UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Desarrollo e implementación de algoritmos de control para un enjambre de drones Crazyflie 2.0 mediante un sistema de visión de cámaras OptiTrack.

Trabajo de graduación presentado por Kenneth Andree Aldana Corado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

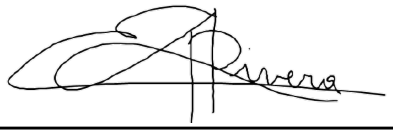
Desarrollo e implementación de algoritmos de control para un enjambre de drones Crazyflie 2.0 mediante un sistema de visión de cámaras OptiTrack.

Trabajo de graduación presentado por Kenneth Andree Aldana Corado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

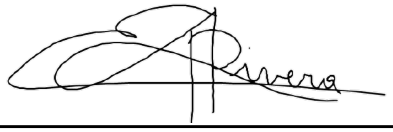
2023

Vo.Bo.:

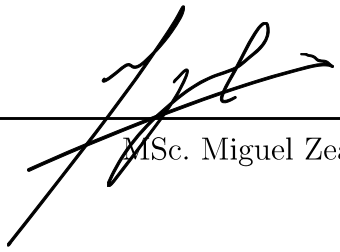
(f) 

Dr. Luis Alberto Rivera Estrada

Tribunal Examinador:

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

MSc. Miguel Zea

(f) 

MSc. Marvin Najarro

Fecha de aprobación: Guatemala, 6 de enero de 2023.

Prefacio

El interés por el área de la robótica y las aplicaciones complejas que en los últimos años desarrollando para uso de los sistemas multi-agente despertaron en mi la pasión para desarrollar este trabajo. Tener la oportunidad de trabajar con temas que siguen siendo áreas de estudio en distintas partes del mundo gracias a todo lo aprendido en los cursos de Sistemas de Control 1 y 2 y Robótica 1 y 2, respaldan mi decisión de desarrollar este proyecto como medio de aprendizaje personal y establecimiento de una base para futuras personas que deseen continuar este trabajo del área de robótica.

Es por esto, que me gustaría agradecer en este espacio a quienes me apoyaron durante todo el proceso, personas que tomaron un papel esencial para que este trabajo se desarrollara:

- A mis papás, Janette y Jorge, porque siempre estuvieron conmigo y me apoyaron en todos los momentos de mi vida. Por motivarme a seguir aprendiendo y estudiar lo que me apasiona.
- A mis hermanos, Marlon y Jorge, gracias porque me han acompañado durante toda mi vida, sacándome risas y dándome consejos cada vez que los necesitaba.
- A mi asesor, Luis Rivera, por estar pendiente y guiarme durante todo el proceso. Por su disponibilidad e interés para resolver mis dudas y darme sugerencias para mejorar mi trabajo.
- A mi catedrático, Miguel Zea, por su disposición a resolver mis dudas y darme información útil para el desarrollo de este trabajo.
- A mis amigos, porque complementaron la experiencia de esta etapa por su disposición de apoyarnos y compartir las habilidades que cada uno tiene para resolver problemas, sin faltar las bromas y risas.

Índice

Lista de figuras

Lista de cuadros

Resumen

En el siguiente proyecto se desarrolló un algoritmos para coordinación de enjambre de drones *Crazyflie 2.0* para la ejecución de formaciones en un ambiente analizado por un sistema de captura de movimiento, OptiTrack. Los algoritmos se desarrollaron a través de la utilización de ecuaciones de consenso, la verificación de estos algoritmos se hizo por medio de simulaciones en Matlab.

La ejecución de formaciones con enjambre de drones presenta diversos retos y por eso este trabajo solucionó los problemas como colisiones y evasión de obstáculos. Esto proporcionará autonomía a cada dron, esto porque cada uno calcula en tiempo real la trayectoria que seguirá para evitar la colisión con otros drones.

Se han desarrollado distintos trabajos que buscan la ejecución de formaciones con *Crazyflie 2.0*, pero se han encontrado con diversos problemas como el aumento del error de la posición conforme aumentan la cantidad de drones. En otros casos se han trabajado proyectos que permiten el calculo de las trayectorias de los drones para ejecutar las formaciones pero sin utilizar control, pues pueden calcular trayectorias, pero no evadir obstáculos.

Este proyecto combina diversos temas, como sistemas de control, algoritmos de consenso que permitan al enjambre comunicarse para poder trasladarse o rotar, conocer las características del *Crazyflie 2.0* y del sistema OptiTrack para la implementación de la información que caracteriza a ambos sistemas y aplicar conceptos de robótica de enjambre para tomar en cuenta factores que reducen el óptimo funcionamiento del enjambre.

Abstract

In the following project, a *Crazyflie 2.0* drone swarm coordination algorithm was developed for the execution of formations in an environment analyzed by a motion capture system, OptiTrack. The algorithms were developed through the use of consensus equations, the verification of these algorithms was done by means of Matlab simulations.

The execution of drone swarm formations presents several challenges and that is why this work solved problems such as collisions and obstacle avoidance. This will provide autonomy to each drone, because each one calculates in real time the trajectory it will follow to avoid collision with other drones.

Different works have been developed that seek the execution of formations with *Crazyflie 2.0*, but they have encountered several problems such as the increase of the position error as the number of drones increases. In other cases, projects have been developed that allow the calculation of drone trajectories to execute formations but without using control, since they can calculate trajectories but not avoid obstacles.

This project combines several topics, such as control systems, consensus algorithms that allow the swarm to communicate in order to move or rotate, to know the characteristics of the *Crazyflie 2.0* and the OptiTrack system for the implementation of the information that characterizes both systems and to apply swarm robotics concepts to take into account factors that reduce the optimal performance of the swarm.

CAPÍTULO 1

Introducción

El estudio y desarrollo de sistemas multi-agente ha tomado un mayor protagonismo en los últimos años. Estos sistemas pueden formarse por distintos tipos de robots, por ello se han encontrado una gran cantidad de aplicaciones para esta rama de la robótica. En la Universidad del Valle de Guatemala se proporciona cursos como sistemas de control y robótica los cuales son la base en la cual se desarrollará el siguiente trabajo, el cual se enfoca en la solución específica de coordinación y formación de un enjambre de drones.

A continuación se presenta el proceso para el desarrollo de un algoritmo de coordinación de drones *Crazyflie 2.0*. Este algoritmo de desarrolla bajo la teoría de control moderno y grafos. Para ello se toma como inicio los algoritmos y resultados de trabajos previos, para su estudio y extracción de información que sea de utilidad. Con la información de utilidad, se realizan pruebas con los algoritmos de mayor simplicidad para analizar el comportamiento de la velocidad, trayectoria y energía utilizada por los agentes según el método utilizado, esto con el propósito de analizar qué cambios significativos presenta una formación en dos dimensiones con un espacio tridimensional.

Una vez realizadas las pruebas iniciales, se procede a realizar las formaciones de prueba con un controlador dinámico que permite, según el caso, utilizar el controlador adecuado. Adicionalmente se implementa un limite de velocidad en los algoritmos para evitar que al implementarse en físico, las velocidad que el algoritmo calcule, no exceda de los límites físicos de los actuadores del dron. Posteriormente se implementa una rutina que permite realizar con mayor efectividad la formación a los drones.

Este proyecto se divide, entonces, en las siguientes partes: estudio y desarrollo de algoritmo: en donde se busca a través de un proceso iterativo encontrar la ecuación que mejor se adapte al problema de la coordinación para formación de agentes; formación de prueba: en donde se va a estudia que tipo de grafo es exitoso para la aplicación y se proponen las formaciones de prueba para los algoritmos seleccionados; La métrica de evaluación para obtener datos estadísticos sobre qué tan exitosa fue la formación, la velocidad y energía de los agentes.

CAPÍTULO 2

Antecedentes

El *Crazyflie 2.0* es un cuadricóptero de código abierto de tamaño reducido [1]. Este dron ha sido utilizado por distintas instituciones educativas y de investigación. Gracias a la versatilidad del dispositivo, se han realizado distintas variaciones en sus aplicaciones de trabajo.

Figura 1: Cuadricóptero *Crazyflie 2.0* [1].

Varias instituciones e individuos que trabajar con estos drones, han implementado sistemas de control para su funcionamiento. Por ello se trabajan distintas simulaciones y pruebas para poder minimizar las pérdidas al momento de realizar las pruebas físicas. El uso de drones tiene un potencial para ejecución de tareas específicas en el mundo, por ello el poder darle instrucciones y que estos las ejecuten sin mayor percance es de suma importancia.

Se han trabajado distintos software que permite su control, tanto como dron individual como enjambre. Estos software han probado ser efectivos pero contienen deficiencias como la coordinación de distintos drones en ambientes con obstáculos.

2.1. Crazyswarm: software de código abierto para control de múltiples drones:

En 2017, Preiss, Wolfgang [2] y colaboradores trabajaron este software que permitía el control de múltiples drones. Este software utilizaba la información del entorno a partir de 14 cámaras VICON. La obtención de estos datos se realizaba a través de VICON Tracker. El desarrollo de esta plataforma se basa en la utilización de herramientas y librerías ROS, con lenguaje C++ en Ubuntu.

Durante la experimentación monitorearon el comportamiento del enjambre según la cantidad de drones. Como resultado se obtuvo que, entré más drones realizaban la tarea, el error de posición aumentaba. Esto se debe a que la aerodinámica del entorno cambiaba debido a los demás drones. En la Figura 2 se observa una formación de cuarenta drones y el sistema de cámaras VICON.

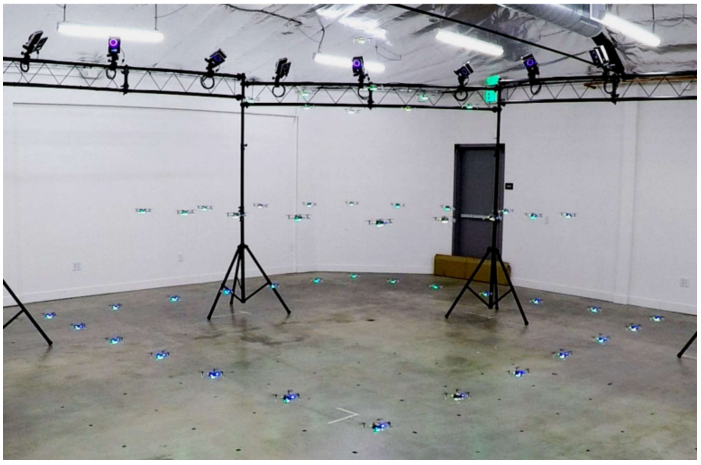


Figura 2: Formación de cuarenta Crazyflies en forma de pirámide [2].

2.2. Uso de distintos Crazyflies como UAV's:

Raúl Zahínos presentó resultados experimentales para el control de 9 Crazyflies 2.0 [3], comunicadas a partir de una computadora por medio de crazyradio PA con un alcance de hasta 1 km. Para poder obtener información del entorno se trabajó en un banco de pruebas de CATEC que les proporciona datos a partir de la convención ENU (East - North - Up). A pesar que el sistema controlado no es lineal, se utilizaron técnicas de control clásico, controlador PID, para poder trabajar con el sistema.

Inicialmente, la implementación del control de estos drones utilizaba el feedback del sistema VICON por medio de la librería Crazyflie-lib-python, software que proporciona bitcraze para el control de distintos drones por antena. El problema que surgió de esta librería es que no es viable utilizarla para trabajos a alta frecuencia. Por estos y demás problemas se migró a trabajar con el proyecto Crazyswarm. Con éste, aún presentaba un aumento en el error euclidiano al momento de trabajar a una mayor frecuencia y con varios drones pero era menor que con el software anterior. El resultado de este trabajo se puede

ver en las figuras 3 y 4.

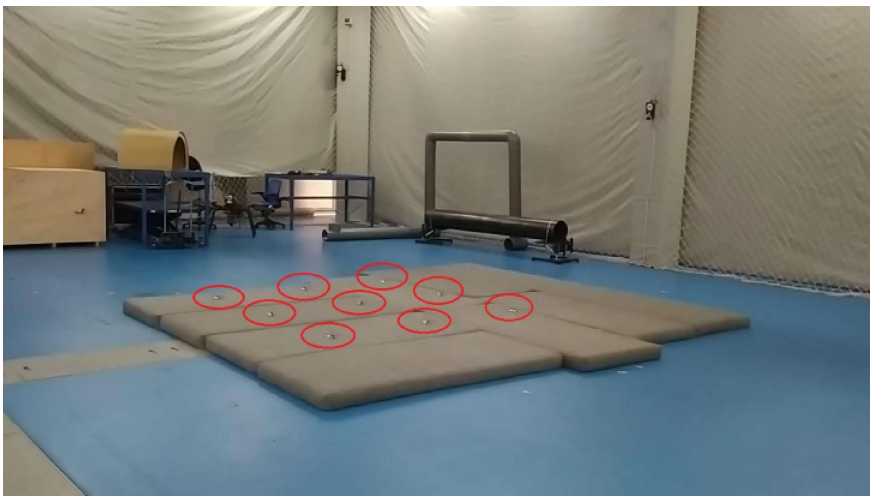


Figura 3: Nueve Crazyflies antes del despegue [3].

Figura 4: Nueve Crazyflies después del despegue [3].

2.3. Simulación de formación de drones:

Luka Jeroncic, en mayo del 2021, desarrolló un conjunto de simulaciones de enjambre de UAVs [4]. Para estas simulaciones, encontró que al trabajar con varios UAVs, se debía considerar que estos deben tener cooperación entre sí. Estas pruebas buscaban evitar colisiones entre ellos y las colisiones con el entorno.

Las simulaciones fueron desarrolladas en Unreal Engine 4 y AirSim, herramientas que permitían el uso del sensor que proporciona la pose del dron (orientación y posición). Estos parámetros pueden ser modificados en AirSim. Los drones cuentan con sensores que permite la lectura de objetos en un rango de 4 metros a la redonda. Se realizaron 15 iteraciones, los

datos obtenidos fueron la desviación estándar, el tiempo promedio en completar la formación y la cantidad de colisiones que sufrieron durante la formación.

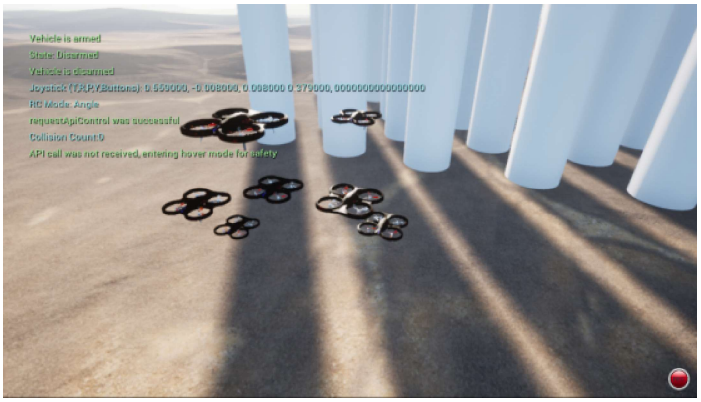


Figura 5: Pruebas de formación inicial con 6 drones [4].

2.4. Algoritmo para trayectorias de UAVs

En el 2020, Hao Zhuo y su equipo trabajaron un algoritmo de planificación de trayectorias para UAVs, este algoritmo se basa en restricciones exactas de posicionamiento para el sistema [5]. Este trabajo garantiza que el UAV pueda llegar a una posición indicada a través de la trayectoria mas corta desde el punto de salida hasta el punto final. Para esto se utilizan múltiples restricciones y la menor cantidad de correcciones de errores. La limitante de este trabajo es que no se tiene la retroalimentación del entorno por medio de un sistema de captura de movimiento, por lo que el algoritmo se basa en considerar el error de posicionamiento durante su vuelo debido a los factores presentes en el entorno. Esto lo trabajan a partir de puntos de corrección en el espacio de vuelo y así corregir el error, como se puede ver en la Figura 6. Con esto en mente, planifican la mejor trayectoria de forma que el UAV sea corregido por los puntos que se definieron como correctos para que el vuelo sea lo más corto y rápido posible.

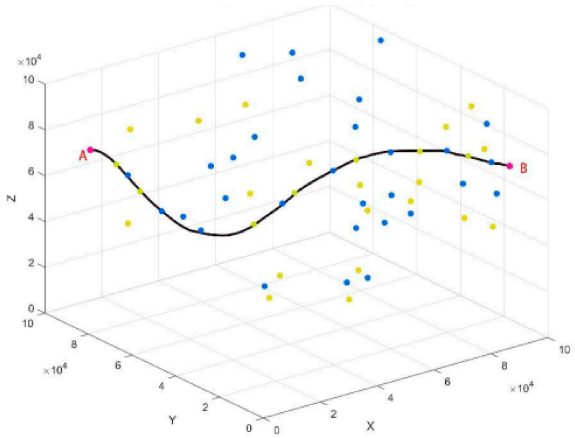


Figura 6: Algoritmo de trayectoria por medio de puntos [5].

2.5. Trabajos previos en UVG

Los trabajos realizados en UVG, permiten tener una base y ver los comportamientos del *Crazyflie 2.0* en distintos escenarios. Los trabajos nos otorgan la ventaja de poder modificar y estudiar los controladores de los drones.

En el 2019, Gabriel Martinez realizó el diseño e implementación de una plataforma para el *Crazyflie 2.0* con el propósito de tener un sistema controlado para poder configurar y realizar pruebas de algoritmos de control de actitud para el dron [6]. Este trabajó fue la base para que en el 2021, Francis Sanabria trabajara en la implementación del cuadricótero a esta plataforma. El diseño de la plataforma con el dron se puede ver en la Figura 7.

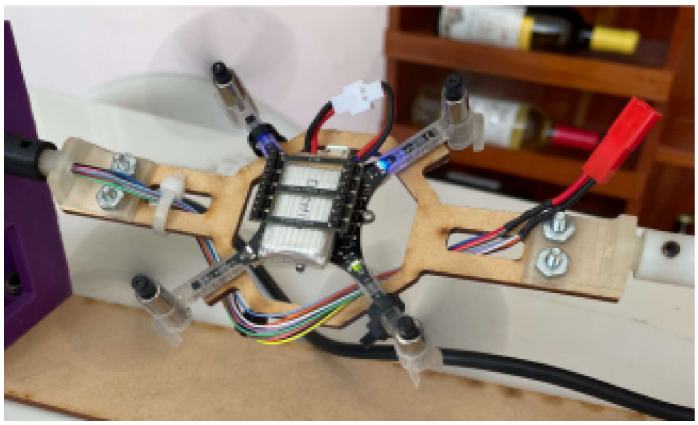


Figura 7: Plataforma con *Crazyflie 2.0* montado [7].

El trabajo de Sanabria permite configurar y estudiar el sistema de control del dron para entender cómo se orienta [7]. Esto lo logró trabajando con el control de actitud del *Crazyflie 2.0*. Para poder cumplir este objetivo, redactó una guía básica de cómo se utiliza el dron, en esta se indicaba qué pasos seguir para poder modificar el firmware del dron. Además, desarrolló una interfaz gráfica para poder controlar el dron de manera sencilla. La interfaz permite al usuario leer en tiempo real el ángulo de banqueo, modificar los parámetros del controlador PID y reiniciarlo a valores predeterminados del mismo y modificar el ángulo pitch al que se desea orientar el dron. En esta interfaz, también permite conectar, desconectar y tarar el dron. Como el objetivo de este proyecto era el estudio y análisis del comportamiento del dron, también cuenta con la función de guardar la data de la iteración. La interfaz se puede ver en la Figura 8. Las limitaciones de este trabajo fueron que no se tiene el comportamiento del dron con todos sus grados de libertad debido al diseño de la plataforma por lo que se tiene información del dron operando en un ambiente completamente controlado.

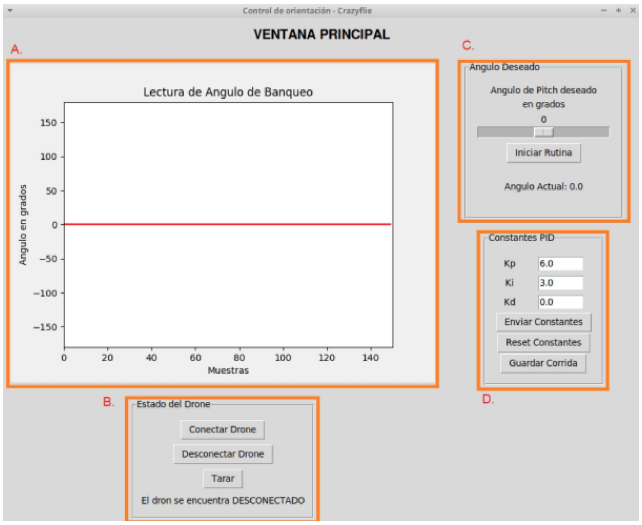


Figura 8: Interfaz gráfica [7].

Steven Castillo, trabajó una plataforma de pruebas en forma de *gimball* [8]. La ventaja de esta plataforma es que presenta 3 grados de libertad, que son los ángulos de vuelo del dron. Para el control de este dron, se utilizó el controlador de vuelo automático Pixhawk 1. Este controlador puede ser modificado a través de una herramienta en Matlab/Simulink. En esta herramienta modificaba los cambios de velocidad de los motores para posteriormente realizar análisis de esfuerzos de manera teórica y por simulaciones para las piezas del sistema.

Andrea Peña, en el 2019 desarrolló un algoritmo para el control de robots de dos ruedas, en su caso utilizó e-puck[9]. Para su trabajo, hizo pruebas con distintos controladores y así comprobar cuál sería más efectivo. Como resultado de su experimentación, utilizó 3 tipos distintos de controlador para aumentar la eficiencia de la energía y tiempo de formación, en dónde destaca el uso de un coseno hiperbólico para optimizar el uso de energía. Para las pruebas del controlador, trabajó el código en matlab para poder realizar simulaciones que le permitieran ver el comportamiento de los agentes. Cómo parte del trabajo, también obtuvo gráficos de cómo se comportaba la velocidad de cada agente con cada controlador y en la Figura 10 se puede observar la trayectoria que siguieron los agentes de inicio a fin. Adicionalmente implementó un control de velocidad para que no excediera los límites físicos del robot. Por último, trabajó en la formación de los robots implementando evasión de obstáculos, lo cuál era útil por su orientación de drones exploradores. Esto no lo pudo implementar de forma física pero, si trabajó la simulación en Webots cómo se puede observar en la Figura 9.

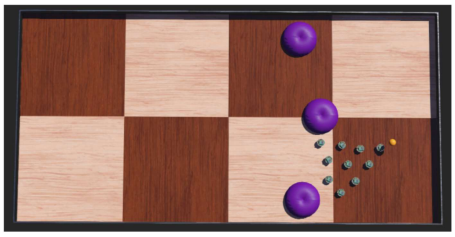


Figura 9: Simulacion en Webots de los algoritmos [9].

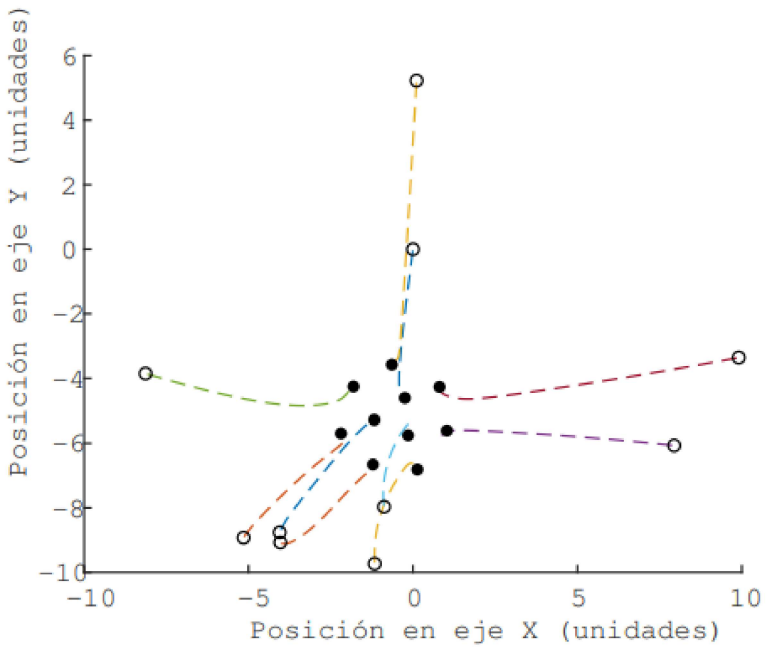


Figura 10: Trayectorias seguidas por los agentes para su formación [9].

Es importante mencionar que los trabajos previos no permiten su implementación para la realización de enjambres de drones, por lo que se restringue la posiblidad de utilizar los drones para ejecutar formaciones o trayectorias específicas.

CAPÍTULO 3

Justificación

El propósito de este proyecto es desarrollar algoritmos de enjambre de drones *Crazyflie 2.0* que utilicen un sistema de captura de movimiento OptiTrack con el fin de realizar las formaciones que se le soliciten de forma autónoma. Esto permite crear enjambres inteligentes utilizando sistemas de control para la ejecución de trayectorias, utilizando la realimentación del ecosistema Robotat.

Estos algoritmos evitan las colisiones entre los drones del enjambre, por lo que se pueden ejecutar distintas formaciones sin importar en qué parte del entorno se encuentra debido a que el algoritmo acerca a los agentes con la intención de mantener un radio entre todos los agentes.

Por medio de las simulaciones, se obtienen datos estadísticos que permitan ver cómo reaccionan los drones al algoritmo cuando se ejecuta, con el fin de ver cuántos drones podrán ser controlados, obtener información de cuánto esfuerzo realizan los drones y que tan precisa fue la formación.

Los enjambres de drones pueden ser utilizados para aplicaciones específicas, por ejemplo: en el campo de la agricultura, para monitorear, vigilar o aplicar plaguicidas al cultivo; en el campo de ingeniería civil, se pueden utilizar para realizar planos topográficos. La principal ventaja que este presenta es que, con un enjambre, se cubre un área de trabajo más grande con distintas perspectivas, por lo que el tiempo que demora en realizar la tarea se reduce.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar algoritmos de coordinación de enjambre de drones *Crazyflie 2.0* para la ejecución de formaciones, utilizando técnicas de control y el sistema de captura de movimiento OptiTrack.

4.2. Objetivos específicos

- Evaluar algoritmos de control y formación para drones *Crazyflie 2.0*.
- Validar los algoritmos desarrollados por medio de simulaciones computarizadas.
- Definir los parámetros que necesita transmitir el sistema OptiTrack al *Crazyflie 2.0* para que el dron pueda conocer su entorno.
- Realizar pruebas físicas con los algoritmos para verificar el correcto funcionamiento del *Crazyflie 2.0*.
- Obtener estadísticas de posición y trayectoria del *Crazyflie 2.0* durante la ejecución de la formación.

CAPÍTULO 5

Alcance

El resultado de este trabajo proporciona un algoritmo funcional que permite la coordinación de un enjambre de drones desde una posición aleatoria en el espacio. Este algoritmo utiliza un sistema de captura de movimiento con el propósito de obtener la posición de cada dron y el punto del entorno en dónde se realizará la formación.

El conjunto de simulaciones fueron desarrolladas y verificadas a través de Matlab en dónde se realizan dos formaciones distintas. Es importante resaltar que en este *software* no se toman en cuenta las limitaciones físicas que proporcionan los drones y el sistema de captura de movimiento a utilizar. Cada formación requiere de 8 agentes cada una.

Se tienen simulaciones con un escenario parecido a la plataforma de pruebas con el sistema OptiTrack que está en la Universidad del Valle de Guatemala, probando así el algoritmo. Estas se desarrollaron para el simulador Webots de Cyberbotics, el cuál contiene una versión del *Crazyflie 2.0*, el cuál fue desarrollado por Bitcraze. Este modelo permite realizar simulaciones con el dron, por lo que se pueda estudiar cómo reacciona ante el algoritmo y obtener resultados que se acerquen más a la realidad. Es importante mencionar que este simulador aún mantiene un entorno ideal, por lo que solo se puede evaluar el éxito del algoritmo descartando fallas por factores externos y fallos relacionados al *Crazyflie 2.0*.

Para el caso de las pruebas físicas, no se pudieron realizar porque la integración del sistema OptiTrack y el firmware del *Crazyflie 2.0* no fue completó en el lapso previsto, por lo que no se contó con el tiempo y el sistema para realizar las pruebas físicas.

CAPÍTULO 6

Marco teórico

El algoritmo desarrollado se basa, principalmente, en la utilización de la teoría de grafos y control moderno. Es por ello que en la siguiente sección se describen conceptos importantes que se utilizarán en el resto del trabajo. Adicionalmente, se proporciona información útil de otros temas como el sistema OptiTrack, el cual también tiene participación en este trabajo.

6.1. Sistema OptiTrack

El sistema de captura de movimiento, OptiTrack, consiste en múltiples cámaras sincronizadas alrededor de un volumen de captura objetivo. Éste captura imágenes 2D con cada cámara. Posteriormente se calculan las posiciones 2D y los datos de posición de cada cámara se comparan para calcular las posiciones en 3D por medio de triangulación. Estas posiciones en 2D se obtienen por medio la detección de la luz reflejada que es emitida por las cámaras, como se puede ver en la Figura 11. Por esto se recomienda minimizar la iluminación ambiental para evitar interferencias como luz solar [10].

El conjunto de cámaras se recomienda instalarse alrededor de la periferia del volumen objetivo, las cámaras a distinta altura para obtener distintas perspectivas del volumen. Las cámaras Primex 41 cuentan con las siguientes especificaciones técnicas:

- **Resolución:** 2048 $\times$ 2048
- **Velocidad de fotogramas:** 180 Hz.
- **Precisión 3D:** +/- 0.10 mm.
- **Rango para marcadores pasivos:** 30 mm.
- **Rango para marcadores activos:** 45 m.

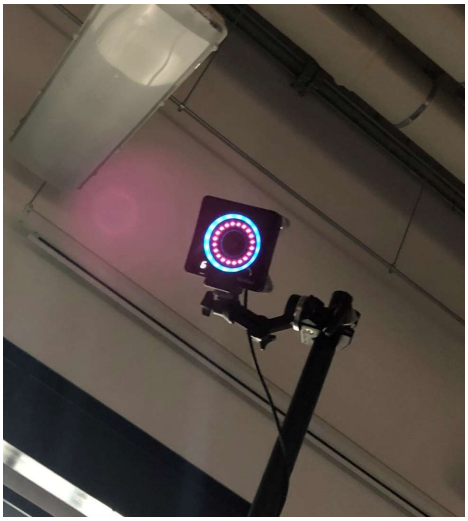


Figura 11: Cámara Prime x 41 del sistema OptiTrack en UVG.

6.2. *Crazyflie 2.0*

El *Crazyflie 2.0* es un dron de 4 motores, cuadricótero, de bajo costo y ligero. Este permite ser controlado por medio de Bluetooth LE y por radiofrecuencia, utilizando un componente adicional, Crazyradio PA. El *Crazyflie 2.0* se carga en 40 minutos y permite un tiempo de vuelo de 7 minutos. Tiene integrado diversos sensores como un giroscopio, acelerómetro y magnetómetro de 3 ejes cada uno, así como un sensor de presión.

6.3. Teoría de grafos

Un grafo se forma por medio de vértices y aristas, las cuales se encargan de conectar los vértices. Como uso general, se presentan como una pareja de conjuntos (V, E), en donde V es el conjunto que define a los vértices y E el de aristas, en donde E se representa a través de pares, de forma que cumple $(a, b) \in V$ [11].

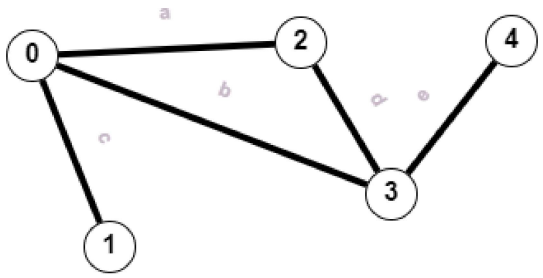


Figura 12: Representación de un grafo no dirigido

Si se observa la Figura 12, la forma formal de escribir este grafo sería: $A = (V, E)$ en dónde $V = 0, 1, 2, 3, 4$, siendo este el conjunto de los vértices y $E = (0, 1), (0, 2), (0, 3), (2, 3), (3, 4)$, siendo este el conjunto de aristas.

6.3.1. Definiciones básicas

Clasificación de los grafos:

- Simple: este grafo consiste en que, cualquier par de vértices tienen, como máximo, una arista que los une.
- Conexo: un grafo es conexo si para cualquier par de vértices (a, b) existe, por lo menos, un camino entre ayb.
- Completo: este tipo de grafo tiene para todo par de vértices (a, b) una arista que los une. Quiere decir que todos los vértices están unidos entre sí.
- Ponderado: este tipo de grafos tienen un número (magnitud) definido en cada arista, es decir, un peso.

Representación de grafos como matrices

Los grafos se utilizan en distintas disciplinas, y es por ello que su representación gráfica no es práctica para su identificación de propiedades. Por esto se utiliza una representación matricial, la cual permite representar relaciones entre vértices y aristas. Existen una variedad de matrices que se encargan de representar alguna propiedad con el propósito de simplificar la manipulación y análisis. Los tipos de matrices asociadas a un grafo son:

- Matriz de incidencia: esta matriz contiene la información asociada con las aristas (columnas), respecto a los vértices (renglones). Su representación cambia si es un grafo dirigido o no. Para el primer caso, si la arista conecta a un vértice $a_{ve} = 1$, de lo contrario $a_{ve} = 0$. Para el segundo caso, para la conexión entre el vértice y la arista, se toma en consideración si esta entra o sale del vértice, si la arista entra $a_{ve} = 1$, en caso contrario $a_{ve} = -1$. Esta matriz se identifica con I. Tomando como ejemplo el grafo de la Figura 12, su matriz de incidencia es:

  No dirigido:

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Matriz de grado: es una matriz diagonal con dimensiones $v \times v$ en donde muestra el grado de cada vértice. Por grado se refiere a cuántos otros vértices está conectado. Esta matriz se identifica con D. Tomando como ejemplo el grafo no dirigido de la Figura 13, su matriz de grado es:

$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

- Matriz de adyacencia: es una matriz de dimensiones $v \times v$, en donde muestra la conexidad entre vértices, es decir, si existe una arista entre el vértice a y b, entonces el elemento $v_{ab} = 1$. Esta matriz se identifica con A. Tomando como ejemplo el grafo no dirigido de la Figura 13, su matriz de adyacencia es:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- Matriz laplaciana: esta matriz es el resultado de realizar operaciones entre otras matrices que representan a un grafo. Esta matriz se identifica con L. Para el caso de los grafos no dirigidos, se obtiene la matriz laplaciana al realizar la operación $L = D - A$, para el caso de los grafos dirigidos se obtiene con $L = II^T$. Tomando como ejemplo el grafo no dirigido de la Figura 13, su matriz de adyacencia es:

$$L = \begin{bmatrix} 2 & -1 & -1 & 0 & 0 & 0 \\ -1 & 4 & -1 & -1 & 0 & -1 \\ -1 & -1 & 3 & 0 & -1 & 0 \\ 0 & -1 & 0 & 2 & 0 & -1 \\ 0 & 0 & -1 & 0 & 2 & -1 \\ 0 & -1 & 0 & -1 & -1 & 3 \end{bmatrix}$$

6.3.2. Grafo de formación

Este grafo representa la configuración deseada según la geometría que la formación requiere. En este grafo se incorpora el concepto de rigidez, esto se refiere a las distancias que deben mantener los vértices para definir una estructura rígida.

Con esto en mente, se incorpora el concepto de grafo mínimo y completamente rígido. Un grafo completamente rígido es un grafo completo que define las distancias entre el vértice i y todos los demás. Este tipo de grafo resulta tener una complejidad alta y una robustez baja puesto que contiene una gran cantidad de restricciones. Mientras que un grafo mínimamente rígido contiene las distancias necesarias para mantener la rigidez de una formación, esto quiere decir que si se elimina alguna arista, la rigidez se pierde.

6.4. Algoritmos de consenso

Cuando se tiene una red de agentes dinámicos, es común encontrarse con el problema de consenso. La solución para este problema consiste en que, cada agente debe llegar a un consenso, esto se refiere a un estado en el que se consigue un acuerdo para poder alcanzar una meta o valor en común según el interés dependiente del estado de los agentes que conforman la red. Este estado se alcanza cuando se tiene constante comunicación entre la red por medio de enlaces de comunicación o sensores. Este intercambio de información se le llama algoritmo de consenso. Los algoritmos de consenso tienen el propósito de que cada agente actualice su información a partir de la información de los otros agentes [12].

Para estas redes multi-agentes, se trabaja la estructura con grafos para indicar que tipo de comunicación habrá entre la red. Se utilizan dos tipos grafos, los dirigidos y los no dirigidos. En la Figura 13 se observan ejemplos de ambos tipos. Para el primer caso, cada arista es representada por una flecha, en donde la cola indica el agente que envía la información, mientras que la punta indica el agente receptor de la información. Para el segundo caso, las aristas son lineas continuas, estos se utilizan en redes en donde la comunicación entre los agentes es bidireccional [13].

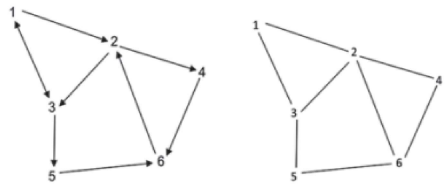


Figura 13: Ejemplos de grafo dirigido (izquierda) y no dirigido (derecha) [13].

En los algoritmos se requiere de una estructura de intercambio de información, la cuál puede ser una estructura sin líder o una con líder y seguidores. La primera estructura funciona de forma que, cada agente se comunica con sus vecinos, y estos no siguen a ningún agente en específico. Esta estructura permite que, aunque no se le de prioridad a una orden deseada a los agentes, eventualmente la salida de cada agente converge a un valor común. En la segunda estructura mencionada, el agente líder puede afectar a los seguidores siempre que se encuentre en su conjunto vecino pero, presenta el problema que no recibe retroalimentación de sus seguidores.

6.4.1. Modelo de votantes

El modelo de votantes hace que un agente considere las situación actual de cada vecino, entonces elige al azar a un vecino y cambia su situación por la del vecino, si es distinta a la propia. Este modelo es bastante robusto, pues este modelo ayuda a converger en la decisión correcta, por lo que presenta un alto grado de precisión [14].

6.4.2. Regla de la mayoría

Este modelo es de los más sencillos para la toma de decisiones colectivas. Para este modelo, el agente comprueba a su grupo de vecinos y cuenta la recurrencia de cada opción. Los agentes pueden cambiar su decisión a la opción más frecuente que encontró entre sus vecinos, esta regla es rápida pero menos precisa que el modelo del votante [14].

6.5. Teoría de control

La rama de control trabaja con el comportamiento de sistemas dinámicos, de esto se forma un sistema de control que consiste en la integración de diversos sistemas y procesos que buscan ser aplicados a sistemas que tienen una respuesta inestable. Este sistema tiene como objetivo proporcionar una salida con un comportamiento deseado. Estos sistemas requieren de una referencia, la cuál es la salida deseada del sistema. Un sistema de control, en pocas palabras, manipula las entradas que tiene un sistema para poder obtener un comportamiento específico en la salida [15]. En la Figura 14, se observa el sistema de control más utilizado, se conoce como lazo canónico de control.

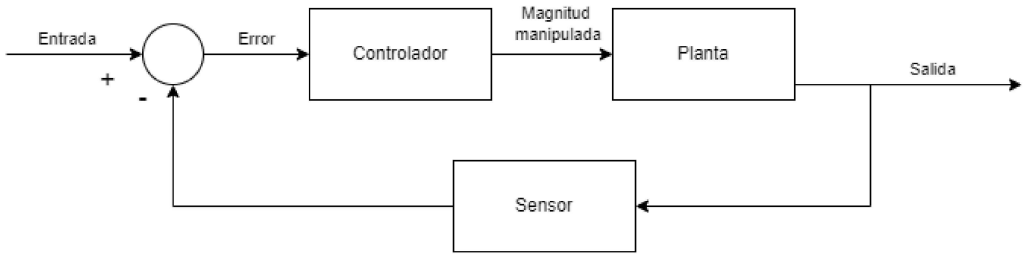


Figura 14: Lazo canónico de control

6.5.1. Control de formación

Para la formación, el controlador se orienta para dotar a los agentes la capacidad de poder reunirse en una estructura determinada. Con esto se busca que se puedan formar según la posición relativa de los demás agentes respecto a sus vecinos. Esto se logra al modificar la velocidad de cada agente respecto al error que existe entre la posición relativa esperada y la posición actual. Por ello se describe a continuación la teoría utilizada para realizar las modificación mencionada a la velocidad de los agentes [15].

De los principales problemas para realizar el control de sistemas multiagentes resulta ser el de consenso, lo cual radica en que un grupo de agentes pueda reunirse en un mismo lugar. Este problema debe tener redes de intercambio de información entre los n agentes que conforman el grupo. La información relevante es la tasa de cambio de cada agente depende de la suma del estado relativo de los otros agentes, de esta afirmación se obtiene la ecuación:

$$\dot{x}_i(t) = \sum_{j \in N(i)} (x_i(t) - x_i(t)) i = 1, 2..., n \tag{1}$$

en donde, $N(i)$ es el conjunto de agentes vecinos del agente i en el conjunto de agentes [16].

La ecuación 1 es la base para obtener la solución del problema canónico en sistemas multiagentes, también conocido como el problema de rendezvous. Este problema tiene como fin que un grupo de agentes coincidan en una misma posición a partir de conocer la posición relativa de sus vecinos o demás agentes. Tomando la ecuación de consenso y añadiendo pesos, se puede trabajar problemas más complejos que el de rendezvous, puesto que ya se trabaja con tensión en las aristas entre agendas. De aquí se trabaja con la función de reducir la tensión entre agentes tomando en cuenta que la tensión de una arista la define la siguiente ecuación:

$$\epsilon = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij} \epsilon_{ij}(||x_i - x_j||) \tag{2}$$

al derivar se obtiene

$$\frac{\partial \epsilon_{ij}}{\partial x_i} = w_{ij}(||x_i - x_j||)(x_i - x_j) \tag{3}$$

De la ecuación 3 se despeja para w_{ij}, este peso varía según el problema que se resolverá. Por lo tanto, se obtienen las siguientes expresiones que definen la ecuación de consenso,

$$\epsilon_{ij} = \frac{1}{2}||x_i - x_j||^2 \tag{4}$$

$$w_{ij} = 1 \tag{5}$$

$$\dot{x}_i = -\sum_{j \in N(i)} (x_i - x_j) i = 1, 2..., n \tag{6}$$

Con 4, se puede tomar como base para distintas aplicaciones en sistemas multiagente, a continuación se presentan algunos ejemplos de las modificaciones aplicadas a la ecuación de consenso para solucionar problemas específicos:

- Velocidad normalizada (consenso bio-inspirado):

$$\epsilon_{ij} = ||x_i - x_j|| \tag{7}$$

$$w_{ij} = \frac{1}{||x_i - x_j||} \tag{8}$$

$$\dot{x}_i = -\sum_{j \in N(i)} \frac{(x_i - x_j)}{||x_i - x_j||} i = 1, 2..., n \tag{9}$$

- Control de formación:

$$\epsilon_{ij} = \frac{1}{2}(||x_i - x_j|| - d_{ij})^2 \tag{10}$$

$$w_{ij} = \frac{||x_i - x_j|| - d_{ij}}{||x_i - x_j||} \tag{11}$$

$$\dot{x}_i = -\sum_{j \in N(i)} \frac{(||x_i - x_j|| - d_{ij})(x_i - x_j)}{||x_i - x_j||} i = 1, 2..., n \tag{12}$$

- Mantenimiento de conectividad:

$$\epsilon_{ij} = \frac{||x_i - x_j||^2}{\triangle - ||x_i - x_j||} \tag{13}$$

$$w_{ij} = \frac{2\triangle - ||x_i - x_j||}{(\triangle - ||x_i - x_j||)^2} \tag{14}$$

$$\dot{x}_i = -\sum_{j \in N(i)} \frac{(2\triangle - ||x_i - x_j||)(x_i - x_j)}{(\triangle - ||x_i - x_j||)^2} i = 1, 2..., n \tag{15}$$

6.6. Robótica de enjambre

La robótica de enjambre es el estudio del diseño de un gran número de agentes relativamente sencillos para que ejecuten un comportamiento colectivo a partir de las interacciones entre los agentes o entre los agentes y el entorno. Las interacciones tienen un papel clave en los enjambres, pues es necesario que cada agente tenga las capacidades de detección local y comunicación [14].

El rendimiento medio de un enjambre, depende del tamaño del enjambre si el espacio de trabajo se mantiene constante. Esto sucede porque, cuando se agregan más robots al sistema, la densidad de agentes va aumentando, y llegada una determinada cantidad, se tendrá el rendimiento óptimo del enjambre. Si pasado este punto, sigue aumentando la cantidad de robots, el rendimiento disminuye porque hay una mayor interferencia entre cada uno y esto ralentiza las acciones, esto puede aumentar hasta el punto que el enjambre no pueda moverse debido a las interferencias [17].

CAPÍTULO 7

Desarrollo del algoritmo de formación

El capítulo a continuación presenta el desarrollo del algoritmo que permitirá por la implementación de control y teoría de grafos la formación de los agentes. Las simulaciones del algoritmo se desarrollaron en Matlab ya que es una herramienta que permite realizar cálculos complejos y permite representar información de forma sencilla.

7.1. Grafo para formaciones

Para la definición del modelo matemático a utilizar, se realizaron distintas pruebas con grafos con distinto nivel de rigidez, esto con el propósito de encontrar el que mejor se adapte para la formación de agentes en un espacio tridimensional. Para esto se propuso un grafo totalmente rígido, mostrado en la Figura 15, de 8 nodos.

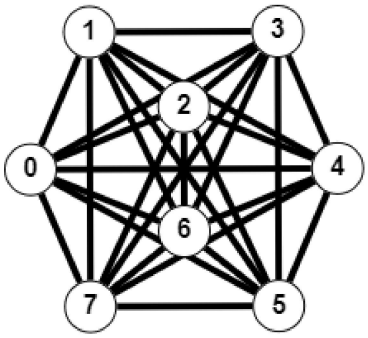


Figura 15: Grafo de prueba para formaciones.

Matriz de incidencia

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

7.1.1. Matrices de adyacencia

Para el grafo de la Figura 15, se definieron dos matrices de adyacencia totalmente rígidas, es decir, dos matrices de adyacencia con distintos pesos para poder ver el comportamiento según la estructura de la formación.

Matriz de adyacencia rígida para la formación 1

$$\begin{bmatrix} 0 & 1 & 1 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{2} & \sqrt{3} \\ 1 & 0 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{3} & 1 & \sqrt{2} \\ 1 & \sqrt{2} & 0 & 1 & \sqrt{2} & 1 & \sqrt{3} & \sqrt{2} \\ \sqrt{2} & 1 & 1 & 0 & \sqrt{3} & \sqrt{2} & \sqrt{2} & 1 \\ 1 & \sqrt{2} & \sqrt{2} & \sqrt{3} & 0 & 1 & 1 & \sqrt{2} \\ \sqrt{2} & \sqrt{3} & 1 & \sqrt{2} & 1 & 0 & \sqrt{2} & 1 \\ \sqrt{2} & 1 & \sqrt{3} & \sqrt{2} & 1 & \sqrt{2} & 0 & 1 \\ \sqrt{3} & \sqrt{2} & \sqrt{2} & 1 & \sqrt{2} & 1 & 1 & 0 \end{bmatrix}$$

Matriz de adyacencia rígida para la formación 2

$$\begin{bmatrix} 0 & 2 & 2 & \sqrt{3} & 2 & \sqrt{8} & \sqrt{3} & \sqrt{7} \\ 2 & 0 & 2 & 1 & \sqrt{8} & 2 & \sqrt{8} & \sqrt{5} \\ 2 & 2 & 0 & 1 & \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} \\ \sqrt{3} & 1 & 1 & 0 & \sqrt{7} & \sqrt{5} & \sqrt{5} & 2 \\ 2 & \sqrt{8} & \sqrt{8} & \sqrt{8} & 0 & 2 & 2 & \sqrt{3} \\ \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} & 2 & 0 & 2 & 1 \\ \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} & 2 & 2 & 0 & 1 \\ \sqrt{7} & \sqrt{5} & \sqrt{5} & 2 & \sqrt{3} & 1 & 1 & 0 \end{bmatrix}$$

La primer configuración presentada es un cubo, el cual se puede ver como ejemplo en la Figura 16, la segunda formación es un prisma triangular, que corresponde a la Figura 17.

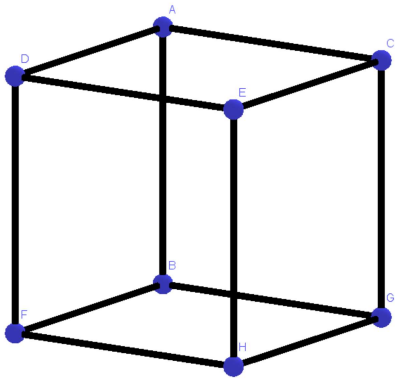


Figura 16: Primera configuración de los agentes, forma cúbica.

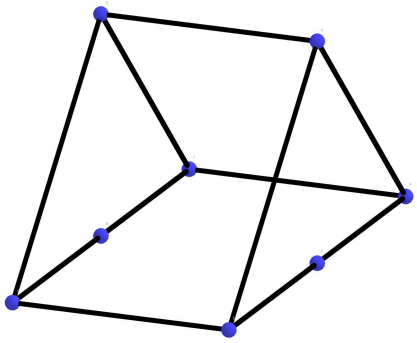


Figura 17: Segunda configuración de los agentes, prisma triangular.

En este caso, cada arista del grafo representa las distancias que deben mantener los agentes para realizar la formación, lo cual determina realmente la comunicación entre cada agente del enjambre. Es importante resaltar que al momento de utilizar los grafos completos, se está definiendo una red de comunicación compleja debido a que cada robot debe saber la posición de todos los agentes. Es por ello que adicional al controlador, en Matlab se realiza un sub algoritmo para que los agentes empiecen en posiciones aleatorias en un espacio de tres dimensiones.

7.2. Métrica de evaluación

El proceso del desarrollo del algoritmo conlleva una gran cantidad de iteraciones para poder comprobar qué tan efectivo está siendo el controlador según la posición en la que empiezan los robots del enjambre. Es por ello que se define un conjunto de métricas para poder comparar y verificar cómo se comporta el algoritmo. Este conjunto consta de cuánto se aproxima la formación realizada por los agentes y la deseada y el consumo de energía de los agentes[9].

7.2.1. Energía utilizada en el sistema

Para esta métrica, se aprovecha la relación entre la energía y la velocidad de los agentes, para ello se trabajó con una aproximación para la integral de la velocidad, presentada en la Ecuación 16

$$\sum_{i=1}^{n}\sum_{k=0}^{T}\dot{x}_i k \tag{16}$$

Esta métrica es relevante puesto que los agentes a utilizar tienen un tiempo de vuelo y energía limitado, por lo tanto es importante reducir el consumo de energía para poder aprovechar al máximo este recurso. Para poder tomar una rúbrica con mayor información, se trabaja con 4 rangos de magnitud. El menos a 50, entre 50 y 100, entre 100 y 200 y el último que pertenece al rango mayor a 200. Con esto se podrá comprobar cómo varía el uso de energía según las modificaciones que se realizaron al algoritmo. Por último, se obtendrá el promedio de la energía total de cada iteración para conocer un dato general de la simulación.

7.2.2. Éxito en la formación

El principal factor para la métrica es si la formación cumple realmente con los pesos que se establecieron en la matriz de adyacencia. Para ello, se utiliza el error cuadrático medio:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_{ij}-d_{ij})^2 \tag{17}$$

en donde x_{ij} representa la distancia instantánea entre el agente i y el agente j , d_{ij} es la distancia que deben tener los agentes entre sus adyacentes, en este caso es la posición ij de la matriz de adyacencia. Para esto, como se trabajó con grafos totalmente rígidos, se tiene la rúbrica que la formación debe tener un error cuadrático medio menor a los límites indicados, en este caso es, si se tiene un error cuadrático menor o igual a 0.001, la formación fue exitosa, de lo contrario, se considera fallida.

7.3. Ecuaciones de consenso básicas

Como base del algoritmo desarrollado, es importante conocer los distintos tipos de ecuaciones existentes para poder conocer el comportamiento que tendrían. Por ello se utilizó la ecuación de consenso, la cual permite la convergencia de los robots en un punto. Posterior a su simulación, se trabajó con la ecuación de control de formaciones para la realizar la geometría indicada en la matriz de adyacencia. Debe mencionarse el problema canónico de Rendezvous, que consiste en el problema de cómo hacer converger en una posición o lugar común y específico a N agentes[16]. Es por ello que se puede observar en ambas simulaciones

que la formación y consenso se lleva a cabo en el centroide de la posición inicial de los robots. Para los siguientes experimentos, la longitud/posición, se mide en unidades arbitrarias (u.a.), para el tiempo, segundos.

7.3.1. Ecuación de consenso lineal

En la Figura 18, se puede observar cómo funciona la ecuación de consenso lineal 4, en dónde no presenta cambios bru respecto a la posición y todos los agentes llegan a converger al mismo punto. En la Figura 19 se tiene el registro de la velocidad de cada agente, aquí se puede observar que, al converger al mismo punto, las velocidades convergen en 0, esto debido a que los agentes se encuentran en el punto deseado, por lo tanto la velocidad tiende a 0.

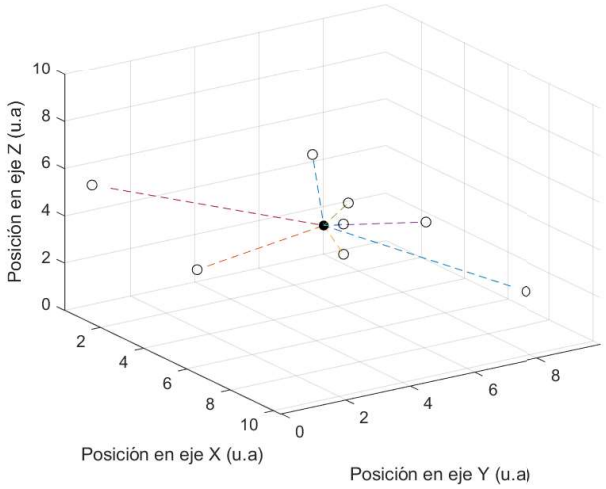


Figura 18: Trayectorias de los agentes con ecuación de consensus.

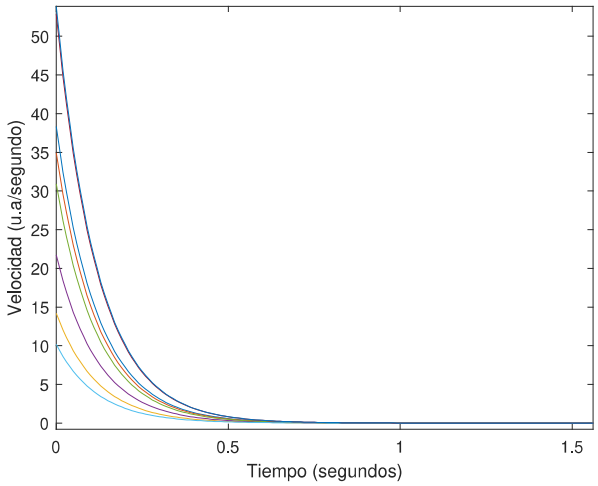


Figura 19: Magnitud de la velocidad de los agentes con ecuación de consensus.

7.3.2. Ecuación de formación

En la Figura 20, se ve el resultado de la ecuación de formación 11, en dónde los agentes cumplen con realizar la formación indicada. Al igual que la ecuación de consenso, sus velocidades también convergen a 0. Para ambos casos se mantiene que se forman en el centroide de la posición inicial de los robots. Para este experimento, se utilizó la matriz de adyacencia de la formación 1.

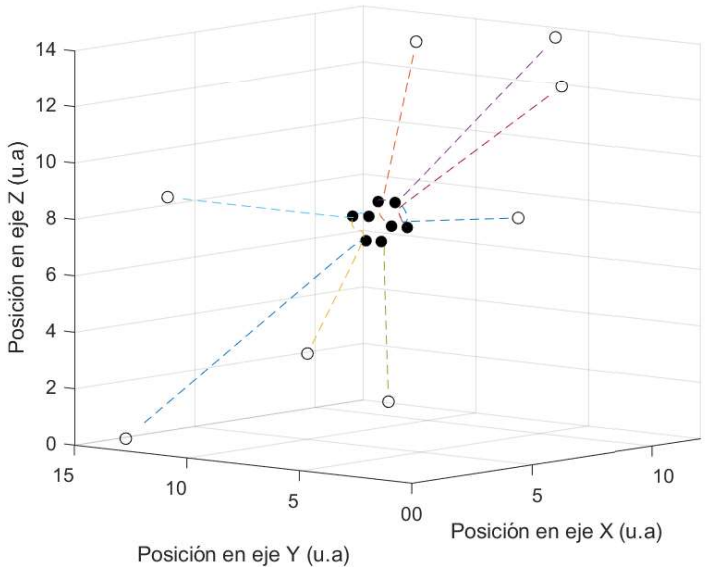


Figura 20: Trayectorias de los agentes con ecuación de formación.

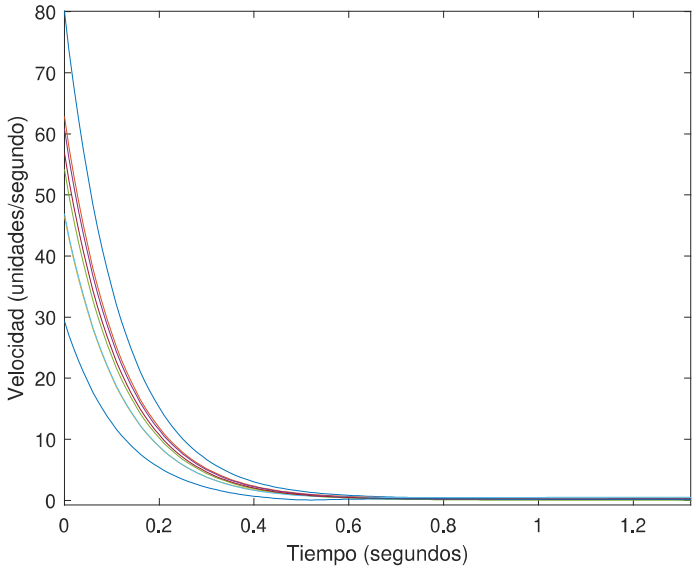


Figura 21: Magnitud de la velocidad de los agentes con ecuación de formación.

Al obtener los resultados de cómo se están comportando los agentes ante las modificaciones de la ecuación de consenso, se continúa realizando pruebas con la combinación de otros modelos para poder obtener más comportamientos y así poder examinar entre los experimentos, cómo optimizar el algoritmo.

7.3.3. Ecuaciones de consenso modificadas

Andrea generó dos ecuaciones a partir de la combinación de las ecuaciones básicas [9]. Estas dos ecuaciones consisten en: La primera ecuación 18 tiene como objetivo principal acercar a todos los agentes hasta una distancia arbitraria, la segunda ecuación 19, consiste en una combinación entre evasión de colisión y control de formación. Tiene como principal función que los agentes realicen la formación deseada sin que interfieran en el radio físico de los otros agentes.

La ecuación de consenso con pesos tiene otras aplicaciones más relevantes, es por ello que se manipuló la tensión entre agentes, esto con el propósito de aprovechar las ventajas que cada aplicación proporcionaba. Para este caso se trabajó con la ecuación de evasión de colisiones, puesto que permitía que se formaran con la restricción que, cada robot no entre en el radio físico del otro y con la ecuación de formación, que permite que los agentes se posicionen al lado de sus vecinos para que puedan formarse según la formación requerida.

$$w_{ij} = \frac{(||x_i - x_j|| - 2r)}{(||x_i - x_j|| - r)^2} \tag{18}$$

$$w_{ij} = \frac{4(||x_i - x_j|| - d_{ij})(||x_i - x_j|| - r) - 2(||x_i - x_j|| - d_{ij})^2}{(||x_i - x_j|| - r)^2(||x_i - x_j||)} \tag{19}$$

Por ello, se trabajó el algoritmo con una ecuación dinámica, esto consiste en que, primero se acciona la ecuación para que los agentes se acerquen hasta un radio en dónde cada uno estuviera lo suficiente cerca del otro para poder empezar a formarse con el peso indicado del grafo, una vez en esta distancia, se cambia al controlador combinado mencionado anteriormente para que se ejecute la formación. El resultado de esta combinación se puede observar en la figuras 22 y 24, en dónde se cumple la formación definida en la matriz de adyacencia. Posteriormente, se puede visualizar en la figuras 23 y 25, cómo se comporta la velocidad de los agentes, en dónde se puede apreciar el cambio de velocidad debido al cambio de la ecuación.

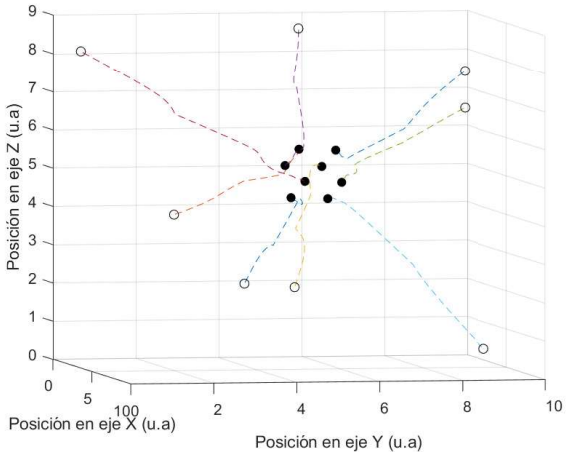


Figura 22: Trayectorias de los agentes con ecuación combinada para la formación 1.

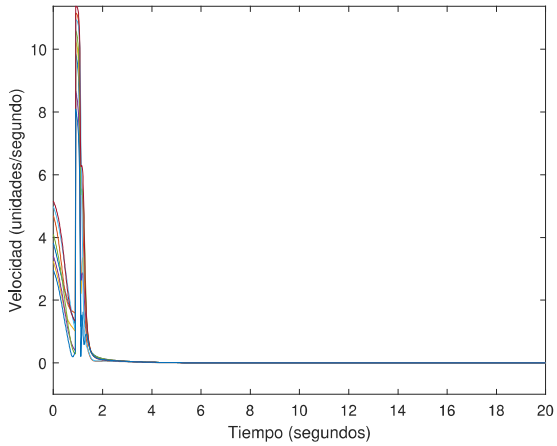


Figura 23: Magnitud de la velocidad de los agentes con ecuación combinada para la formación 1.

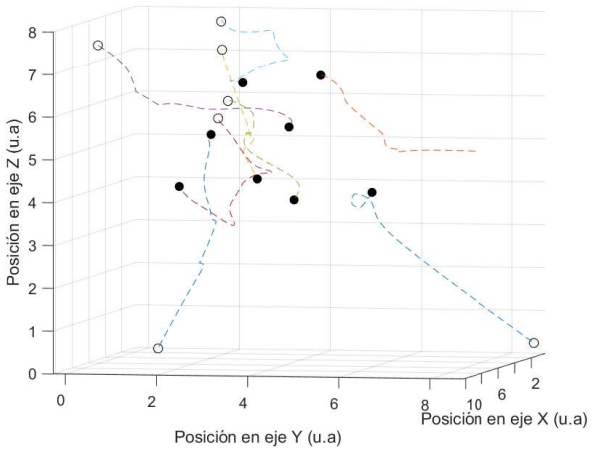


Figura 24: Trayectorias de los agentes con ecuación combinada para la formación 2.

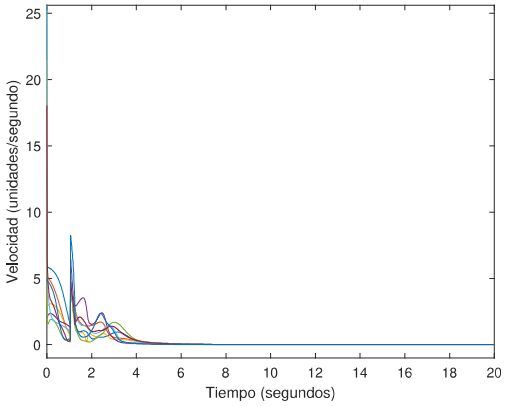


Figura 25: Magnitud de la velocidad de los agentes con ecuación combinada para la formación 2.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificado	1	100	68	32
Modificado	2	100	69	31

Cuadro 1: Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico.

Modelo	Grafo	$E < 50$	$50 > E < 100$	$100 < E < 200$	$E > 200$	Promedio
Modificado	1	29	56	6	9	1.23 e+03
Modificado	2	6	74	9	11	1.42 e+03

Cuadro 2: Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación.

Estos resultados nos permiten observar que con el algoritmo planteado están funcionando satisfactoriamente, en el Cuadro 1 se puede observar que se obtuvo un 71 % de éxito para la primera formación y un 63 % para la segunda, esto se debe a que la segunda formación es una geometría con mayor complejidad.

En las figuras 23 y 25 se puede observar que las magnitudes iniciales de la velocidad de los agentes son altas a comparación del comportamiento que presentan durante el resto de la simulación, por lo tanto, se requiere la implementación de un límite de velocidad para poder estudiar el comportamiento con agentes no ideales. En el Cuadro 2 podemos observar qué la mayoría de formaciones se encuentra en el segundo rango más bajo de uso de energía, pero el consumo promedio se ve disparado puesto que, los datos del rango mayor a pesar de ser pocos, resultan consumiendo una magnitud atípica de energía, por lo que, si en alguna prueba ocurre esto primero, la energía almacenada del robot puede consumirse por completo. Esto se respalda al observar que, al inicio, algunos agentes tienen una velocidad muy alta al inicio, por lo que se traduce a un consumo alto de energía.

Límite de velocidad

Partiendo de los resultados obtenidos anteriormente, se agregó un límite de velocidad de manera que, la velocidad del agente se mueva en la dirección que corresponde sin que la velocidad sea mayor al límite establecido. Esto se trabajará con el vector unitario de la velocidad obtenida por la ecuación de consenso. Por lo que, se realiza un mapeo de la velocidad con el límite establecido [18].

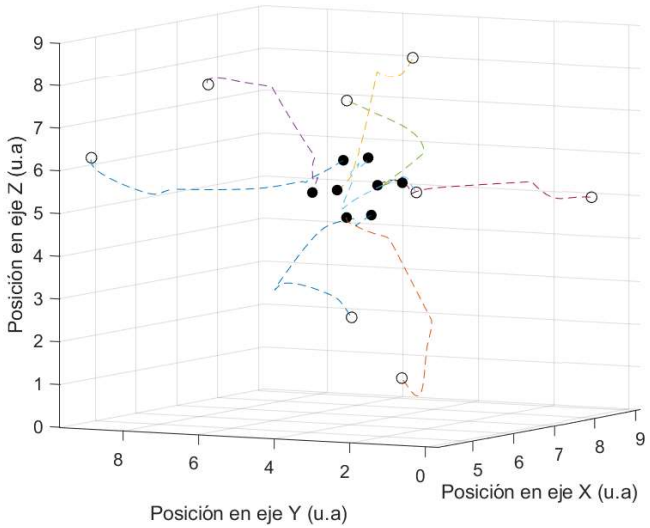


Figura 26: Trayectorias de los agentes con la ecuación dinámica y límite de velocidad para la formación 1.

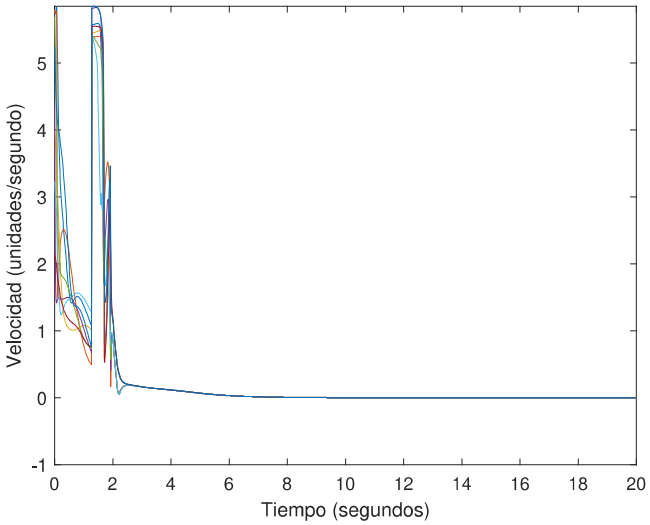


Figura 27: Magnitud de la velocidad de los agentes con la ecuación dinámica y límite de velocidad para la formación 1.

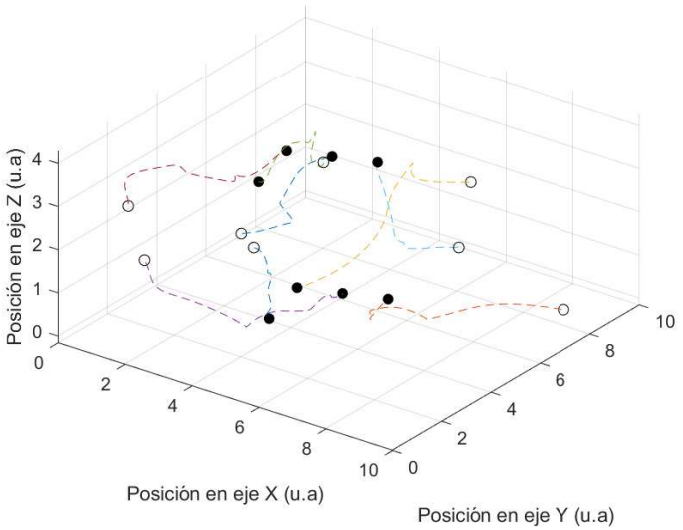


Figura 28: Trayectorias de los agentes con la ecuación dinámica y límite de velocidad para la formación 2.

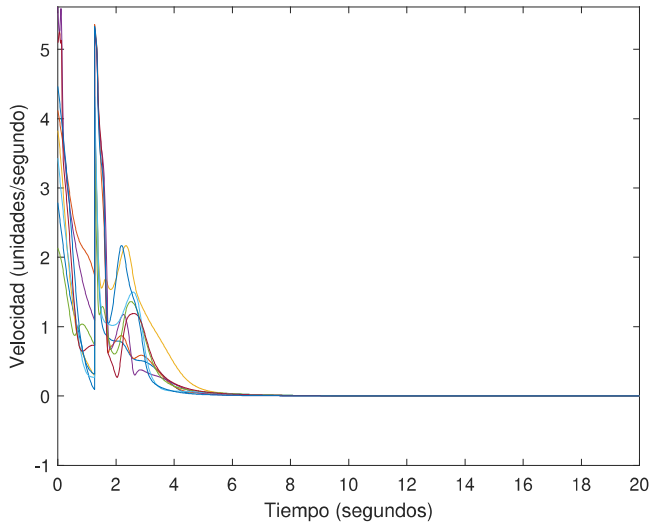


Figura 29: Magnitud de la velocidad de los agentes con la ecuación dinámica y límite de velocidad para la formación 2.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad	1	100	76	24
Modificada con límite de velocidad	2	100	66	34

Cuadro 3: Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad.

Estos resultados nos indican que la implementación de un límite de velocidad fue exitoso, pues los resultados de las 100 iteraciones realizadas por cada formación no aumentó en la

Modelo	Grafo	E $<$ 50	50 $>$ E $<$ 100	100 $<$ E $<$ 200	E $>$ 200	Promedio
Modificado	1	95	5	0	0	41.73
Modificado	2	84	16	0	0	42.66

Cuadro 4: Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación con límite de velocidad.

cantidad de formaciones fallidas, cómo se puede observar en el Cuadro 3. En las figuras 27 y 29 se observa que la velocidad ya no presenta cambios bruscos en magnitud durante toda la trayectoria a comparación del modelo sin límite de velocidad, por lo que se considera un algoritmo seguro para la implementación física. En las figuras 26 y 28 se pueden observar la formación exitosa de los agentes en la simulación del modelo con la implementación del límite de la velocidad en dónde respalda la capacidad de los agentes de formarse aunque no dispongan de velocidades no limitadas.

Pruebas en planos para formación en 3D

La idea de esta simulación es obtener los comportamientos lo más cercano a la realidad, es por ello que se trabajó con una posición aleatoria en el plano $Z = 0$, y así poder observar el comportamiento de los agentes al momento de la inicialización del algoritmo. El resultado de esta prueba fue que, los agentes no pudieron formarse debido a que, partiendo del problema canónico, este permite el consenso de agentes en en el centroide de la posición inicial de los agentes, acá debemos considerar que, al momento de inicializar los agentes en un plano, el centroide en un eje termina siendo la posición del plano en ese eje, por lo que les resulta imposible salir de este plano para mantener los pesos previamente definidos para la formación. El resultado de esta prueba fallida puede observarse en las figuras 30 y 31, en donde los agentes intentaron generar la formación sin salir del plano. Se realizaron 100 iteraciones para cada caso, el resultado de esto se puede observar en el Cuadro 5

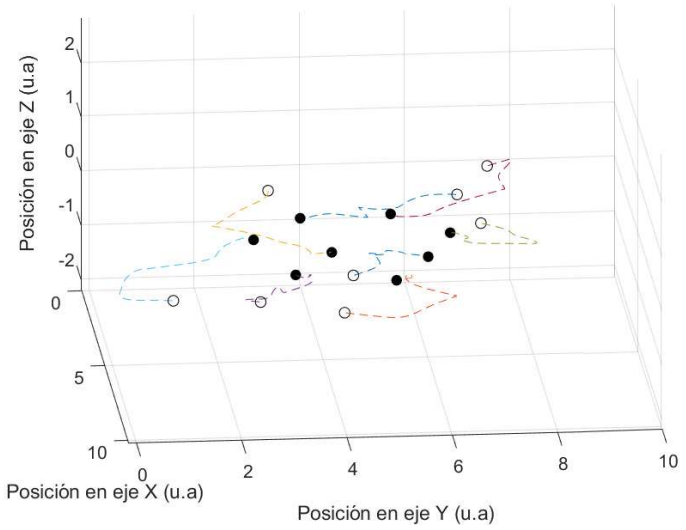


Figura 30: Trayectorias de los agentes en el plano $Z = 0$ para la formación 1.

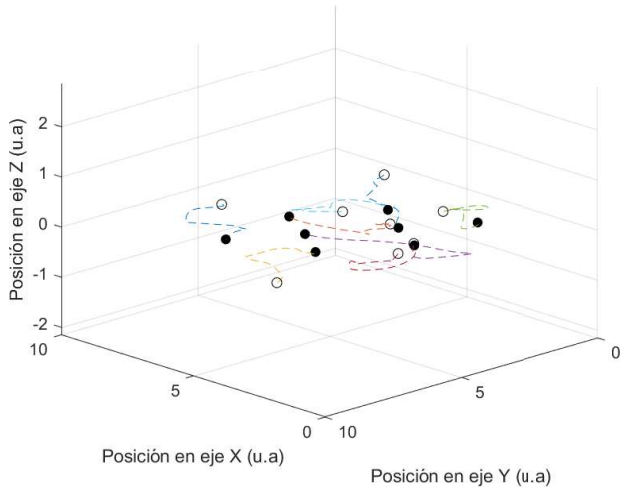


Figura 31: Trayectorias de los agentes en el plano $Z = 0$ para la formación 2.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad	1	100	0	100
Modificada con límite de velocidad	2	100	0	100

Cuadro 5: Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad en el plano $Z = 0$.

Producto de esta situación, se hizo otra prueba con un plano a una inclinación arbitraria para comprobar el comportamiento de los agentes ante plano. Para este caso, se obtuvo dos comportamientos distintos para la formación de cubo y prisma triangular. Para el primer caso, prisma triangular, los agentes se pudieron formar exitosamente; Para el segundo caso, cubo, los agentes no completaron la formación con éxito. El resultado puede observarse en las figuras 32 y 33.

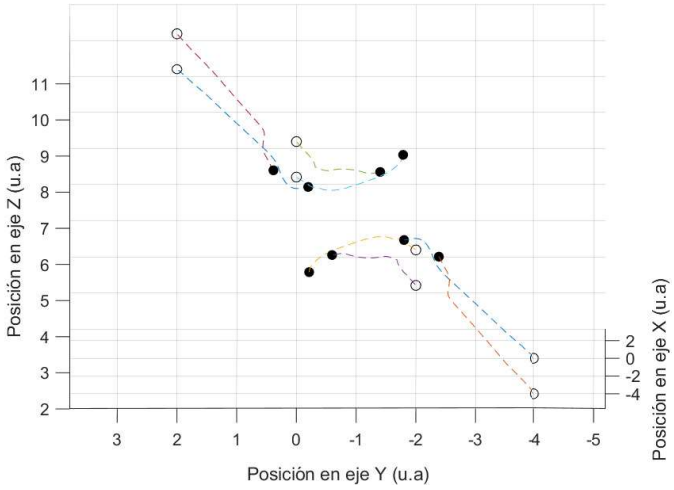


Figura 32: Trayectorias de los agentes en el plano arbitrario para la formación 1.

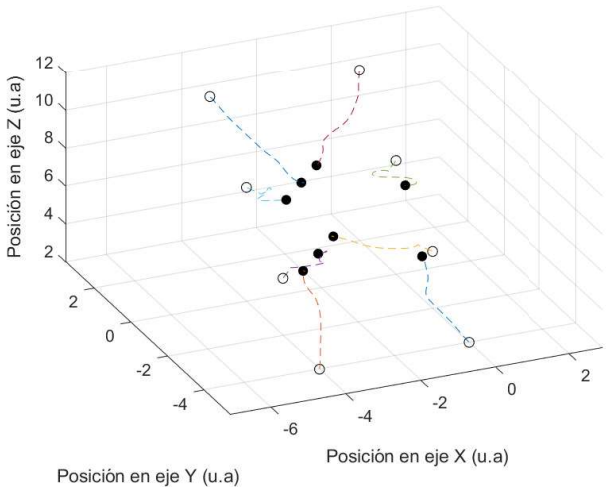


Figura 33: Trayectorias de los agentes en el plano arbitrario para la formación 2.

Para poder solucionar el problema con la inicialización en un plano, se implementa una dinámica inicial para que los agentes al momento de iniciar se eleven durante un lapso arbitrario de tiempo y con ello, salgan del plano. Esto resulta exitoso debido a que, el algoritmo trabaja agente por agente, por lo tanto, estos no se elevan al mismo tiempo por lo que, desde que el primer agente se eleva, rompe la formación de plano y pueden iniciar con la formación. Esto se comprueba con las figuras 34 y 35, en dónde se puede observar cómo los agentes obtienen una velocidad vertical a distinto tiempo. Es importante resaltar que, se continúa con la restricción del centroide, por lo que los agentes deben elevarse lo suficiente para que la base no interfiera con los mismos. Esta dinámica resultó efectiva, debido a que se mantuvo un porcentaje aceptable de intentos exitosos. Se puede observar la trayectoria de los agentes para las dos formaciones en las figuras 36 y 37.

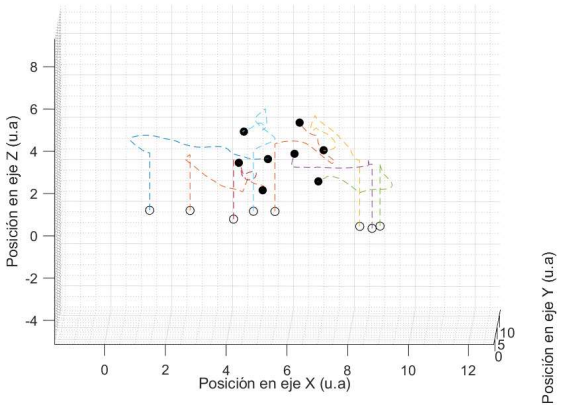


Figura 34: Trayectorias de los agentes al iniciar en el plano $Z = 0$ con la dinámica para romper el plano para la formación 1.

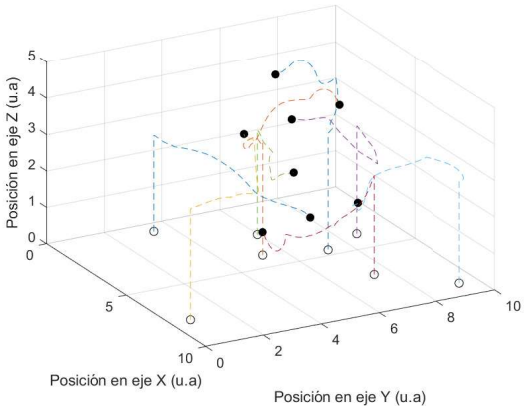


Figura 35: Trayectorias de los agentes al iniciar en el plano $Z = 0$ con la dinámica para romper el plano para la formación 2.

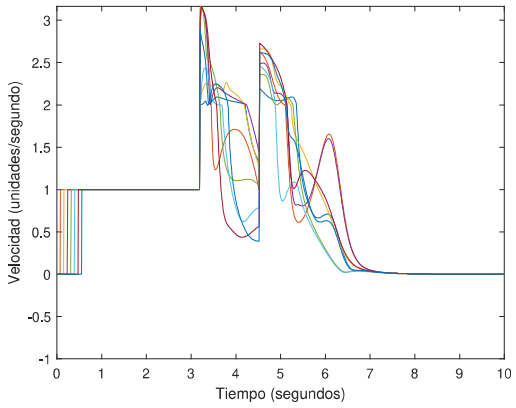


Figura 36: Velocidad de los agentes al iniciar en el plano $Z = 0$ con la dinámica para romper el plano para la formación 1.

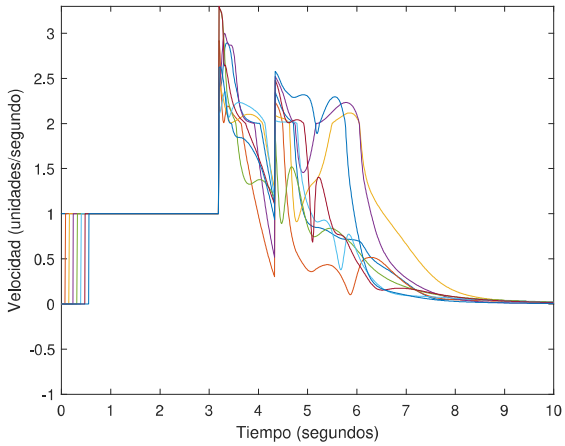


Figura 37: Velocidad de los agentes al iniciar en el plano $Z = 0$ con la dinámica para romper el plano para la formación 2.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad y rutina extra	1	100	72	28
Modificada con límite de velocidad y rutina extra	2	100	65	35

Cuadro 6: Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad en el plano $Z = 0$ con rutina para romper plano.

Modelo	Grafo	$E < 50$	$50 > E < 100$	$100 < E < 200$	$E > 200$	Promedio
Modificado	1	0	100	0	0	65.61
Modificado	2	0	100	0	0	64.69

Cuadro 7: Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación con límite de velocidad con rutina para romper el plano $Z = 0$.

En el Cuadro 6 se puede observar que para el primer modelo, la reducción de formaciones redujo en un 4 %, para el segundo caso redujo únicamente un 1 %. Esto nos dice que la rutina implementada no representa un cambio significativo para la ejecución de formaciones por lo que se considera una correcta implementación. Es importante mencionar que, al observar el Cuadro 6, si presenta un aumento de consumo de energía del 50 %, esto ocurre debido a que la rutina le indica a los agentes durante 3 unidades de tiempo para garantizar que los agentes no colisionen con el plano en $Z = 0$.

CAPÍTULO 8

Implementación en el entorno físico de pruebas

El algoritmo desarrollado se implementó en el entorno de simulación de Cyberbotics, Webots. Este simulador modela los parámetros físicos de los *Crazyflie 2.0*. Este conjunto de simulaciones pondrán a prueba los algoritmos para verificar que los drones no colisionen entre sí, que puedan realizar la formación indicada y si se trabaja con el límite de velocidad. Esto se basa en el modelo del *Crazyflie 2.0* desarrollado por Bitcraze [19].

8.1. Simulaciones

El algoritmo y resultados obtenidos en Matlab, demostraron ser resultados exitosos. Por lo tanto, se realizaron las transformaciones necesarias del lenguaje utilizado en Matlab, a controladores y funciones auxiliares en python. Esto se realizó por medio de un proceso iterativo para poder comprobar que cada función funcionara correctamente.

En el simulador se trabaja con un observador, llamado supervisor, el cuál simula al OptiTrack para el entorno de Webots.Esto debido a que el supervisor funciona como un robot pero con capacidades extras, cómo obtener información de otros robots, movilizarlos, etc. [20]. Este es el que obtiene la información de cada dron y realiza los cálculos para posteriormente enviarle la información a cada *Crazyflie 2.0* y que estos se movilicen. Adicionalmente, se le asigna la capacidad de movilizar a cada dron en una posición aleatoria en la plataforma.

En este simulador se requiere de identificar cuál es la velocidad que deben tener los motores para que los *Crazyflie 2.0* puedan desplazarse y posteriormente se queden en equilibrio en algún punto elevado. El propósito de conocer este dato es modificar el algoritmo de control de formaciones para que las velocidades no converjan a una magnitud 0 cuando ya estén formados, puesto que, al realizar las pruebas físicas, si están formados los drones, estos caerían al suelo. Para esto, se trabajó con una variable que asignaba cuál era la altura

deseada, que utilizaba los valores proporcionados por el controlador y que se mantuviese la velocidad final.

Al tener una cantidad de parámetro para modificar, se procedió a hacer pruebas sencillas para poder observar el comportamiento de los *Crazyflies* ante los valores proporcionados por el algoritmo, así poder mapear los datos a forma que cada dron se mueva. La primer prueba se realizó utilizando 4 drones, por lo que se utilizó el grafo de la Figura 38, considerando de igualmente un grafo completo o totalmente rígido.

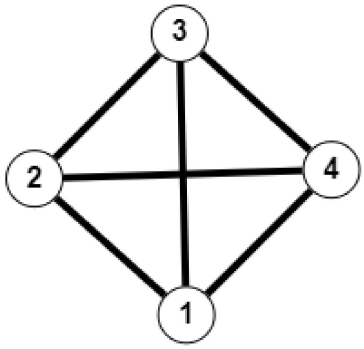


Figura 38: Grafo de prueba para formaciones en simulador Webots.

Para el grafo de la Figura 38, se definió la siguiente matriz de adyacencia:

$$\begin{bmatrix} 0 & 3 & 3 & \sqrt{18} \\ 3 & 0 & \sqrt{18} & 3 \\ 3 & \sqrt{18} & 0 & 3 \\ \sqrt{18} & 3 & 3 & 0 \end{bmatrix}$$

La configuración anterior representada es un cuadrado, el cual se puede ver como ejemplo en la Figura 39

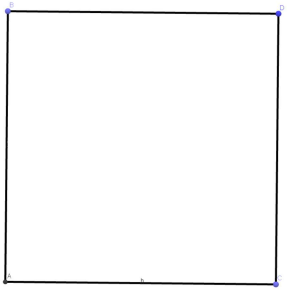


Figura 39: Formación de prueba para el simulador Webots.

Para realizar las pruebas, se consideró aplicar una posición aleatoria al igual que en las simulaciones de Matlab, por lo que al aplicar esto en Webots, se agregó una verificación

extra para que, al momento de generar las posiciones aleatorias, estas alejen más a los drones entre sí, si alguno está dentro del radio de otro. Por lo que la posición inicial de los drones se observa en la Figuras 40, mientras que la posición aleatoria generada en el controlador del supervisor se observa en la Figura 41.

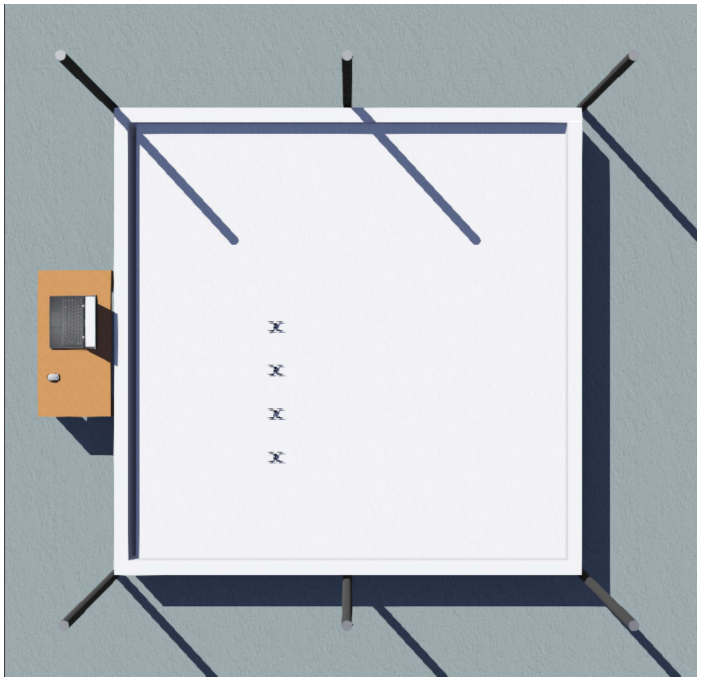


Figura 40: Posición inicial en simulador de Webots.

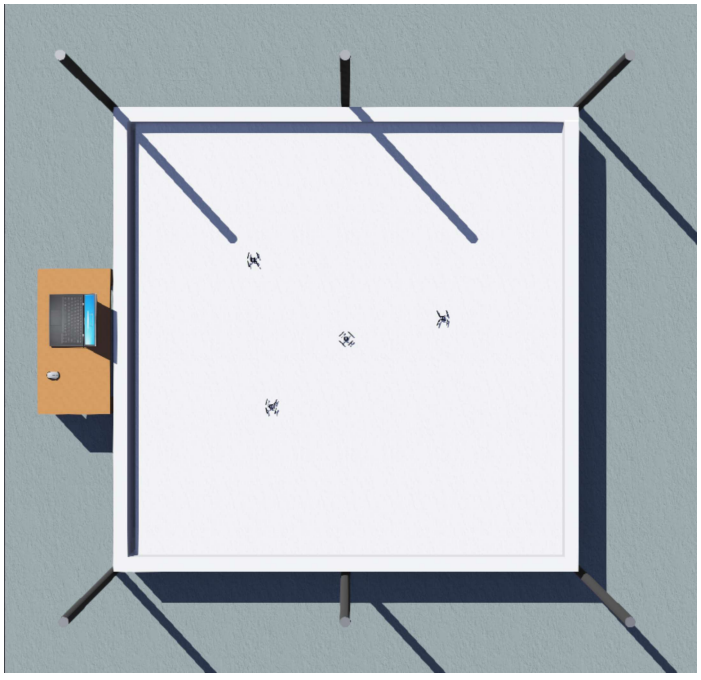


Figura 41: Posición final para iniciar formación en Webots.

Posteriormente se asigna las velocidades obtenidas a partir del controlador dinámico, para que realice la formación, como se puede observar en la Figura 42.

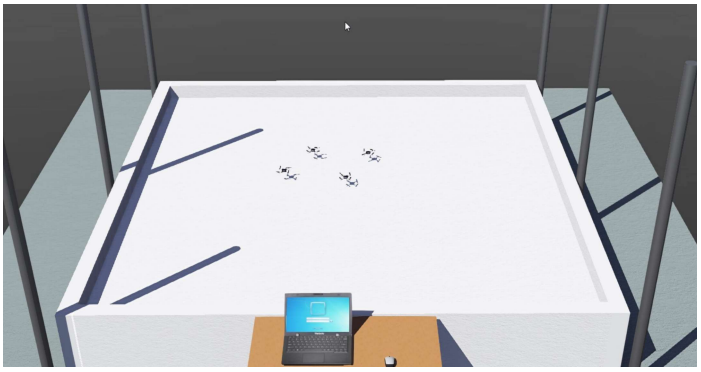


Figura 42: Formación en forma de cuadrado en Webots.

Estos resultados nos muestran un funcionamiento aceptable al momento de realizar la formación, es importante mencionar que, aunque esto resultado termina siendo una formación en 2D, los drones son capaces de intercambiar de lugar al pasar uno por encima del otro. Por lo que el resultado con 4 agentes se considera exitoso.

Para poner a prueba los resultados ideales de Matlab, se procede a hacer la prueba con 8 drones *Crazyflie 2.0*, por lo que se retoman las formaciones de las figuras 16 y 17. Se implementa nuevamente la asignación aleatoria de posiciones, en la Figura 43 se observa la posición inicial de todos los drones, mientras que en la Figura 44 se observa la posición asignada por el supervisor.

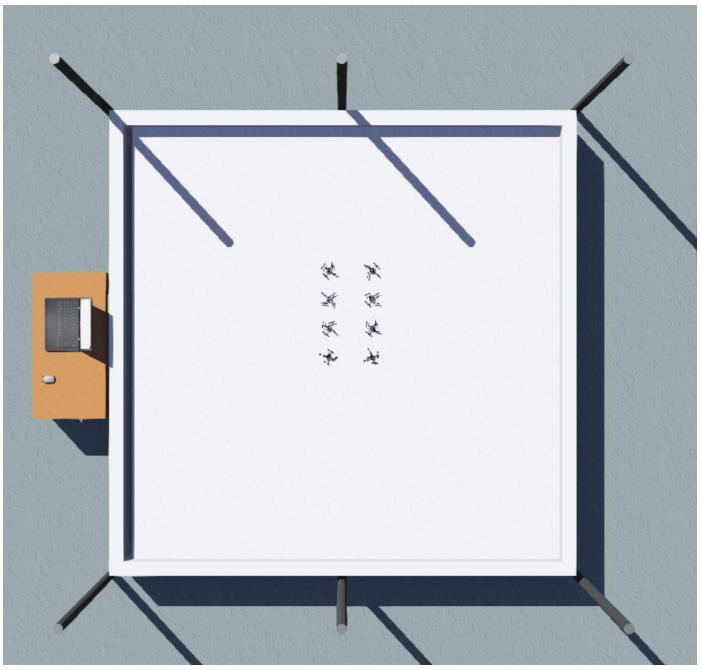


Figura 43: Posición inicial en simulador de Webots para 8 *Crazyflies 2.0*.

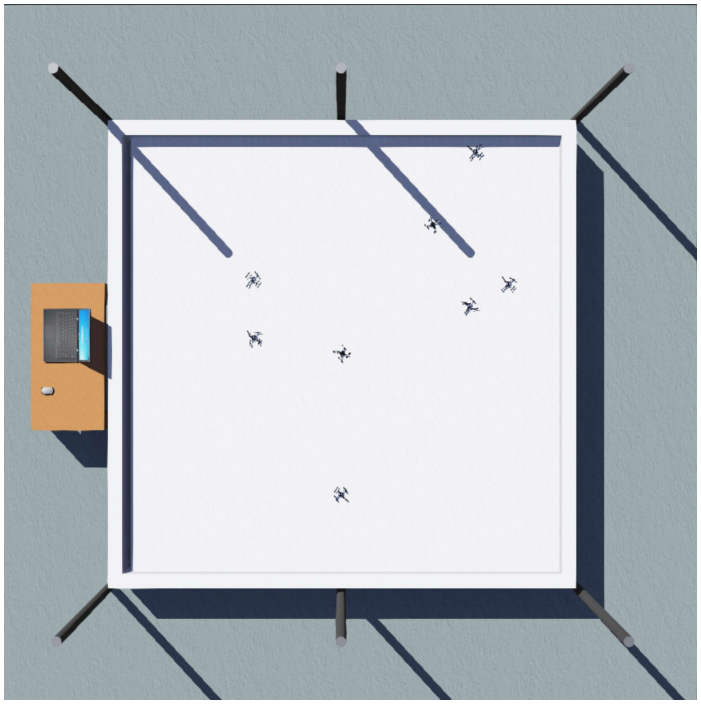


Figura 44: Posición final para realizar formación en simulador de Webots para 8 *Crazyflies 2.0*.

Los resultados para estas simulaciones no fueron efectivos, al utilizar un grafo totalmente rígido, los drones se procuraban acercar a los demás con una mayor velocidad y esto ocasionaba colisiones, para el caso del controlador, resulta que, si se disminuye la velocidad máxima en el supervisor, los drones no se movilizaban, esto debido a que no era un cambio significativo para que los motores del dron no movilizaran al mismo. El resultado fallido por colisión se puede observar en la Figura 45, mientras que el fallido por disminución de velocidad se observa en la Figura 46.

Figura 45: Formación fallida para 8 *Crazyflies 2.0*, error por colision debido al aumento de velocidad.

Figura 46: Formación fallida para 8 *Crazyflies 2.0*, error disminución de velocidad máxima.

CAPÍTULO 9

Conclusiones

- El algoritmo desarrollado permite formarse en alguna configuración específica a un conjunto de agentes, otorgándole a cada uno la capacidad de recalcular su trayectoria según la posición de los demás agentes.
- El uso de grafos completamente rígidos presenta una red de comunicación compleja para la formación en 3 dimensiones pero, resulta ser una herramienta útil para esta aplicación debido a que, por la cantidad de grados de libertad que presenta cada agente en tres dimensiones, se asegura que se tienen las restricciones suficientes para que la formación sea la esperada.
- Los algoritmos en las simulaciones no toman en consideración las limitantes físicas de los actuadores de los agentes, por lo que implementar el límite de velocidad a las simulaciones nos presenta un comportamiento más cercano a las pruebas físicas.
- El uso de un control dinámico permite utilizar los beneficios que presenta cada aplicación de la ecuación de consenso para aumentar el éxito de la formación.
- La rutina de elevación para los drones funciona durante un tiempo especificado por el usuario, por lo que es relevante tomar en cuenta que este varía según la formación utilizada para evitar que los agentes salgan del área capturada por el sistema OptiTrack.
- Los resultados de las trayectorias y estadísticas de posición nos indican que el algoritmo desarrollado permite que cada agente se comporte según lo esperado.
- El controlador incorporado del *Crazyflie 2.0* en el simulador de Webots converge en un lapso corto, por lo que cambios elevados de velocidad, afectan la estabilidad del dron.

CAPÍTULO 10

Recomendaciones

- Implementar un control para la aceleración de los agentes para evitar cambios drásticos en la velocidad.
- Evaluar si se obtiene alguna ventaja al utilizar grafos mínimamente rígidos en la formación de agentes en 3 dimensiones.
- Realizar pruebas con otro tipo de grafos y otras formaciones con rigidez mínima local comprobada.
- Se recomienda modificar las constantes del controlador PID incorporado en el simulador de Webots para el *Crazyflie 2.0*, para disminuir la velocidad de convergencia y así no desestabilizar al dron.
- Para obtener simulaciones sin retraso, se recomienda generar un solo controlador para todos los drones, así evitar que el software ejecute un controlador por *Crazyflie 2.0*.
- Para las pruebas físicas se recomienda realizar pruebas con una cantidad menor de drones al inicio para observar el comportamiento que presentan al estar cerca de otros drones, así evitar que haya daño a los actuadores o al dron completo.

CAPÍTULO 11

Bibliografía

[1] «Crazyflie 2.0.» dirección: https://store.bitcraze.io/products/crazyflie2.

[2] J. A. Preiss, W. Honig, G. S. Sukhatme y N. Ayanian, «Crazyswarm: A large nano-quadcopter swarm,» en *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, págs. 3299-3304.

[3] R. Zahínos, «Sistema de coordinación y control de múltiples vehículos aéreos no tripulados en testbed de interiores,» Universidad de Sevilla, 2020.

[4] L. Jeroncic, «Drone Swarm Simulator,» Tesis de mtría., The Arctic University of Norway, Tromsø, Norway, 2021.

[5] H. Zhou, H.-L. Xiong, Y. Liu, N.-D. Tan y L. Chen, «Trajectory planning algorithm of UAV Based on System Positioning Accuracy Constraints,» *Electronics*, vol. 9, n.º 2, pág. 250, 2020. DOI: 10.3390/electronics9020250.

[6] G. Martínez, «Plataforma de pruebas y ajustes de sistemas de control para vehículos multirrotores,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[7] F. J. Sanabria, «Diseño e implementación de una plataforma de pruebas para sistemas de control para el dron Crazyflie 2.0,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[8] S. J. Castillo Lou, «Implementación y desarrollo de plataforma tipo gimbal para pruebas de control de drones de mediana potencia,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[9] A. M. Peña Echeverría, «Algoritmo de sincronización y control de sistemas de robots multi-agente para misiones de búsqueda,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[10] «Optitrack for Robotics.» dirección: www.optitrack.com.

[11] K. Ruohonen, *Graph Theory*. Tampere University of Technology, 2008.

[12] E. Hernández, «Sistema multi-cámara para la reconstrucción volumétrica de objetos 3D,» Tesis de mtría., Universidad Zaragoza, 2019.

[13] J. Ávila, «Diseño e implementación de estrategias de consenso líder-seguidor para un sistema multi-agentes,» Tesis de mtría., Universidad autónoma del estado de Hidalgo, 2019.

[14] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.

[15] N. Nise, *Control system engineering*. Wiley, 2014.

[16] M. Mesbahi y M. Egerstedt, *Graph Theoretic Methods in Multiagent Networks*. Wiley, 2014.

[17] M. Egerstedt, *RobotEcology*. 2021.

[18] D. Poole, *Álgebra lineal: una introducción moderna*. Cengage Learning Latin America, 2017.

[19] «Crazyflie Summer Project with ROS2 and Mapping.» dirección: `https://www.bitcraze.io/tag/webots/`.

[20] «Webots documentation: Supervisor.» dirección: `https://cyberbotics.com/doc/reference/supervisor`.

CAPÍTULO 12

Anexos

12.1. Repositorio

Enlace al repositorio con simulaciones para los modelos en Matlab y modelo final en Webots: https://github.com/Kaldana/AlgoritmoSwarmDrones