dot{\eta} - W\dot{\eta} \times IW\dot{\eta} + \tau) \tag{2.5}$$

Sustituyendo (2.5) en (2.1) podemos representar el modelo dinámico genérico de un cuadrirrotor como,

$$m\ddot{x} = u(\sin\theta\cos\phi\cos\psi + \sin\phi\sin\psi) \tag{2.6a}$$
$$m\ddot{y} = u(\sin\theta\cos\phi\sin\psi - \sin\phi\cos\psi) \tag{2.6b}$$
$$m\ddot{z} = u(\cos\theta\cos\phi) + mg \tag{2.6c}$$
$$\ddot{\theta} = \tilde{\tau}_\theta \tag{2.6d}$$
$$\ddot{\phi} = \tilde{\tau}_\phi \tag{2.6e}$$
$$\ddot{\psi} = \tilde{\tau}_\psi \tag{2.6f}$$

Donde u y los pares $\tilde{\tau}_\theta$, $\tilde{\tau}_\phi$ y $\tilde{\tau}_\psi$ son las entradas de control del sistema.

2.1. Identificación de parámetros del controlador interno

Se considera un cuadrirrotor con un controlador interno que se encarga de estabilizar sus ángulos de Euler de cabeceo y alabeo, su velocidad de rotación en guiñada y su velocidad de ascenso. En [2] los diseñadores del AR.Drone muestran la arquitectura de su controlador interno, sin embargo hay muchos parámetros que no se muestran a detalle. En [1] se hace un análisis experimental del AR.Drone para obtener dichos parámetros. Las ecuaciones diferenciales que se proponen son:

$$\ddot{z} = -a_1\dot{z} + a_3u_z \tag{2.7a}$$
$$\ddot{\theta} = -b_1\dot{\theta} - b_2\theta + b_3u_\theta \tag{2.7b}$$
$$\ddot{\phi} = -c_1\dot{\phi} - c_2\phi + c_3u_\phi \tag{2.7c}$$
$$\ddot{\psi} = -d_1\dot{\psi} + d_3u_\psi \tag{2.7d}$$

donde a_i, b_i, c_i y d_i son los parámetros característicos del sistema que se desea obtener y las entradas de control u_z, $u_\theta(t)$, $u_\phi(t)$ y $u_\psi(t)$ son, respectivamente, la referencia

de velocidad de ascenso/descenso, los ángulos de referencia de cabeceo y alabeo y la referencia de velocidad de rotación en guiñada.

Los datos obtenidos experimentalmente desde los sensores a bordo del cuadrirrotor se analizaron con el comando **tfest** del software MATLAB, con lo que se obtienen las siguientes funciones de transferencia,

$$\frac{Z(s)}{U_z(s)} = \frac{3.384}{s^2 + 3.613s} \tag{2.8a}$$

$$\frac{\Theta(s)}{U_\theta(s)} = \frac{60.40}{s^2 + 4.302s + 28.24} \tag{2.8b}$$

$$\frac{\Phi(s)}{U_\phi(s)} = \frac{-67.30}{s^2 + 6.542s + 25.50} \tag{2.8c}$$

$$\frac{\Psi(s)}{U_\psi(s)} = \frac{3.828}{s^2 + 4.225s} \tag{2.8d}$$

Al realizar la conversión de las funciones de transferencia (2.8) a las ecuaciones diferenciales (2.7), se obtienen los parámetros característicos para el controlador interno del AR.Drone 2.0,

$$\begin{array}{lll} a_1 = 3.613 & a_3 = 3.384 & \\ b_1 = 4.302 & b_2 = 28.24 & b_3 = 60.40 \\ c_1 = 6.542 & c_2 = 25.50 & c_3 = -67.30 \\ d_1 = 4.224 & d_3 = 3.828 & \end{array}$$

Al combinar (2.6) con (2.7), es posible obtener un modelo dinámico que contempla el controlador interno del AR.Drone 2.0. Debido a que las entradas de control del cuadrirrotor seleccionado son diferentes a las de un cuadrirrotor genérico, es necesario modificar las entradas de control del modelo genérico para que coincidan con las de nuestro cuadrirrotor comercial [1]. Se sustituye (2.7a) en (2.6c) para obtener una función para el empuje total u en términos de la entrada de control u_z dada por

$$u = \frac{m(-a_1\dot{z} + a_3 u_z - g)}{\cos\theta\cos\phi} \tag{2.9}$$

Para obtener el modelo dinámico completo del AR.Drone 2.0, se sustituye (2.9) en (2.6a)-(2.6b) y se reemplaza (2.6c)-(2.6f) por (2.7). Finalmente se obtiene el modelo

dinámico no lineal completo para el AR Drone 2.0 dado por

$$\ddot{x} = (-a_1\dot{z} + a_3u_z - g)(\tan\theta\cos\psi + \frac{\tan\phi}{\cos\theta}\sin\psi) \tag{2.10a}$$

$$\ddot{y} = (-a_1\dot{z} + a_3u_z - g)(\tan\theta\sin\psi - \frac{\tan\phi}{\cos\theta}\cos\psi) \tag{2.10b}$$

$$\ddot{z} = -a_1\dot{z} + a_3u_z \tag{2.10c}$$

$$\ddot{\theta} = -b_1\dot{\theta} - b_2\theta + b_3u_\theta \tag{2.10d}$$

$$\ddot{\phi} = -c_1\dot{\phi} - c_2\phi + c_3u_\phi \tag{2.10e}$$

$$\ddot{\psi} = -d_1\dot{\psi} + d_3u_\psi \tag{2.10f}$$

Podemos notar que las entradas de control del AR.Drone 2.0, u_z, u_θ, u_ϕ y u_ψ, ahora controlan a todo el sistema (2.10).

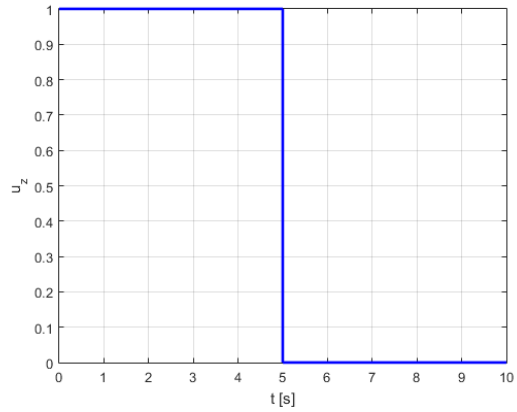
Vale la pena observar que la dinámica de la coordenada de altura z, dada por (2.10c) y la dinámica del ángulo de guiñada ψ dada por (2.10f) están completamente desacopladas de las otras variables de estado. Este hecho nos permitirá diseñar los controladores para z y ψ de manea independiente al controlador del movimiento en el plano horizontal X-Y.

2.2. Simulación del modelo dinámico en lazo abierto

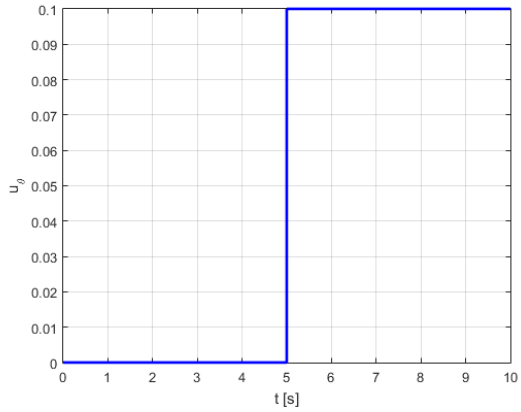
Para validar el modelo dinámico, primero se prueba en lazo abierto para notar si el comportamiento es el esperado. Las señales de control que se le envían al sistema se muestran en la Fig. 2.2.

Se puede notar que las señales de control u_ϕ y u_ψ son cero a propósito, además la señal de control u_z se mantiene activa los primeros 5s y después se desactiva el tiempo restante, la señal de control u_θ permanece inhabilitada los primeros 5s y después se habilita con cierto valor el tiempo restante. El comportamiento del AR.Drone con esas señales de control resulta que en los primeros 5s asciende (desplazamiento sobre el eje Z) a una velocidad constante hasta que u_z se deshabilita y al habilitar u_θ después de los 5s, el ángulo de cabeceo varía y por lo tanto se produce el movimiento de avance (desplazamiento sobre el eje X). Este comportamiento se muestra en la Fig. 2.3, el punto inicial es el punto negro y el punto final es el punto rojo.

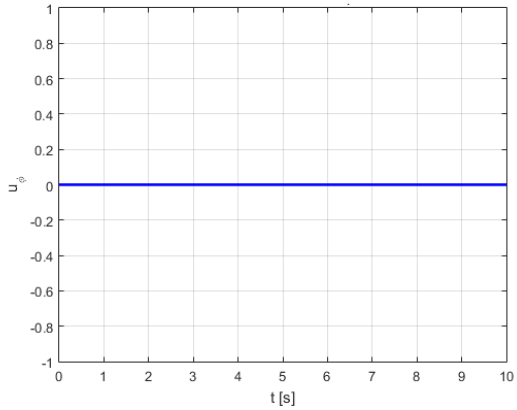
Ahora se le envían al sistema las señales de control mostradas en la Fig. 2.4 y en la Fig. 2.6.



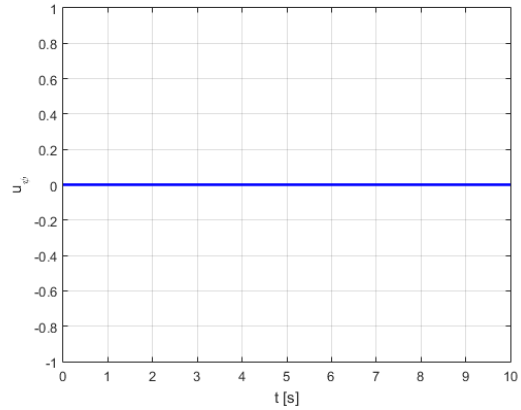
(a) Señal de control u_z.



(b) Señal de control u_θ.



(c) Señal de control u_ϕ.



(d) Señal de control u_ψ

Figura 2.2: Señales de control enviadas al sistema.

La señal de control u_z se mantiene habilitada los primeros 5s, esto genera el movimiento de ascenso del Drone, luego, se desabilita u_z y desde los 5s hasta las 8s, u_ϕ es habilitada, lo que provoca variación en el ángulo de alabeo y produce movimiento lateral. Después de los 8s se desabilita u_ϕ y se habilita u_ψ esto afecta la velocidad de rotación de guiñada del Dron (rotación sobre el eje Z). Este comportamiento se muestra en la Fig. 2.6a, el punto inicial es el punto negro y el punto final es el punto rojo. La rotación en guiñada no se aprecia porque en el punto no es posible apreciar una orientación.

Al ser u_ψ constante, el ángulo de rotación en guiñada siempre incrementará tal y como se muestra en la Fig. 2.6b.

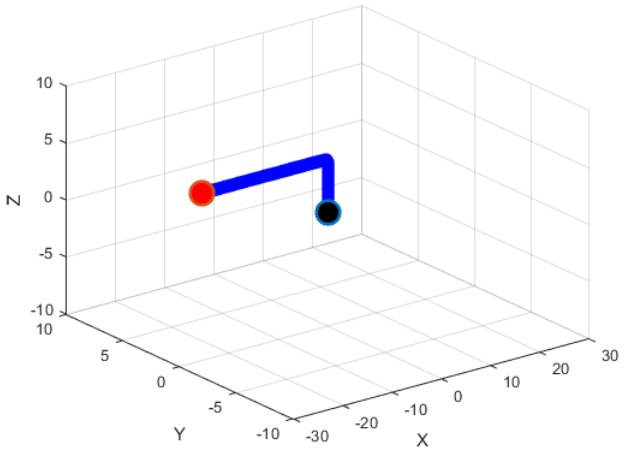
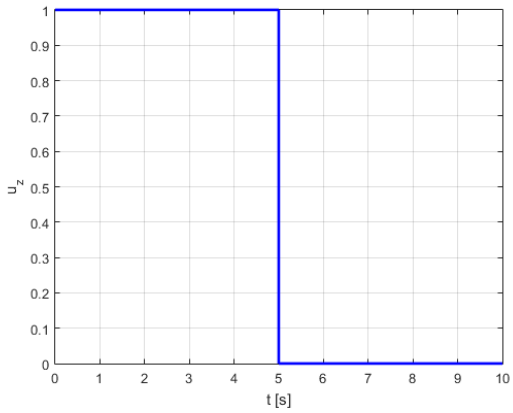
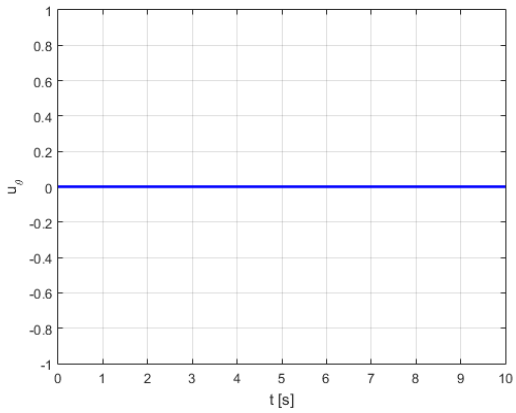


Figura 2.3: Trayectoria Real

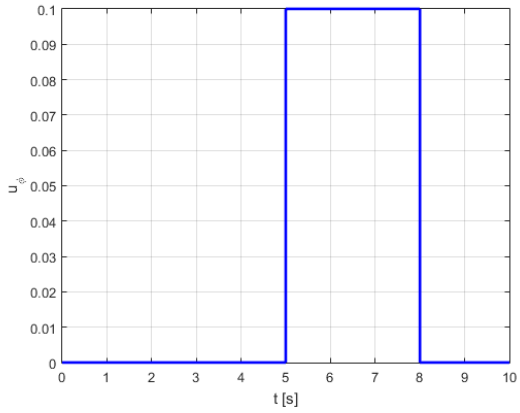


(a) Señal de control u_z.

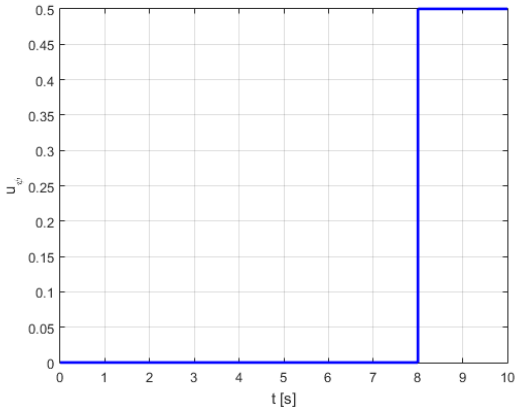


(b) Señal de control u_θ.

Figura 2.4: Señales de control u_z y u_θ.

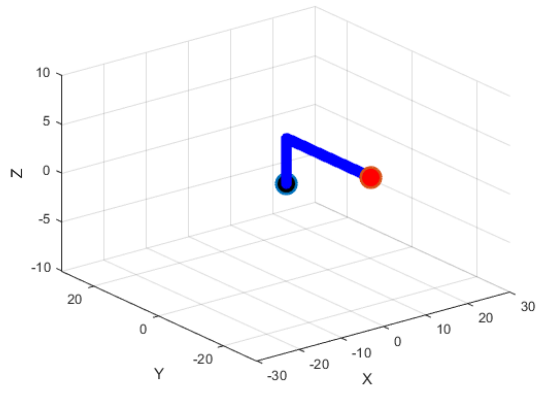


(a) Señal de control u_ϕ.

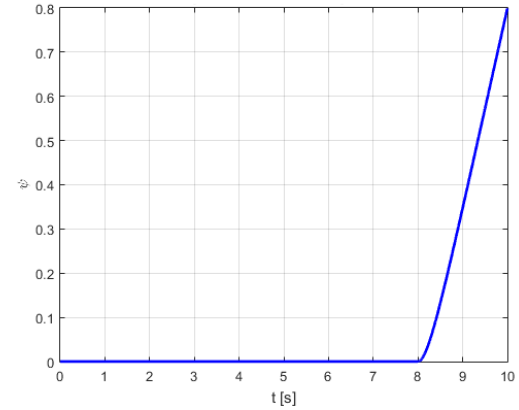


(b) Señal de control u_ψ

Figura 2.5: Señales de control u_ϕ y u_ψ.



(a) Trayectoria Real



(b) Ángulo de guiñada ψ

Figura 2.6: Señales de control u_ϕ y u_ψ.

Capítulo 3

Plataforma experimental

Originalmente, este trabajo de tesis estaba motivado por su implementación y evaluación en una plataforma experimental, desafortunadamente, debido a la pandemia de COVID-19, el acceso a la plataforma fue imposible, por lo que solamente se presentarán resultados en simulación en el software Simulink-Matlab. De cualquier manera, se describe la plataforma experimental, que fue la motivación original de este trabajo.

La plataforma experimental esta conformada por los siguientes componentes principales:

- Sistema de medición de estados
- Sistema de control
- Agentes a controlar

3.1. Sistema de medición de estados

Para obtener el estado de los agentes del sistema (posición y orientación) se utiliza un sistema de visión artificial fabricado por OptiTrack, el cual consiste en 12 cámaras modelo Flex 13, con resolución de 1.3 MP a 120 FPS, como la mostrada en la figura 3.1. Cada cámara cuenta con hardware de procesamiento de imágenes, lo cual permite reducir la carga computacional del procesador. Las cámaras se encuentran organizadas en un arreglo 3×4.

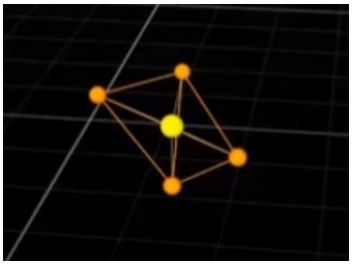
Para obtener la posición espacial de los agentes se utiliza el software Motive: Tracker, diseñado por OptiTrack. Motive: Tracker es capaz de obtener la posición de objetos en su espacio de visión por medio del seguimiento de marcadores reflejantes, como el mostrado en la fig. 3.2a. Dentro del software es necesario definir objetos delimitados por tres marcadores como mínimo. Para evitar ambigüedades en las posiciones angulares de los objetos, es necesario distribuir los marcadores reflectantes de forma que

Figura 3.1: Cámara Flex 13

no se generen figuras simétricas.

También es importante tomar en cuenta que si se definen más de un cuerpo se debe usar arreglos únicos de marcadores, esto para evitar que el software se confunda e intercambie la información. En la figura 3.2b se muestra un objeto definido dentro del software.

(a) Marcadores reflectantes.



(b) Objeto definido en Motive: Tracker

Figura 3.2: Seguimiento de objetos en un sistema OptiTrack

La misma computadora que ejecuta el programa Motive: Tracker ejecuta a la vez un programa de tipo servidor programado en el lenguaje Visual C++. Dicho programa se encarga de crear y actualizar estructuras de datos que incluyen las posiciones de los cuadrirrotores en el espacio de visión de las cámaras y de enviarlas a cualquier cliente que las solicite dentro de una red de tipo ethernet. Tanto las velocidades lineales como las posiciones y velocidades angulares de los cuadrirrotores son obtenidas directamente de los sensores de los cuadrirrotores.

3.2. Sistema de control

Cada uno de los agentes del sistema cuenta con su propia computadora de control. Las computadoras de control cuentan con procesadores i3, o superiores, y con sistema operativo Linux Ubuntu 16.04. Para enviar las leyes de control a cada cuadrirrotor se hace uso de ROS (Robot Operating System) en su version Indigo. ROS es una colección de herramientas, librerías y convenciones de código abierto para el control de diversos tipos de robots. Los programas de control serán escritos en lenguaje C++

y otros en Python.

Adicionalmente, cada computadora de control ejecuta un programa de tipo cliente que se conecta por medio de una red ethernet al servidor que esta siendo ejecutado en la computadora de visión artificial. Este programa obtiene las estructuras de datos correspondientes al cuadrirrotor que se desea controlar y las publica en nodos de ROS de tal forma que sean visibles para el programa que ejecuta las leyes de control. La conexión entre cada computadora de control y el cuadrirrotor que desea controlar es por medio de wi-fi.

3.3. Agentes del sistema

Para este trabajo de tesis se consideran como agentes del sistema cuadrirrotores comerciales fabricados por la compañía francesa Parrot. Se considera el uso de cuadrirrotores modelo AR.Drone 2.0. Se eligieron estos cuadrirrotores debido a su costo accesible y a la gran cantidad de sensores de a bordo con los que cuentan.

El Ar.Drone 2.0, mostrado en la fig. 3.3, es un cuadrirrotor comercial lanzado en 2012. La estructura del AR.Drone 2.0 esta compuesta por tubos de fibra carbono y un núcleo recubierto por unicel para proteger sus circuitos de control. Posee dos cubiertas de unicel intercambiables, una para interiores que protege sus hélices y otra para exteriores que no tiene dicha protección. El AR.Drone tiene un peso de 380 g. al usar la cubierta de exteriores y de 420 g. al usar la cubierta de interiores.

(a) Interiores. (b) Exteriores.

Figura 3.3: AR.Drone 2.0.

El AR.Drone 2.0 cuenta con las siguientes especificaciones técnicas [3]:

- Micro-computadora interna
  - Procesador ARM Cortex A8 de 32 bits a 1 GHz.
  - 1GB de memoria DDR2 RAM a 200MHz.
  - Sistema operativo Linux Busybox 2.6.32.
  - 1 puerto USB 2.0.
  - Wi-Fi 802.11b/g/n.

- Sensores
  - Giroscopio de 3 ejes con precisión de $2,000^o/s$.
  - Acelerómetro de 3 ejes con precisión de ± 50 mg.
  - Magnetómetro de 3 ejes con precisión de $\pm 6^o$.
  - Barómetro con precisión de ± 10 Pa.
  - Sensor ultrasónico de altura.
  - Cámara vertical QVGA a 60 FPS orientada hacia el suelo.
  - Cámara frontal 720p HD a 30 fps.

- Motores
  - 4 motores sin escobillas de 28,500 rev/min a 14.5 w.
  - Engranes de nilón.
  - Bujes de bronce.

El puerto USB del AR.Drone 2.0 puede ser utilizado para conectar memorias para guardar vídeos o para conectar un modulo GPS, el cual se vende por separado. El AR.Drone es energizado por una batería de polímero de litio de 11.1 v, 1500 mAh.

3.4. Sistema Operativo Robótico (ROS)

ROS (en inglés Robot Operating System) [4], [5] es un meta-sistema operativo robótico que provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes. Se basa en una arquitectura de grafos donde el procesamiento toma lugar en los nodos que pueden recibir, mandar y multiplexar mensajes de sensores, control, estados, planificaciones y

actuadores, entre otros. La librería está orientada para un sistema UNIX (Ubuntu (Linux)). La diferencia principal entre un sistema operativo y un meta-sistema operativo es que el meta-sistema operativo se construye en el nivel alto del sistema operativo y permite diferentes procesos (nodos) que se comunican entre sí en tiempo de ejecución.

ROS tiene dos partes básicas: la parte del sistema operativo, ROS, como se ha descrito anteriormente y ros-pkg, una suite de paquetes aportados por la comunidad de usuarios (organizados en conjuntos llamados pilas o en inglés stacks). ROS es software libre bajo términos de licencia BSD (Berkeley Software Distribution). Esta licencia permite libertad para uso comercial y de investigador. Las contribuciones de los paquetes en ros-pkg están bajo una gran variedad de licencias diferentes.

Se elige trabajar con ROS ya que cuenta con las librerías necesarias para recibir y procesar los datos que transmite el Optitrack. Además, es posible programar nodos en distintos lenguajes de programación y aún así intercambiar información entre esos nodos.

3.5. Gazebo

Gazebo, que nace con el nombre de Gazebo Project, es un simulador 3D, cinemático, dinámico y multi-robot que permite realizar simulaciones de robots en entornos complejos, interiores o exteriores, realistas y tridimensionales [6]. Es una buena herramienta para simulaciones que se desean probar en una plataforma experimental, ya que se programa en ROS y se utilizan los comandos de las entradas de control del agente.

Entre sus características se pueden nombrar las siguientes:

- Gazebo es un software libre financiado, en parte, por Willow Garage, pudiendo ser reconfigurado, ampliado y modificado.
- Compatible con ROS y Player. Podemos ejecutar Gazebo desde ROS y utilizar las APIs de este último para controlar los robots en las simulaciones, es decir, enviar y recibir datos de éstas.
- Simulación realista de la física de los cuerpos rígidos. Los robots pueden interactuar con el mundo (pueden coger y empujar cosas, rodar y deslizarse por el suelo) y viceversa (les afecta la gravedad y pueden colisionar con obstáculos del mundo).
- Capacidad de desarrollar y simular modelos de robots propios (URDF) e incluso cargarlos en tiempo de ejecución.

- Posibilidad de crear escenarios (mundos) de simulación, variando las características de los contactos con el suelo, los obstáculos e incluso los valores de la gravedad en las tres dimensiones. También es posible variar las características de contacto de cada agente individualmente.
- Contiene diversos plugins para añadir sensores al modelo del robot y simularlos, como sensores de odometría (GPS e IMU), de fuerza, de contacto, lásers y cámaras estéreo.

En cuanto a su integración en ROS, el simulador posee su propio stack denominado ***simulator_gazebo***, que se divide en una serie de paquetes con diferentes funcionalidades:

- *gazebo*: Contiene la versión más reciente de Gazebo Project integrada en ROS, de forma que es ejecutable como un nodo más de ROS, contiene sus propios topicos y servicios.
- *gazebo_msgs*: Contiene los tipos de servicios (srv) y mensajes (msg) para interactuar con Gazebo desde ROS. Por tanto los tipos de mensajes y servicios a utilizar para enviar o recibir del simulador deben ser incluidos a partir de este paquete.
- *gazebo_plugins*: Contiene los plugins para utilizar los diferentes sensores.
- *gazebo_tools*: Contiene, entre otras herramientas, la que nos permiten enviar o eliminar modelos URDF en el simulador, mediante el nodo gazebo_model, del mismo paquete.
- *gazebo_worlds*: Contiene los archivos de configuración estándares del entorno (world) desarrollados por Willow Garage, así como diferentes modelos URDF de objetos (muros, mesas, sillas,etc) a modo de ejemplo.

En la Fig. 3.4 se muestra el entorno de simulación para el AR.Drone.

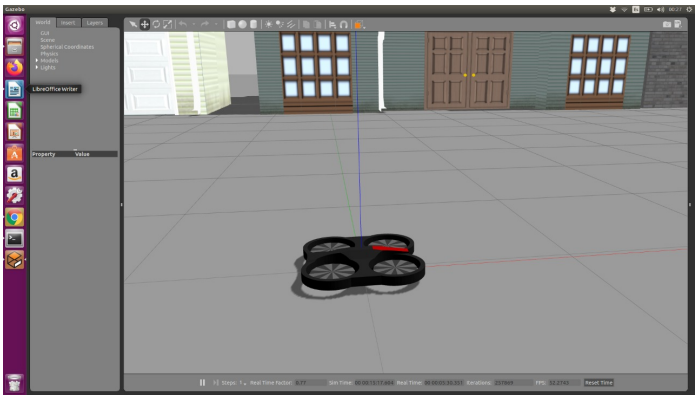


Figura 3.4: Entorno de simulación para AR.Drone

Capítulo 4

Control de altura y control de guiñada

En este capítulo se diseña un control de altura y un control de guiñada para seguimiento de trayectorias sobre el eje Z y el ángulo de guiñada ψ. A partir del modelo completo del AR.Drone 2.0, observamos que las ecuaciones (2.10c) y (2.10f), correspondientes a las dinámicas de la altura z y del ángulo de guiñada ψ respectivamente, están completamente desacopladas de las otras variables de estado. Por ello, es posible diseñar estas leyes de control independientemente del movimiento en el plano horizontal.

4.1. Control de altura

La dinámica de la altura está dada por:

$$\ddot{z} = -a_1 \dot{z} + a_3 u_z$$

Definiendo los estados:

$$z_1 = z, \qquad z_2 = \dot{z}$$

La dinámica en el espacio de estados resulta:

$$\begin{aligned} \dot{z}_1 &= z_2 \\ \dot{z}_2 &= -a_1 z_2 + a_3 u_z \end{aligned}$$

Definiendo los errores de seguimiento de trayectoria en z_1 y z_2,

$$\begin{aligned} e_{z_1} &= z_1 - z_1^d \\ e_{z_2} &= z_2 - \dot{z}_1^d \end{aligned}$$

El sistema en las coordenadas del error es,

$$\dot{e}_{z_1} = e_{z_2} \tag{4.1a}$$
$$\dot{e}_{z_2} = -a_1 z_2 + a_3 u_z - \ddot{z}_1^d \tag{4.1b}$$

Para estabilizar el sistema, primero se propone una ley de control auxiliar dada por:

$$u_z = \frac{1}{a_3}[\ddot{z}_1^d + a_1 z_2 + w_z] \tag{4.2}$$

Al realimentar (4.2) con (4.1), el sistema se reescribe como:

$$\dot{e}_{z_1} = e_{z_2} \tag{4.3a}$$
$$\dot{e}_{z_2} = w_z \tag{4.3b}$$

Para estabilizar el sistema (4.3) se propone la ley de control,

$$w_z = -k_{pz} e_{z_1} - k_{dz} e_{z_2}$$

Con k_{pz}, $k_{dz} > 0$. Finalmente se obtiene,

$$u_z = \frac{1}{a_3}[\ddot{z}_1^d + a_1 z_2 - k_{pz} e_{z_1} - k_{dz} e_{z_2}] \tag{4.4}$$

Ahora, se considera el sistema (4.1) en lazo cerrado con la ley de control (4.4), el sistema en lazo cerrado resulta,

$$\dot{e}_{z_1} = e_{z_2} \tag{4.5a}$$
$$\dot{e}_{z_2} = -k_{pz} e_{z_1} - k_{dz} e_{z_2} \tag{4.5b}$$

el cual puede ser expresado en forma matricial como

$$\begin{bmatrix} \dot{e}_{z_1} \\ \dot{e}_{z_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{pz} & -k_{dz} \end{bmatrix} \begin{bmatrix} e_{z_1} \\ e_{z_2} \end{bmatrix} = H_z \begin{bmatrix} e_{z_1} \\ e_{z_2} \end{bmatrix} \tag{4.6}$$

Debido a que la matriz H_z es Hurtwitz, el sistema en lazo cerrado es exponencialmente estable.

4.2. Control de guiñada

La dinámica del ángulo de guiñada está dada por:

$$\ddot{\psi} = -d_1 \dot{\psi} + d_3 u_\psi$$

Definiendo los estados:

$$\psi_1 = \psi, \qquad \psi_2 = \dot{\psi}$$

La dinámica en el espacio de estados resulta:

$$\dot{\psi}_1 = \psi_2$$
$$\dot{\psi}_2 = -d_1\psi_2 + d_3 u_\psi$$

Definiendo los errores de seguimiento de trayectoria en ψ_1 y ψ_2,

$$e_{\psi_1} = \psi_1 - \psi_1^d$$
$$e_{\psi_2} = \psi_2 - \dot{\psi}_1^d$$

El sistema en las coordenadas del error es,

$$\dot{e}_{\psi_1} = e_{\psi_2} \tag{4.7a}$$
$$\dot{e}_{\psi_2} = -d_1\psi_2 + d_3 u_\psi - \ddot{\psi}_1^d \tag{4.7b}$$

Para estabilizar el sistema (4.7), primero se propone una ley de control auxiliar dada por:

$$u_\psi = \frac{1}{d_3}[\ddot{\psi}_1^d + d_1\psi_2 + w_\psi] \tag{4.8}$$

Al realimentar (4.8) con (4.7), el sistema se reescribe como:

$$\dot{e}_{\psi_1} = e_{\psi_2} \tag{4.9a}$$
$$\dot{e}_{\psi_2} = w_\psi \tag{4.9b}$$

Para estabilizar el sistema (4.9) se propone la ley de control,

$$w_\psi = -k_{p\psi} e_{\psi_1} - k_{d\psi} e_{\psi_2}$$

Con $k_{p\psi}$, $k_{d\psi} > 0$. Finalmente se obtiene,

$$u_\psi = \frac{1}{d_3}[\ddot{\psi}_1^d + d_1\psi_2 - k_{p\psi} e_{\psi_1} - k_{d\psi} e_{\psi_2}] \tag{4.10}$$

Ahora, se considera el sistema (4.7) en lazo cerrado con la ley de control (4.10), el sistema en lazo cerrado resulta,

$$\dot{e}_{\psi_1} = e_{\psi_2} \tag{4.11a}$$
$$\dot{e}_{\psi_2} = -k_{p\psi} e_{\psi_1} - k_{d\psi} e_{\psi_2} \tag{4.11b}$$

el cual puede ser expresado en forma matricial como

$$\begin{bmatrix} \dot{e}_{\psi_1} \\ \dot{e}_{\psi_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\psi} & -k_{d\psi} \end{bmatrix} \begin{bmatrix} e_{\psi_1} \\ e_{\psi_2} \end{bmatrix} = H_\psi \begin{bmatrix} e_{\psi_1} \\ e_{\psi_2} \end{bmatrix} \tag{4.12}$$

Debido a que la matriz H_ψ es Hurtwitz, el sistema en lazo cerrado es exponencialmente estable.

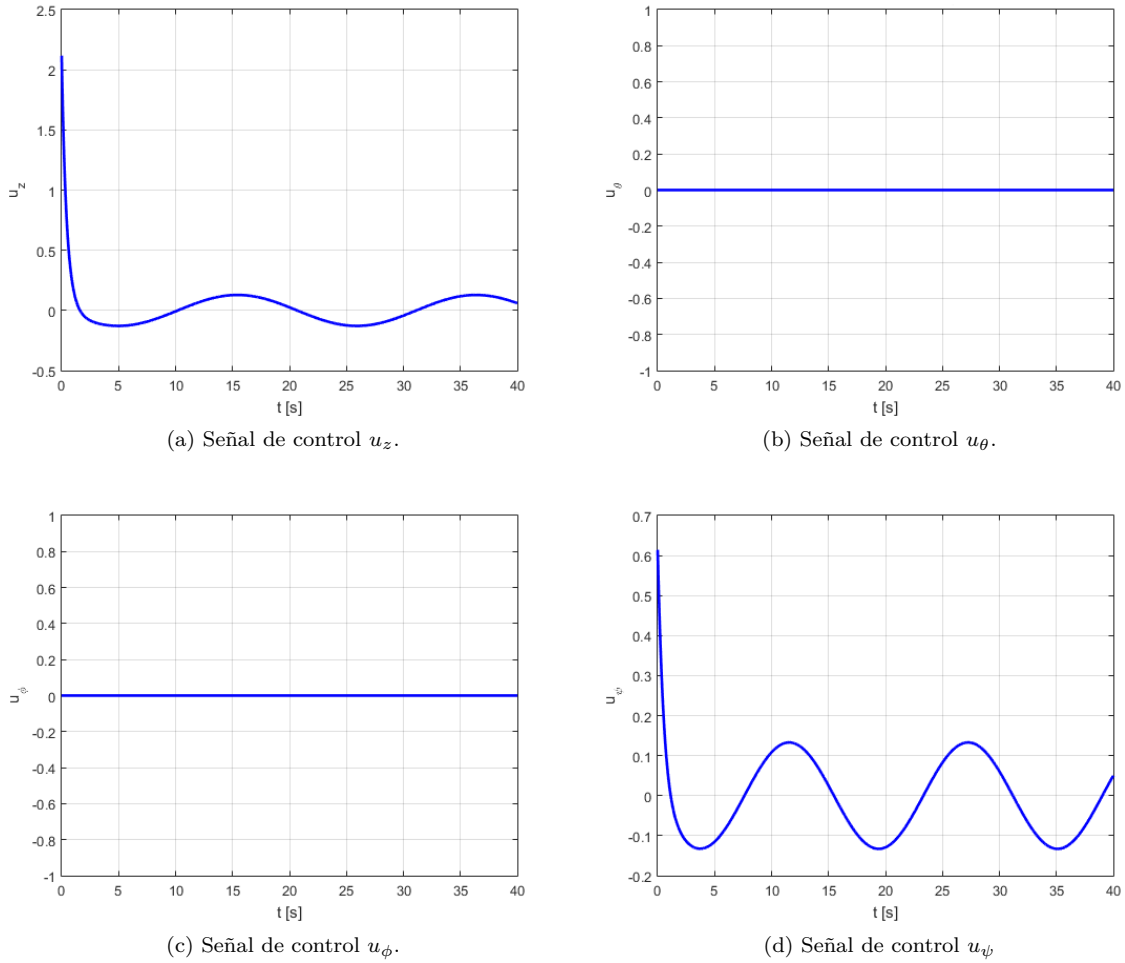
4.3. Simulación de los controles propuestos

Se proponen las siguientes trayectorias deseadas,

$$z_d = 1.5 + 0.4\cos(0.3t) \tag{4.13a}$$
$$\psi_d = 0.3\cos(0.4t) \tag{4.13b}$$

Las condiciones iniciales para la simulación son: $z = 1$ y $\psi = 0$. Las ganancias: $k_p = 8$ y $k_d = 6$.

El control actúa para realizar el seguimiento de trayectoria y genera las señales de control mostradas en la figura 4.1. Se recuerda que la dinámica tanto de z y ψ están desacopladas y no dependen de las señales de control u_θ y u_ϕ. Debido a eso, no son necesarias las señales de control u_θ y u_ϕ, y se observa en las figuras 4.1b y 4.1c, que en efecto están inactivas y a pesar de eso, se logra el seguimiento en altura y en guiñada de manera satisfactoria.



(a) Señal de control u_z.

(b) Señal de control u_θ.

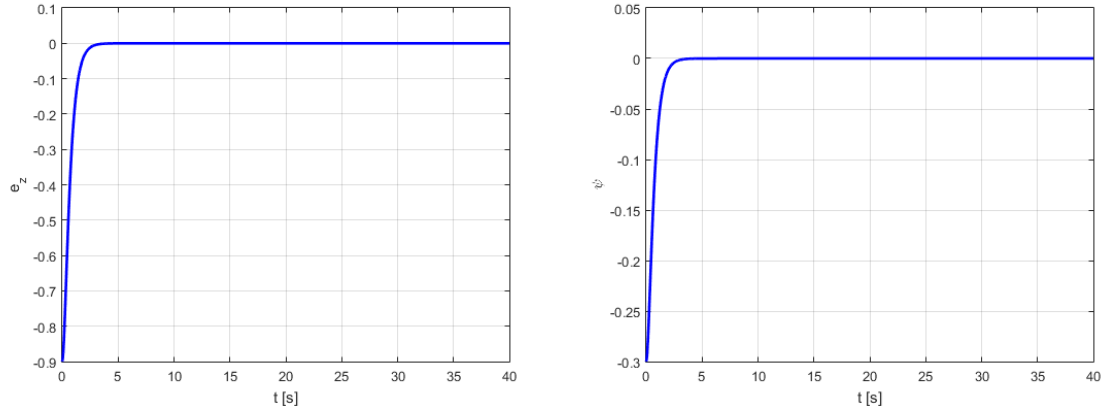
(c) Señal de control u_ϕ.

(d) Señal de control u_ψ

Figura 4.1: Señales de control generadas.

En la Fig. 4.1 se muestran las señales enviadas al sistema para lograr la trayectoria deseada, el valor que toman las señales de control son apropiados y aceptables. En la Fig. 4.2 se muestran los errores de seguimiento en z y ψ, se nota que el error

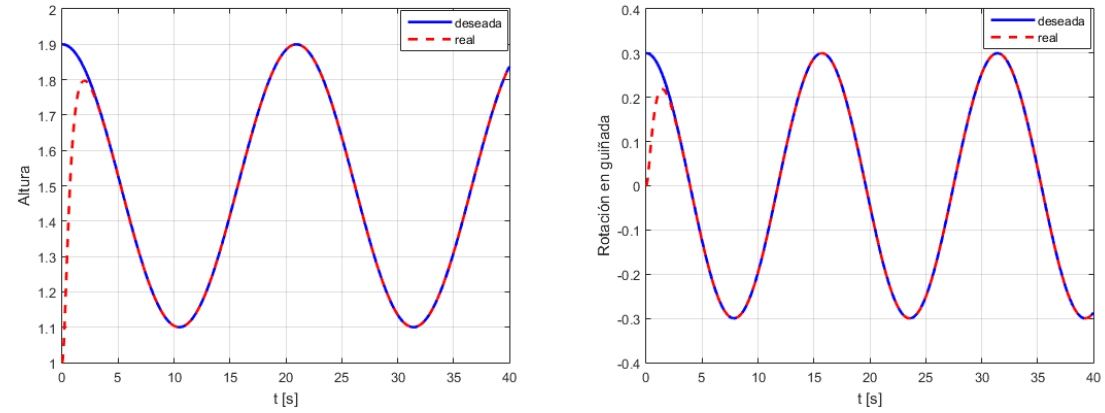
converge a cero, el seguimiento de trayectoria para altura y guiñada es bueno para la implementación en el mundo físico.



(a) Error en z. (b) Error en ψ.

Figura 4.2: Errores de seguimiento en z y ψ.

También se muestran las gráficas de las trayectorias deseadas contra las reales para cada caso; para el control de altura y para el control de rotación en guiñada. En la Fig. 4.3 se muestra el seguimiento de trayectoria, la línea azul es la trayectoria deseada y la línea roja es la trayectoria real del AR.Drone 2.0.



(a) Trayectoria de z. (b) Trayectoria de ψ.

Figura 4.3: Trayectorias de z y ψ.

Las leyes de control tienen buen desempeño en seguimiento de trayectoria. Ahora que se tiene el control de altura y guiñada, es momento de desarrollar un esquema de control para movimiento en el plano X-Y, que tenga la capacidad de evitar colisiones en su entorno con obstáculos conocidos o con algún otro agente dentro del sistema.

Capítulo 5

Control de movimiento en el plano XY con evasión de colisiones

En este capítulo abordamos el problema de control de movimiento en el plano horizontal (X-Y) con evasión de colisiones. La ley de control que proponemos consta de dos partes: (i) una parte que resuelve el control de movimiento, basada en el algoritmo fundamental de consenso para agentes de segundo orden y (ii) una parte reactiva, basada en campos vectoriales tipo foco repulsivo que evitan que los agentes se encuentren a menos de una cierta distancia predeterminada.

Recordemos que la dinámica en el plano X-Y está dada por:

$$\ddot{x} = f_1 = \Gamma_z(\tan\theta\cos\psi + \frac{\tan\phi}{\cos\theta}\sin\psi) \tag{5.1a}$$

$$\ddot{y} = f_2 = \Gamma_z(\tan\theta\sin\psi - \frac{\tan\phi}{\cos\theta}\cos\psi) \tag{5.1b}$$

Con $\Gamma_z = (-a_1\dot{z} + a_3 u_z - g)$

Las ecuaciones (5.1) muestran claramente la alta no linealidad presente en la dinámica del movimiento en el plano horizontal. Sin embargo, el cuadrirrotor es capaz de generar trayectorias muy agresivas con pequeñas variaciones en los ángulos de alabeo y cabeceo. Típicamente, dichos angulos son menores a 3 o 4 grados sexagesimales. Por otra parte, el ángulo de guiñada puede variar arbitrariamente. Las consideraciones anteriores nos permiten considerar una aproximación de (5.1) con θ y ϕ alrededor de cero, pero conservando las no linealidades debidas a ψ.

Linealizando la dinámica alrededor del punto de equilibrio $\theta, \phi = 0$ resulta,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \left. \begin{bmatrix} \dfrac{\partial f_1}{\partial \theta} & \dfrac{\partial f_1}{\partial \phi} \\ \dfrac{\partial f_2}{\partial \theta} & \dfrac{\partial f_2}{\partial \phi} \end{bmatrix} \right|_{\theta,\phi=0} \begin{bmatrix} \theta \\ \phi \end{bmatrix}$$

Entonces,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \underbrace{\Gamma_z \begin{bmatrix} \cos\psi & \sin\psi \\ \sin\psi & -\cos\psi \end{bmatrix}}_{\tilde{A}(\psi)} \begin{bmatrix} \theta \\ \phi \end{bmatrix}$$

En una primera etapa se proponen como entradas de control virtuales a las variables θ y ϕ para lograr el comportamiento deseado. Específicamente, proponemos

$$\begin{bmatrix} \theta_i \\ \phi_i \end{bmatrix} = \tilde{A}_i(\psi)^{-1} u_i = \tilde{A}_i(\psi)^{-1}(\gamma_i + \beta_i) \tag{5.2}$$

donde el subíndice i se refiere al número de agente y definimos:

$$u_i = \underbrace{-k_0 \sum_{j=1}^{N_i} \widetilde{X}_i - k_1 \sum_{j=1}^{N_i} \widetilde{\upsilon}_i}_{\gamma_i} + \underbrace{\varepsilon \sum_{j \in M_i} \delta_{ij} \begin{bmatrix} (x_i - x_j) - (y_i - y_j) \\ (x_i - x_j) + (y_i - y_j) \end{bmatrix}}_{\beta_i} \tag{5.3}$$

Donde:

$$X_i = [x_i, y_i]^T, \qquad \upsilon_i = \dot{X}_i$$

$$\begin{aligned} \widetilde{X}_i &= X_i - X_i^* \\ \widetilde{\upsilon}_i &= \upsilon_i - \upsilon_i^* \\ u_i &= \dot{\upsilon}_i \\ N_i &: \text{conjunto de vecinos del nodo } i \end{aligned}$$

$$M_i = \{X_j | \, \|X_i(t) - X_j(t)\| \leq d\} \tag{5.4}$$

$$\delta_{ij} = \begin{cases} 1 & \text{si } \|X_i(t) - X_j(t)\| \leq d \\ 0 & \text{en otro caso} \end{cases} \tag{5.5}$$

d : umbral de conmutación para aplicar la ley de control reactiva.

ε : aceleración opuesta requerida para evitar la colisión.

5.1. Primera etapa de control para dos agentes

Se tomará el caso más simple, es decir, $N_1 = \{2\}$, $N_2 = \{1\}$. El gráfico de comunicación de los dos agentes se muestra en la figura 5.1.

Figura 5.1: Gráfico de comunicación para dos agentes.

Del gráfico de comunicación se tienen los errores de posición de cada agente,

$$\widetilde{X}_1 = X_1 - X_2 - C_{21}$$

$$\widetilde{X}_2 = X_2 - X_1 - C_{12}$$

Asumiendo que C_{ij} son constantes, los errores de velocidad están dados por:

$$\widetilde{v}_1 = \dot{\widetilde{X}}_1 = \dot{X}_1 - \dot{X}_2$$

$$\widetilde{v}_2 = \dot{\widetilde{X}}_2 = \dot{X}_2 - \dot{X}_1$$

De lo anterior, se puede escribir la forma de u_i, lo que resulta,

$$u_1 = \ddot{X}_1 = -k_0(X_1 - X_2 - C_{21}) - k_1(\dot{X}_1 - \dot{X}_2) + \varepsilon\delta_{12}\begin{bmatrix}(x_1 - x_2) - (y_1 - y_2)\\(x_1 - x_2) + (y_1 - y_2)\end{bmatrix}$$

$$u_2 = \ddot{X}_2 = -k_0(X_2 - X_1 - C_{12}) - k_1(\dot{X}_2 - \dot{X}_1) + \varepsilon\delta_{21}\begin{bmatrix}(x_2 - x_1) - (y_2 - y_1)\\(x_2 - x_1) + (y_2 - y_1)\end{bmatrix}$$

Ahora ya se tiene una ley de control asumiendo que las entradas de control son θ y ϕ para cada cuadrirrotor. Se mostrarán los resultados en simulación del desempeño de esta ley de control obtenida.

Figura 5.2: Formación deseada.

5.1.1. Simulación de la primera etapa de control con dos agentes

Ahora, se muestran los resultados de la simulación de nuestros agentes, para la simulación se utilizó una formación muy simple, la cual solo hace que un agente esté a la izquierda del otro en línea recta a una distancia de $4[m]$ como se muestra en la figura 5.2.

Para la simulación se utilizaron los siguientes parámetros:

$$x_0 = [-4, 4] \qquad y_0 = [0, 0]$$

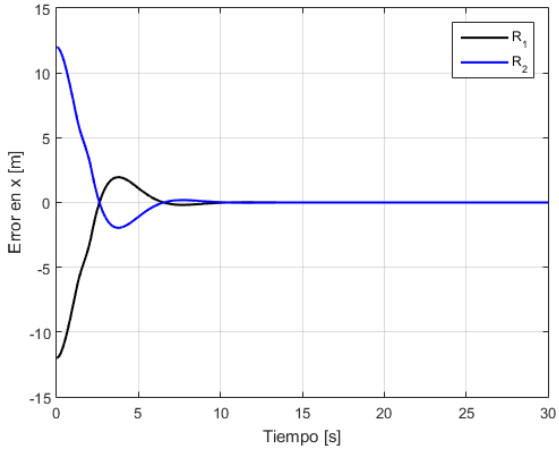
$$C_{21} = [4, 0] \qquad C_{12} = [-4, 0]$$

Con ganancias $k_0 = 0.5$, $k_1 = 0.6$ y los parámetros $\varepsilon = 2.1$ y $d = 0.9$.

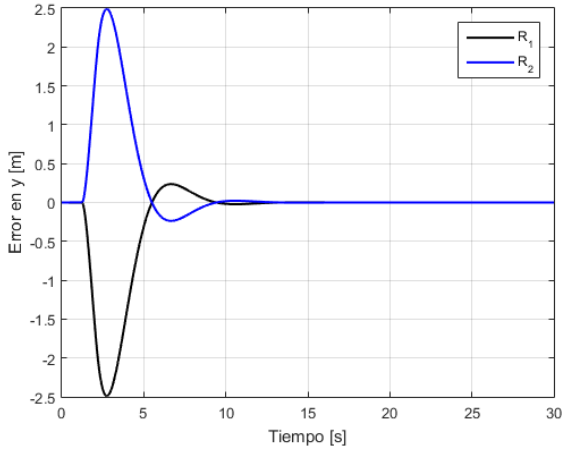
Las trayectorias de seguimiento deseadas para altura y guiñada son:

$$z_{id} = 1 + 0.4\cos(2\pi t)/45 \qquad \psi_{id} = \sin(2\pi t)/45 \qquad i = 1, 2$$

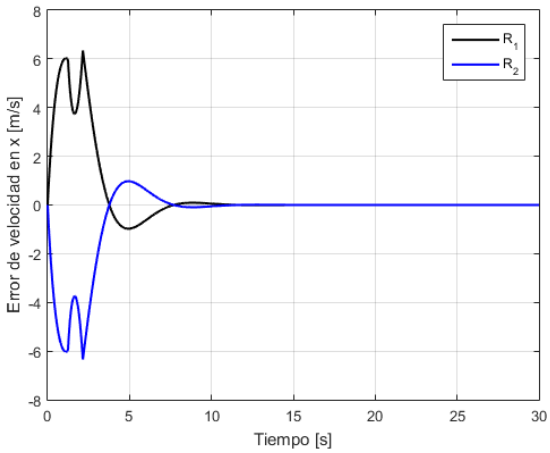
Se propusieron las trayectorias deseadas de altura y guiñada de la misma forma para ambos agentes. Ahora se muestran los resultados obtenidos.



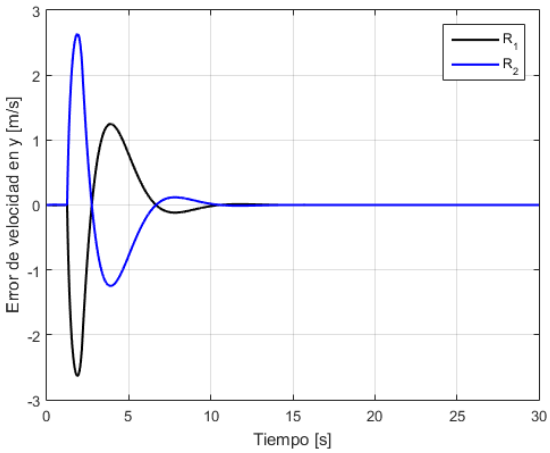
(a) Error de posición de x



(b) Error de posición de y



(c) Error de velocidad de x



(d) Error de velocidad de y

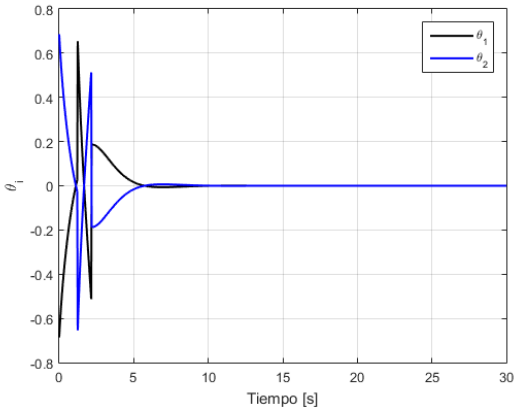
Figura 5.3: Errores de posición y velocidad.

En la figura 5.3 se observan los errores de posición y de velocidad en X y Y. Se puede notar como convergen a cero, esto quiere decir que los agentes llegan a la formación deseada en el plano X-Y sin importar que hagan seguimiento de trayectoria en altura y guiñada. Los cambios más abruptos en la velocidad ocurren al momento de evitar la colisión.

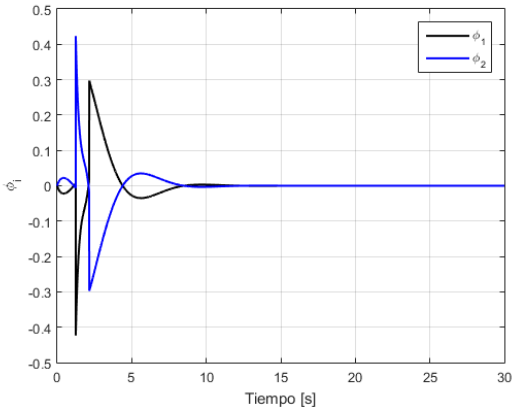
Las señales de control en esta fase fueron θ_i y ϕ_i y se muestran en la figura 5.4. Son algo bruscos los cambios repentinos en los ángulos pero son aceptables los ángulos que toma el sistema para lograr la evasión de colisión.

En la figura 5.5 se puede observar la distancia entre los agentes y se muestra la trayectoria generada por cada agente en el plano X-Y. Se puede ver que si se logra el efecto de evasión de colisión. En la figura 5.6 se muestra la trayectoria en el espacio

que generan los agentes.

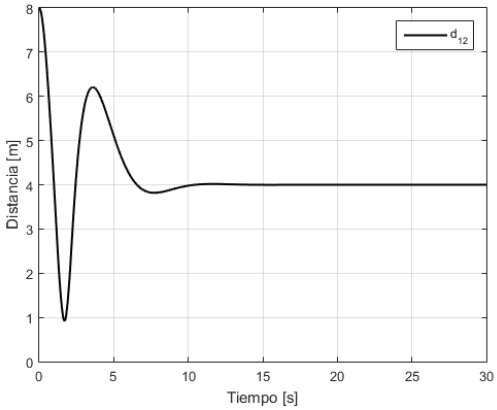


(a) Señal de control para x

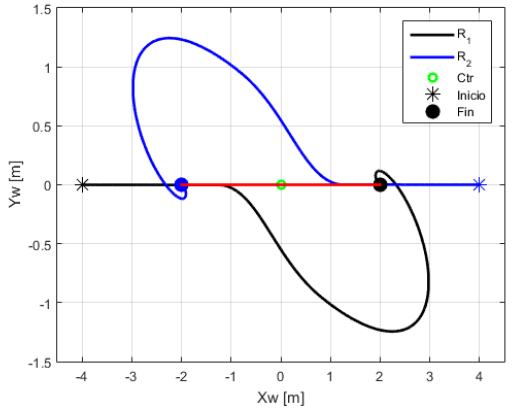


(b) Señal de control para y

Figura 5.4: Señales de control para X y Y.



(a) Distancia entre agentes



(b) Trayectoria generada en X-Y

Figura 5.5: Trayectorias y distancia de los agentes.

Ahora que se ha logrado la primera etapa de control, se procede a realizar la siguiente etapa. Se logró la evasión de colisión tomando como entradas de control los ángulos θ y ϕ pero debemos recordar que la referencia para esos ángulos se controlan mediante las entradas u_θ y u_ϕ, entonces, en esta etapa se desarrolla una ley de control, tal que, u_θ y u_ϕ converjan a θ y ϕ que se obtuvieron en la etapa uno. Para esto, llamaremos a θ y ϕ de la etapa uno como θ^* y ϕ^* respectivamente.

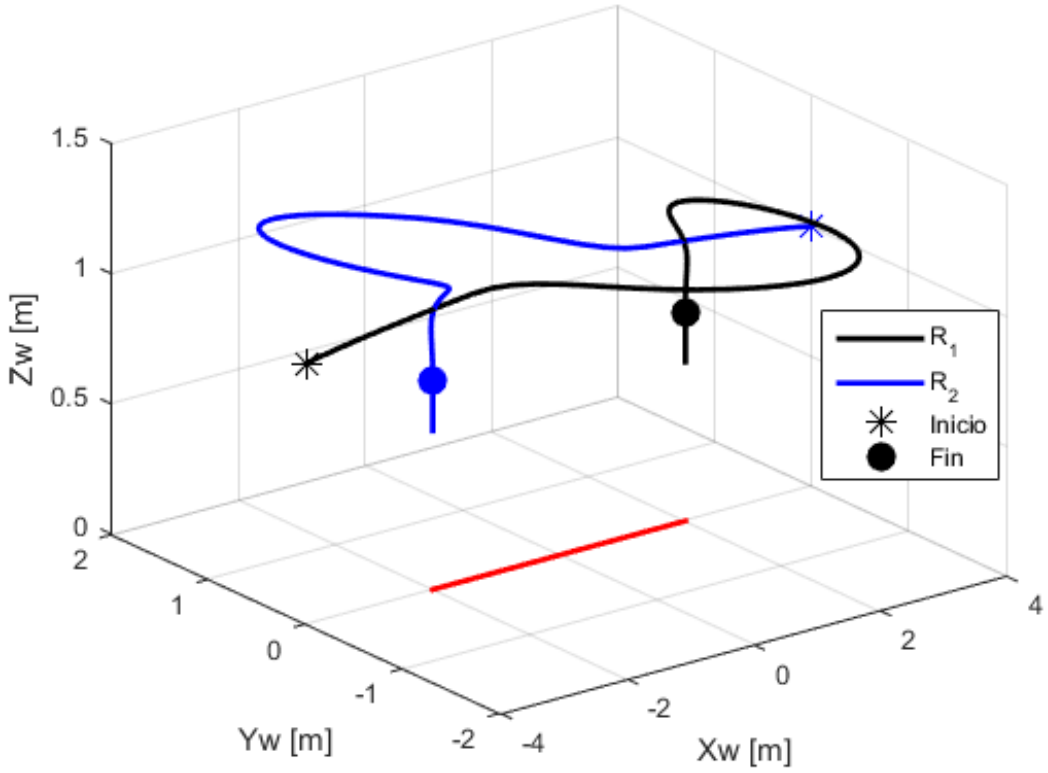


Figura 5.6: Trayectorias en XYZ.

5.2. Segunda etapa de control para dos agentes

5.2.1. Control para u_θ

La dinámica del ángulo de cabeceo está dada por:

$$\ddot{\theta} = -b_1\dot{\theta} - b_2\theta + b_3 u_\theta$$

Definiendo los estados:

$$\theta_1 = \theta, \qquad \theta_2 = \dot{\theta}$$

La dinámica en el espacio de estados resulta:

$$\dot{\theta}_1 = \theta_2$$
$$\dot{\theta}_2 = -b_1\theta_2 - b_2\theta_1 + b_3 u_\theta$$

Definiendo los errores de seguimiento de trayectoria en θ_1 y θ_2,

$$e_{\theta_1} = \theta_1 - \theta_1^*$$
$$e_{\theta_2} = \theta_2 - \dot{\theta}_1^*$$

donde $\theta_1^* = \theta^*$ que es el valor deseado de la entrada virtual θ (5.2), obtenida en la primera etapa. El sistema en las coordenadas del error es,

$$\dot{e}_{\theta_1} = e_{\theta_2} \tag{5.6a}$$
$$\dot{e}_{\theta_2} = -b_1\theta_2 - b_2\theta_1 + b_3 u_\theta - \ddot{\theta}_1^* \tag{5.6b}$$

Para estabilizar el sistema (5.6), primero se propone una ley de control auxiliar dada por:

$$u_\theta = \frac{1}{b_3}[\ddot{\theta}_1^* + b_1\theta_2 + b_2\theta_1 + w_\theta] \tag{5.7}$$

Al realimentar (5.7) con (5.6), el sistema se reescribe como:

$$\dot{e}_{\theta_1} = e_{\theta_2} \tag{5.8a}$$
$$\dot{e}_{\theta_2} = w_\theta \tag{5.8b}$$

Para estabilizar el sistema (5.8) se propone la ley de control,

$$w_\theta = -k_{p\theta} e_{\theta_1} - k_{d\theta} e_{\theta_2}$$

Con $k_{p\theta}$, $k_{d\theta} > 0$. Finalmente se obtiene,

$$u_\theta = \frac{1}{b_3}[\ddot{\theta}_1^* + b_1\theta_2 + b_2\theta_1 - k_{p\theta} e_{\theta_1} - k_{d\theta} e_{\theta_2}] \tag{5.9}$$

Ahora, se considera el sistema (5.6) en lazo cerrado con la ley de control (5.9), el sistema en lazo cerrado resulta,

$$\dot{e}_{\theta_1} = e_{\theta_2} \tag{5.10a}$$
$$\dot{e}_{\theta_2} = -k_{p\theta} e_{\theta_1} - k_{d\theta} e_{\theta_2} \tag{5.10b}$$

el cual puede ser expresado en forma matricial como

$$\begin{bmatrix} \dot{e}_{\theta_1} \\ \dot{e}_{\theta_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\theta} & -k_{d\theta} \end{bmatrix} \begin{bmatrix} e_{\theta_1} \\ e_{\theta_2} \end{bmatrix} = H_\theta \begin{bmatrix} e_{\theta_1} \\ e_{\theta_2} \end{bmatrix} \tag{5.11}$$

Debido a que la matriz H_θ es Hurtwitz, el sistema en lazo cerrado es exponencialmente estable.

5.2.2. Control para u_ϕ

La dinámica del ángulo de alabeo está dada por:

$$\ddot{\phi} = -c_1\dot{\phi} - c_2\phi + c_3 u_\phi$$

Definiendo los estados:

$$\phi_1 = \phi, \qquad \phi_2 = \dot{\phi}$$

La dinámica en el espacio de estados resulta:

$$\dot{\phi}_1 = \phi_2$$
$$\dot{\phi}_2 = -c_1\phi_2 - c_2\phi_1 + c_3 u_\phi$$

Definiendo los errores de seguimiento de trayectoria en ϕ_1 y ϕ_2,

$$e_{\phi_1} = \phi_1 - \phi_1^*$$
$$e_{\phi_2} = \phi_2 - \dot{\phi}_1^*$$

donde $\phi_1^* = \phi^*$ que es el valor deseado de la entrada virtual ϕ (5.2), obtenida en la primera etapa. El sistema en las coordenadas del error es,

$$\dot{e}_{\phi_1} = e_{\phi_2} \tag{5.12a}$$
$$\dot{e}_{\phi_2} = -c_1\phi_2 - c_2\phi_1 + c_3 u_\phi - \ddot{\phi}_1^* \tag{5.12b}$$

Para estabilizar el sistema (5.12), primero se propone una ley de control auxiliar dada por:

$$u_\phi = \frac{1}{c_3}[\ddot{\phi}_1^* + c_1\phi_2 + c_2\phi_1 + w_\phi] \tag{5.13}$$

Al realimentar (5.13) con (5.12), el sistema se reescribe como:

$$\dot{e}_{\phi_1} = e_{\phi_2} \tag{5.14a}$$
$$\dot{e}_{\phi_2} = w_\phi \tag{5.14b}$$

Para estabilizar el sistema (5.14) se propone la ley de control,

$$w_\phi = -k_{p\phi} e_{\phi_1} - k_{d\phi} e_{\phi_2}$$

Con $k_{p\phi}$, $k_{d\phi} > 0$. Finalmente se obtiene,

$$u_\phi = \frac{1}{c_3}[\ddot{\phi}_1^* + c_1\phi_2 + c_2\phi_1 - k_{p\phi} e_{\phi_1} - k_{d\phi} e_{\phi_2}] \tag{5.15}$$

Ahora, se considera el sistema (5.12) en lazo cerrado con la ley de control (5.15), el sistema en lazo cerrado resulta,

$$\dot{e}_{\phi_1} = e_{\phi_2} \tag{5.16a}$$
$$\dot{e}_{\phi_2} = -k_{p\phi} e_{\phi_1} - k_{d\phi} e_{\phi_2} \tag{5.16b}$$

el cual puede ser expresado en forma matricial como

$$\begin{bmatrix} \dot{e}_{\phi_1} \\ \dot{e}_{\phi_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k_{p\phi} & -k_{d\phi} \end{bmatrix} \begin{bmatrix} e_{\phi_1} \\ e_{\phi_2} \end{bmatrix} = H_\phi \begin{bmatrix} e_{\phi_1} \\ e_{\phi_2} \end{bmatrix} \tag{5.17}$$

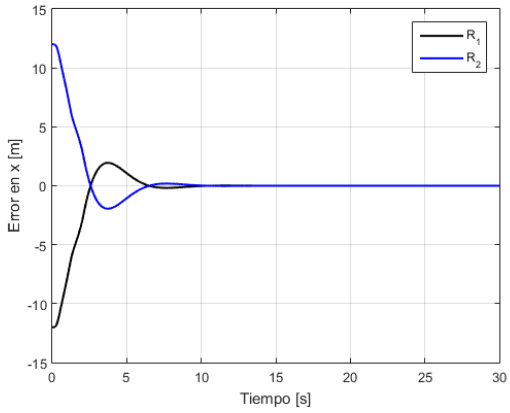
Debido a que la matriz H_ϕ es Hurtwitz, el sistema en lazo cerrado es exponencialmente estable.

5.2.3. Simulación de la segunda etapa de control con dos agentes

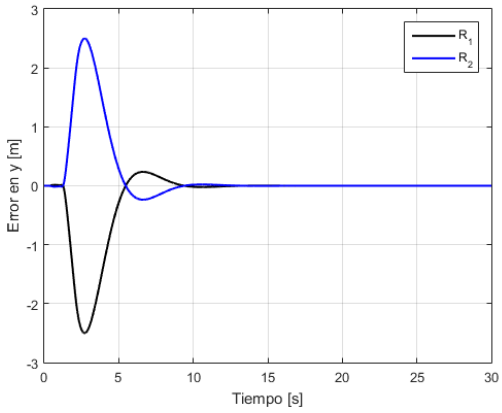
La segunda etapa se simulará con los mismos parámetros que la etapa uno, y con

$$k_{p\theta} = 100; \qquad k_{d\theta} = 5;$$
$$k_{p\phi} = 100; \qquad k_{p\phi} = 5;$$

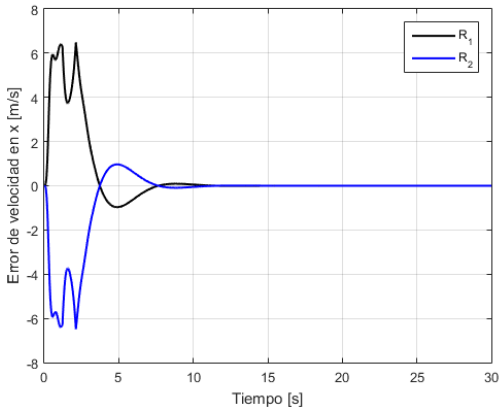
Ahora se muestran los resultados obtenidos.



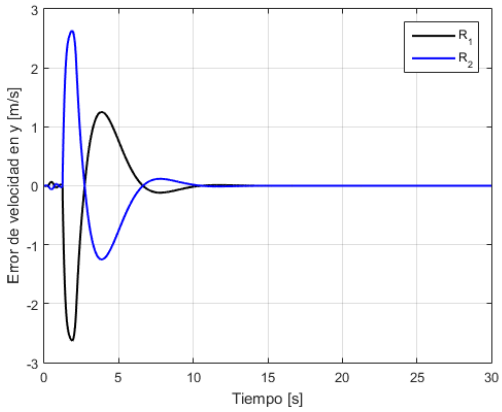
(a) Error de posición de x



(b) Error de posición de y



(c) Error de velocidad de x



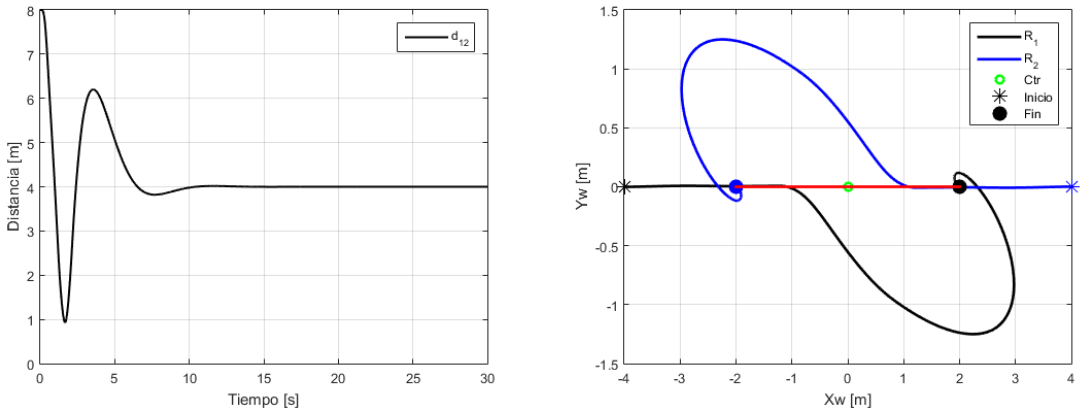
(d) Error de velocidad de y

Figura 5.7: Errores de posición y velocidad.

En la figura 5.7 se observan los errores de posición y de velocidad en X y Y. Se puede notar como convergen a cero, esto quiere decir que los agentes llegan a la formación deseada en el plano X-Y sin importar que hagan seguimiento de trayectoria en altura y guiñada.

En la figura 5.8 se puede observar la distancia entre los agentes y se muestra la trayectoria generada por cada agente en el plano X-Y. Se puede ver que si se logra el

efecto de evasión de colisión. En la figura 5.9 se muestra la trayectoria en el espacio que generan los agentes.



(a) Distancia entre agentes

(b) Trayectorias generadas en X-Y

Figura 5.8: Trayectorias en X-Y y distancia entre agentes.

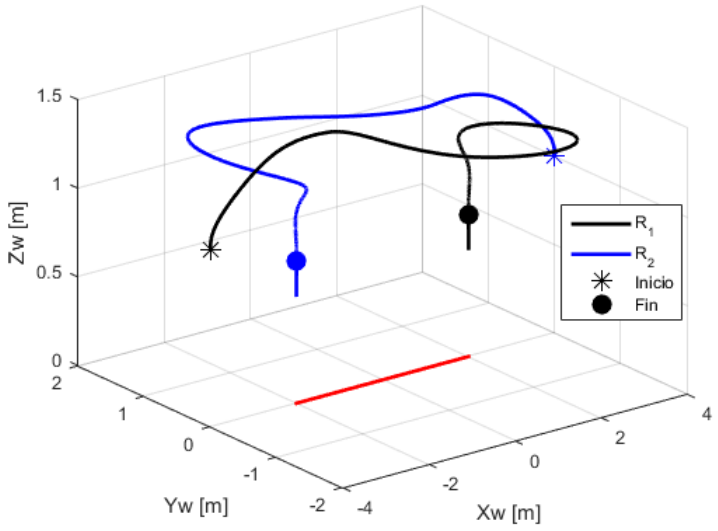


Figura 5.9: Trayectorias en XYZ.

5.3. Primera etapa de control para tres agentes

En este punto, la ley de control propuesta resuelve el problema de formación de los agentes del sistema, además permite la evasión de colisiones entre obstáculos conocidos y entre otros agentes del entorno. Ahora, se propone una ley de control en la que además de resolver el problema de formación y evasión de colisiones también se resuelva el problema de seguimiento de trayectoria. Para este caso, se añadirá un agente más al sistema y se propondrá un esquema líder-seguidor. Para lograr que el sistema multiagente se desplace sobre el plano horizontal X-Y conservando una formación deseada, es necesario seleccionar un agente líder, ℓ, en la formación. El agente líder será el que siga la trayectoria deseada para desplazarse y consecuentemente la formación hará lo mismo.

Se maneja el esquema de control en dos etapas, en la primera se pretende lograr el comportamiento deseado del sistema multiagente, con las entradas de control virtuales propuestas. Una vez que se obtiene dicho comportamiento, se procede a diseñar una segunda etapa de control que asegure la convergencia de las entradas de control reales a las entradas de control virtuales de la primera etapa del sistema.

Recordando la dinámica linealizada para XY,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \underbrace{\Gamma_z \begin{bmatrix} \cos\psi & \sin\psi \\ \sin\psi & -\cos\psi \end{bmatrix}}_{\tilde{A}(\psi)} \begin{bmatrix} \theta \\ \phi \end{bmatrix}; \qquad \Gamma_z = (-a_1\dot{z} + a_3 u_z - g)$$

Proponiendo θ y ϕ como entradas de control virtuales para el sistema. Tenemos un sistema de la forma:

$$\ddot{X} = \tilde{A}(\psi)u_a \tag{5.18}$$

Si $u_a = \tilde{A}^{-1}(\psi)u$. Al sustituir u_a en el sistema, se tiene,

$$\ddot{X} = u \tag{5.19}$$

Lo que para cada agente sería,

$$\ddot{X}_i = u_i \qquad ; \qquad X_i : \text{Posición en XY para cada agente.}$$

Ahora se considera, un conjunto de n agentes denotado por $N = \{R_1, R_2, ..., R_n\}$ que se mueven sobre un plano. Cada agente es capaz de detectar la posición de cierto subconjunto de agentes $N_i \subset N$. La posición deseada para cada agente R_i respecto a N_i en una formación dada, está definida por,

$$X_i^* = \frac{1}{n_i} \sum_{j \in N_i} (X_j + C_{ji})$$

donde $C_{ij} = [h_{ji}, v_{ji}]^T$ es el vector de posición relativa deseada del agente R_i respecto al agente R_j y n_i es la cardinalidad del subconjunto N_i.

Al derivar la posición deseada respecto al tiempo, se tiene que la velocidad deseada para cada agente está dada por:

$$\dot{X}_i^* = \frac{1}{n_i} \sum_{j \in N_i} \dot{X}_j$$

Ahora, se define el error de posición como $e_{X_i} = X_i - X_i^*$ y el error de velocidad como $\dot{e}_{X_i} = \dot{X}_i - \dot{X}_i^*$. Para lograr el control de formación se propone utilizar el algoritmo fundamental de consenso para sistemas tipo doble integrador para cada agente[8].

$$\gamma_i = -k_0 e_{X_i} - k_1 \dot{e}_{X_i} \tag{5.20}$$

Debido a que los agentes tipo cuadrirrotor tienen limitaciones físicas, se necesita tener un control acotado. Para garantizar que la ley de control (5.20) esté acotada se propone:

$$\gamma_i = -k_0 \tanh(e_{X_i}) - k_1 \tanh(\dot{e}_{X_i}) \tag{5.21}$$

Para el agente líder, primero se define la trayectoria que se desea seguir en el plano X-Y como $m(t)$. Entonces, el error de posición para el agente líder se define como:

$$e_{X_\ell} = X_\ell - m(t) \tag{5.22}$$

y su error de velocidad como:

$$\dot{e}_{X_\ell} = \dot{X}_\ell - \dot{m}(t) \tag{5.23}$$

Es necesario prealimentar la ley de control de movimiento para los agentes con la aceleración deseada. Entonces la ley de control que soluciona el problema de formación, de marcha y la no colisión para esta primera etapa está dada por las siguientes ecuaciones.

Para el agente líder:

$$\begin{aligned} \ddot{X}_\ell &= u_\ell = \ddot{m}(t) + \gamma_\ell + \beta_\ell \\ u_\ell &= \ddot{m}(t) \underbrace{-k_0 e_{X_\ell} - k_1 \dot{e}_{X_\ell}}_{\gamma_\ell} + \underbrace{\varepsilon \sum_{j \in M_\ell}^{n} \delta_{\ell j} \begin{bmatrix} (X_{x_\ell} - X_{x_j}) - (X_{y_\ell} - X_{y_j}) \\ (X_{x_\ell} - X_{x_j}) + (X_{y_\ell} - X_{y_j}) \end{bmatrix}}_{\beta_\ell} \end{aligned} \tag{5.24}$$

Y para los agentes seguidores:

$$\begin{aligned}\ddot{X}_i &= u_i = \ddot{X}_i^* + \gamma_i + \beta_i \\ u_i &= \ddot{X}_i^* \underbrace{-k_0 e_{X_i} - k_1 \dot{e}_{X_i}}_{\gamma_i} + \underbrace{\varepsilon \sum_{j \in M_i}^{n} \delta_{ij} \begin{bmatrix} (X_{x_i} - X_{x_j}) - (X_{y_i} - X_{y_j}) \\ (X_{x_i} - X_{x_j}) + (X_{y_i} - X_{y_j}) \end{bmatrix}}_{\beta_i}\end{aligned} \tag{5.25}$$

Las leyes de control (5.24) y (5.25) son similares a la ley de control que se utilizó para los dos agentes, la diferencia es que al cambiar el error de un agente asignado como líder en función de la trayectoria deseada, se logra que el sistema siga una trayectoria. Se hará una simulación numérica con las leyes de control (5.24) y (5.25) para ver el desempeño de la primera etapa de control. Para la simulación se pretende usar los parámetros de [7] que es una simulación numérica con robot tipo terrestre y así ver su desempeño en robot tipo cuadrirrotor.

Para la simulación numérica se tienen los siguientes parámetros y la gráfica de formación deseada que se muestra en la Fig. 5.10.

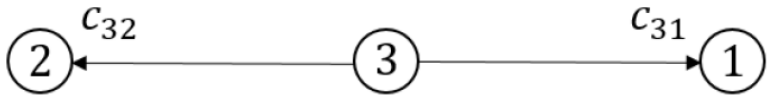


Figura 5.10: Formación deseada.

De la gráfica de comunicación se obtienen los siguientes datos,

Los errores de posición para cada agente:

$$\begin{aligned} e_{X_1} &= X_1 - X_3 - C_{31} \\ e_{X_2} &= X_2 - X_3 - C_{32} \\ e_{X_\ell} &= e_{X_3} = X_3 - m(t) \end{aligned} \tag{5.26}$$

Los errores de velocidad para cada agente:

$$\begin{aligned} \dot{e}_{X_1} &= \dot{X}_1 - \dot{X}_3 \\ \dot{e}_{X_2} &= \dot{X}_2 - \dot{X}_3 \\ \dot{e}_{X_\ell} &= \dot{e}_{X_3} = \dot{X}_3 - \dot{m}(t) \end{aligned} \tag{5.27}$$

Los campos vectoriales repulsivos para cada agente:

$$\beta_1 = \varepsilon\delta_{12}\begin{bmatrix}(x_1-x_2)-(y_1-y_2)\\(x_1-x_2)+(y_1-y_2)\end{bmatrix} + \varepsilon\delta_{13}\begin{bmatrix}(x_1-x_3)-(y_1-y_3)\\(x_1-x_3)+(y_1-y_3)\end{bmatrix}$$

$$\beta_2 = \varepsilon\delta_{21}\begin{bmatrix}(x_2-x_1)-(y_2-y_1)\\(x_2-x_1)+(y_2-y_1)\end{bmatrix} + \varepsilon\delta_{23}\begin{bmatrix}(x_2-x_3)-(y_2-y_3)\\(x_2-x_3)+(y_2-y_3)\end{bmatrix} \tag{5.28}$$

$$\beta_\ell = \beta_3 = \varepsilon\delta_{31}\begin{bmatrix}(x_3-x_1)-(y_3-y_1)\\(x_3-x_1)+(y_3-y_1)\end{bmatrix} + \varepsilon\delta_{32}\begin{bmatrix}(x_3-x_2)-(y_3-y_2)\\(x_3-x_2)+(y_3-y_2)\end{bmatrix}$$

Con las ecuaciones (5.26), (5.27) y (5.28) se pueden formar las leyes de control u_i y u_ℓ. Recordar que al obtener u_i se tiene que, $u_{a_i} = \tilde{A}_i(\psi)^{-1}u_i$. Entonces se procede a mostrar los resultados de la simulación numérica.

5.3.1. Simulación de la primera etapa de control para tres agentes

Los parámetros de simulación son los siguientes. La trayectoria deseada para el líder es una lemniscata de Gerono, dada por:

$$m(t) = \begin{bmatrix}m_x(t)\\m_y(t)\end{bmatrix} = \begin{bmatrix}\sin(0.15t)\cos(0.15t)\\-0.699+\sin(0.15t)\end{bmatrix}$$

Las condiciones iniciales de los agentes están dadas por:

$$\begin{bmatrix}x_1, & y_1, & z_1, & \theta_1, & \phi_1, & \psi_1\end{bmatrix}^T = \begin{bmatrix}-0.699, & 0.75, & 1, & 0, & 0, & 0\end{bmatrix}^T$$
$$\begin{bmatrix}x_2, & y_2, & z_2, & \theta_2, & \phi_2, & \psi_2\end{bmatrix}^T = \begin{bmatrix}-0.699, & -0.75, & 1, & 0, & 0, & 0\end{bmatrix}^T$$
$$\begin{bmatrix}x_3, & y_3, & z_3, & \theta_3, & \phi_3, & \psi_3\end{bmatrix}^T = \begin{bmatrix}0.6, & 0, & 1, & 0, & 0, & 0\end{bmatrix}^T$$

Los vectores de posición relativa:

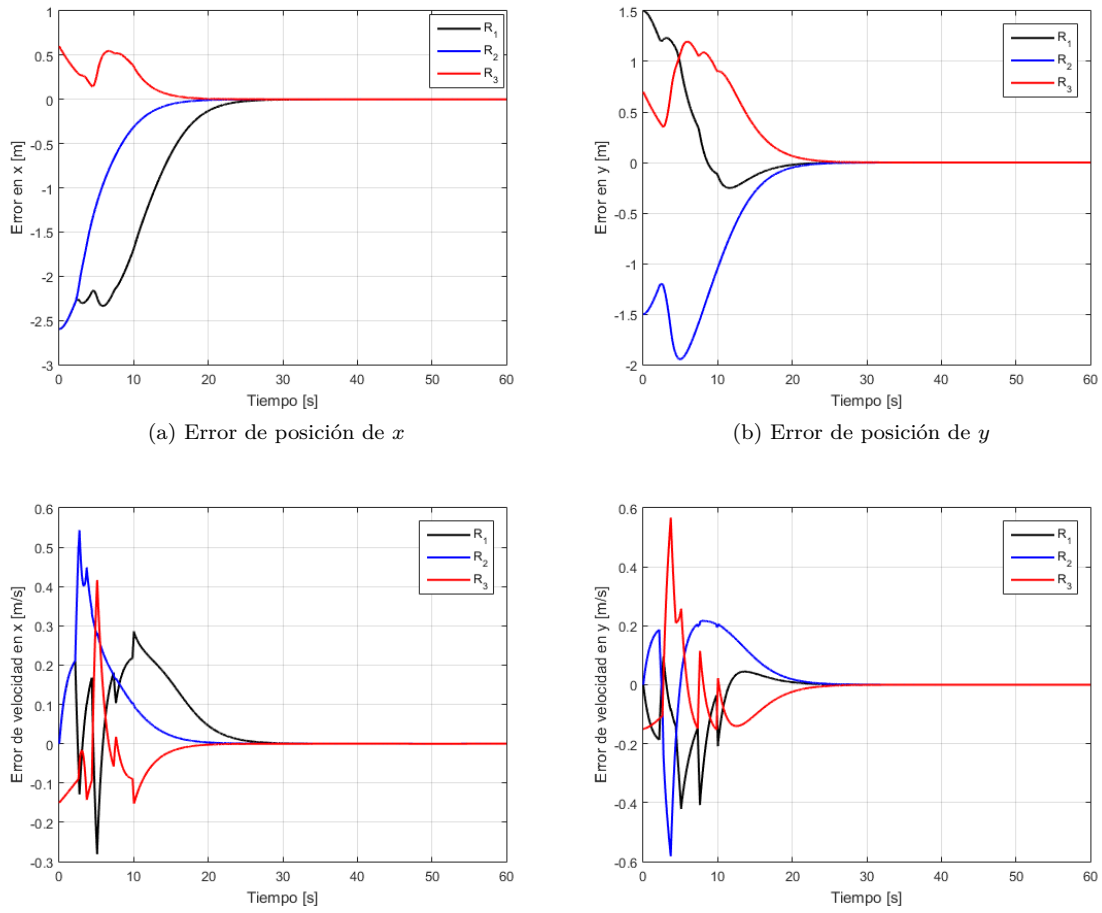
$$c_{31} = \left[1.5\sin(\frac{\pi}{3}), -1.5\cos(\frac{\pi}{3})\right]^T$$
$$c_{32} = \left[1.5\sin(\frac{\pi}{3}), 1.5\cos(\frac{\pi}{3})\right]^T$$

Además de moverse en el plano X-Y los agentes son capaces de realizar seguimiento de trayectoria en altura y guiñada. La trayectoria deseada para altura y guiñada se muestra a continuación,

$$\begin{bmatrix}z_{1d} & z_{2d} & z_{3d}\end{bmatrix} = \begin{bmatrix}0.5 & 1.2 & 1.7\end{bmatrix}$$
$$\begin{bmatrix}\psi_{1d} & \psi_{2d} & \psi_{3d}\end{bmatrix} = \begin{bmatrix}0.6\sin(0.08t) & 0.4\sin(0.1t) & 0.2\sin(0.04)\end{bmatrix}$$

y los parámetros $k_0 = 0.24$, $k_1 = 1.0$, $d = 0.8$ y $\varepsilon = 0.8$

Los resultados de la simulación se muestran a continuación.



(a) Error de posición de x

(b) Error de posición de y

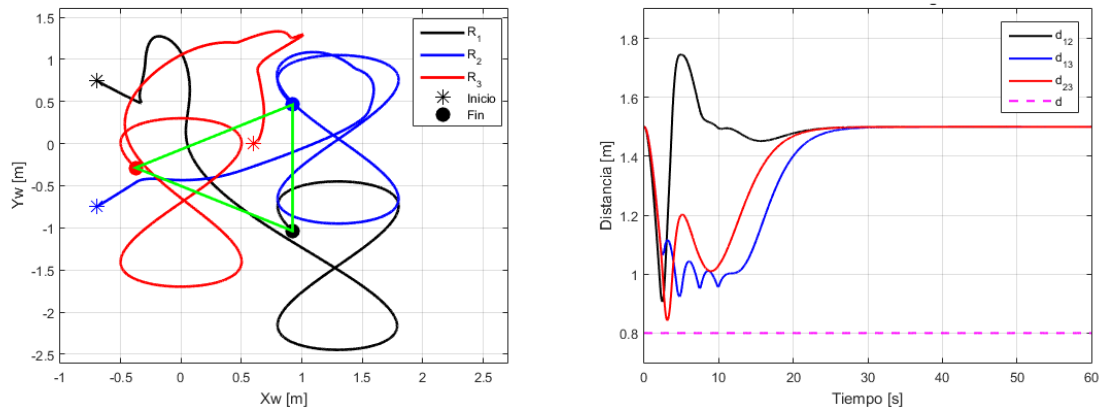
(c) Error de velocidad de x

(d) Error de velocidad de y

Figura 5.11: Errores de posición y velocidad.

En la Figura 5.11 se muestra que los errores de velocidad y posición convergen a cero, esto sucede después de los primeros treinta segundos. Que los errores de posición de los agentes lleguen a cero, significa que han alcanzado la posición deseada uno respecto del otro. Además, como el error de posición del agente líder (R_3) está en función de la trayectoria deseada en el plano, cuando éste error se hace cero significa que el agente líder está siguiendo la trayectoria que se le definió.

En la Figura 5.12b se muestra la distancia entre los agentes y se puede notar que en ningún momento se acercan más de lo que está determinado por el parámetro d. En la Figura 5.12a se muestran las trayectorias que generan los agentes en el plano XY y es posible notar que se logra la no colisión. La ley de control del sistema multi-agente es capaz de realizar formación, marcha y evasión de colisiones.



(a) Trayectorias generadas en X-Y

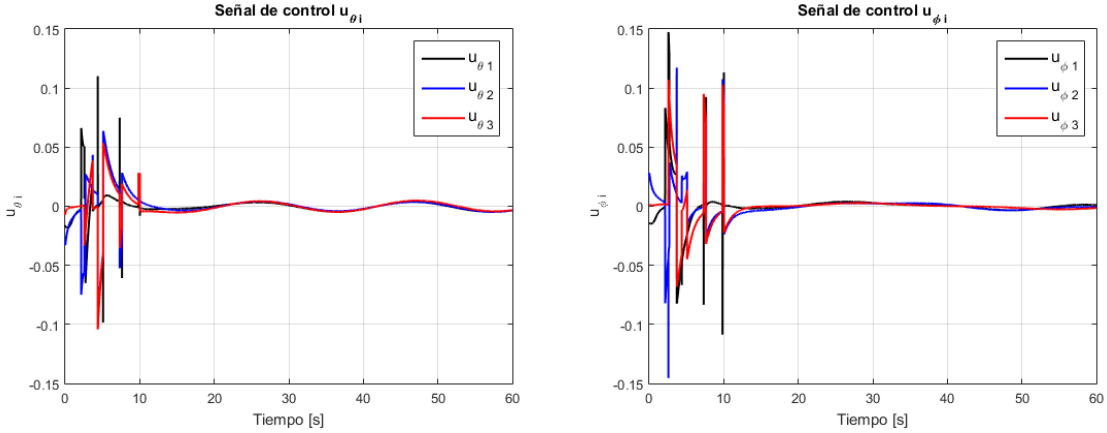
(b) Distancia entre los agentes

Figura 5.12: Trayectorias en XY y distancia entre agentes.

Las entradas de control se muestran a continuación. En la Figura 5.13 se muestran las entradas de control que generan el movimiento en X-Y (alabeo y cabeceo). A pesar de que las señales se ven algo bruscas, los valores que toman son aceptables para una posible implementación física. En la Figura 5.14 se muestran las señales de control para la altura y guiñada de cada agente.

Se puede notar que los valores son aceptables y que mientras los agentes realizan formación, marcha y evasión de colisiones en un plano horizontal también son capaces de hacer seguimiento de trayectoria en altura y guiñada.

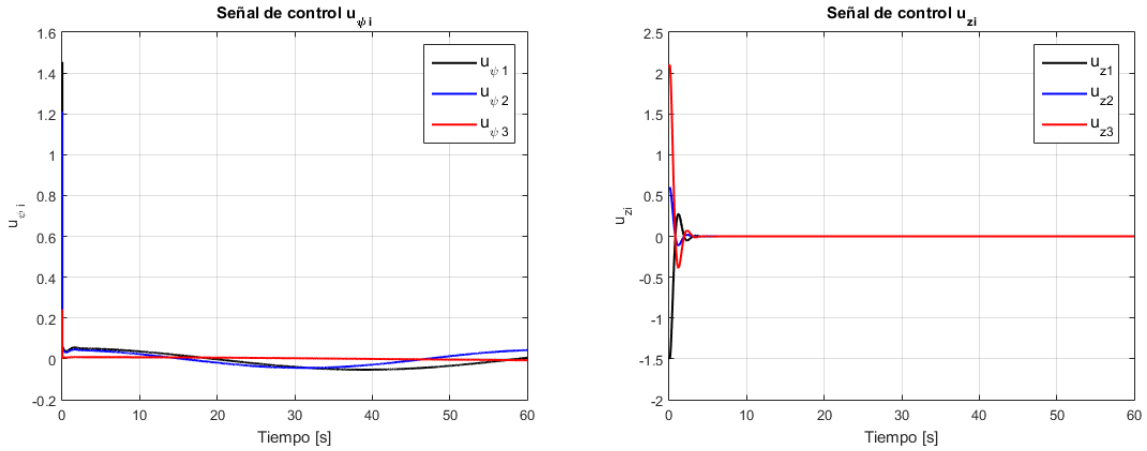
Se debe recordar que en esta primera etapa las entradas de control no son las entradas de control reales del AR Drone 2.0, sino que se tomaron como entradas de control virtuales a θ_i y ϕ_i, entonces las llamaremos θ_i^* y ϕ_i^*.



(a) u_θ

(b) u_ϕ

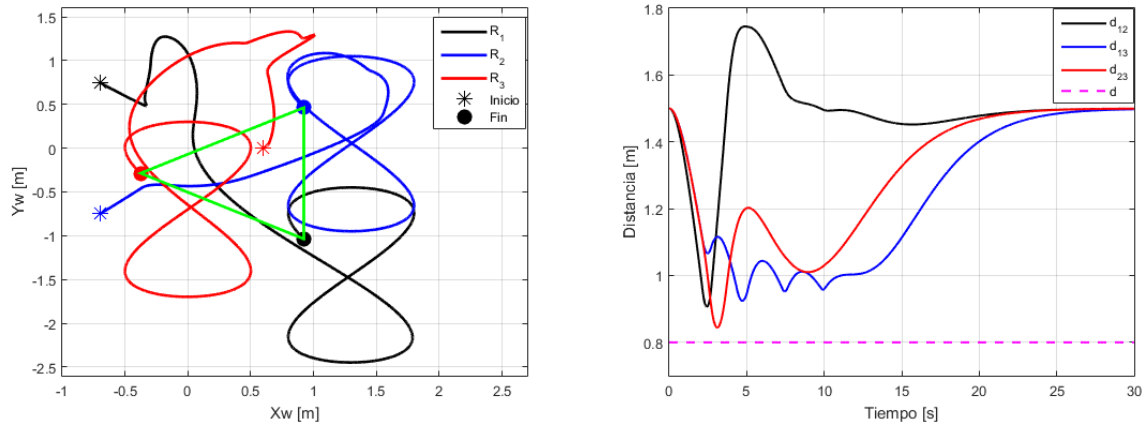
Figura 5.13: Señales de control para movimiento en X-Y



(a) u_ψ (b) u_z

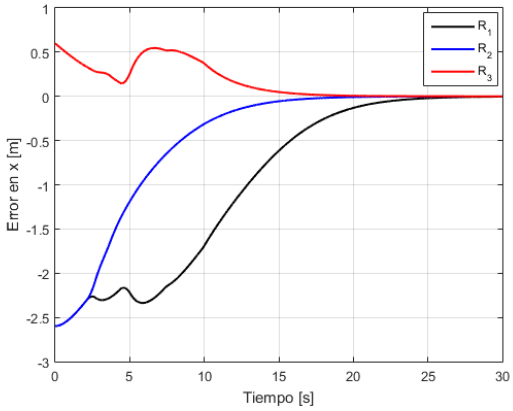
Figura 5.14: Señales de control para altura y guiñada

Para mostrar un poco más a detalle las señales de control mostradas anteriormente, se presentan las siguientes figuras en donde se muestran las señales en los primeros segundos y no necesariamente hasta el final de la simulación. Esto porque en los primeros segundos, cuando los agentes están en peligro de colisión, la ley de control demanda valores mayores en las entradas de control. En las siguientes gráficas se aprecia mejor cómo actua la ley de control cuando la evasión sucede. En la figura 5.15b se puede notar como los agentes comienzan a alcanzar la formación deseada después de los 25s y en ningún momento las distancias están por debajo del parámetro d.

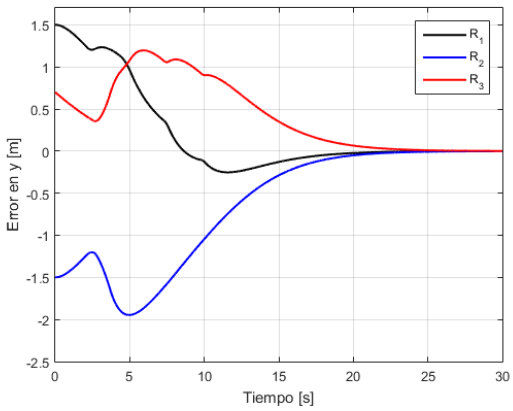


(a) Trayectorias generadas en X-Y (b) Distancia entre los agentes

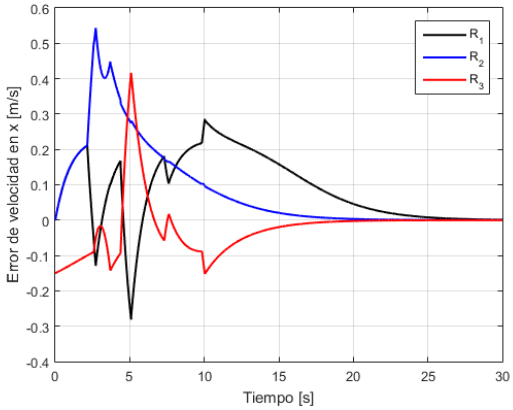
Figura 5.15: Trayectorias en X-Y y distancia entre agentes



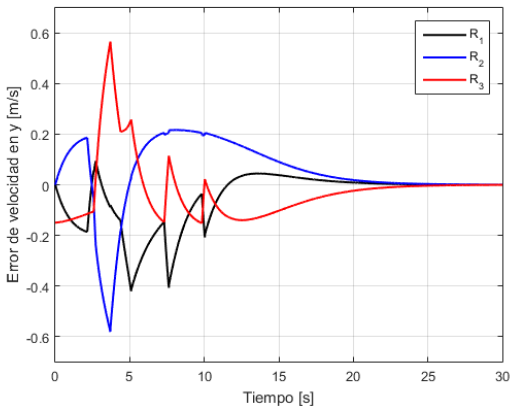
(a) Error de posición de x



(b) Error de posición de y

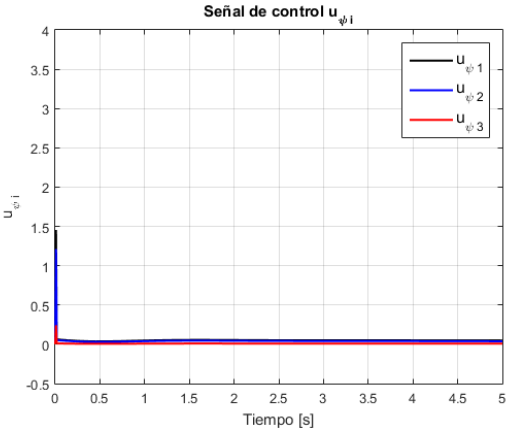


(c) Error de velocidad de x

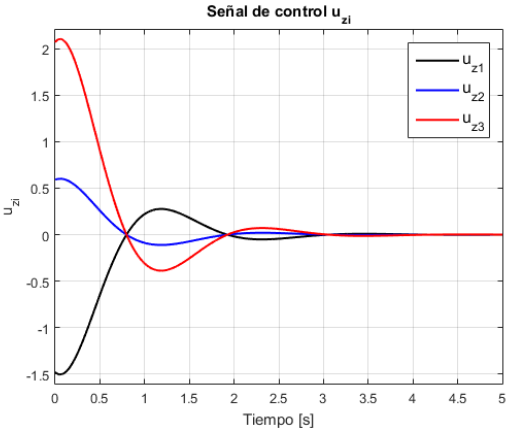


(d) Error de velocidad de y

Figura 5.16: Errores de posición y velocidad.

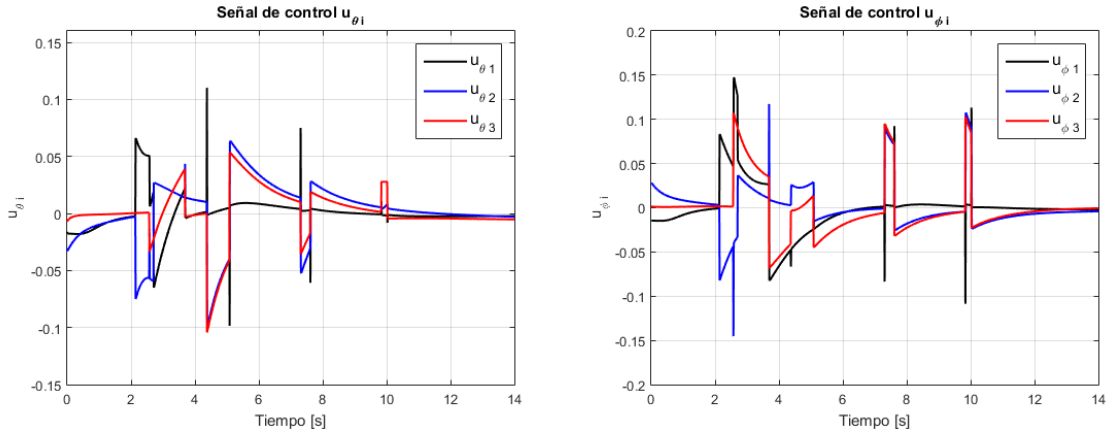


(a) u_ψ



(b) u_z

Figura 5.17: Señales de control para altura y guiñada



(a) u_θ (b) u_ϕ

Figura 5.18: Señales de control para movimiento en X-Y

5.3.2. Simulación de la segunda etapa de control para tres agentes

Como anteriormente en la primera etapa de control se puede notar que el control funciona correctamente, entonces, es posible utilizar las mismas ecuaciones (5.9) y (5.15) que se utilizaron para dos agentes, a pesar de que para tres agentes se incluyó el seguimiento de trayectoria. Recordando las ecuaciones, se muestran a continuación.

$$u_\theta = \frac{1}{b_3}[\ddot{\theta}_1^* + b_1\theta_2 + b_2\theta_1 - k_{p\theta}e_{\theta_1} - k_{d\theta}e_{\theta_2}]$$

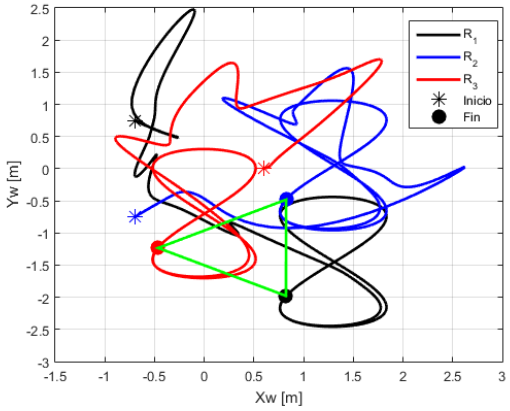
$$u_\phi = \frac{1}{c_3}[\ddot{\phi}_1^* + c_1\phi_2 + c_2\phi_1 - k_{p\phi}e_{\phi_1} - k_{d\phi}e_{\phi_2}]$$

Donde,

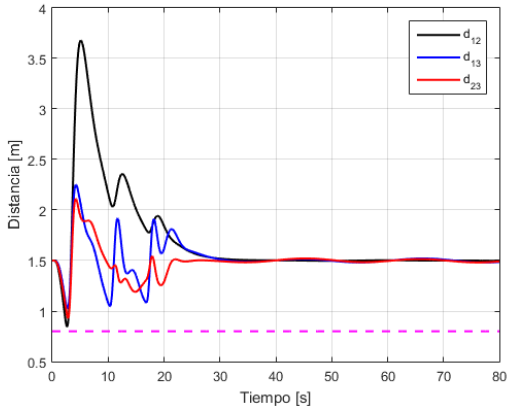
$$\theta_1 = \theta, \qquad \theta_2 = \dot{\theta}, \qquad e_{\theta_1} = \theta - \theta^*, \qquad e_{\theta_2} = \dot{\theta} - \dot{\theta}^*, \qquad \ddot{\theta}_1^* = \ddot{\theta}^*$$
$$\phi_1 = \phi, \qquad \phi_2 = \dot{\phi}, \qquad e_{\phi_1} = \phi - \phi^*, \qquad e_{\phi_2} = \dot{\phi} - \dot{\phi}^*, \qquad \ddot{\phi}_1^* = \ddot{\phi}^*$$

Con θ^* y ϕ^* como las referencias deseadas que se obtienen en la primera etapa de control. El control de altura y de guiñada se mantienen de la misma forma para esta simulación ya que no afecta directamente al control en el plano X-Y debido a que se consideran como sistemas desacoplados.

Se utilizarán los mismos parámetros de simulación que en la primera etapa para darnos cuenta del desempeño del control propuesto. A continuación se muestran los resultados.

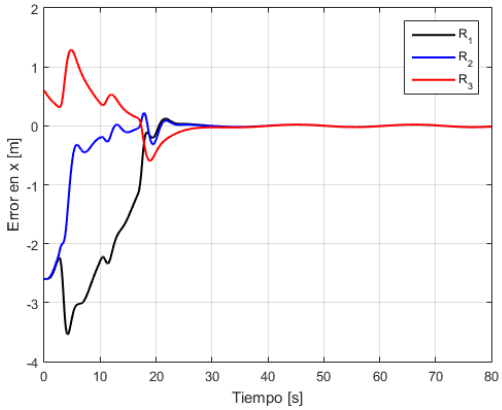


(a) Trayectorias generadas en X-Y

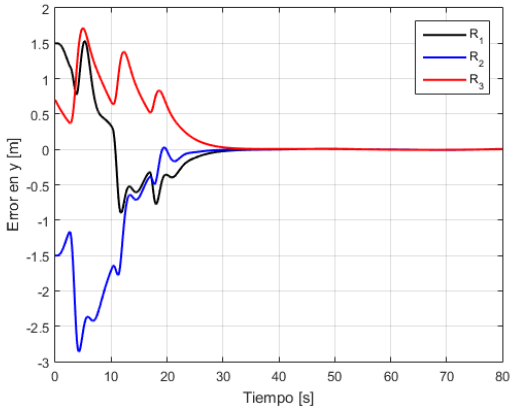


(b) Distancia entre los agentes

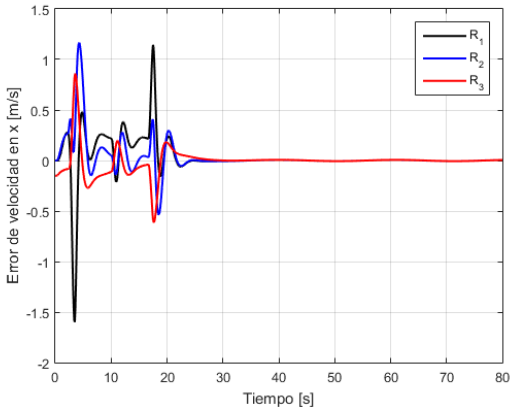
Figura 5.19: Trayectorias en X-Y y distancia entre agentes



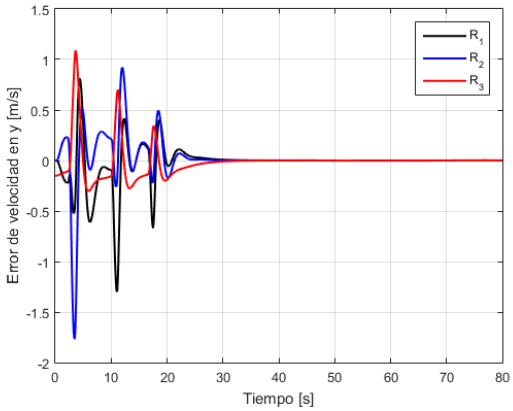
(a) Error de posición de x



(b) Error de posición de y



(c) Error de velocidad de x

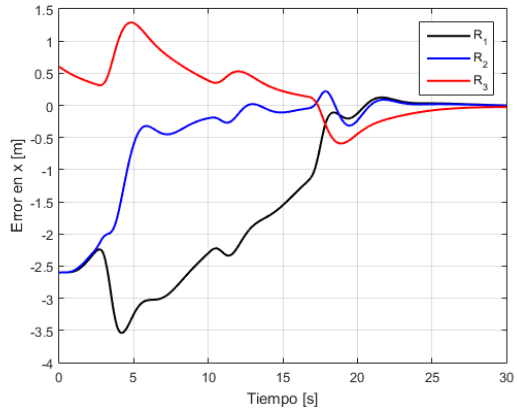


(d) Error de velocidad de y

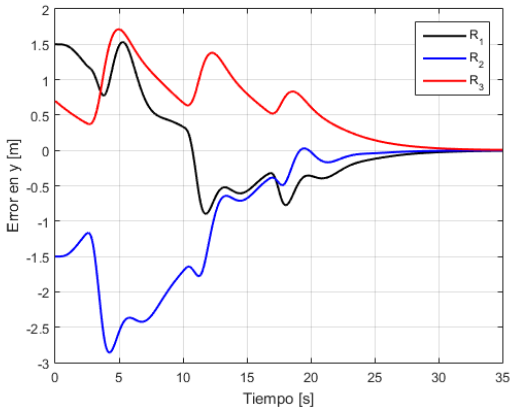
Figura 5.20: Errores de posición y velocidad.

Para mostrar un poco más a detalle las gráficas mostradas anteriormente, se presentan las siguientes figuras en donde se muestran las trayectorias, además de los errores en los primeros segundos y no necesariamente hasta el final de la simulación. Esto porque en los primeros segundos, cuando los agentes están en peligro de colisión, es cuando la parte de la no colisión hace presencia. En las siguientes gráficas se aprecia mejor cómo actua la ley de control cuando la evasión sucede. En la figura 5.15b se puede notar como los agentes comienzan a alcanzar la formación deseada después de los 30s y en ningún momento las distancias están por debajo del parámetro d.

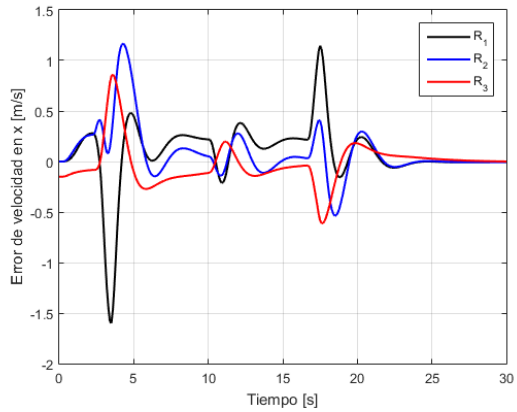
En la figura 5.20 se puede ver que los errores se mantienen muy cerca del cero después de los cuarenta segundos, esto significa que el agente líder (R_3) está siguiendo la trayectoria deseada y que el sistema multi-agente a su vez mantiene la fotmación deseada que se le asignó.



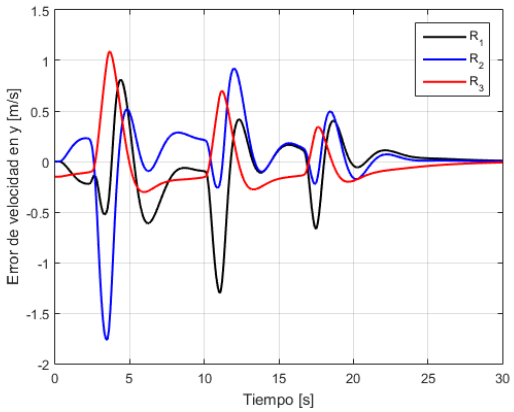
(a) Error de posición de x



(b) Error de posición de y

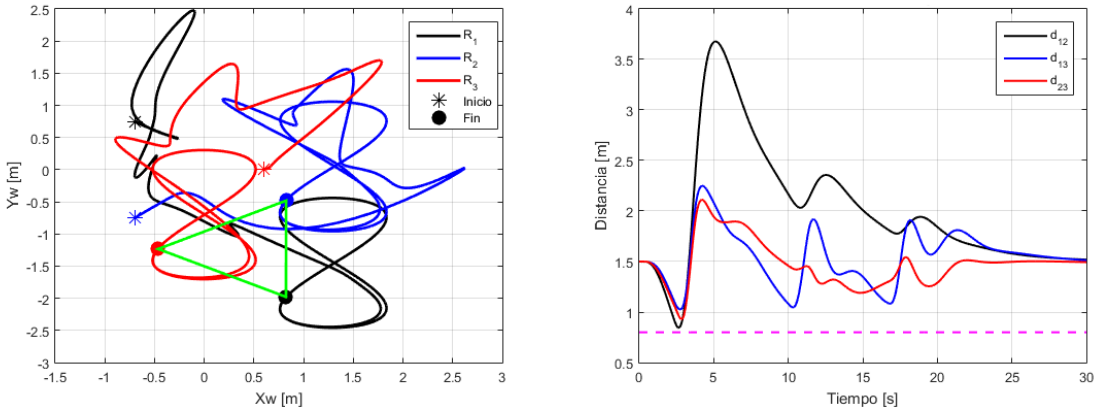


(c) Error de velocidad de x



(d) Error de velocidad de y

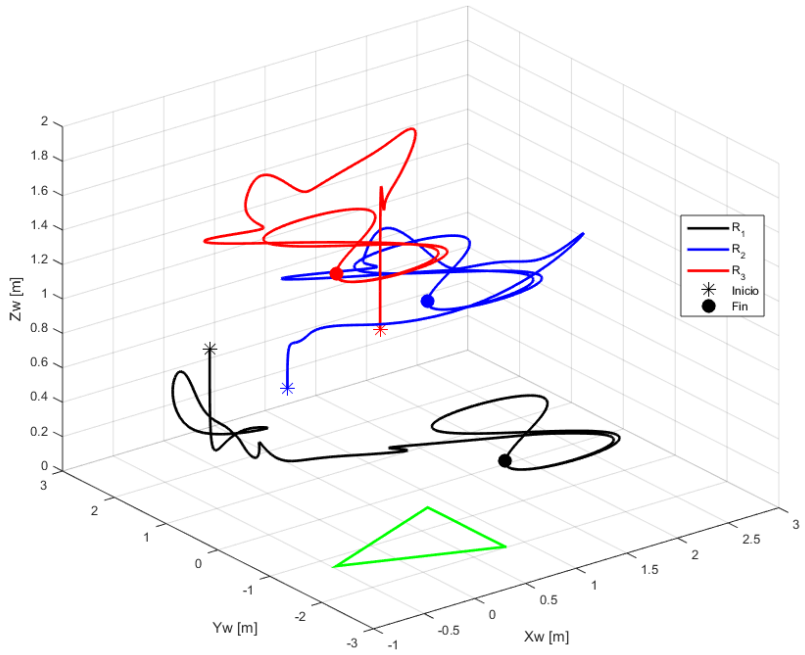
Figura 5.21: Errores de posición y velocidad.



(a) Trayectorias generadas en X-Y

(b) Distancia entre los agentes

Figura 5.22: Trayectorias en X-Y y distancia entre agentes



(a) Trayectorias generadas en XYZ

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo se propuso una alternativa al problema de control de movimiento con evasión de obstáculos para sistemas multi-agente conformados por drones, específicamente, cuadrirrotores AR.DRone 2.0 de la compañía francesa Parrot.

Los algoritmos de control propuestos se basan en el algoritmo fundamental de consenso para el control de movimiento y en la adición de campos vectoriales repulsivos para la evasión de colisiones.

Ambas técnicas de control han sido aplicadas a agentes de primer orden y han comenzado a ser empleadas para sistemas multi-agente de segundo orden modelados mediante grupos de dobles integradores. En el caso de sistemas compuestos por drones tipo cuadrirrotor, existen dos dificultades principales:

1. El modelo matemático de un cuadrirrotor es altamente no lineal.
2. Las coordenadas en el plano horizontal X-Y del cuadrirrotor tienen grado relativo 4, por lo que no es posible afectar instantáneamente la aceleración en dichas coordenadas; mucho menos afectar instantáneamente las velocidades. En el caso de agentes de segundo orden, aun previendo una colisión, la velocidad no puede ser afectada instantáneamente debido a la inercia. Este mismo efecto se presenta para cuadrirrotores, pero es aún más complejo debido al grado relativo de las coordenadas en el plano horizontal.

La propuesta que desarrollamos en este trabajo de tesis consiste en considerar en una primera etapa a los ángulos de alabeo y guiñada como entradas "virtuales" de control para que el cuadrirrotor siga las trayectorias prescritas en el plano X-Y. En una segunda etapa, se diseñan las entradas "reales" que aseguran que los ángulos converjan a los valores deseados. En algún sentido, este proceso es similar a la técnica de backstepping, pero resolviendo cada paso para grupos de dobles integradores, en lugar de integradores simples.

El desempeño de los algoritmos propuestos fue evaluado mediante simulaciones numéricas y los resultados son satisfactorios.

Este trabajo de tesis deja algunas líneas abiertas para trabajo futuro:

1. Desde luego, implementar las leyes de control obtenidas en una plataforma experimental para evaluar su desempeño en tiempo real.
2. Diseñar e implementar un algoritmo para que la transición a la ley de control reactiva (campos vectoriales tipo foco repulsivo) sea suave. Actualmente esta transición es discontinua. Como es conocido, las leyes de control discontinuas pueden conllevar efectos indeseados, como castañeo y/o deterioro de los actuadores.
3. Estudiar el efecto de la discretización de las leyes de control para su implementación en tiempo real.
4. Las leyes de control propuestas no toman en cuenta perturbaciones tales como desbalanceo de las hélices, zona muerta en los engranes y efectos aerodinámicos tales como efecto piso y/o interacción entre corrientes de aire. Entonces, surge de manera natural estudiar la robustez de los algoritmos propuestos o bien diseñar observadores para cierto tipo de perturbaciones.

Bibliografía

[1] Rosaldo, M.A.: *Control de grupos de cuadrirrotores empleando observadores de estado*. Ph.D. dissertation, CINVESTAV. Mayo 2019.

[2] P. J. Bristeau, F. Callou, D. Vissière y N. Petit: *The Navigation and Control technology inside the AR.Drone micro UAV*. IFAC Proceedings Volumes, 44(1):1477 – 1484, 2011, ISSN 1474-6670. `http://www.sciencedirect.com/science/article/pii/S1474667016438188`, 18th IFAC World Congress.

[3] *Especificaciones del AR.Drone 2.0*. `https://www.parrot.com/global/drones/parrot-ardrone-20-power-edition`.

[4] Aaron Martinez and Enrique Fernández.: *Learning ROS for Robotics Programming*. September 2013. Libro publicado por Packt Publishing Ltd., Birmingham, UK.

[5] YoonSeok Pyo, HanCheol Cho, RyuWoon Jung y TaeHoon Lim: *ROS Robot Programming*. December, 2017. Libro publicado por ROBOTIS Co.,Ltd.

[6] *Entorno de simulación ROS/Gazebo*. `http://www.ros.org/wiki/simulator_gazebo`.

[7] J. Meza-Herrera, E. Aranda-Bricaire y J. F. Flores-Resendiz: *Control de movimiento con evasión de colisiones para sistemas multi-agente de segundo orden*. En *XX Congreso Mexicano de Robótica - COMRob 2018*. Ensenada, Baja California, Septiembre 2018.

[8] W. Ren and R. W. Beard: *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications*. En *Communications and Control Engineering Series*. Springer, 2008.