UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Generación de trayectorias automáticas para el dron *CrazyFlie2.0* utilizando algoritmos de inteligencia computacional

Trabajo de graduación presentado por Carlos Roberto Efrain Avendaño Quinteros para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

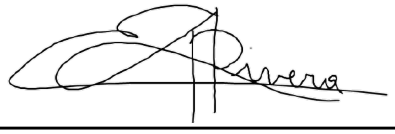
Generación de trayectorias automáticas para el dron *CrazyFlie2.0* utilizando algoritmos de inteligencia computacional

Trabajo de graduación presentado por Carlos Roberto Efrain Avendaño Quinteros para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2023

Vo.Bo.:

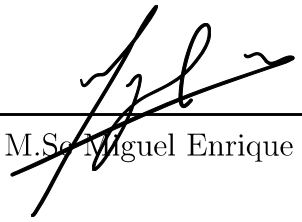
(f) 

Dr. Luis Alberto Rivera Estrada

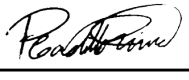
Tribunal Examinador:

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

M.Sc Miguel Enrique Zea Arenales

(f) 

M.Sc Pedro Iván Castillo Rivera

Fecha de aprobación: Guatemala, 3 de enero de 2023.

Prefacio

El presente trabajo de graduación requirió de múltiples disciplinas que abarcan distintos conocimientos que convergen en un trabajo que muestra como se pueden aplicar conocimientos avanzados de programación, Sistemas de control, Algoritmos de inteligencia computacional entre otros. Todo esto se pudo realizar gracias a los muchos conocimientos adquiridos durante la carrera. En el presente trabajo de graduación se presenta la aplicación de algoritmos de inteligencia computacional e inteligencia de enjambre para el dron *CrazyFlie2.0*. El deseo de crear este trabajo es dejar una rama para que futuros trabajos de graduación se basen en ella para ir creando poco a poco proyectos más interesantes y fomenten la inversión en proyectos de ciencia e ingeniería como éste.

Agradezco de primera a Dios por la oportunidad de estudiar en esta casa de estudios y a mis padres por trabajar duro durante estos años invirtiendo en mi educación superior. Le agradezco a mi asesor de tesis Dr. Luis Alberto Rivera por tenerme paciencia y guiarme en el proceso de este complicado trabajo. De la misma manera agradezco a mi coasesor de tesis Msc. Miguel Zea por orientarme también en dicho proceso con el dron *CrazyFlie2.0*

Índice

Lista de figuras

Lista de cuadros

Resumen

El objetivo principal de este trabajo de graduación fue la implementación de algoritmos de inteligencia computacional en la planificación automática de trayectorias para encontrar caminos óptimos a diferentes trayectos propuestos para uno o varios drones en un espacio tridimensional. Para esto se investigó y evaluó varios algoritmos con la finalidad de tener por lo menos dos. Cuando se habla de camino óptimo se refiere a evasión de obstáculos y menor tiempo en la realización del trayecto. Con este código hecho se valida la trayectoria con una simulación que despliega en un plano tridimensional la posición de los puntos de la trayectoria a seguir por el dron.

Al tener una simulación satisfactoria con los algoritmos validados se procedió a realizar una práctica real en un ambiente controlado donde el dron tuvo un espacio de trabajo delimitado por el sistema de captura de movimiento OptiTrack. El sistema OptiTrack dio información acerca de las coordenadas en el sistema tridimensional del dron *CrazyFlie2.0* y de los obstáculos que se colocaron en la pista de obstáculos. Esta información es importante para que los algoritmos generaran trayectorias desde la posición del dron hasta la meta. Estas trayectorias se generaron automáticamente por medio de los algoritmos, los cuales se encargaron de determinar si la ruta es la óptima entre las muchas que se generaron.

Abstract

The main objective of this research thesis is the implementation of computational intelligence algorithms and swarm robotics. To achiveve this two algorithms were used, Ant Colony Optimization and Particle Swarm Optimization. Both algorithms were modified to use them to find trayectories point to point and check if those trayectories were optimized. To check if those algorithms were optimized a series of obstacles were put in the Robotat system in order to challenge the algorithm to find the best route with that obstacle.

Two algorithms were made to accomplish this goal, Ant Colony Optimization was the hardest one to make and also the one who takes most of the time. Particle Swarm Optimization is a lot easier algorithm and was the one who were used the most due to the time it takes is lower than Ant Colony Algorithm.

The Robotat ecosystem was used to make a live practice were the drone *CrazyFlie2.0* was used to see in real life how this works. The drone flew and looked forward to follow the trayectories to catch the end point.

CAPÍTULO 1

Introducción

Muchos algoritmos de inteligencia computacional que se usan en la vida cotidiana están basados en el comportamiento del reino animal. En este caso se usaron algoritmos que usan inteligencia de enjambre como lo usan las abejas, hormigas, peces, termitas entre otros. Los algoritmos también se pueden usar en agentes robóticos para toma de decisiones individuales que ayuden a otros agentes en un comportamiento de enjambre.

El Ant Colony Optimization (ACO) se basa en el comportamiento de hormigas buscando alimento, es decir buscar alimento desde el hormiguero hasta un punto donde se encuentre este. Existen diferentes versiones de este algoritmo como el Ant System (AS) y el Simple Ant Colony Optimization (SACO). Entre las aplicaciones más usadas para esté algoritmo se encuentran diseño de rutas para automóviles, procesamiento de señales entre otras.

El Particle Swarm Optimization (PSO) se basa en el comportamiento de aves que buscan un punto de convergencia el cual puede ser alimento o un lugar de migración. Todos las partículas dependen de una a otra, es decir que las decisiones que tome un individuo afectan el comportamiento del grupo para encontrar una solución óptima que normalmente en el algoritmo se trata de una ecuación.

El sistema OptiTrack es un ambiente para agentes robóticos o humanos donde a partir de pelotas reflectivas se pueden determinar posiciones (x,y,z) o ángulos *yaw*, *pitch* y *roll* lo cuál es ideal para ágentes robóticos voladores como los drones. Existen varios tipos como los ópticos y los no ópticos los cuales usan acelerómetros pero cuentan con desventajas. En la Universidad del Valle de Guatemala se cuenta con un sistema de captura de movimiento OptiTrack el cual es óptico debido al uso de cámaras que capturan bolas reflectivas para determinar posiciones y ángulos.

En el presente trabajo de graduación se buscó implementar todas estas herramientas. El sistema OptiTrack identifica las posiciones del dron *CrazyFlie2.0* y de algunos obstáculos en el Robotat los cuales son enviadas al algoritmo para determinar puntos importantes para los cálculos de los algoritmos. Estos algoritmos indican el trayecto óptimo del punto inicial del

drone hasta la meta propuesta por el usuario y generan una simulación donde presenta una visualizacion del trayecto. Con estos valores se comunica al *CrazyFlie2.0* para entregarle los set points calculados por los algoritmos y que este vuele por estos puntos hasta llegar a la meta. Esto se logró con el aporte de diferentes conocimientos de trabajos de graduación en [1], [2], [3], [4], [5].

CAPÍTULO 2

Antecedentes

2.1. Dron de vuelo autónomo con inteligencia artificial capaz de reconocer patrones para labores de rescate.

En este trabajo de fin de grado a cargo de Juan Carlos De Alfonso Juliá [6]. Se trabajó en un algoritmo necesario para una navegación autónoma de un dron sin necesidad de intervención humana para labores de rescate. Este trabajo consiste en poder programar un dron con un marco de coordenadas en una zona específica. Con las coordenadas establecidas el trabajo de dicho dron es poder manejarse solo en estas trayectorias y buscar posibles objetivos de rescate sin intervención humana dando notificación de los rescatistas de la ubicación de los posibles lugares que necesiten rescate de personas. Esta tarea de búsqueda se basa en algoritmos de inteligencia artificial programados para que el dron por medio de una cámara pueda identificar patrones o imágenes como el de una mano humana pidiendo ayuda, fuego u otro tipo de señales de auxilio. Para esta tarea del dron se utilizó una *Raspberry Pi* a bordo para poder realizar las tareas de inteligencia artificial además de controlar el dron con un módulo hat NAVIO2 que se adapta a la *Raspberry Pi* el cual cuenta con un barómetro de alta resolución (MS5611), dos unidades inerciales (IMU) (MPU9250 y LSM9DS1), un módulo GPS (Ublox M8N) y 14 salidas PWM para controlar los motores del dron.

2.2. Generación de trayectorias automáticas con condiciones iniciales extremas

En la tesis a cargo de Daniel Warren [7] se presenta el control de un cuadricóptero. Se trabajó en la elaboración de un controlador capaz de manejar uno y varios cuadricópteros simultaneamente. Este controlador tiene la tarea de manejar los ángulos *'roll'* y *'pitch'* que permiten al cuadricóptero seguir trayectorias que requieran grandes aceleraciones y poderse

recuperar o estabilizar de condiciones iniciales extremas para el sistema. También se trabajó en una programación capaz de hacer que los cuadricópteros puedan trabajar en conjunto para poder cargar en conjunto pesos que solo un dron no podría. Se hizo un parámetro para que el cuadricóptero pueda saber por medio de sus capacidades físicas la dimensión de la carga que esté tratando de levantar para determinar cuántos compañeros se necesitan para levantar la carga. Se trabajó en una generación de trayectorias automática que logra hacer que los cuadricópteros manejen en presencia de diferentes obstáculos determinando cuál es la trayectoria óptima.

2.3. Manufactura, modelado y control de cuadricópteros con capacidad de comunicación inalámbrica Wi-Fi con el ecosistema Robotat

En el trabajo de graduación a cargo de Carlos Alonzo [8] se hizo el desarrollo de un dispositivo volador de cuatro motores con el fin de ser utilizado dentro del ecosistema Robotat ubicado en el laboratorio de robótica de la Universidad del Valle de Guatemala. Todos estos componentes tuvieron una rigurosa selección siempre tomando en cuenta el factor de calidad versus precio. En la primera sección se consideraron elementos que representen un bajo costo en componentes como los motores sin escobillas, controladores electrónicos de velocidad, hélices, diseño del marco estructural de la aeronave entre otros componentes. En otra parte se desarrolló un entorno de simulación Matlab en donde se pueden modificar los parámetros del controlador de vuelo. Este controlador de vuelo es un controlador PD (proporcional derivativo) el cual tiene la tarea de hacer que el dron permanezca en estado de *'hovering'* por medio de un microcontrolador ESP32 en lenguaje C. Para este proyecto se hicieron varias pruebas de vuelo en donde en la interfaz se observaba la velocidad de los cuatro motores con respecto a los ángulos de entrada de una unidad de medida inercial. Como última sección se hizo una integración del sistema de captura de movimiento OptiTrack y el drone para recibir datos en tiempo real de la posición y orientación dentro del sistema abarcado por el OptiTrack. El sistema OptiTrack es un ecosistema de diferentes cámaras de captura de movimiento utilizadas para determinar posiciones utilizando unas pelotas reflectivas para determinar poses o posiciones de cuerpos rígidos o puntos en el espacio.

2.4. Diseño de un controlador de vuelo para cuadricópteros con la capacidad de usar el ecosistema Robotat

En el trabajo de graduación a cargo de Hans Burmester [9] se elaboró un controlador de vuelo el cual permite la interacción entre motores BLDC, módulos ESP32, MPU-9250 y el sistema de captura de movimiento OptiTrack. Los protocolos de comunicación utilizados en estos procesos fueron I2C y MQTT. Todo esto montado en una placa electrónica que se encargara de dar alimentación a los diferentes módulos y repartir las señales de controles a sus sistemas correspondientes. Para este software se hicieron diferentes pruebas obteniendo datos del modulo MPU-9250 con las cuales se obtuvieron resultados de los cálculos de *'roll'*,

'pitch' y *'yaw'* dando comprobación del correcto funcionamiento del software. También se hizo la prueba de si la batería de litio lograba ser suficiente para dar alimentación al sistema. Esta batería es de tipo Li-Ion con capacidad de 6800mAh con peso de 287 gramos y un voltaje nominal de 12V. Para comprobar que esta batería lograba suplir la demanda del drone se hicieron diferentes pruebas a diferentes velocidades y se determinó que la batería lograba suplir cada una de ellas.

2.5. Desarrollo e implementación del ecosistema Robotat y comunicación inalámbrica

El trabajo de graduación a cargo de Camilo Perafan [10], consiste en una red de comunicación WiFi para varios agentes, que al funcionar junto al sistema de captura de movimiento OptiTrack se obtiene un ecosistema de experimentación robótico denominado Robotat. En los resultados de este trabajo de graduación se corroboro que existe un error del 5.28 % con respecto de las medidas reales en los resultados del sistema. Posterior a esto se realizó una librería en C para un microcontrolador ESP32 para lograr una conexión a la red WiFi del ecosistema y así poder recibir datos leídos del sistema OptiTrack por medio de un protocolo denominado MQTT y del mismo modo poder mandar datos por este mismo medio. También se trabajó en una librería en Python implementado en la computadora que esta asignada al ecosistema el cual tomaba los datos del OptiTrack de las poses y posteriormente publicaba los datos al microcontrolador. Junto a estas tareas también se necesitaba determinar cuantos agentes pueden estar en el ecosistema haciendo que el sistema trabajara de forma óptima teniendo un resultado a un valor máximo de 11 agentes antes de tener una latencia menor de 10Hz. Como último paso en este trabajo de graduación se desarrollo una antena inteligente la cual permite a cualquier agente no robótico poder interactuar con el ecosistema. Junto a esto también se determinó que se puede mejorar la calidad y exactitud de los datos obtenidos por el OptiTrack agregando un filtro Kalman y una unidad de medida inercial lo que permite reducir el ruido de los datos obtenidos del OptiTrack.

2.6. Diseño de una plataforma de pruebas para controlar el dron *CrazyFlie2.0*

En la fase anterior se desarrolló un controlador de vuelo y estabilización para los drones *CrazyFlie2.0*. Este trabajo de graduación a cargo de Francis Sanabria [11] cuenta con una máquina virtual con los programas y recursos necesarios para manipular el *CrazyFlie2.0* de manera óptima. De igual forma se describió la API de una librería de Python la cual permite el control de dron a través de una computadora por medio de funciones de alto nivel. Con esta API se desarrollo una interfaz grafica que permite el control del dron de manera más sencilla y amigable con el usuario. Esta interfaz gráfica permite manipular el ángulo de cabeceo del dron y visualizar y guardar los datos del codificador rotacional para poder ser usados posteriormente por el controlador.

Estos controladores de vuelo para el *CrazyFlie2.0* se basan en sistemas de control moderno y clásico dependiendo la necesidad del usuario. El control moderno aplicado al dron

establece variables de estado las cuales miden la posición (x, y, z) del dron, mide el ángulo de balanceo, cabeceo y guiñada del dron, las velocidades lineales (x, y, z) del dron con respecto al marco de referencia inercial y finalmente las velocidades de cabeceo, balanceo y guiñada del dron con respecto al marco de referencia inercial. Cada una de estas variables de estado es manejada por el controlador moderno LQR, en control clásico un PID no puede controlar tantas variables y condiciones como el LQR.

CAPÍTULO 3

Justificación

En proyectos previos se desarrollaron controladores para el dron *CrazyFlie2.0* para tenerlos en posiciones estables configurados en la interfaz dada. También se trabajó con el sistema de captura OptiTrack para poder tener un sistema funcional que pueda ser los sensores que utilicen diferentes agentes robóticos en la pista del laboratorio. Aprovechando estos recursos previos se buscó usar algoritmos de inteligencia computacional para generar trayectorias las cuales el dron debe de seguir para completar un trayecto.

La finalidad de todo este proyecto es poder empezar a incursionar en el desarrollo de algoritmos, programación avanzada y agentes robóticos autómatas con la capacidad de hacer tareas sin supervisión como en misiones de rescate, búsqueda, seguridad entre otras aplicaciones. El hecho de hacer drones autómatas en una aplicación de la vida real se puede ver en el ejemplo de labores de rescate donde se pueden tener drones en búsqueda de sobrevivientes mientras el equipo de rescate en vez de volarlos se concentra en otras tareas haciendo más eficientes y rápidos estos procesos. También se puede analizar el caso de seguridad en donde un dron pueda determinar un delito y notificarlo a las autoridades correspondientes.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Generar automáticamente trayectorias para el dron *CrazyFlie2.0* en una pista de obstáculos utilizando algoritmos de inteligencia computacional.

4.2. Objetivos específicos

- Utilizar los algoritmos de inteligencia computacional para generación automática de trayectorias.
- Determinar las posiciones del dron y de los obstáculos por medio del sistema OptiTrack.
- Validar los algoritmos desarrollados por medio de simulaciones computarizadas.
- Validar las simulaciones computarizadas en pruebas físicas con el dron *CrazyFlie2.0* en el entorno del OptiTrack.

CAPÍTULO 5

Alcance

El alcance de este trabajo de graduación abarcó la implementación simulada y real para algoritmos de inteligencia computacional y robótica de enjambre, Particle Swarm Optimization y Ant Colony Optimization. Para esto se tomó de referencia los antecedentes de trabajos de graduación de fase previa los cuales se cambiaron para adaptarlos algoritmos a entornos en tres dimensiones. Esto con el fin de evitar obstáculos y encontrar rutas óptimas a diferentes entornos. Para la optimización de trayectorias el algoritmo Ant Colony Optimization es capaz de encontrar la trayectoria entre dos puntos de un mapa tridimensional representado con grafos y aristas. En la evasión de obstáculos se evita que tome los grafos que representan el espacio ocupado por un cuerpo que obstruye el paso y hace que el algoritmo calcule sobre como evitarlo.

Estos algoritmos se desarrollaron en Matlab donde se hizo una conexión con el sistema Robotat con las cámaras de captura de movimiento OptiTrack para mandar datos de la posición de los obstáculos y de la posición del dron. Se mandan las trayectorias al dron por medio de la antena del dron *CrazyFlie2.0* para que este las siga y evite los obstáculos.

Para futuros trabajos de graduación se pueden implementar estos algoritmos en otros lenguajes de programación para hacer una verificación de eficiencia debido a la complejidad de estos algoritmos.

CAPÍTULO 6

Marco teórico

6.1. Características del dron *CrazyFlie2.0*

El dron *CrazyFlie2.0* [11] es un dron elaborado por bitcraze es un cuatrimotor de 27 gramos que cuenta con un microcontrolador STM32F405 como el controlador principal, Cuenta con un microcontrolador nRF51822 encargado de la alimentación y las señales de radio, tiene un conector micro USB, Baterías LiPo de que van desde 100mA hasta 980mA, Conector USB para carga de baterías, una interfaz ultra rápida para USB, 8KB de EEPROM, Unidades de medición inercial IMU MPU-9250 y un sensor de presión LPS25H.

Las especificaciones de las señales de radio son las siguientes:

- Un ancho de banda de 2.4GHz ISM
- 20 dBm amplificador de radio.
- Módulos Bluetooth de bajo consumo energético

Las dimensiones del dron son de $92 \times 92 \times 29$ mm de motor a motor contando el tren de aterrizaje de este. Para determinar si el dron está en buenas condiciones se tiene que hacer una verificación de integridad con el siguiente proceso:

Conectar la batería y presionar el botón de encendido a la espera de observar LED's azules completamente iluminados y un LED frontal derecho de color rojo parpadeando en intervalos de 2 segundos.

Para hacer pruebas de vuelo se utilizan las siguientes aplicaciones proporcionadas por Bitcraze. La primera es *Crazyflie2.0* disponible para *IOS* y *Android*. Esta aplicación manda datos del celular al dron por medio de comunicación *Bluetooth*. La siguiente aplicación se

utiliza desde la computadora en donde se emplea una antena llamada *Crazyradio PA* y una aplicación para computadora llamada *CrazyFlie client*. Para esto se necesita de una máquina virtual en donde se pueden apreciar muchas más variables de vuelo para el control del dron.

Figura 1: Dron *CrazyFlie2.0* [12].

6.2. Particle Swarm Optimization

La optimización por enjambre de partículas [13] es un método de optimización heurística orientado a encontrar mínimos o máximos globales cuyo funcionamiento esta basado en comportamiento de bandas de animales como pájaros o bancos de peces en donde el movimiento de los individuos es el resultado de combinar las decisiones de cada uno de los individuos con el comportamiento del resto.

6.2.1. Algoritmo

Crear un enjambre inicial de n partículas aleatorias en donde cada partícula contara con 4 elementos necesarios. La posición que representa una combinación de valores de las variables, el valor de la función objetivo en la posición donde se encuentra la partícula, una velocidad que indica como y hacia donde se desplaza la partícula y un registro de la mejor posición en la cual ha estado la partícula hasta el momento. Con esto definido se procede a evaluar cada n partícula con la función objetivo. Se procede a actualizar la posición y velocidad de cada partícula en donde se proporciona al algoritmo la capacidad de optimización. Cuando no se cumple con un criterio dado se repite el proceso.

6.2.2. Creación de la partícula

Cada partícula se define por su posición y velocidad y valor que van variando a medida que esta partícula se encuentra en movimiento. Esta debe tener la capacidad de almacenar la mejor posición en la que ha estado hasta el momento. En condiciones de inicio para las partículas solo se tienen los valores de posición y velocidad asumiendo el resto de las variables como cero.

6.2.3. Mover la partícula

Cuando se habla de mover una partícula esto implica ir cambiando su velocidad y posición dando información al algoritmo para poder optimizar. La velocidad de cada partícula del enjambre se actualiza empleando la siguiente ecuación.

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

Donde

- $v_i(t+1)$ Es la velocidad de la partícula n en el momento t+1 que quiere decir nueva velocidad
- $v_i(t)$ Es la velocidad de la partícula n en el momento t es decir la velocidad actual.
- w Es el coeficiente de inercia dando la posibilidad de incrementar o disminuir la velocidad de la partícula.
- c_1 Es el coeficiente cognitivo
- $r1$ Es el vector de valores aleatorios entre uno y cero de longitud igualo al de vector de velocidad.
- $\hat{x}_i(t)$ Es la mejor posición en la que ha estado la partícula n hasta el momento.
- $x_i(t)$ Es la posición de la partícula n en el momento t.
- c_2 Es el coeficiente social
- r_2 Es el vector de valores aleatorios entre uno y cero de longitud igual a la del vector velocidad.
- $g(t)$ Es la posición de todo el enjambre en el momento t, el mejor valor global.

Uno de los principales problemas del algoritmo PSO es que las partículas tienden a adquirir velocidades muy altas haciendo que estas salgan de sus límites de búsqueda o no puedan converger a una región óptima.

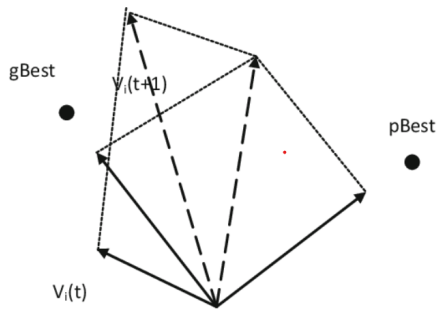


Figura 2: Algoritmo de inteligencia computacional *Particle Swarm Optimization* [13].

6.3. Ant Colony Optimization

En el año 1991 Marco Dorigo [14] [15] propuso en su tesis doctoral un algoritmo denominado *"ant system"* que explicaba y simula el comportamiento de una colonia de hormigas para buscar objetivos o comida. Este Algoritmo tiene muchas variaciones a conveniencia de los intereses de los desarrolladores. Al tratarse de simular una colonia de hormiga simula una inteligencia de enjambre en donde diferentes sujetos trabajan en conjunto que poco a poco se va organizando de forma autónoma.

Como se mencionó anteriormente este algoritmo tiene el objetivo de simular un enjambre de hormigas y para esto los investigadores en los años 40's y 50's se detuvieron a observar con detalle el comportamiento de termitas, para ser específicos las especies *"natalensis"* y *"Cubitermes"*. Descubrieron que estos insectos son capaces de reaccionar a diferentes estímulos causados por un sujeto del enjambre que afecta al resto de la colonia.

Esto se puede apreciar no solo en esas especies de termitas si no también en muchas especies de hormigas e insectos que poseen el modelo de una colmena. En donde estas hormigas van buscando alimento para la colmena en un trayecto desconocido hasta llegar a dicha meta.

Cuando un individuo encuentra alimento despierta feromonas que alertan a los demás individuos para indicar que se encontró un objetivo en donde estas también generan esa reacción haciendo una cadena. Las hormigas entonces van creando un camino óptimo de la colonia al alimento, es decir el camino más corto para transportar a casa su alimento.

Este ejemplo se llevo a cabo en un experimento llamado el puente binario a cargo del científico Deneubourg usando hormigas *"Linepithema humile"* o también conocidas como hormigas argentinas. Este experimento consistía en colocar una colmena de hormigas y alimento en dos secciones apartadas y conectadas por dos puentes de misma longitud asegurándose de no haber feromonas en el trayecto. Se dejaron las hormigas a libertad por un tiempo especifico hasta notar resultados en los que las hormigas cruzaron ambos puentes hasta encontrar el alimento y regresarlo al hormiguero. Estas decisiones de las hormigas fueron espontáneas, pero con el pasar del tiempo uno de los dos puentes a pesar de tener la misma longitud era el preferido, esto resulta de esta manera porque con el tiempo un puente tiende a tener un camino de feromonas cada vez mas fuerte que el otro. Esto hace poco a poco que las hormigas converjan a elegir ese camino pese a que los dos puentes tengan la misma distancia.

6.4. Simple ant colony optimization (SACO)

Este algoritmo es un modelo que describe el comportamiento del experimento del puente binario hecho por Deneubourg. En donde el problema es encontrar el camino óptimo entre dos nodos en un grafo G = (N, A), en donde N es el conjunto de vértices o nodos y A es la matriz que representa la conexión entre los nodos. El algoritmo también tiene un set de

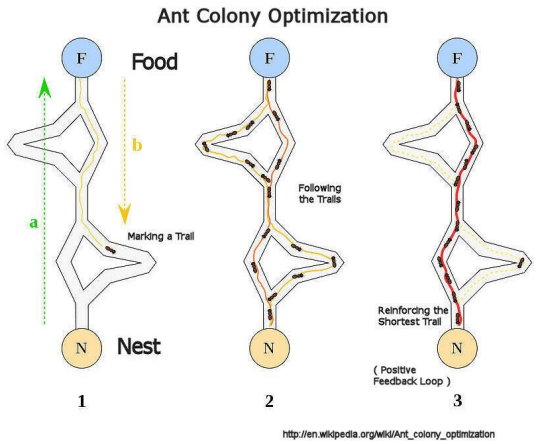


Figura 3: Algoritmo de inteligencia computacional *Ant Colony Optimization* [14].

variables $\tau = \tau_{ij}(t)$ llamada feromona artificial asociada a los arcos (i, j) del grafo G. La intensidad de cada camino con feromonas es proporcional a la utilidad calculada por las hormigas de usar el arco correspondiente para encontrar soluciones óptimas.

En el algoritmo SACO cada hormiga elabora desde el nodo inicial una solución candidata para el camino óptimo aplicando una decisión paso a paso. En cada nodo existe información local de la feromona que es guardada en el nodo y que se usan en los arcos de ese nodo para que sea percibida por las hormigas y usadas por ellas de una forma estocástica para decidir que camino seguir después. Cuando se localiza un nodo i la hormiga k usa los caminos de feromonas τ_{ij} para calcular la probabilidad p_{ij}^k de escoger j como el siguiente nodo.

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{j \in \mathcal{N}_i^k} \tau_{ij}^\alpha} & \text{if } j \in \mathcal{N}_i^k \\ 0 & \text{if } j \notin \mathcal{N}_i^k \end{cases}$$

Donde $\mathcal{N}_i^k$ es una vecindad factible de una hormiga k cuando se encuentra en un nodo i. Para el inicio del ciclo se asegura que $\tau_0 = 1$ para evitar una división entre cero de la expresión. En este algoritmo el vecino factible $\mathcal{N}_i^k$ de una hormiga k en un nodo i contiene todos los nodos directamente conectados a i a excepción del predecesor del nodo i, el cual fue el nodo que la hormiga k visito antes de moverse a i. De esta forma las hormigas evitan regresar a los mismos nodos que visitaron con anterioridad a i. Esto puede omitirse cuando existe un callejón sin salida en donde sería el caso $\mathcal{N}_i^k$ este vacío, nótese que esto puede ocasionar a las hormigas en entrar a un bucle infinito.

6.5. Sistema de captura de movimiento OptiTrack

El OptiTrack [10] es un sistema de captura de movimiento. Este cuenta con tecnología con la capacidad de grabar movimiento de personas, robots y objetos en un área delimitada. Estos datos se transfieren desde las cámaras a una aplicación de computadora en la cual

se puede visualizar en una interfaz gráfica en tiempo real los movimientos y poses de los individuos en el área. Para que las cámaras puedan captar estos movimientos es necesario de unas pelotas con pintura reflectora que tienen como objetivo reflejar la luz emanada desde las cámaras para que estas puedan determinar a partir de ese reflejo la posición de esta pelota en un plano tridimensional. Estos sistemas de captura de movimiento son ampliamente utilizados en la industria del cine y de los videojuegos.

Las cuatro diferentes técnicas que existen para los sistemas de captura movimiento son:

- Técnicas ópticas pasivas
- Técnicas ópticas activas
- Técnicas sin necesidad de pelotas reflectoras
- Técnicas inerciales

Hablando de las características de estos sistemas se puede empezar con la técnica óptica pasiva las que usan pelotas reflectivas que se colocan en la persona u objeto que será grabado. Estos van a reflejar la luz infrarroja emanada por las cámaras del sistema de captura. Una vez las cámaras identifiquen el reflejo van a determinar la posición del objeto en un plano tridimensional y transmitir estos datos a la aplicación de la computadora.

La técnica óptica activa por otra parte hace exactamente lo mismo que la pasiva con la diferencia que ahora las pelotitas son las que emanan la luz para que las cámaras detecten la posición de estas. Esto implica que estas pelotitas deben de contar con una fuente de poder para cada una de ellas.

Para el caso de las técnicas no ópticas se encuentran aquellas que no necesitan de estas pelotitas reflectivas. Estas cámaras tienen sensibilidad a la profundidad y tiene algoritmos para detectar y seguir cuerpos u objetos para grabarlos. Esta técnica es bastante mas cómoda de utilizar que las dos técnicas antes explicadas, pero con el costo de que no son tan precisas como las anteriores.

Finalmente se tiene la técnica inercial la cual no necesita de cámaras en cambio utiliza unidades de medición inercial que se caracterizan por tener giroscopios, magnetómetros, y acelerómetros los cuales envían sus datos para poder determinar posición y movimiento en un espacio tridimensional.

CAPÍTULO 7

Algoritmo Ant Colony Optimization para planificación de rutas y evasión de obstáculos

En este capítulo se presenta la metodología empleada para el algoritmo de planificación de trayectorias y evasión de obstáculos. Se detallan parámetros y exponen ejemplos hechos para diferentes puntos.

Se utilizó el entorno de MATLAB 2018 para desarrollar este algoritmo donde se busca la optimización de trayectorias de diferentes entornos posibles en los que podemos situar al dron *CrazyFlie2.0*.

7.1. Resultados antecedentes

Estos resultados cumplen con encontrar una ruta óptima entre dos puntos en un mapa tridimensional. A comparación del trabajos de graduación de Daniela Baldizón donde se presentan resultados en un mapa de dos dimensiones o plano.

7.1.1. Ruta del punto 1 al punto 100

En la Figura 4 se puede apreciar como el algoritmo encuentra la ruta óptima entre el punto 1 y el 100. Obstáculos presentes en los nodos 75, 66, 57, 44, 35, 26 y 17

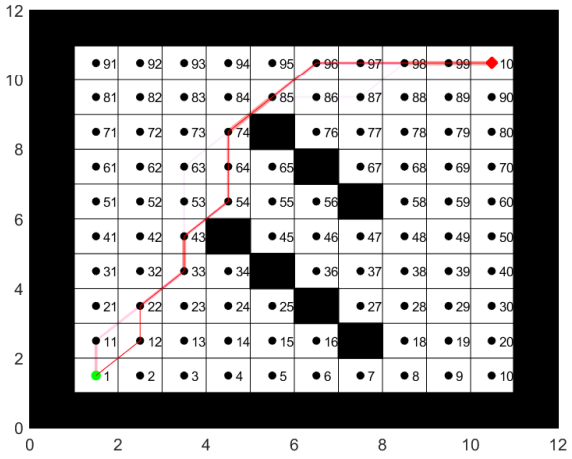


Figura 4: Trayectoria generada por Ant Colony Optimization. [16]

7.1.2. Ruta del punto 1 al punto 100 con mapa 2

En la Figura 5 se puede apreciar como el algoritmo encuentra la ruta óptima entre el punto 1 y el 100. Obstáculos presentes en los nodos 12,13,22,23,37,38,47,48,75,76,85 y 86

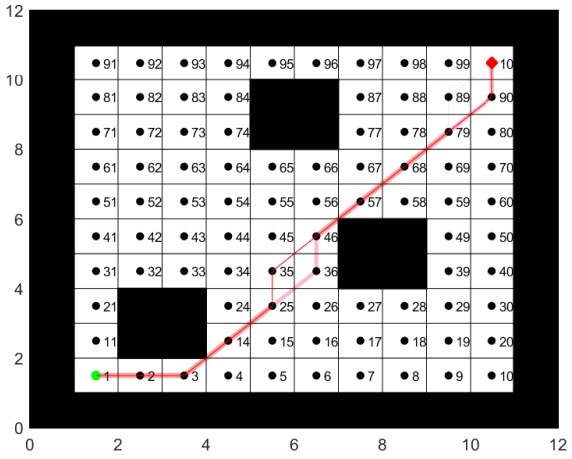


Figura 5: Trayectoria generada por Ant Colony Optimization. [16]

7.1.3. Ruta del punto 1 al punto 100 con mapa 3

En la Figura 6 se puede apreciar como el algoritmo encuentra la ruta óptima entre el punto 1 y el 100. Obstáculos presentes en los nodos 23,24,27,28,32,62,67,72,73,74,78,79

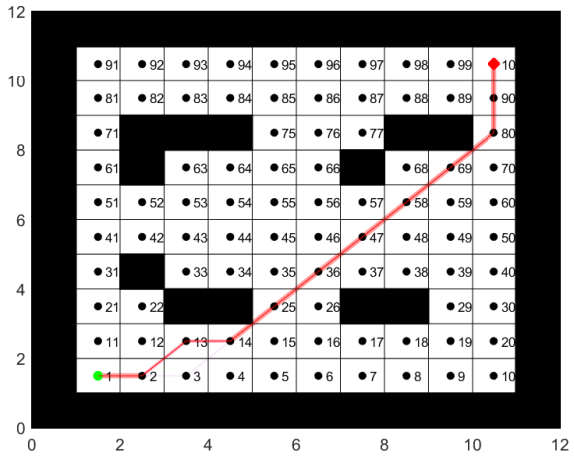


Figura 6: Trayectoria generada por Ant Colony Optimization. [16]

7.2. Metodología

Para el Ant Colony Optimization Se tomó de base el algoritmo de las fases anteriores [16] [17] y se modificó de acuerdo al la aplicación de optimización de trayectorias para un dron en tres dimensiones. La base de este algoritmo son los grafos, los cuales forman espacios de 4×4×4. Para simular los obstáculos se hicieron planos a partir de cuatro grafos o figuras sólidas que usan varios grafos para formarlas. En la Figura 7 se puede apreciar el mapa tridimensional con todas las trayectorias posibles o aristas entre nodos sin obstáculos.

El mapa tridimensional se define de la siguiente manera. Primero se establecen la cantidad nodos del mapa donde al tratarse de un cubo se sabe que tiene cuatro nodos de ancho, cuatro nodos de profundidad y cuatro nodos de altura. Esto para dar un resultado de sesenta y cuatro nodos en el mapa como se presenta en la Figura 8.

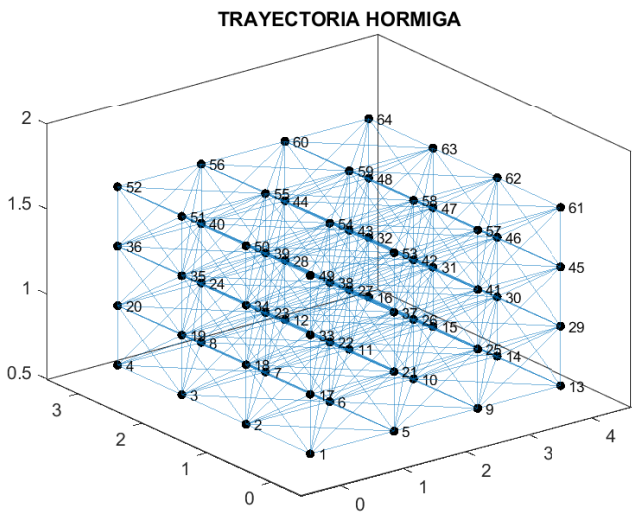


Figura 7: Espacio tridimensional compuesto por grafos y aristas.

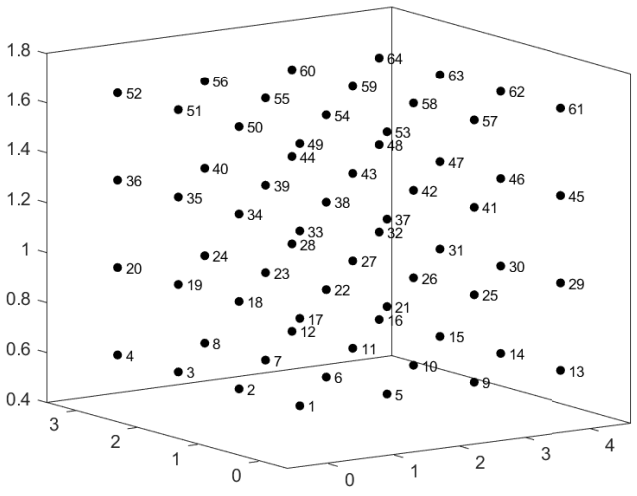


Figura 8: Nodos en el mapa tridimensional.

Ya teniendo los nodos definidos se proceden a definir las aristas que une las posibles rutas. Con estos parámetros definidos se puede crear un grafo en MATLAB que se compone a base de dos tablas una con el nombre de los nodos y las posiciones (x, y, z) en vectores columna para cada una y otra tabla con la matriz peso, feromona y conexión nodo a nodo, que viene siendo los nodos que queremos unir con una arista. Ya teniendo todos estos parámetros definidos se aprecia el resultado en la Figura 7.

7.3. Parámetros

$Parametros$	$Valor$
ρ	0.35
α	0.7
β	1
Q	0.7
$Hormigas$	500
$iteraciones$	2500

Cuadro 1: Esta tabla muestra los valores ingresados en el algoritmo Ant Colony Optimization.

7.4. Validación del algoritmo Ant Colony Optimization

A continuación se presentan resultados del algoritmo generando rutas, sin obstáculos, para diferentes puntos de meta a partir de un punto de partida 1. Esto se elaboró a partir de los parámetros mencionados en el Cuadro 2.

7.4.1. Punto 1 a punto 64

En la Figura 9 se puede observar como se presenta una linea recta desde inicio hasta la meta deseada al tratarse de la ruta óptima al no haber obstáculos.

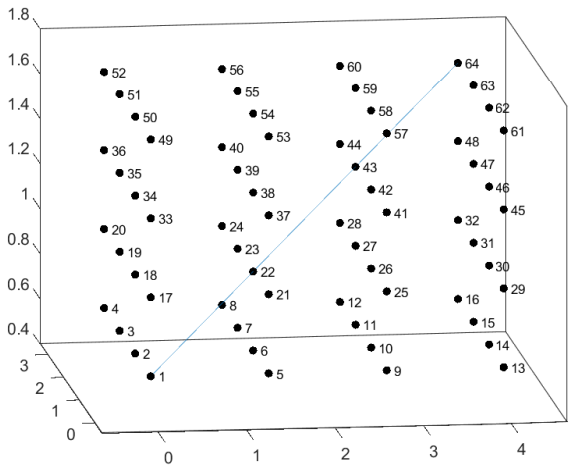


Figura 9: Trayectoria generada por Ant Colony Optimization.

7.4.2. Punto 5 a punto 36

En la Figura 10 se puede observar como se presenta una ruta óptima entre puntos sin obstáculos.

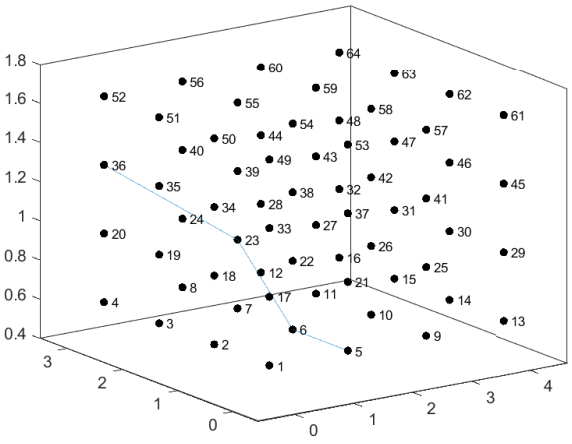


Figura 10: Trayectoria generada por Ant Colony Optimization.

7.4.3. Punto 3 a punto 62

En la Figura 11 se puede observar como se presenta una ruta óptima entre puntos sin obstáculos.

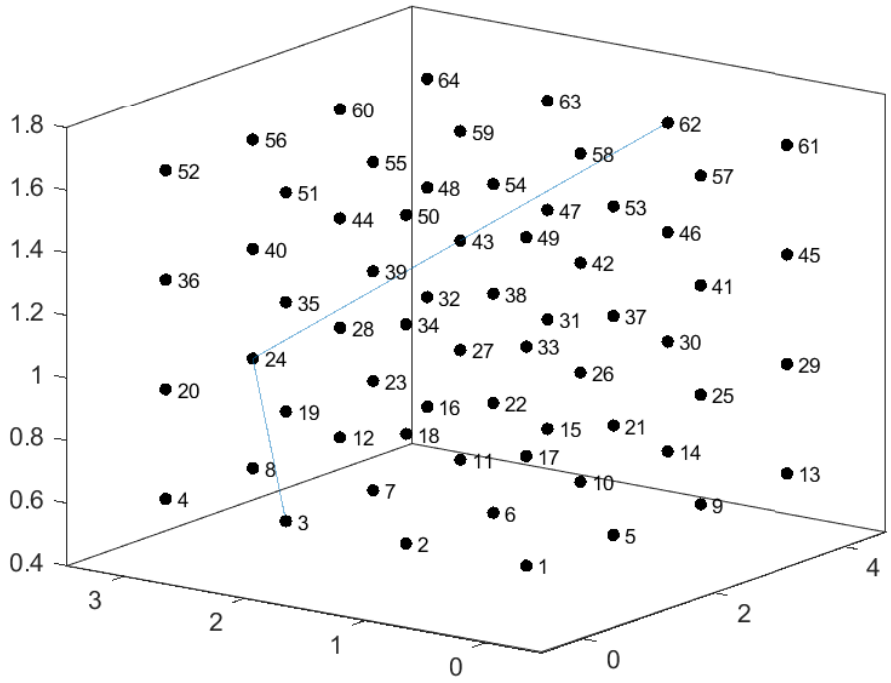


Figura 11: Trayectoria generada por Ant Colony Optimization.

7.5. Validación del algoritmo Ant Colony Optimization con obstáculos

7.5.1. Punto 1 a punto 64 con obstáculos

En la Figura 12 se puede observar como se presenta una ruta óptima entre puntos esquivando los obstáculos verdes representados por los nodos 22, 26, 43 y 47.

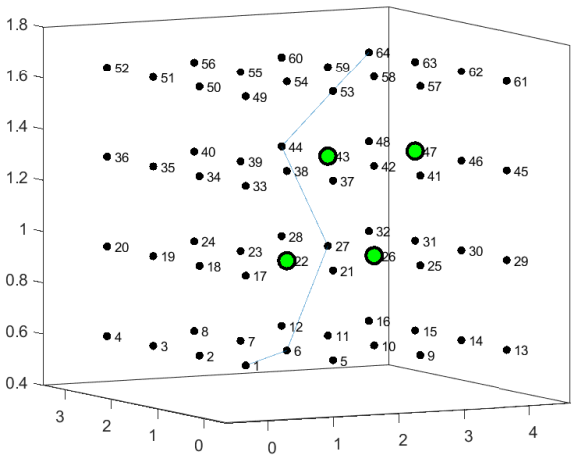


Figura 12: Trayectoria generada por Ant Colony Optimization.

7.5.2. Punto 5 a punto 44 con obstáculos

En la Figura 13 se puede observar como se presenta una ruta óptima entre puntos esquivando los obstáculos verdes representados por los nodos 23, 26, 39 y 47.

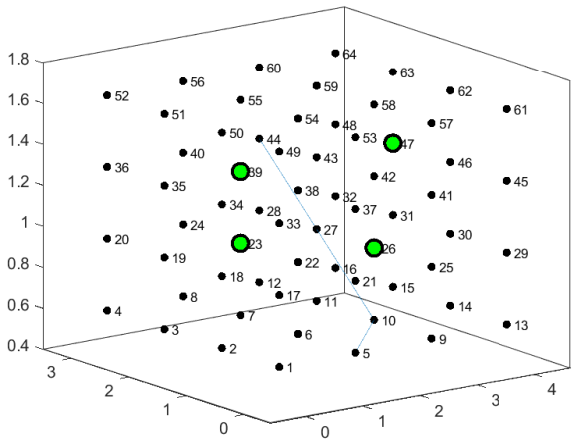


Figura 13: Trayectoria generada por Ant Colony Optimization.

7.5.3. Punto 2 a punto 48 con obstáculos

En la Figura 14 se puede observar como se presenta una ruta óptima entre puntos esquivando los obstáculos verdes representados por los nodos 23, 26, 39 y 47.

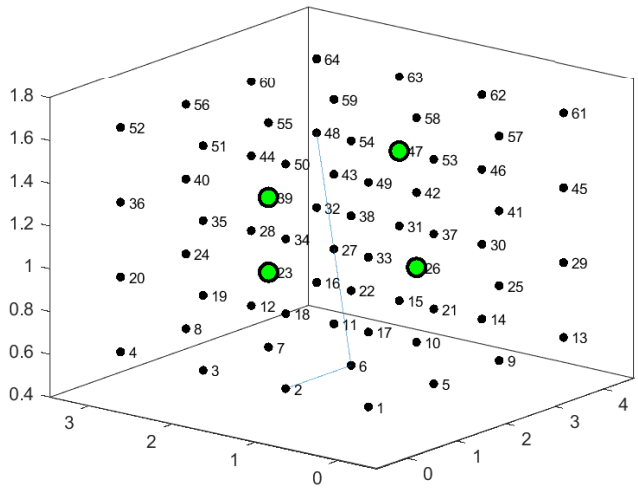


Figura 14: Trayectoria generada por Ant Colony Optimization.

CAPÍTULO 8

Algoritmo Particle Swarm Optimization para planificación de rutas y evasión de obstáculos.

En este capítulo se presenta el algoritmo de Particle Swarm Optimization para encontrar la solución de una ecuación de costo buscando los mínimos de dicha función. Esto con el objetivo de poder encontrar un punto, en un mapa tridimensional,donde se quiere que llegue a converger el dron sobre un punto solución de mínimos. Se empezó haciendo una adaptación del modelo en un plano bidimensional donde se utilizan dos variables para encontrar el mínimo. Este algoritmo en dos dimensiones se suele utilizar para agentes robóticos que se desplazan sobre un plano como los robots E-Pucks usados en los trabajos de graduación antecesores [16] y [17]. Pero para este caso se utilizan los 3 planos de un mapa tridimensional y tres variables para adaptarse a las necesidades del dron *CrazyFlie2.0*.

8.1. Resultados Particle Swarm Optimization en dos dimensiones

8.1.1. Buscando el costo mínimo de la función $f = (x-40)^2 + (y-20)^2$

En las siguientes imágenes se tienen dos momentos de la simulación, donde en la Figura 15 se aprecia el algoritmo calculando el punto mínimo de la función. Este punto mínimo se establece en $(x = 40, y = 20)$. Después de ciertas iteraciones el algoritmo encuentra el punto mínimo como se aprecia en la Figura 16.

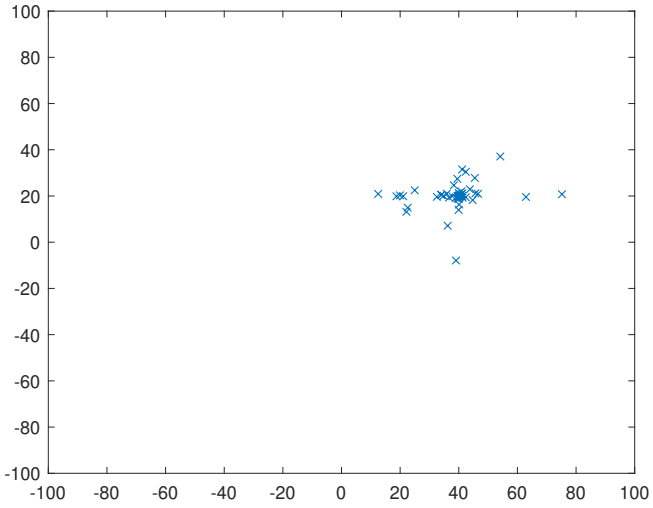


Figura 15: Particle Swarm Optimization ejecutándose.

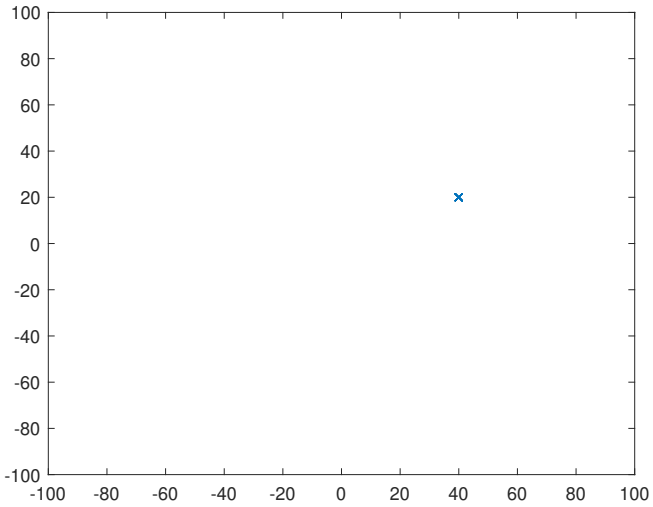


Figura 16: Particle Swarm Optimization solución.

8.1.2. Buscando el costo mínimo de la función $f = (x + 50)^2 + (y - 60)^2$

En las siguientes imágenes se tienen dos momentos de la simulación, donde en la Figura 17 se aprecia el algoritmo calculando el punto mínimo de la función. Este punto mínimo se establece en $(x = -50, y = 60)$. Después de ciertas iteraciones el algoritmo encuentra el punto mínimo como se aprecia en la Figura 18.

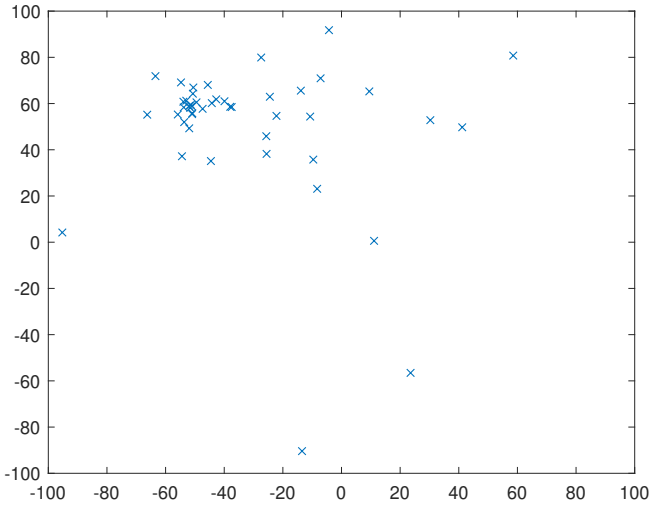


Figura 17: Particle Swarm Optimization ejecutándose.

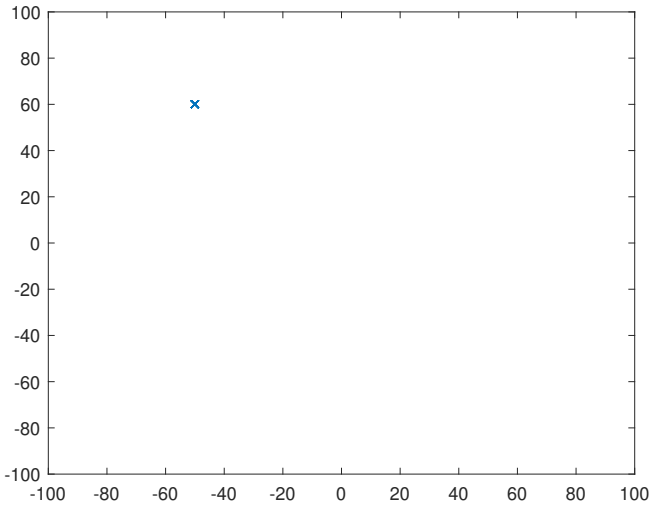


Figura 18: Particle Swarm Optimization solución.

8.2. Metodología

Se definen parámetros iniciales del algoritmo entre los cuales se tienen la inercia de las partículas, el factor de corrección, la cantidad de iteraciones y el número de partículas.

Se define la ecuación a la cual se le encontrará los mínimos. Para este trabajo de graduación se utilizará un hiper-paraboloide elíptico $f = (u-1)^2 + (v-1)^2 + (w-1)^2$ [19]. Esto con el fin de poder definir el punto mínimo como el punto donde se quiere que llegue el dron como meta y donde eventualmente todas las partículas convergerán.

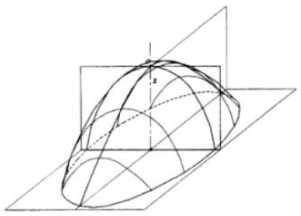


Figura 19: Función paraboloide elíptico[18].

Se define una matriz donde irán guardados los valores de las posiciones (u,v,w). las mejores posiciones (u,v,w), las velocidades de (u,v,w) y el valor mínimo.

8.3. Parámetros

$Parametros$	$Valor$
$iteraciones$	100
$inercia$	1.0
$factor - correccion$	2.0
$partculas$	20

Cuadro 2: Esta tabla muestra los valores ingresados en el algoritmo Particle Swarm Optimization.

8.4. Validación de la primera versión del algoritmo Particle Swarm Optimization

A continuación se muestran resultados donde se aprecia como las partículas convergen, en un mapa tridimensional, a la solución de la ecuación propuesta. Se señalan los valores a los cuales tienen que llegar cada coordenada.

8.4.1. Función $f = (u - 45)^2 + (v - 16) +^2 (w - 32)^2$

En las siguientes imágenes se tienen dos momentos de la simulación, donde en la Figura 20 se aprecia el algoritmo calculando el punto mínimo de la función. Este punto mínimo se establece en $(u = 45, v = 16, w = 32)$. Después de ciertas iteraciones el algoritmo encuentra el punto mínimo como se aprecia en la Figura 21.

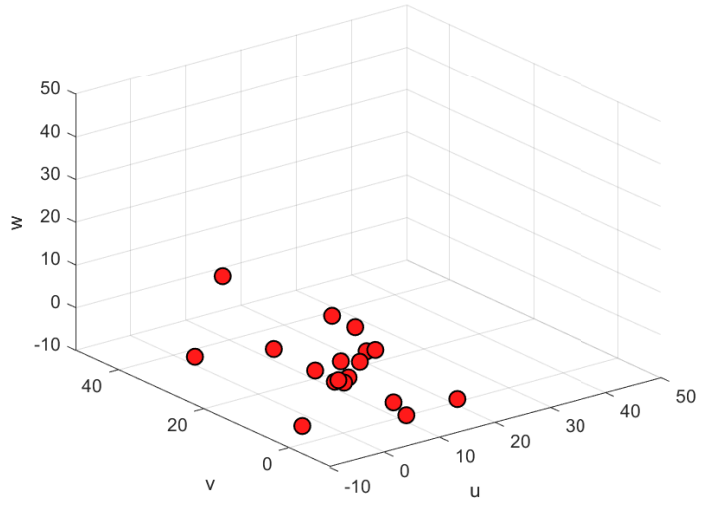


Figura 20: Código Particle Swarm Optimization ejecutándose.

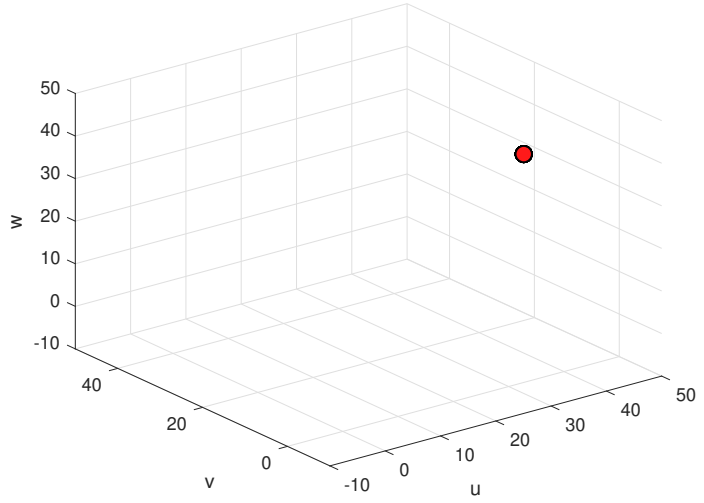


Figura 21: Solución encontrada con PSO.

8.4.2. Función $f = (u-18)^2 + (v-50) +^2 (w-35)^2$

En las siguientes imágenes se tienen dos momentos de la simulación, donde en la Figura 22 se aprecia el algoritmo calculando el punto mínimo de la función. Este punto mínimo se establece en $(u = 18, v = 50, w = 35)$. Después de ciertas iteraciones el algoritmo encuentra el punto mínimo como se aprecia en la Figura 23.

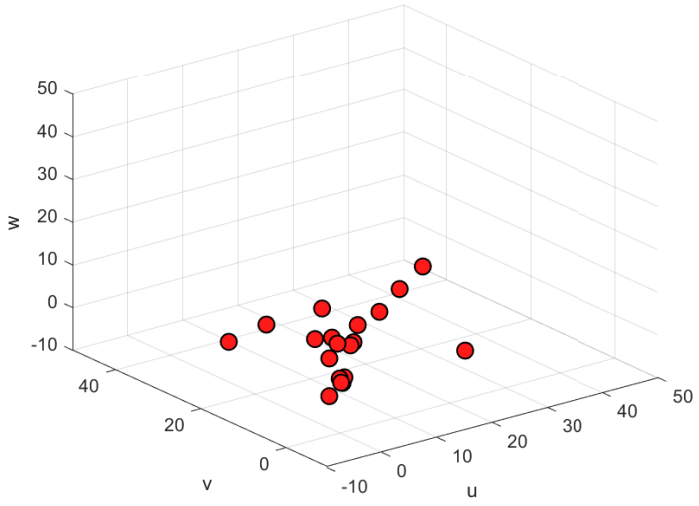


Figura 22: Código Particle Swarm Optimization ejecutándose.

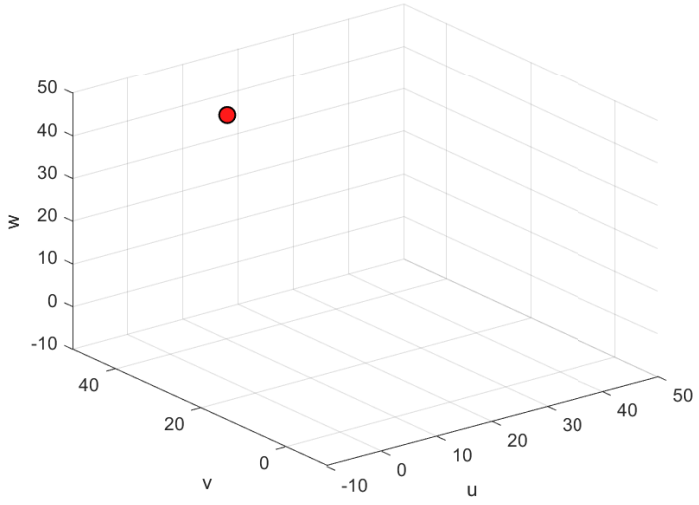


Figura 23: Solución encontrada con PSO.

8.5. Segunda versión del algoritmo Particle Swarm Optimization

A continuación se muestra como se adaptó la primera versión algoritmo, que funciona para encontrar el mínimo de un hiper-paraboloide elíptico, a una aplicación de generación de múltiples trayectorias posibles para que el dron *CrazyFlie2.0* pueda escoger dependiendo cual de los nodos iniciales este mas cerca de el. Además como las partículas ocupan básicamente todo el volumen del espacio que usara el dron, estos no realizaran una tarea de búsqueda del punto de referencia gracias a que siempre habrá una partícula cerca. Gracias a esto el resto del enjambre desde el inicio sabe el punto a donde debe de dirigirse.

Los obstáculos se simulan como nodos verdes en el espacio con una órbita de tolerancia de 0.5m de radio, que hace una repulsión a las partículas que cruzan dicha órbita.

8.5.1. Nodo de meta $x = 4$ $y = 3$ $z = 1.65$

En las siguientes imágenes se tienen tres momentos de la simulación, donde en la Figura 24 se aprecia la población inicial en punto de partida para empezar a generar las trayectorias múltiples. En la Figura 25 se aprecia como las partículas se dirigen a la meta siempre evitando los obstáculos modelados como nodos verdes. Finalmente en la Figura 26 se aprecia como todas las partículas llegan a la meta final ya teniendo guardado su trayectoria. En la Figura 27 se pueden apreciar todas las trayectorias de cada partícula hasta la referencia $x = 4$ $y = 3$ $z = 1.65$.

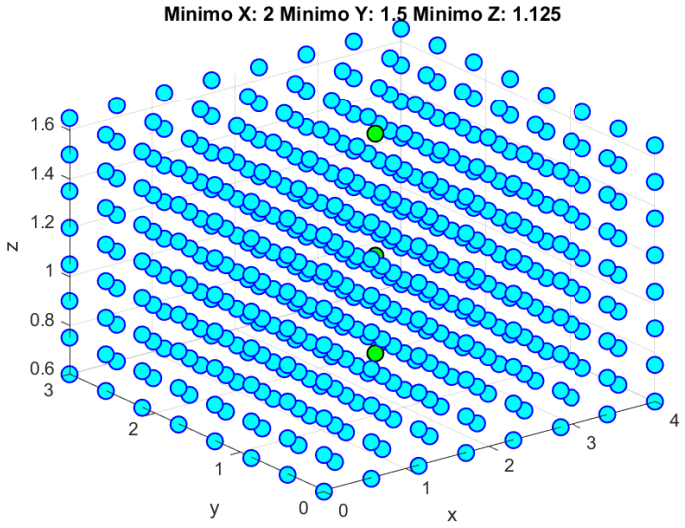


Figura 24: Trayectorias múltiples utilizando Particle Swarm Optimization.

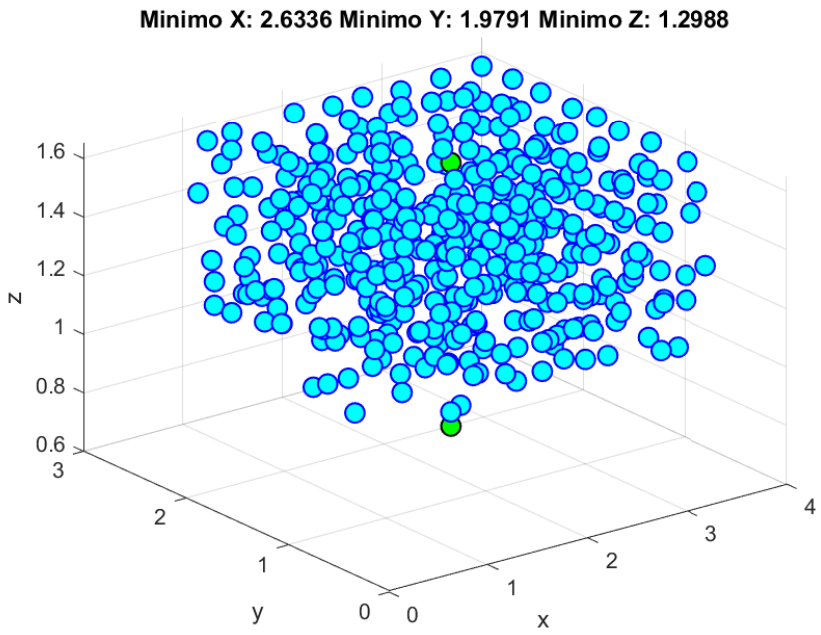


Figura 25: Trayectorias múltiples utilizando Particle Swarm Optimization.

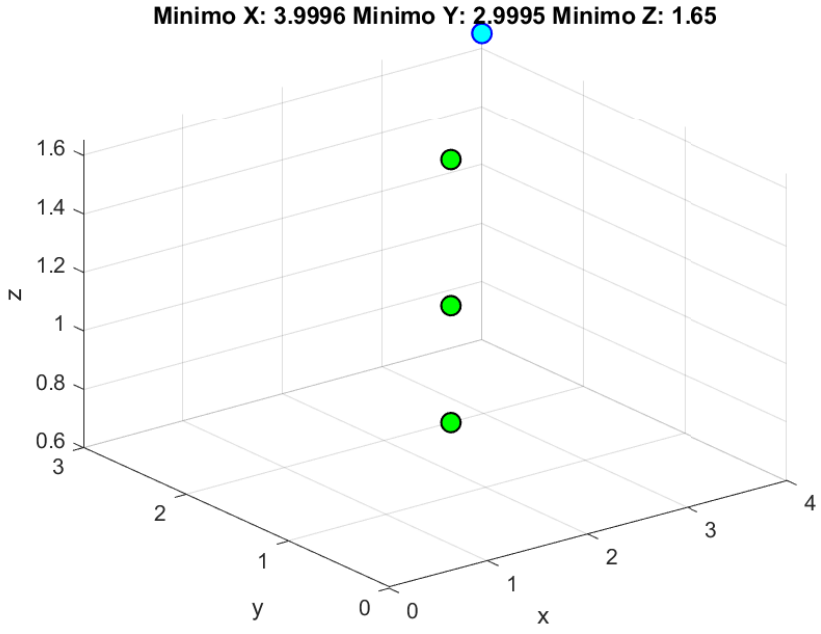


Figura 26: Trayectorias múltiples utilizando Particle Swarm Optimization.

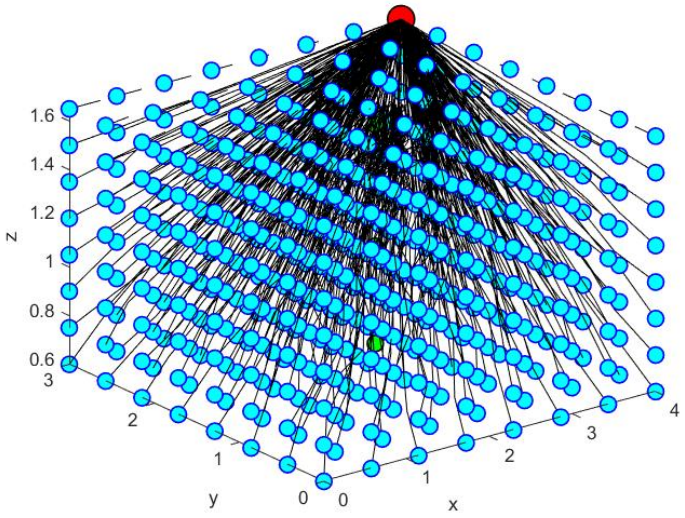


Figura 27: Trayectorias múltiples utilizando Particle Swarm Optimization.

8.5.2. Nodo de meta $x = 1$ $y = 3$ $z = 1$

En las siguientes imágenes se tienen tres momentos de la simulación, donde en la Figura 28 se aprecia la población inicial en punto de partida para empezar a generar las trayectorias múltiples. En la Figura 29 se aprecia como las partículas se dirigen a la meta siempre evitando los obstáculos modelados como nodos verdes. Finalmente en la Figura 30 se aprecia como todas las partículas llegan a la meta final ya teniendo guardado su trayectoria. En la Figura 31 se pueden apreciar todas las trayectorias de cada partícula hasta la referencia $x = 1$ $y = 3$ $z = 1$.

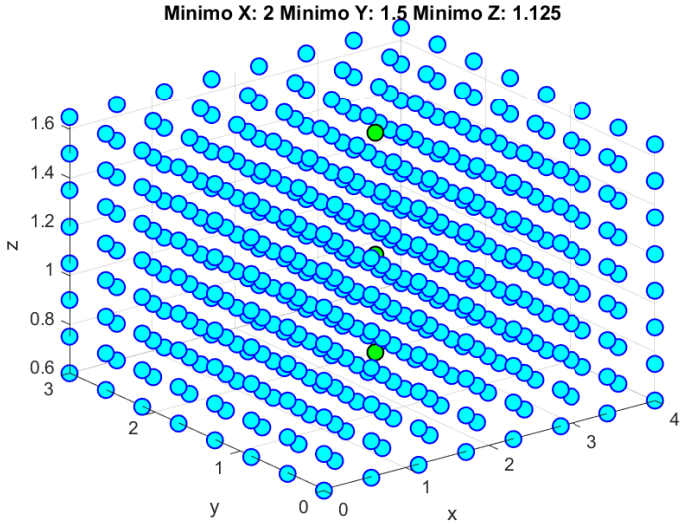


Figura 28: Trayectorias múltiples utilizando Particle Swarm Optimization.

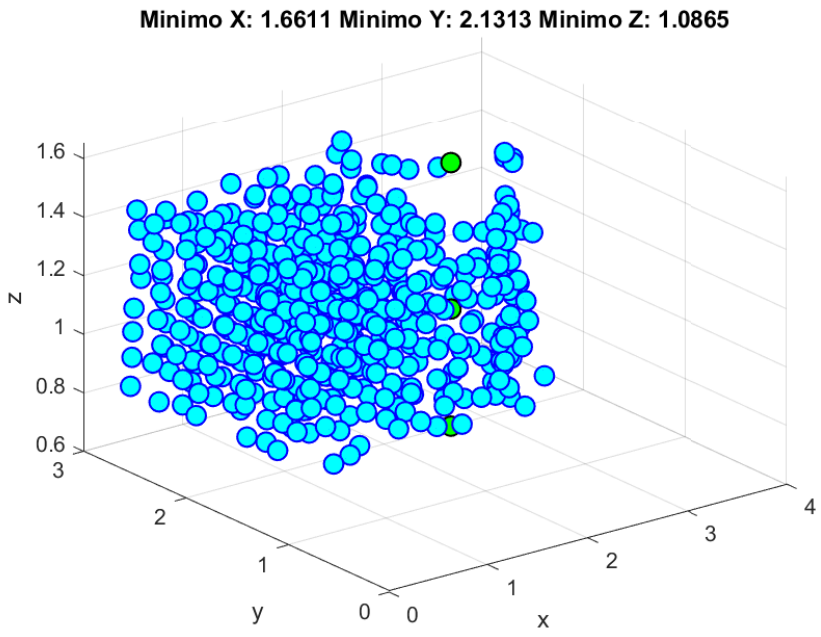


Figura 29: Trayectorias múltiples utilizando Particle Swarm Optimization.

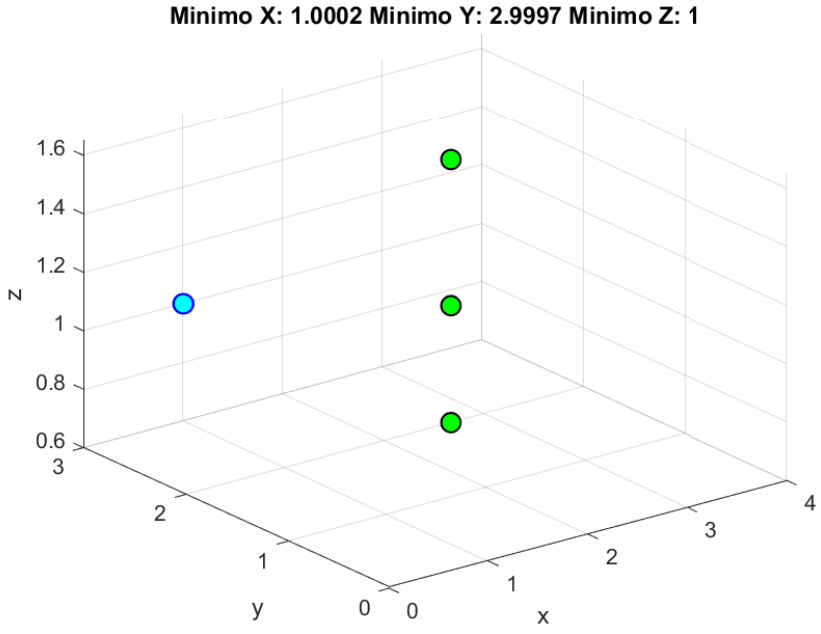


Figura 30: Trayectorias múltiples utilizando Particle Swarm Optimization.

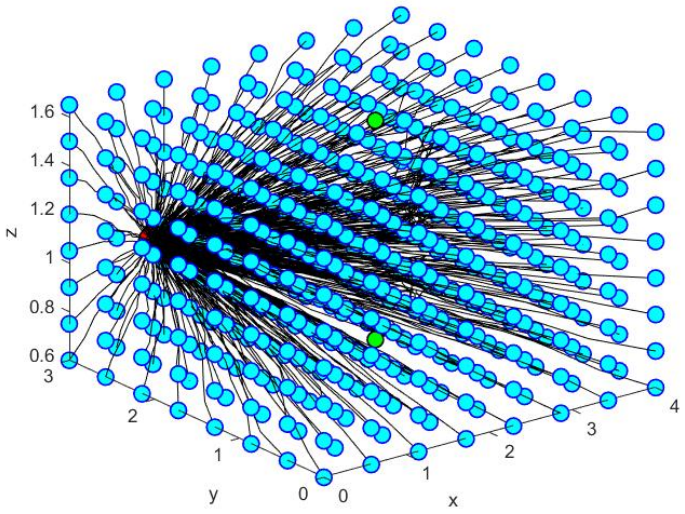


Figura 31: Trayectorias múltiples utilizando Particle Swarm Optimization.

CAPÍTULO 9

Identificación de marcadores con el sistema OptiTrack

En este capítulo se presenta una serie de resultados donde se utiliza el sistema de captura de movimiento OptiTrack. Esto tiene el objetivo de poder exponer el uso de este sistema el cual se utilizó para saber la posición de los obstáculos y la posición inicial del drone *CrazyFlie2.0*. Se utilizo el ecosistema Robotat [10] en conjunto con un servidor montado en una computadora que recibe los datos de las cámaras para posteriormente poderse enviar por medio de un router. El código fue elaborado en MATLAB y es propio del laboratorio de robótica de la Universidad del Valle de Guatemala 32 elaborado por el Catedrático de robótica Msc. Miguel Zea. Las dimensiones que maneja el sistema OptiTrack son de 4.2 metros en el eje x, 3.4 metros en el eje z y 1.60 metros en el eje y, estas medidas ya fueron establecidas en los algoritmos.

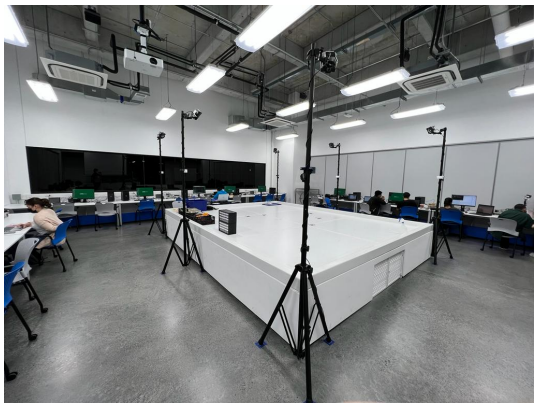


Figura 32: Laboratorio de robótica de la Universidad del Valle de Guatemala.

9.1. Sistema de captura de movimiento OptiTrack de la Universidad del Valle de Guatemala

9.1.1. Cámaras del OptiTrack

En las siguientes imágenes se aprecian como están puestas y como son las cámaras del sistema de captura de movimiento. Son seis cámaras colocadas simétricamente en las esquinas y los lados de la plataforma 33. En 34 se puede observar cómo es una de estas cámaras del sistema de captura de movimiento OptiTrack la cual en su parte de enfrente despliega un número con la cual ella se numera en el programa. Como se mencionó en el marco teórico estas cámaras utilizan infrarrojos que se disparan de alrededor del lente que se puede apreciar como puntos rojos en 35. Además de una leve luz azul que se aprecia en 34 y 35 que será la que llegue a las pelotas reflectivas de los marcadores para determinar sus posiciones.

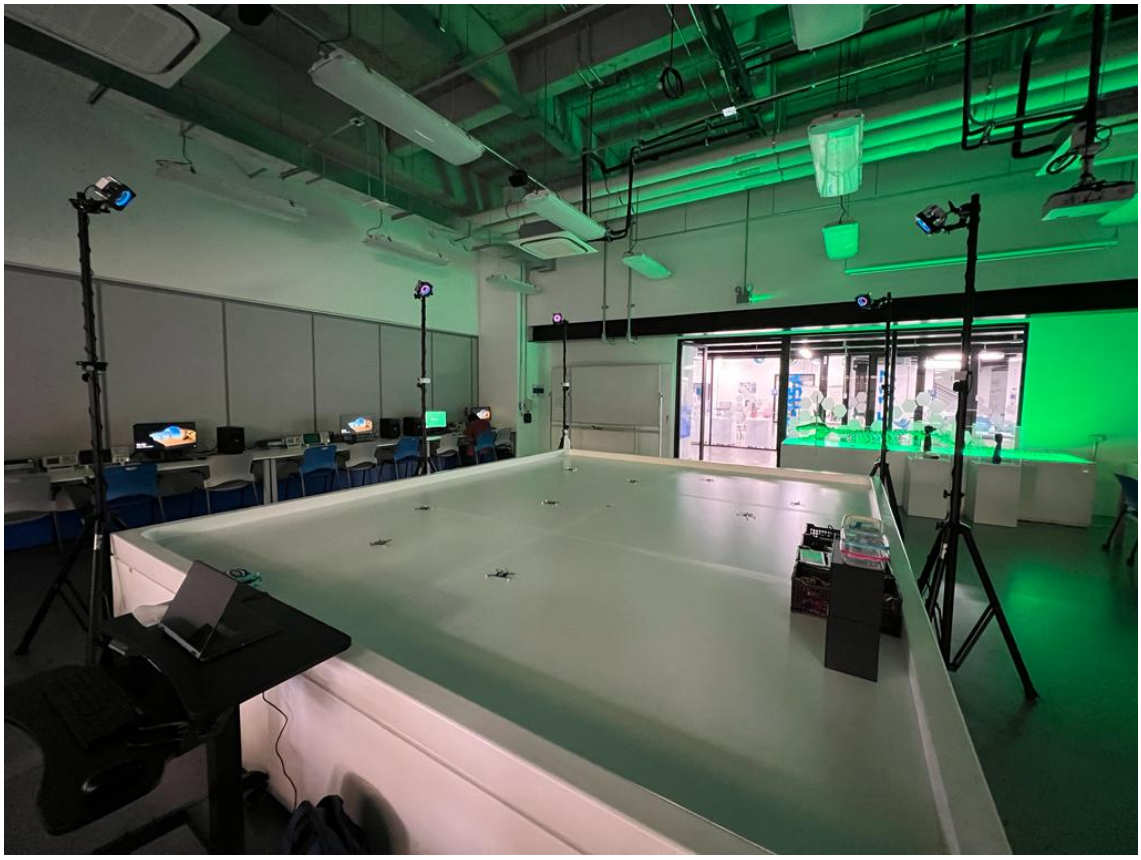


Figura 33: Sistema Robotat de la Universidad del Valle de Guatemala.

Figura 34: Cámara de captura de movimiento OptiTrack.

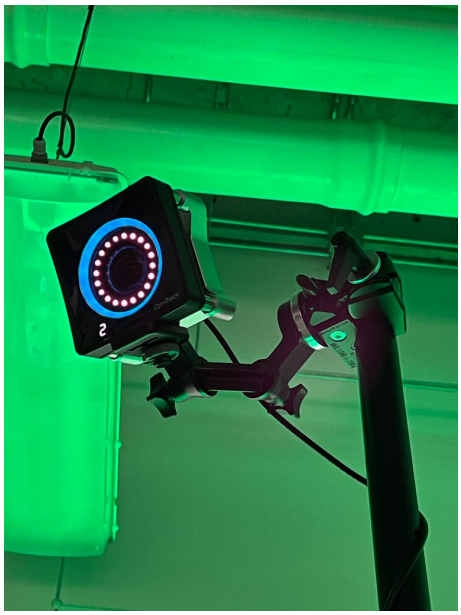


Figura 35: Cámara de captura de movimiento OptiTrack.

9.1.2. Marcadores para posiciones dentro del Robotat

Dentro de la plataforma 36 se colocan diferentes marcadores los cuales pueden simplemente dejarse dentro de esta y representar un obstáculo o colocárselos a agentes robóticos. Para el caso del dron *CrazyFlie2.0* se colocaran solo las pelotas reflectivas en el para no colocarle mucha carga al agente. y para los obstáculos se pueden colocar elementos dentro del entorno con uno de estos marcadores pegados a él.

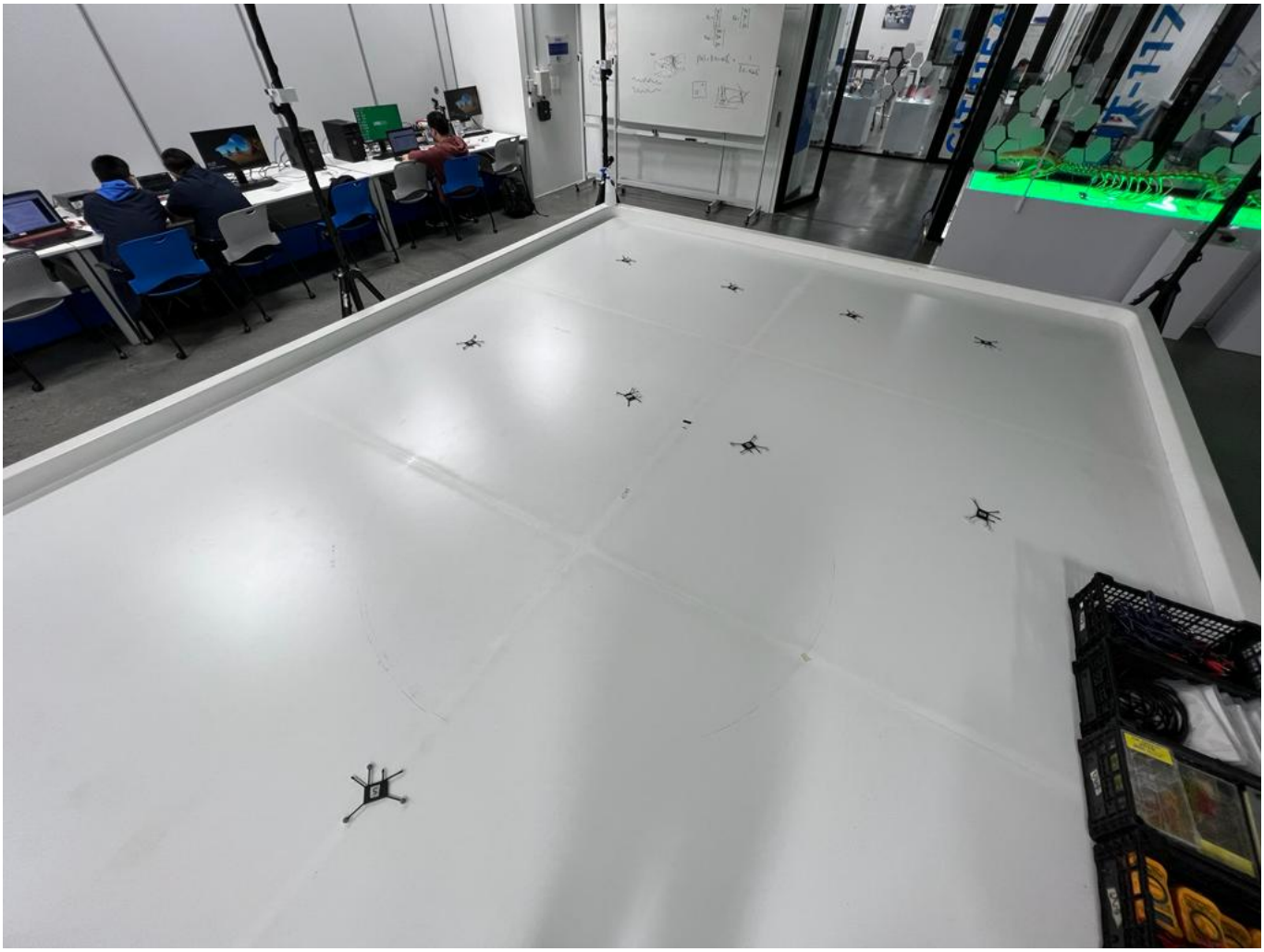


Figura 36: Marcadores dentro de la plataforma.

Los marcadores tienen una forma bastante básica la cual esta conformada por una estructura plástica con tornillos que se enroscan a las pelotas reflectivas. Estas se pueden apreciar como reflejan la luz del flash de la cámara en la Figura 37.

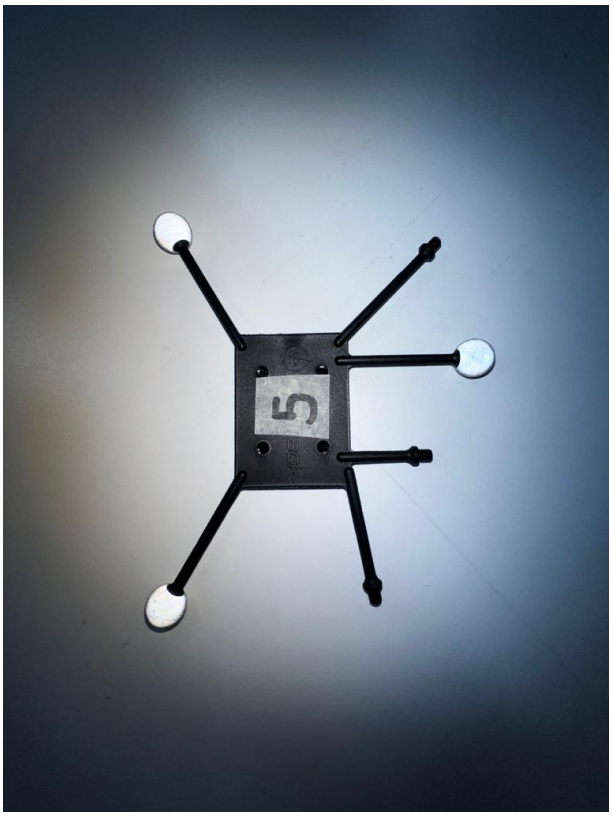


Figura 37: Marcador.

9.2. Envió de datos del OptiTrack por medio del ecosistema Robotat

El sistema Optitrack no puede mandar datos de forma inalámbrica a servidores que necesitan saber información de marcadores de la plataforma. Por lo tanto se lleva un proceso de envió de datos por medio de un servidor en una computadora con el programa controlador de las cámaras a un router, en donde el es el encargado de mandar estos datos a computadoras conectadas a su red.

9.2.1. Programa Motive Body

Motive Body 38 es el software encargado de poder controlar las cámaras de captura de movimiento dando una visualización de los cuerpos rígidos y de las cámaras en una visualización discreta. Además de poder entregar las coordenadas (x,y,z) de cada marcador y sus ángulos *Roll*, *Pitch* y *Yaw*

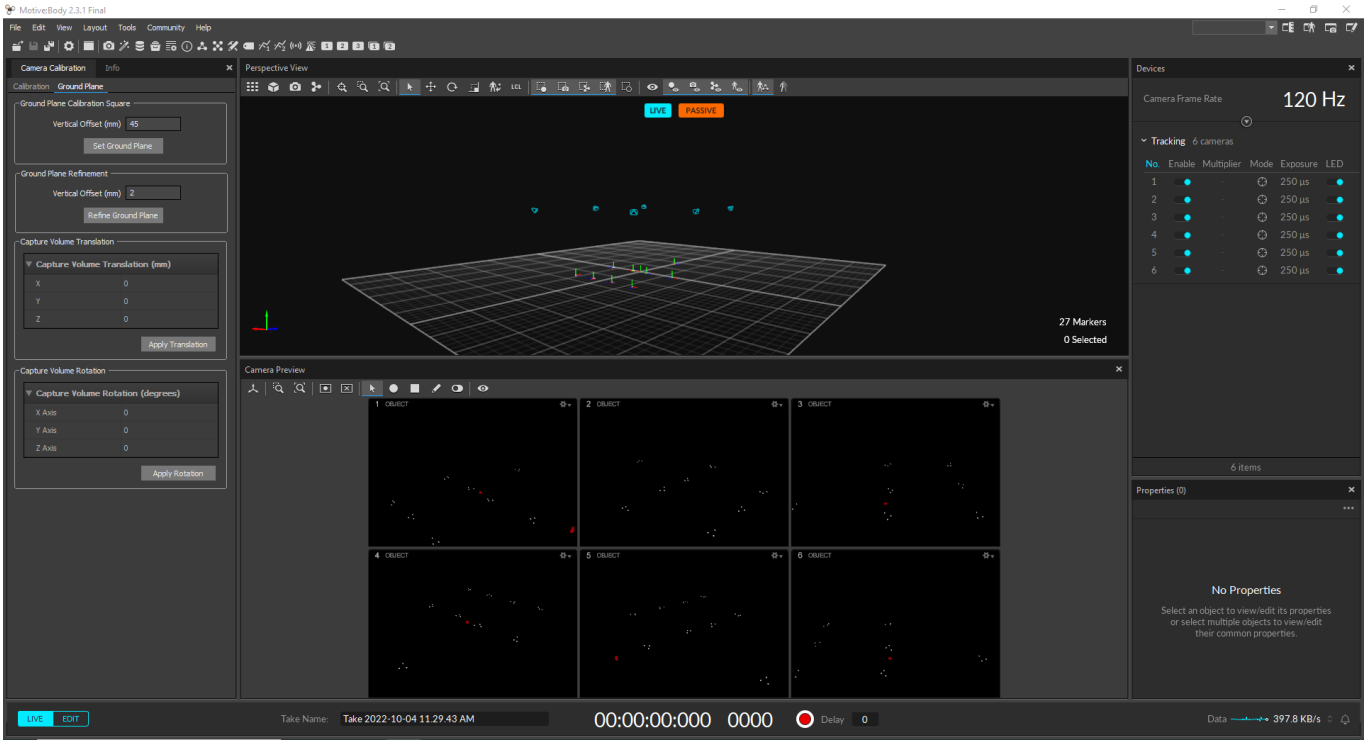


Figura 38: Software Motive Body.

En la Figura 39 se puede apreciar la sección del software donde se puede visualizar el entorno de una forma discreta donde hace las siluetas de las cámaras y de los marcadores en un entorno tridimensional. También otorga un conteo de la cantidad de marcadores y la dirección de los ejes.

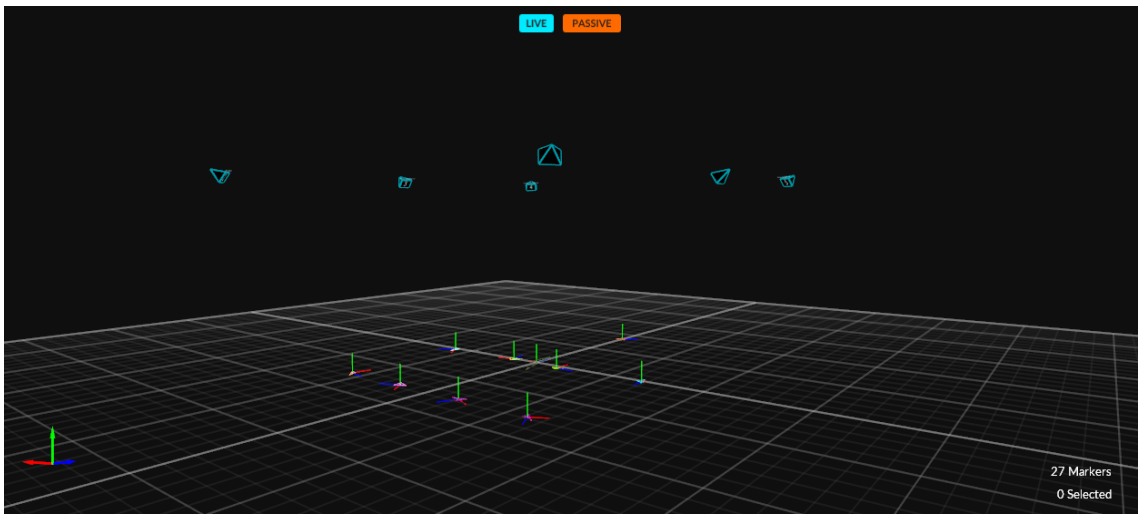


Figura 39: Mapa tridimensional discreto.

En otra sección 40 se pueden observar la visualización de cada cámara a la plataforma con los marcadores reflejando la luz que sale de las cámaras.

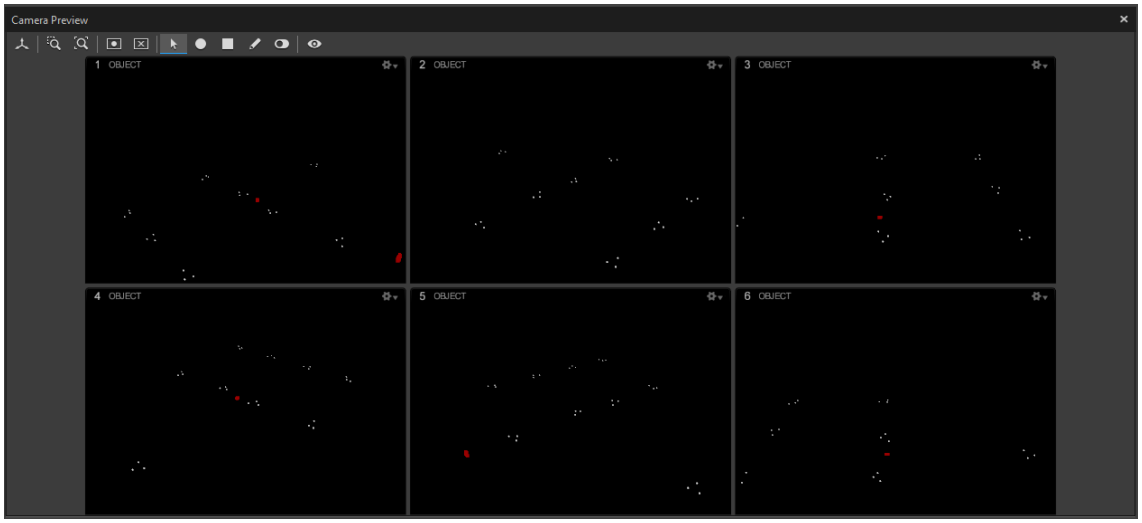


Figura 40: Vista de cada cámara.

9.2.2. Solicitud de datos del OptiTrack

Con el código proporcionado por el catedrático del laboratorio de robótica solo es necesario colocar la IP del router del Robotat y estar conectado a su red wifi para establecer una linea directa de comunicación con este. Una vez conectados se solicitan los datos por medio de una función que requiere el numero del marcador a solicitar.

Datos del marcador 5

En las Figuras 41 y 42 se pueden apreciar los datos del marcador 5 donde se aprecian los datos tanto en el software Motive Body como en el de MATLAB enviados por WIFI desde el router del Robotat.

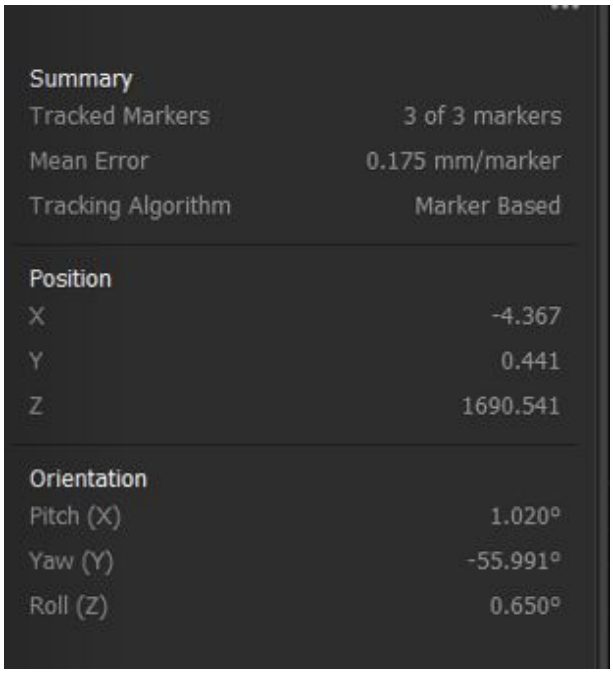


Figura 41: Datos del marcador 5 en Motive Body en milímetros y grados.

```
posiciones5 =

   -0.0041    0.0009    1.6904

>> angulosdeg5

angulosdeg5 =

   -0.2951
  -55.9794
    0.8105
```

Figura 42: Datos del marcador 5 recibidos en MATLAB en metros y grados.

Datos del marcador 10

En las Figuras 41 y 42 se pueden apreciar los datos del marcador 10 donde se aprecian los datos tanto en el software Motive Body como en el de MATLAB enviados por WIFI desde el router del Robotat.

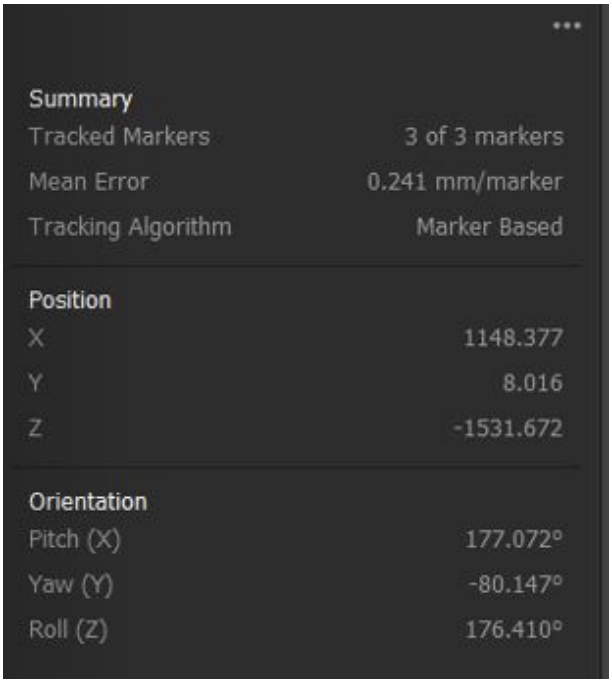


Figura 43: Datos del marcador 10 en Motive Body en milímetros y grados.

```
posiciones10 =

    1.1485    0.0084   -1.5312

>> angulosdeg10

angulosdeg10 =

 -175.9436
  -80.1333
  176.5080
```

Figura 44: Datos del marcador 10 recibidos en MATLAB en metros y grados.

CAPÍTULO 10

Validación de resultados de los algoritmos de inteligencia computacional en el entorno Webots

En este capitulo se presentan los resultados de las simulaciones de Webots [45] con el dron *CrazyFlie2.0*. El modelo del dron y el mundo se consiguieron en [19] donde nos presentan un programa sencillo para manipular al dron con el teclado. Este archivo fue modificado conforme lo necesitado para hacer pruebas con los algoritmos. Se añadieron tres controladores proporcionales de constante 1 para controlar la posición del dron en una rutina de trayectorias de punto a punto.

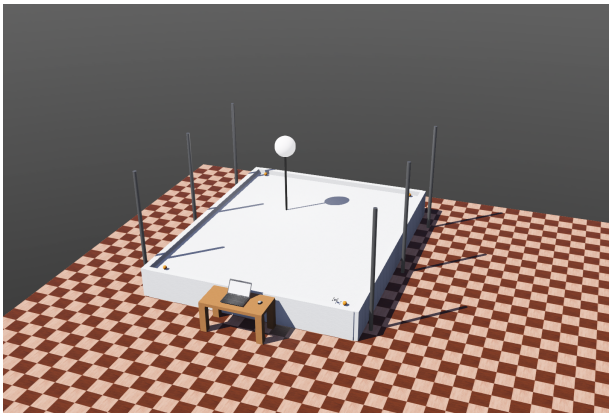


Figura 45: Simulación entorno Robotat en webots para el dron *CrazyFlie2.0*.

10.1. Simulación de rutas en Webots con el dron *CrazyFlie2.0* para el algoritmo Ant Colony Optimization

10.1.1. Ruta del nodo 1 al nodo 64 con lámpara como obstáculo

En la siguiente Figura 46 se presenta el resultado del algoritmo Ant Colony Optimization en el cual calcula una trayectoria de punto a punto del nodo 1 al nodo 64 con cuatro nodos simulando el obstáculo que en Webots se mostrara como una lampara. En las Figuras 48, 49, 50, 51 y 52 se muestran los puntos en donde el dron converge para desplazarse hasta la meta.

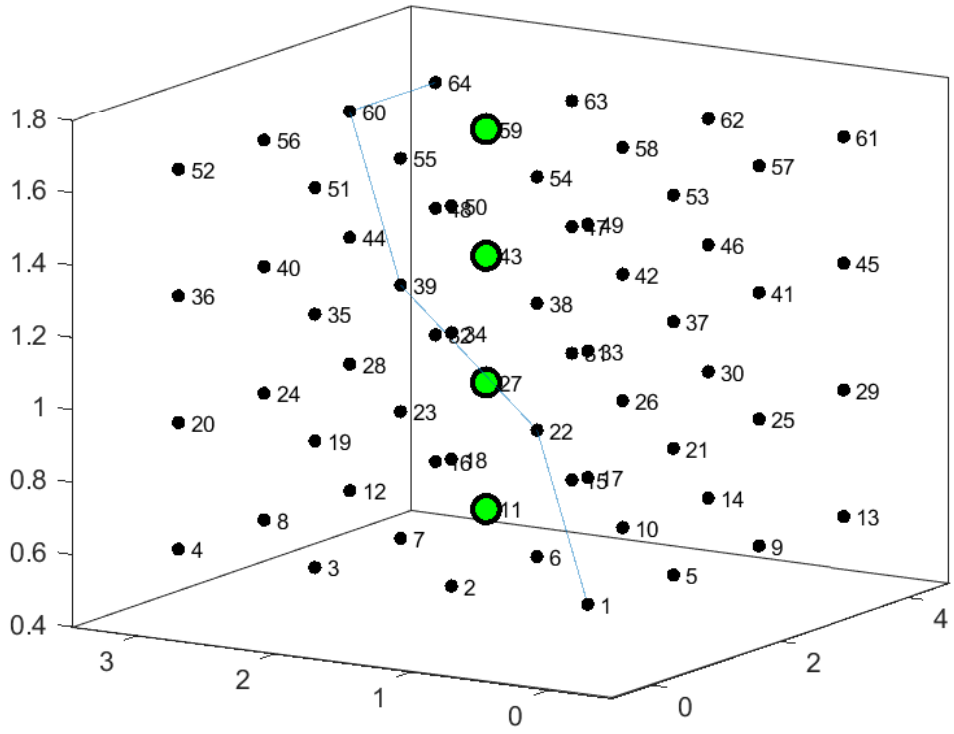


Figura 46: Trayectoria óptima del nodo 1 al nodo 64 con lámpara como obstáculo.

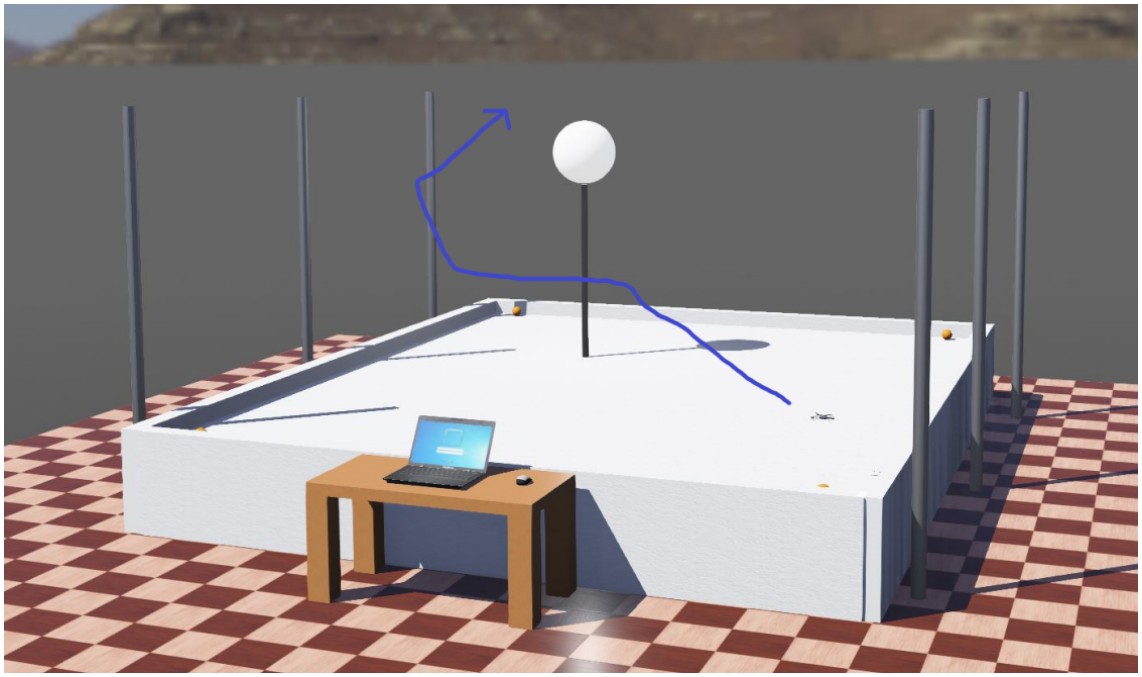


Figura 47: Trayectoria esperada.

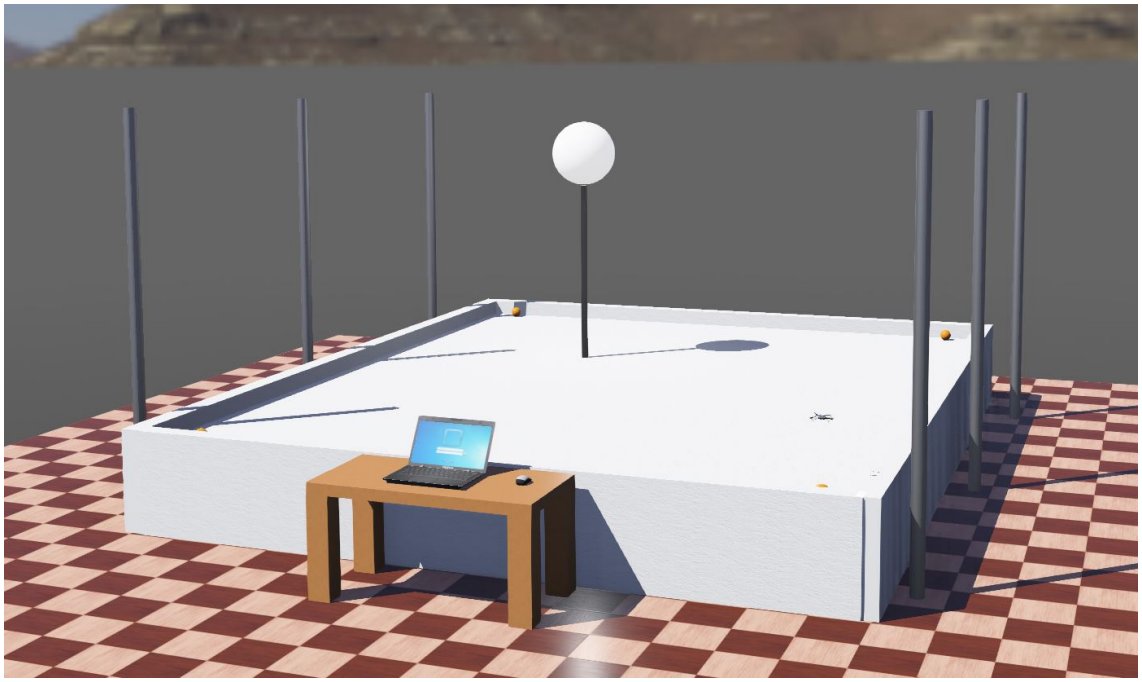


Figura 48: Dron *CrazyFlie2.0* en nodo 1.

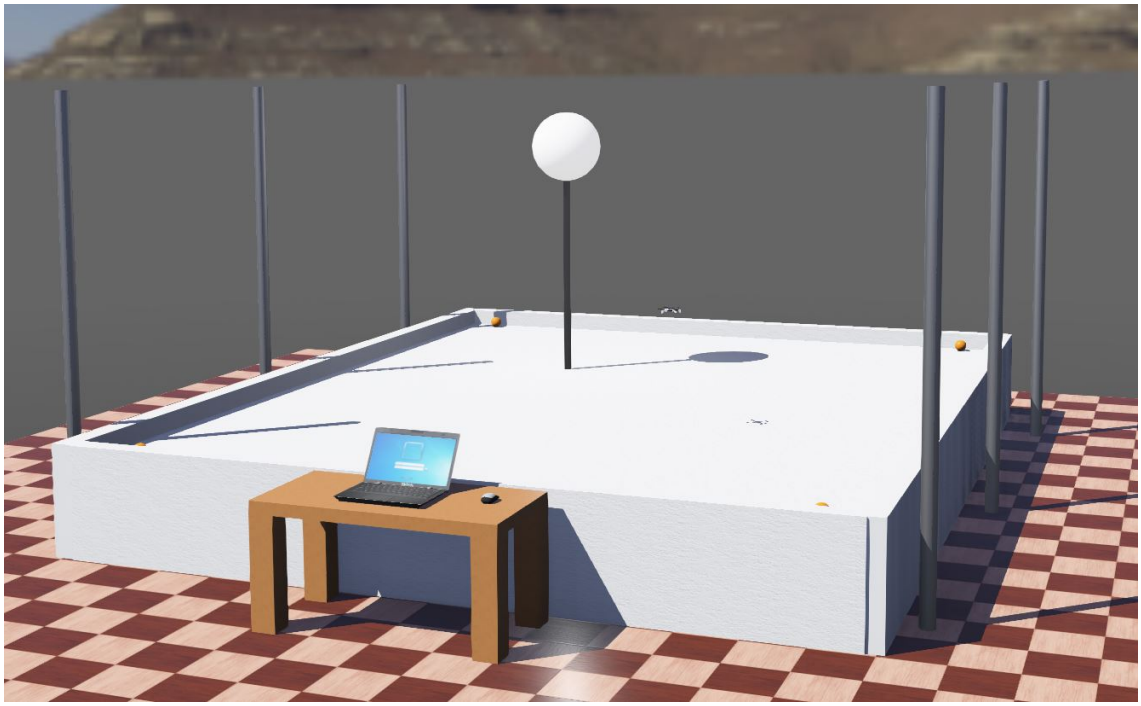


Figura 49: Dron *CrazyFlie2.0* en nodo 22.

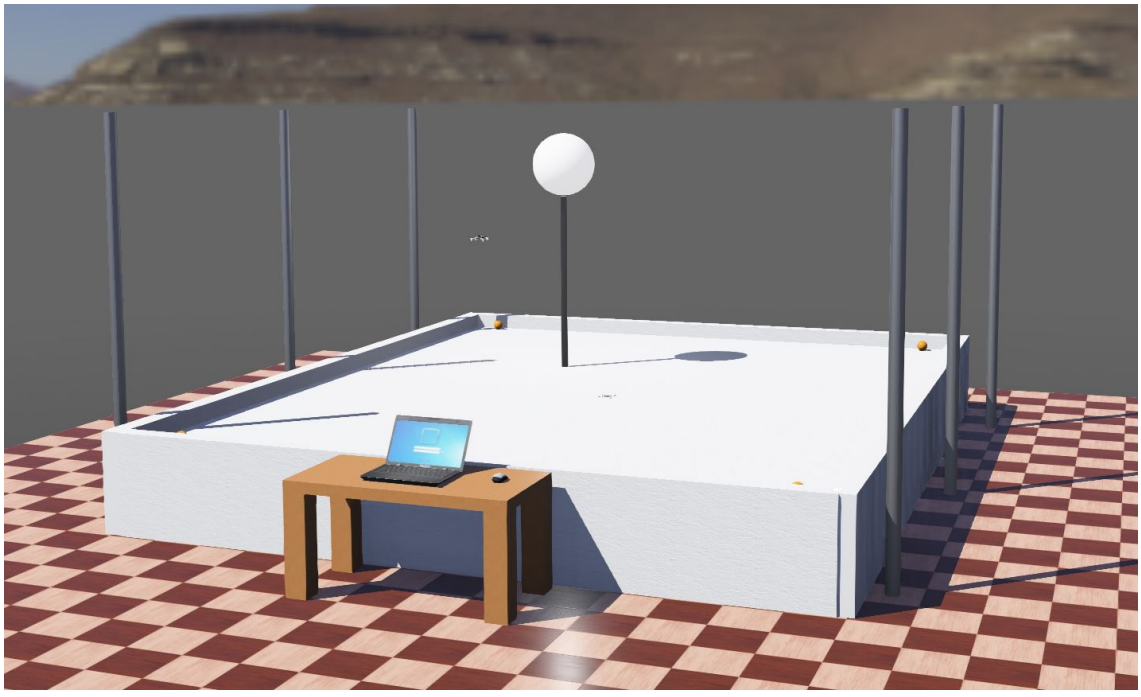


Figura 50: Dron *CrazyFlie2.0* en nodo 39.

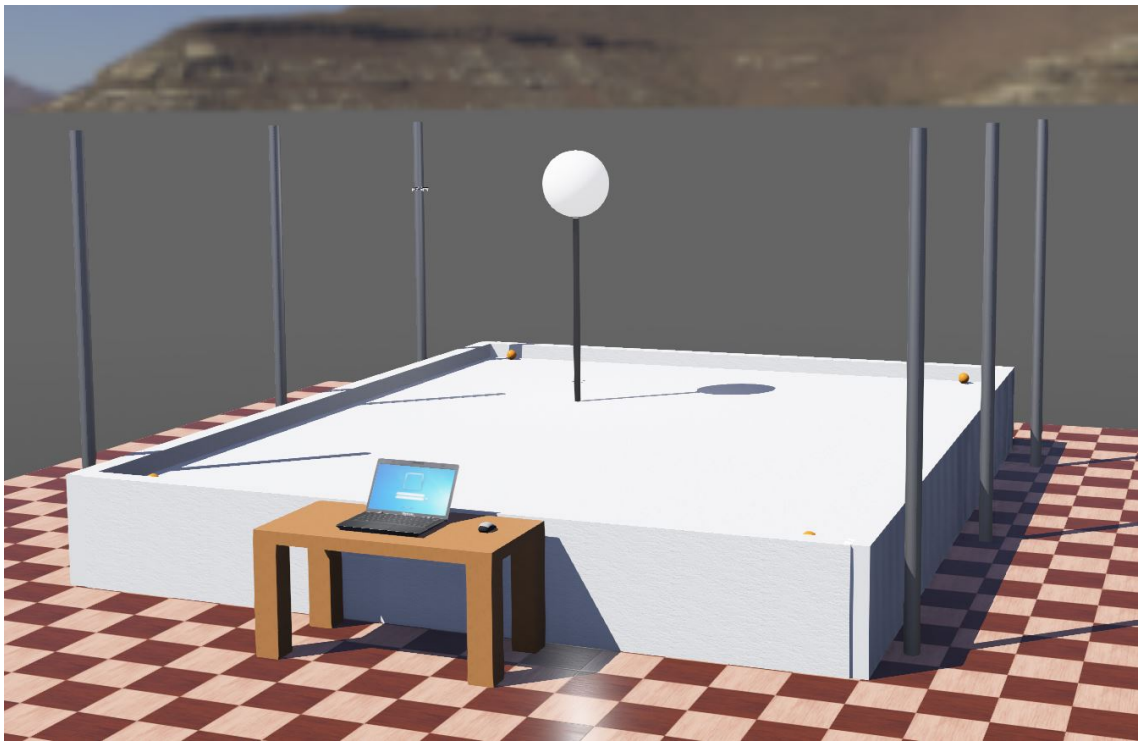


Figura 51: Dron *CrazyFlie2.0* en nodo 60.

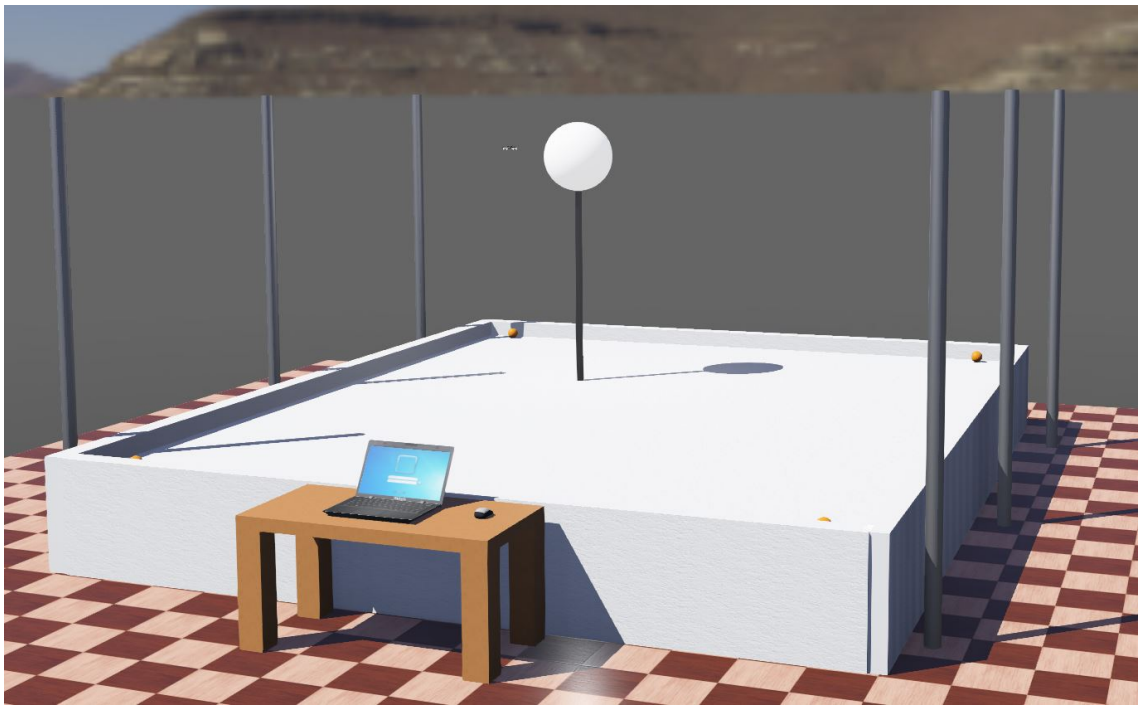


Figura 52: Dron *CrazyFlie2.0* en nodo 64.

10.2. Simulación de rutas en Webots con el dron *CrazyFlie2.0* para el algoritmo Particle Swarm Optimization

10.2.1. Ruta hasta punto $x = 4$, $y = 3$, $z = 1.65$ con lámpara como obstáculo

En la siguiente Figura 53 se puede apreciar la trayectoria óptima hasta la meta calculada por el algoritmo Particle Swarm Optimization. Esta trayectoria del punto de inicio hasta la meta al igual que el Ant Colony Optimization se trata de una trayectoria de punto a punto que el dron *CrazyFlie2.0* ira siguiendo. La lámpara de Webots se representa por medio de tres nodos verdes colocados en una configuración donde ocupen el espacio de la lámpara en el cálculo. En las Figuras 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65 se puede apreciar como el dron *CrazyFlie2.0* sigue la trayectoria de punto a punto hasta la meta.

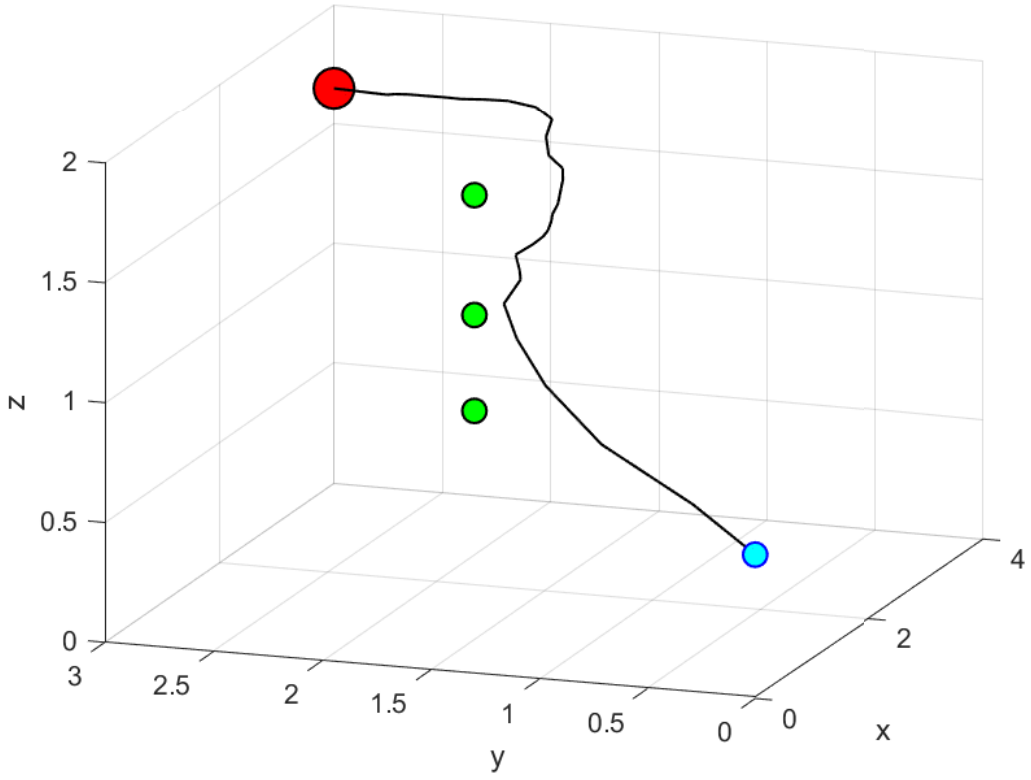


Figura 53: Trayectoria óptima hasta punto $x = 4$, $y = 3$, $z = 1.65$ con lámpara como obstáculo.

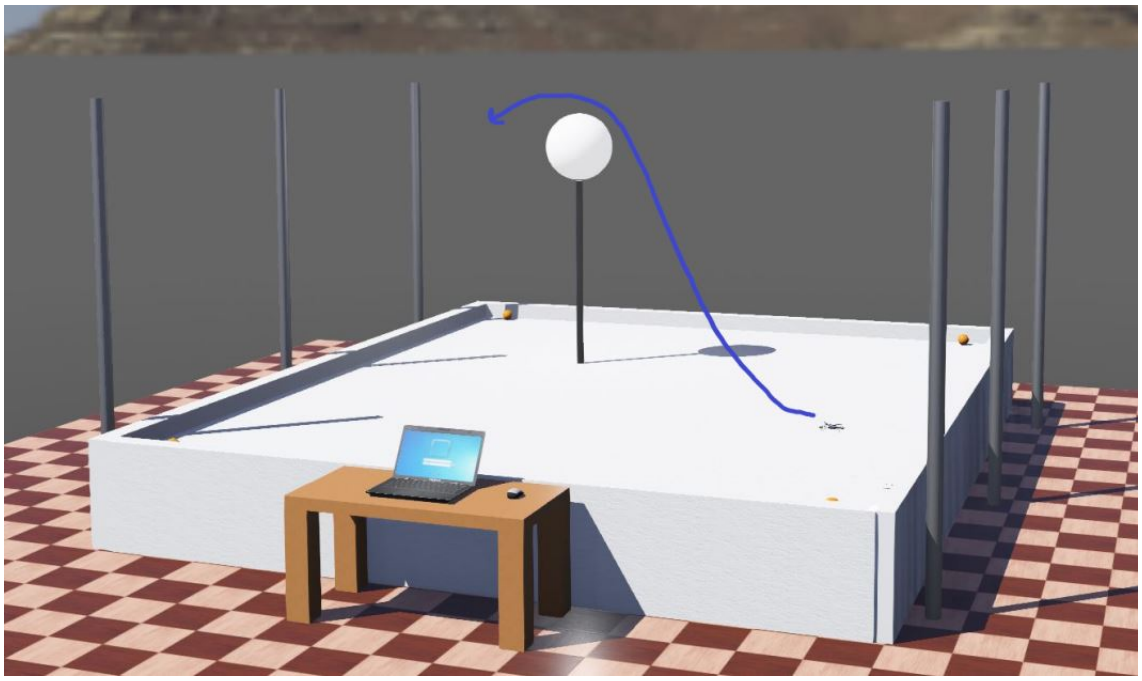


Figura 54: Trayectoria esperada.

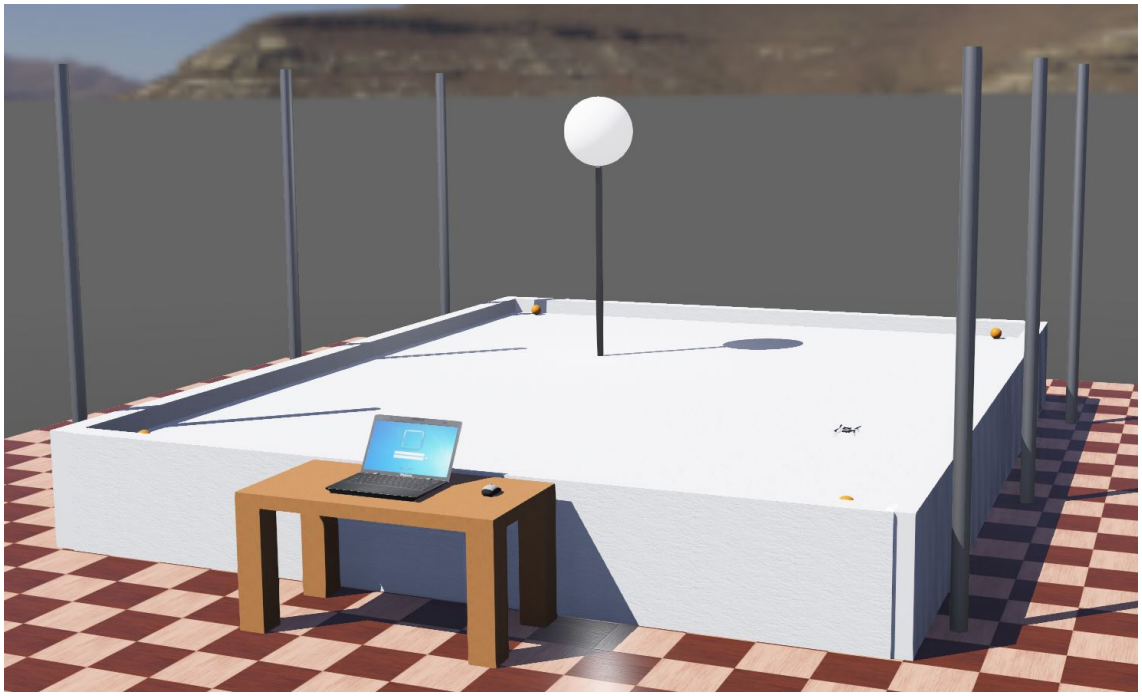


Figura 55: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

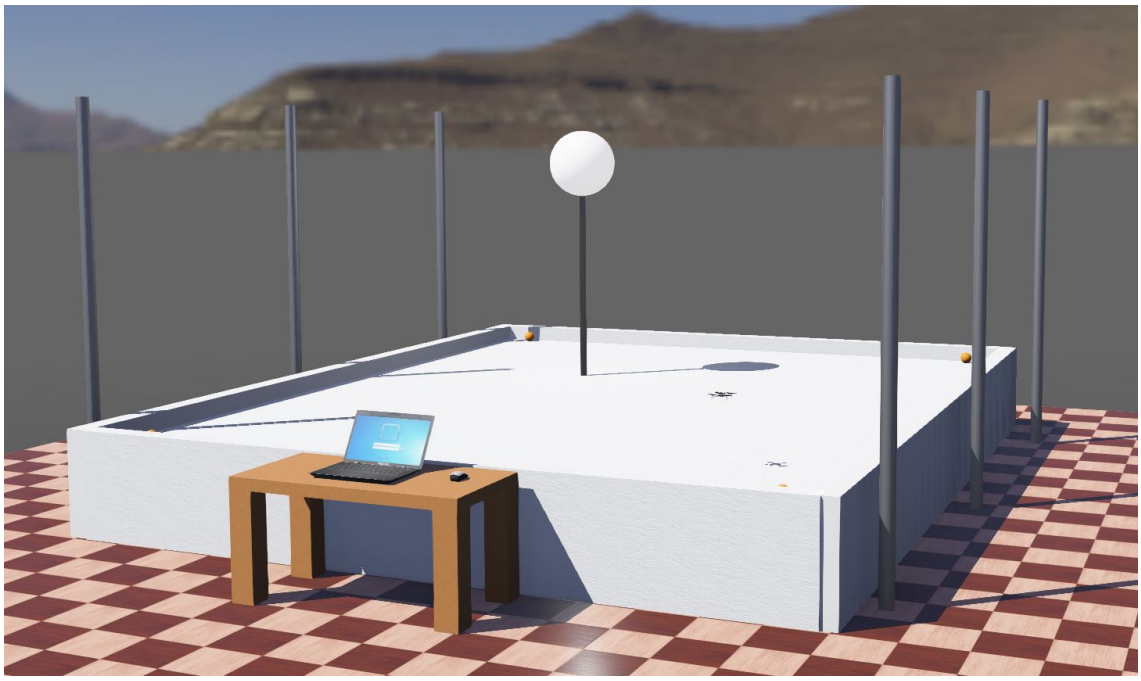


Figura 56: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

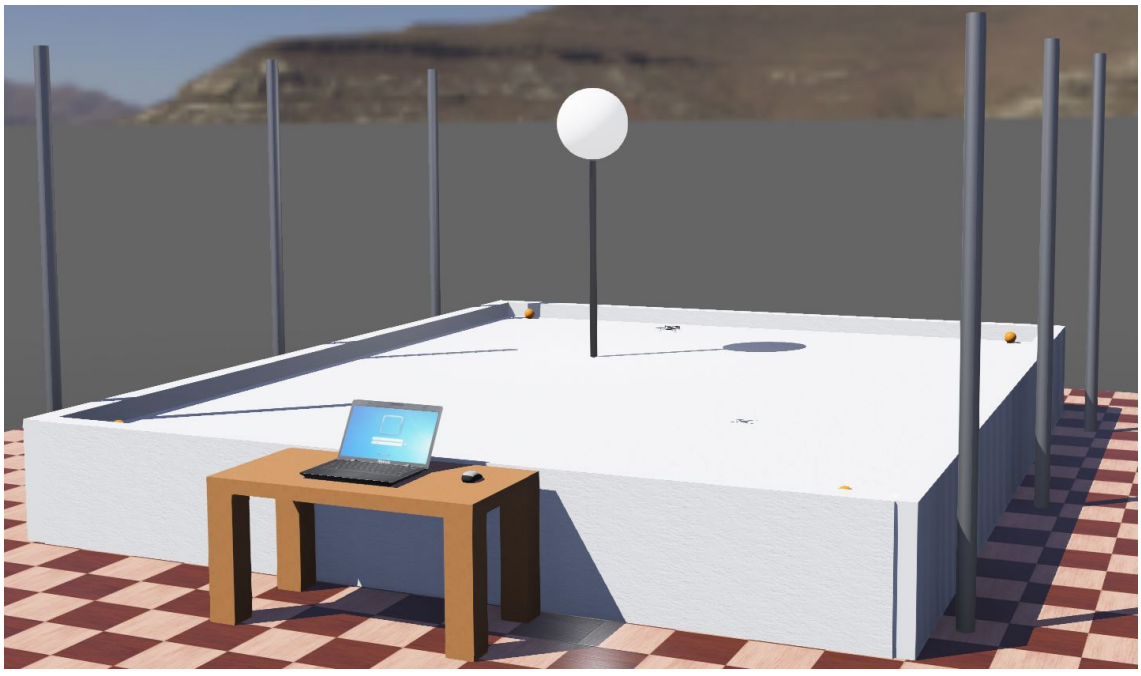


Figura 57: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

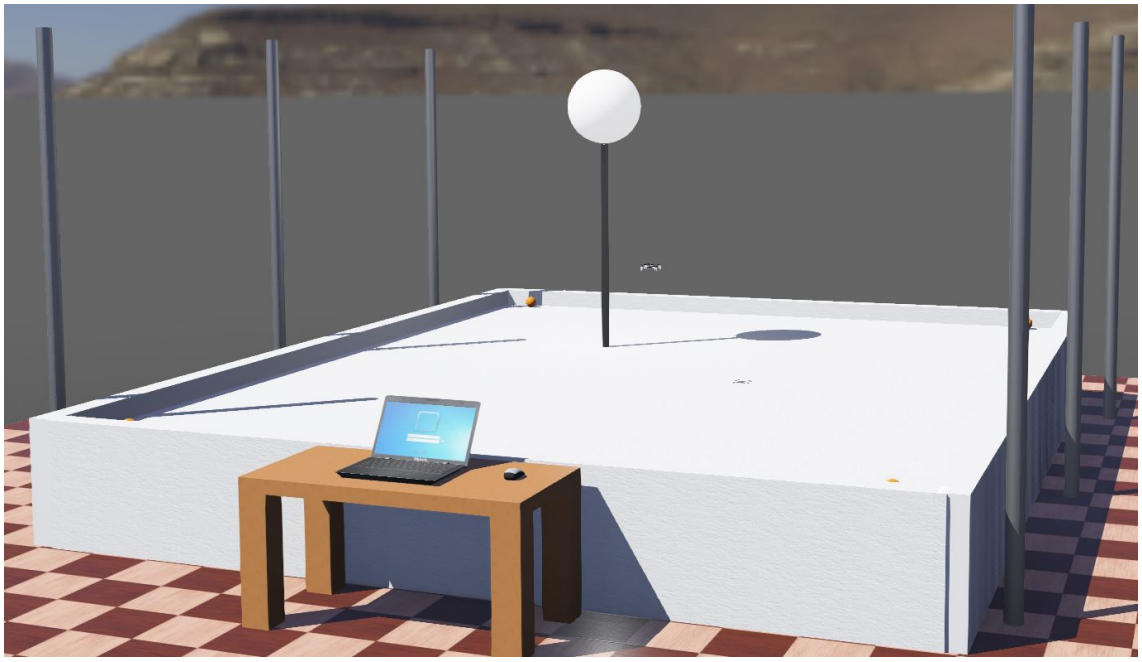


Figura 58: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

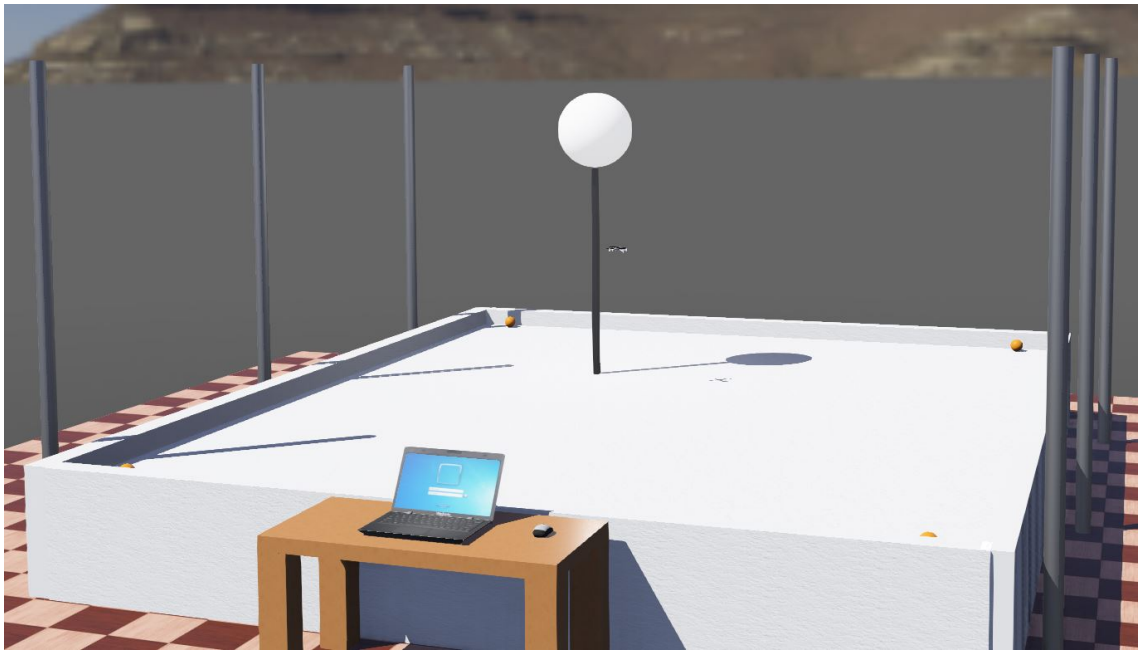


Figura 59: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

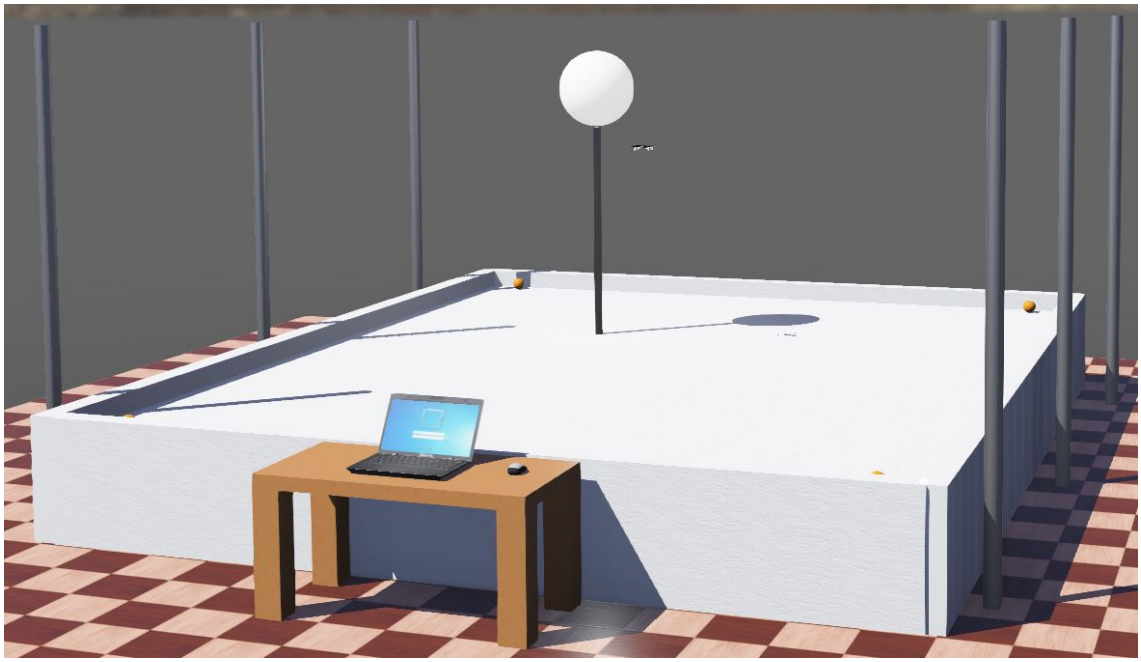


Figura 60: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

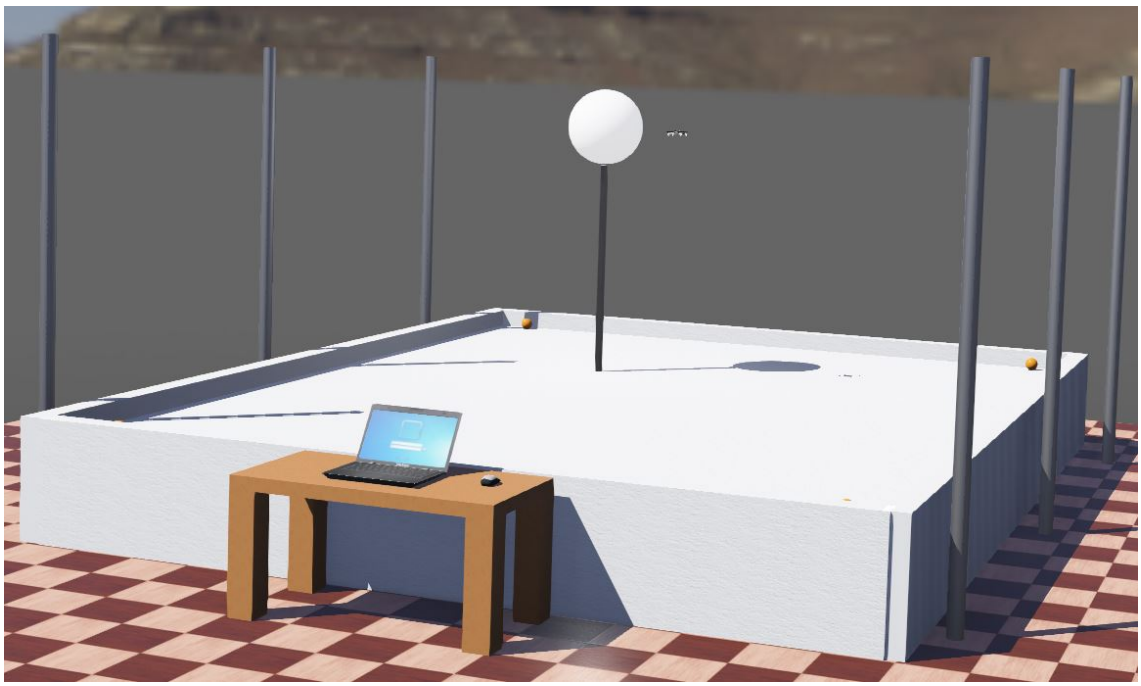


Figura 61: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

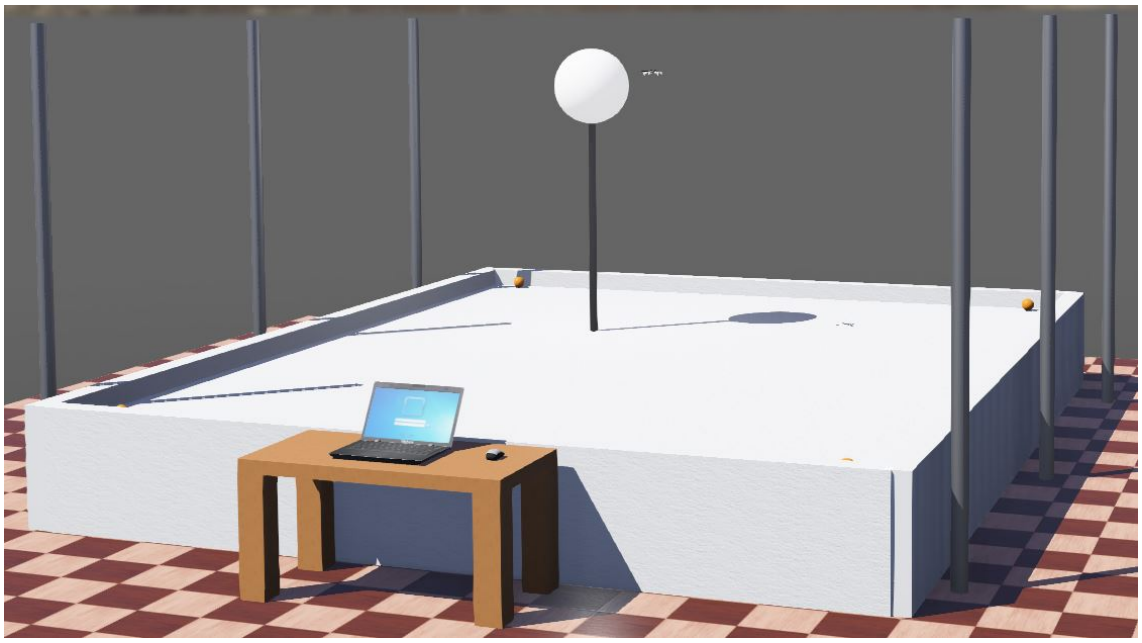


Figura 62: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

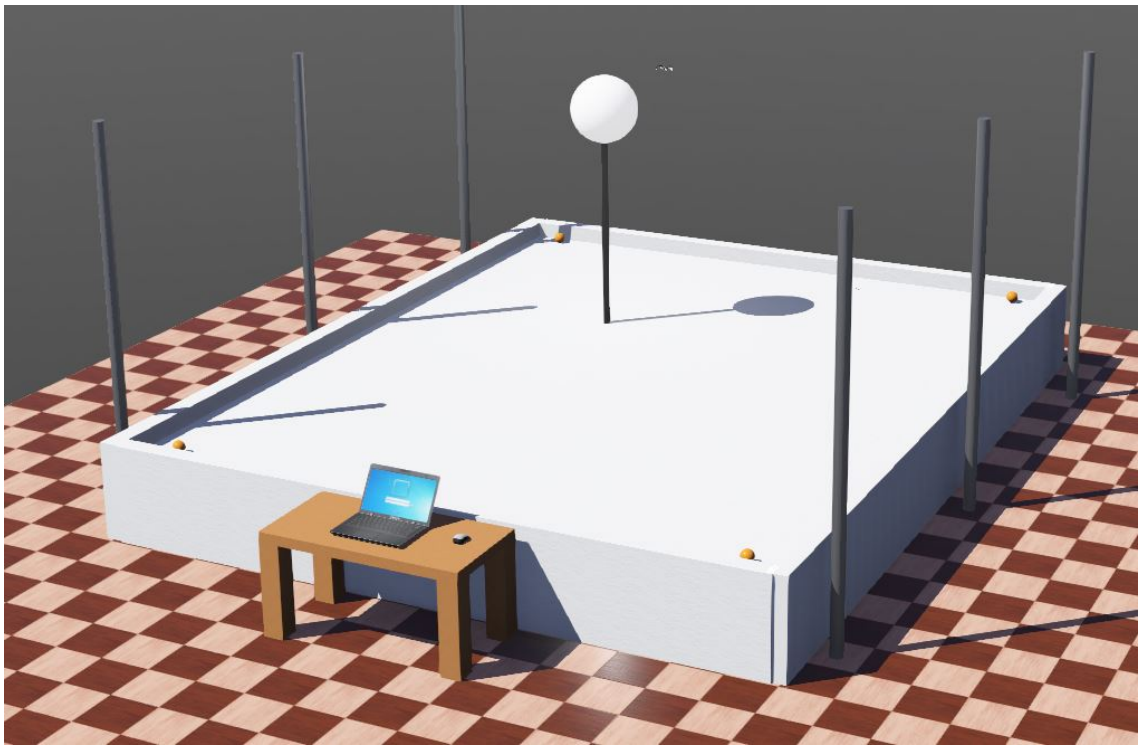


Figura 63: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

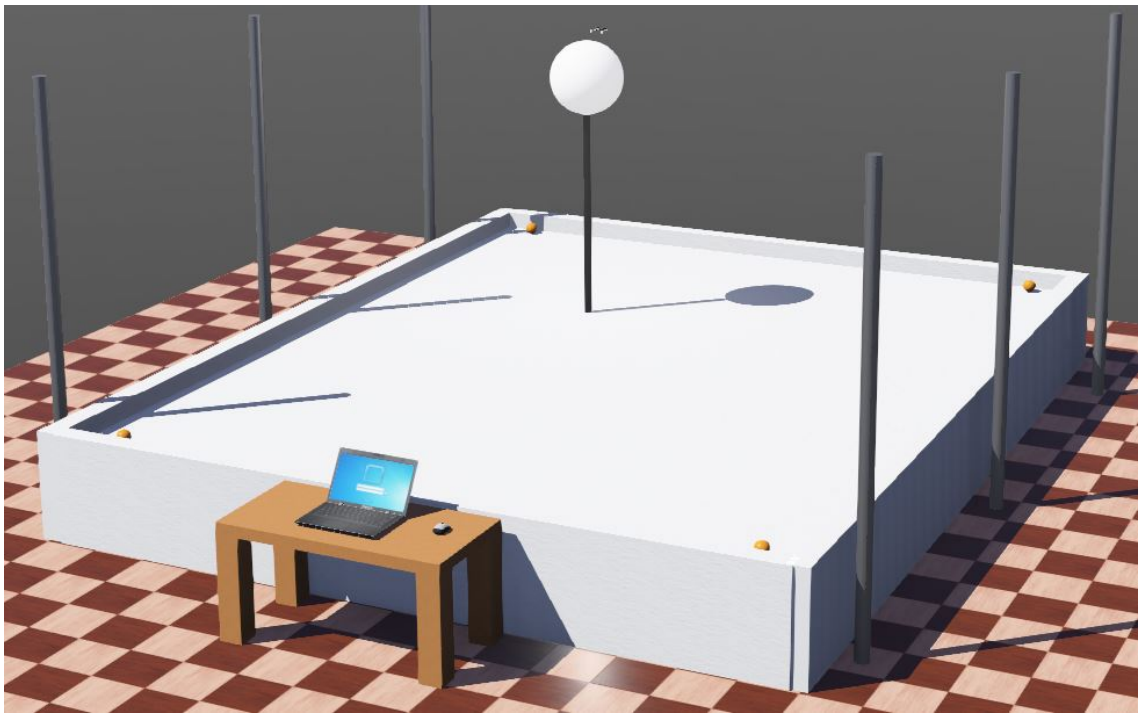


Figura 64: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

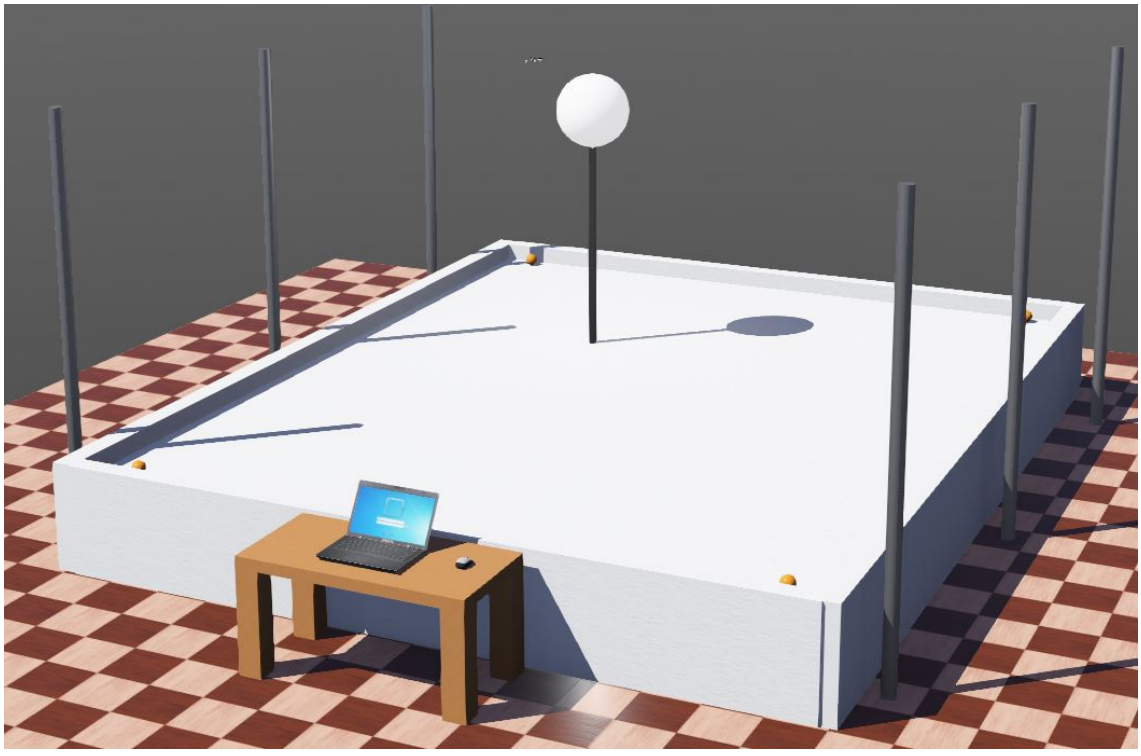


Figura 65: Dron *CrazyFlie2.0* desplazándose por la trayectoria calculada.

CAPÍTULO 11

Conclusiones

- Encontrar rutas óptimas para el dron *CrazyFlie2.0* es caro computacionalmente para el algoritmo Ant Colony Optimization.
- El algoritmo Particle Swarm Optimization muestra ser menos caro computacionalmente para encontrar rutas óptimas para el dron *CrazyFlie2.0*.
- El algoritmo Particle Swarm Optimization demostró ser un algoritmo más rápido que el Ant Colony Optimization.
- El algoritmo Ant Colony Optimization entrega menos puntos de trayectoria que el Particle Swarm Optimization.
- El algoritmo Particle Swarm Optimization entrega bastantes puntos de trayectoria debido a la complejidad de sus trayectorias.
- Los datos recibidos en Matlab del OptiTrack varían con respecto a los de Motive Body debido a la tolerancia de las cámaras.

CAPÍTULO 12

Recomendaciones

- Hacer un barrido de parámetros para los códigos de los algoritmos y comprobar con cuáles de estos convergen más rápido.
- Probar otros lenguajes de programación para verificar eficiencia en tiempo y recursos contra MATLAB.
- Usar la computadora de alto rendimiento del departamento para tener resultados más rápidos gracias a su alta potencia de procesamiento.
- Buscar optimizar el código del algoritmo para disminuir tiempos de cálculo y recursos utilizados.
- Buscar unir ambos algoritmos para mezclar sus fortalezas y compensar las debilidades de cada uno.
- Buscar algoritmos de optimización distintos a los propuestos en el trabajo de graduación para proponer alternativas al Ant Colony Optimization y al Particle Swarm Optimization.
- Buscar la posibilidad de utilizar el drone *CrazyFlie2.1* en vez del *CrazyFlie2.0* para aprovechar mejoras en el diseño.

CAPÍTULO 13

Bibliografía

[1] W. Sierra, "Implementación y Validación del Algoritmo de Robótica de Enjambre *Ant Colony Optimization* en Sistemas Físicos," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[2] J. Cahueque, "Implementación de enjambre de robots en operaciones de búsqueda y rescate," Tesis de licenciatura, Universidad del Valle de Guatemala, 2019.

[3] A. Aguilar, "Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[4] E. Santizo, "Aprendizaje Reforzado y Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre," Tesis de licenciatura, Universidad del Valle de Guatemala, 2020.

[5] A. Maas, "Implementación y Validación del Algoritmo de Robótica de Enjambre *Particle Swarm Optimization* en Sistemas Físicos," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[6] J. C. D. A. Juliá, "Dron de vuelo autónomo con reconocimiento basado en inteligencia artificial," Tesis de licenciatura, Universidad COMPLUTENSE Madrid, 2020.

[7] D. W. Mellinger, "Trajectory Generation and Control for Quadrotors," University of Pennsylvania, 2012.

[8] C. Alonzo, "Manufactura, modelado y control de un cuadricóptero con comunicación inalámbrica Wi-Fi integrado al ecosistema Robotat," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[9] H. Burmester, "Diseño de controlador de vuelo para cuadricóptero con capacidad de integrarse a ecosistema robotat vía inalámbrica Wi-Fi," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[10] C. Perafan, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[11] F. Sanabria, "Diseño y disposición de una plataforma de pruebas para sistemas de control para el dron Crazyflie 2.0," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[12] Bitcraze, "Sitio de informacion acerca del *Crazyflie2.0*," https://www.bitcraze.io/products/old-products/crazyflie-2-0/.

[13] J. A. Rodrigo, "Optimización con enjambre de partículas (Particle Swarm Optimization," 2019, Artículo de investigación.

[14] T. Gonzalez, *An Introduction to Ant Colony Optimization*. London: Taylor Francis group, 2008, Handbook of approximation Algorithms and Metaheuristics.

[15] M. Pedemonte, "Ant Colony Optimization," 2007, Artículo de investigación.

[16] D. Baldizón, "Aplicaciones Prácticas para Algoritmos de Inteligencia y Robótica de Enjambre," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[17] G. Iriarte, "Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," Tesis de licenciatura, Universidad del Valle de Guatemala, 2021.

[18] C. C. Gutiérrez, A. R. Domínguez y J. M. Salinas, *Cuádricas elípticas e hiperbólicas*. Madrid: Tebar, Geometría para ingenieros.

[19] Bitcraze. (2021), dirección: `https://www.cyberbotics.com/doc/guide/crazyflie?version=master&tab-os=linux&tab-language=python`.

CAPÍTULO 14

Anexos

14.1. Repositorio Github

Para obtener el codigo desarrollado para los algoritmos ingresar al siguiente link: https://github.com/CAVENDANO17192/Carlos-Avenda-o-Tesis-2022