UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Generación de trayectorias automáticas para el dron *CrayzFlie2.0* 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,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Generación de trayectorias automáticas para el dron *CrayzFlie2.0* 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,

2022

Vo.Bo.:

(f) _____

Ing. Estuardo Mancio

Tribunal Examinador:

(f) _____

Ing. Estuardo Mancio

(f) _____

MSc. Carlos Esquit

(f) _____

Ing. Luis Pedro Montenegro

Fecha de aprobación: Guatemala, 5 de diciembre de 2018.

Prefacio

El presente trabajo de graduacion requirio de multiples 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 mas interesantes y fomenten la inversión en proyectos de ciencia e ingeniería como este.

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 e irme guiando en el proceso de este complicado trabajo. De la misma manera algradezco a mi co-asesor de tesis Msc. Miguel Zea por orientarme tambien en dicho proceso con el dron *CrazyFlie2.0*

Índice

Prefacio v

Lista de figuras ix

Lista de cuadros xi

Resumen xiii

Abstract xv

1. Introducción 1

2. Antecedentes 3

3. Justificación 7

4. Objetivos 9

5. Alcance 11

6. Marco teórico 13

7. Algoritmo de planificación de trayectorias y evasión de obstáculos 21
- 7.1. Metodología . . . 21
- 7.2. Parametros . . . 24
- 7.3. Validación del algoritmo Ant Colony Optimization . . . 25
  - 7.3.1. Punto 1 a punto 64 . . . 25
  - 7.3.2. Punto 5 a punto 36 . . . 26
  - 7.3.3. Punto 3 a punto 62 . . . 26

8. Algoritmo de verificación de costos mínimos 27
- 8.1. Metodología . . . 27
- 8.2. Parametros . . . 28
- 8.3. Validación del algoritmo Particle Swarm Optimization . . . 28

8.3.1. minimo u = 10, v = 5, W = 15 28
8.3.2. mínimo u = 40, v = 10, W = 30 29
8.3.3. mínimo u = 1, v = 40, W = 40 30

9. Conclusiones **31**

10. Recomendaciones **33**

11. Bibliografía **35**

12. Anexos **37**
12.1. Repositorio Github 37

Lista de figuras

1. Dron *CrazyFlie2.0* [7] 14
2. Algoritmo de inteligencia computacional *Particle Swarm Optimization* [8] . 16
3. Algoritmo de inteligencia computacional *Ant Colony Optimization* [9] 18

4. Espacio tridimensional compuesto por grafos y aristas 22
5. Trayectoria generada por Ant Colony Optimization 25
6. Trayectoria generada por Ant Colony Optimization 26
7. Trayectoria generada por Ant Colony Optimization 26

8. Código Particle Swarm Optimization ejecutándose 28
9. Solución encontrada con PSO 28
10. Código Particle Swarm Optimization ejecutándose 29
11. Solución encontrada con PSO 29
12. Código Particle Swarm Optimization ejecutándose 30
13. Solución encontrada con PSO 30

Lista de cuadros

1. Parámetros de simulación 24

2. Parámetros de simulación 28

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 circuitos propuestos. 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 circuito. Con estos algoritmos se hizo simulaciones computarizadas para validar dichos algoritmos de inteligencia artificial modificados para nuestra aplicación.

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 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 were 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 real están basados en el comportamiento del reino animal. Esto buscando una solución para nosotros lo que seria para los animales su alimento, hogar o migración entre otros objetivos. En este caso se usaron algoritmos que usan inteligencia de enjambre como lo usan las abejas, hormigas, peces, termitas entre otros. Estos algoritmos necesitan computadoras de grandes recursos para no tardarse mucho en procesar una solución óptima debido a las largas iteraciones que se ejecutan. Los algoritmos también se pueden usar en agentes roboticos para toma de decisiones individuales que ayuden a otros agentes en un comportamiento de enjambre.

El Ant Colony Optimization (ACO) esta basado 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 mas usadas para este algoritmo se encuentran las redes neuronales, diseño de rutas para automoviles, procesamiento de señales entre otras.

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

El sistema OptiTrack es un ambiente para agentes roboticos o humanos donde a partir de pelotas reflectivas se pueden determinar posiciones (x,y,z) o angulos *yaw*, *pitch* y *roll* lo cual es ideal para agentes roboticos voladores como los drones. Existen varios tipos como los opticos y los no opticos los cuales usan acelerometros pero cuentan con desventajas. En la Universidad se cuenta con un Sistema de captura de movimiento OptiTrack el cual es optico debido al uso de camaras que capturan bolas reflectivas para determinar posiciones y angulos.

CAPÍTULO 2

Antecedentes

Antecedentes externos a UVG

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 [1]. 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 modulo 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.

Generación de trayectorias automáticas con condiciones iniciales extremas

En la tesis a cargo de Daniel Warren [2] 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 cúal es la trayectoria óptima.

Antecedentes internos UVG

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

En este trabajo de graduación a cargo de Carlos Alonzo [3] 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 ultima 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.

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 [4] se realizo 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.

Desarrollo e implementación del ecosistema Robotat y comunicación inalámbrica

En este trabajo de graduación a cargo de Camilo Perafan [5], 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 de realizo 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 ultimo 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.

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 [6] cuenta con una maquina 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 mas 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* están basados 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 controlado por el controlador moderno LQR, en control clásico un controlador PID no puede controlar tantas variables y condiciones como el controlador 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 planea usar algoritmos de inteligencia computacional para generar trayectorias las cuales el dron debe de seguir para completar un circuito.

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

Objetivo General

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

Objetivos Específicos

- Utilizar los algoritmos de inteligencia computacional para generación automática de trayectorias.
- Determinar las posiciones del dron y de los obstaculos 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 abarco la implementación simulada y real para algoritmos de inteligencia computacional y robótica de enjambre, Particle Swarm Optimization y Ant Colony Optimization. Para esto se tomo 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 pensar mas al algoritmo sobre rutas para evitarlo.

Para la optimización con el Particle Swarm Optimization toma una ecuación de evaluación la cual dependiendo la posición de la partícula devuelve valores en tres dimensiones que se ingresan en la ecuación esperando el mínimo de la función.

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 obstaculos.

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

Características del dron *CrazyFlie2.0*

El dron *CrazyFlie2.0* [6] 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 esta 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* [7]

Particle Swarm Optimization

La optimización por enjambre de partículas [8] 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.

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.

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.

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.

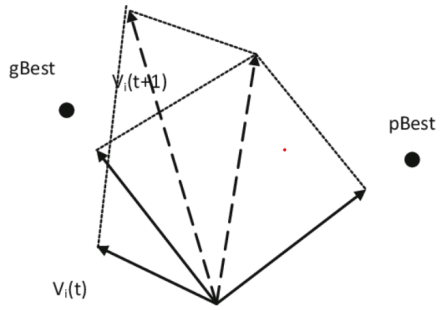


Figura 2: Algoritmo de inteligencia computacional *Particle Swarm Optimization* [8]

Ant Colony Optimization

En el año 1991 Marco Dorigo [9] [10] 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 menciono 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.

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 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.

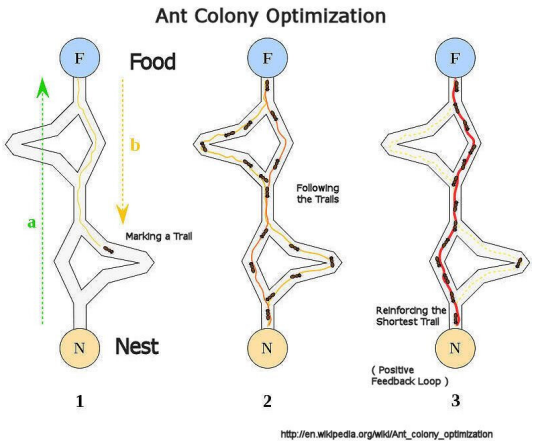


Figura 3: Algoritmo de inteligencia computacional *Ant Colony Optimization* [9]

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.

Sistema de captura de movimiento OptiTrack

El OptiTrack [5] 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 de planificación de trayectorias y evasión de obstáculos

En este capítulo se habla sobre la metodología empleada para el algoritmo de planificación de trayectorias y evasión de obstáculos. Detallando parámetros y ejemplos hechos para diferentes puntos.

Se utilizo 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. Metodología

Para el Ant Colony Optimization Se tomo de base el algoritmo de la fase anterior [11] 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 4x4x4. Para simular los obstáculos se hicieron planos a partir de 3 o 4 grafos o figuras solidas que usan varios grafos para formarlas. En la siguiente figura se puede apreciar el mapa tridimensional con todas las trayectorias posibles o aristas entre nodos.

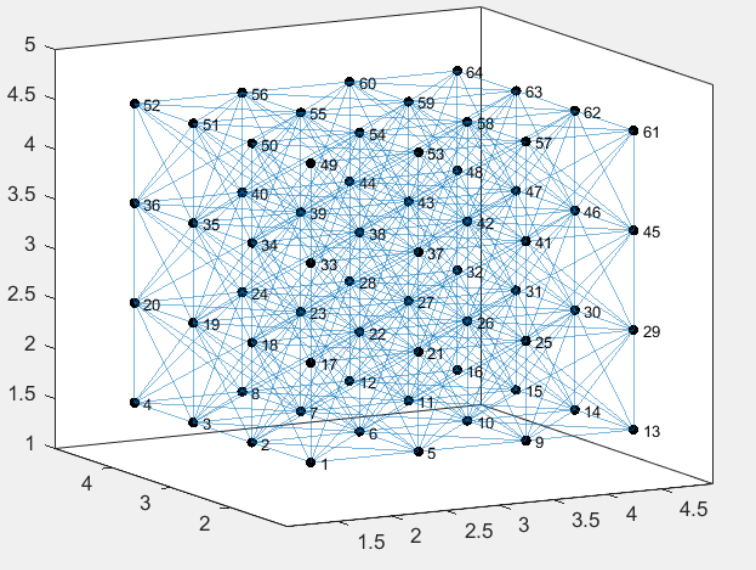
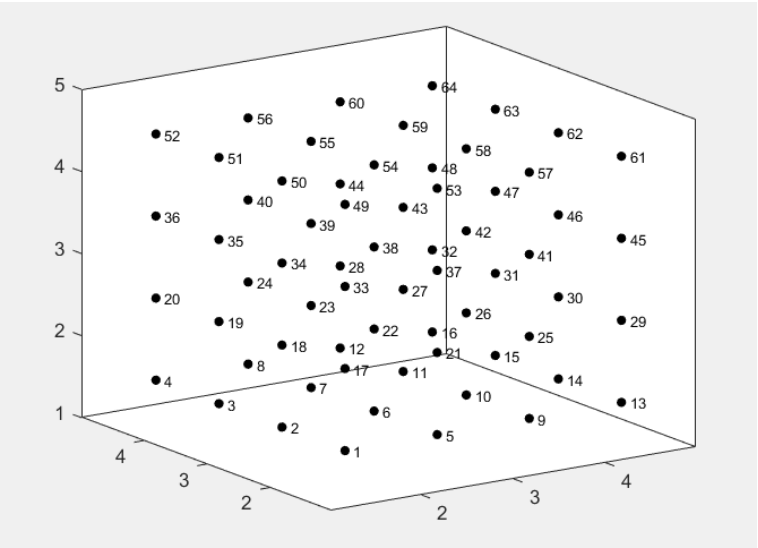


Figura 4: Espacio tridimensional compuesto por grafos y aristas

El plano tridimensional se define de la siguiente manera. Primero se establece los limites donde empezarían los nodos en cada eje que normalmente se establece en 1 y el final que seria el tamaño del grid al tratarse de un cubo perfecto.

```
%grid = 4

x_lim_pos = grid;
y_lim_pos = grid;
z_lim_pos = grid;

x_lim_neg = 1;
y_lim_neg = 1;
z_lim_neg = 1;

grid_x = x_lim_pos - x_lim_neg + 1;
grid_y = y_lim_pos - y_lim_neg + 1;
grid_z = z_lim_pos - z_lim_neg + 1;

n = grid_x * grid_y * grid_z;


[X1, Y1, Z1] = meshgrid(1:grid_x, 1:grid_y, 1:grid_z);

Coords = [reshape(X1, [n, 1]),reshape(Y1, [n, 1]),  reshape(Z1, [n, 1])];
```

Ya teniendo estos valores se tienen definidos las coordenadas en cada eje para los grafos. Con esto ya se puede crear un grafo en MATLAB agregando una tabla con estas posiciones con el nombre de los nodos en un vector columna y otra tabla con la matriz peso, feromona y EndNodes, que seria la matriz que define las aristas. Después se puede desplegar en una ventana donde se puede apreciar las diferentes posiciones en el plano tridimensional sin aristas.

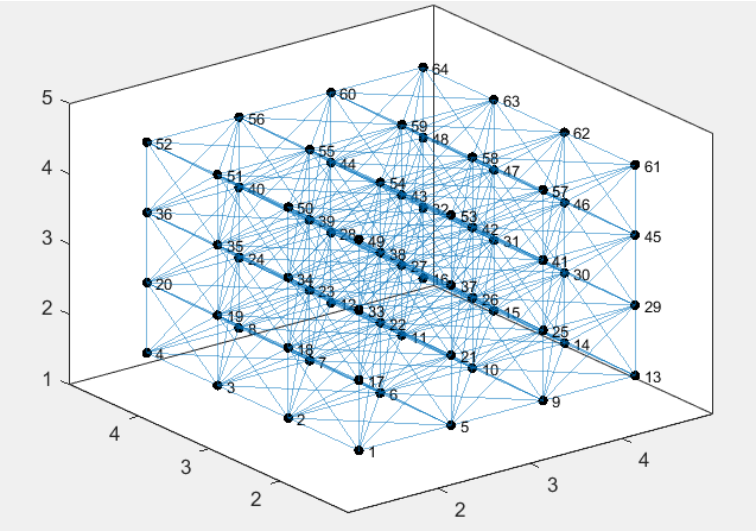


Seguido de esto se crearon las funciones correspondientes para definir las aristas entre los grafos siempre que estas queden entre los subcubos y no hagan saltos muy pronunciados.

```
for nivel = 0:16:48
for columna = 0:4:12
for fila  = 1:1:3

    res1 = fila+columna+nivel
    res2 = res1 +1
    EndNodes= [EndNodes; res1,res2]
    Weight=[Weight;1];

end
end
end

%eje 2
for nivel = 0:16:48
for columna = 1:1:4
for fila = 0:4:8

    res1 = fila+columna+nivel
    res2 = res1 + 4
    EndNodes = [EndNodes; res1, res2]
    Weight=[Weight;1];
end
end
end
```

Con estas aristas definidas se puede ir definiendo la matriz EndNodes, la cual define de que nodo a que nodo se define una arista. El vector Peso también se define el cual seria la distancia que existe entre los grafos y el vector feromona siempre se inicializa en uno.

Esto se crea para diferentes aristas tanto horizontales, verticales y diagonales teniendo como resultado un entorno tridimensional totalmente definido.

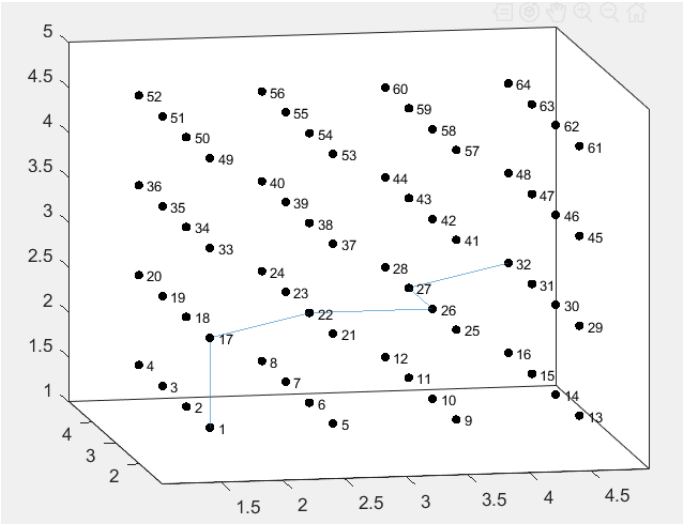


7.2. Parametros

$PARAMETROS$	$VALOR$
ρ	0.35
α	1
β	1
γ	1
Q	0.7
$Hormigas$	250

Cuadro 1: Esta tabla muestra los valores ingresados en el algoritmo para su correcto funcionamiento dando resultados aceptables en encontrar rutas óptimas.

Al definir estos parámetros y correr el algoritmo se define el punto de partida y el punto de meta para empezar a buscar trayectorias hasta encontrar la ruta óptima donde el algoritmo devuelve una imagen donde se visualiza esta trayectoria, para el ejemplo de una ruta del punto 1 al punto 32.



7.3. Validación del algoritmo Ant Colony Optimization

A continuación se muestran resultados del algoritmo generando trayectorias para diferentes puntos de meta a partir de un punto de partida '1'. Esto se elaboro a partir de los parámetros elegidos a partir del cuadro uno.

7.3.1. Punto 1 a punto 64

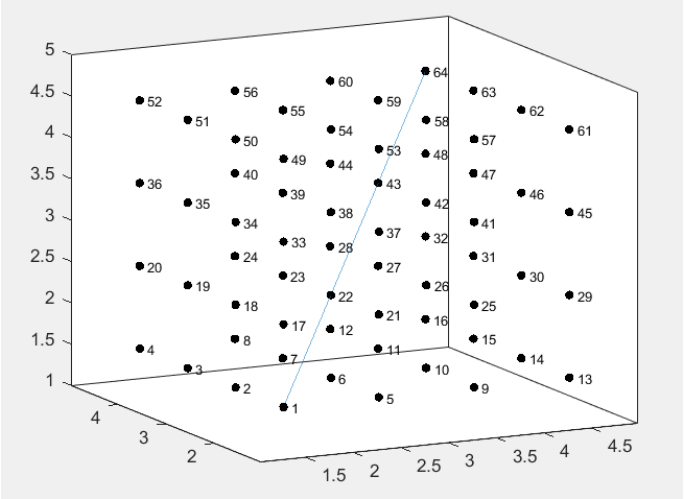


Figura 5: Trayectoria generada por Ant Colony Optimization

7.3.2. Punto 5 a punto 36

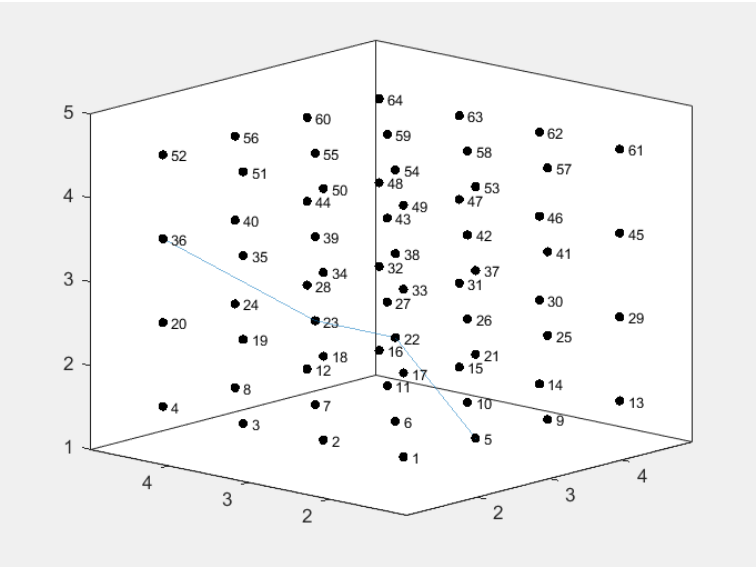


Figura 6: Trayectoria generada por Ant Colony Optimization

7.3.3. Punto 3 a punto 62

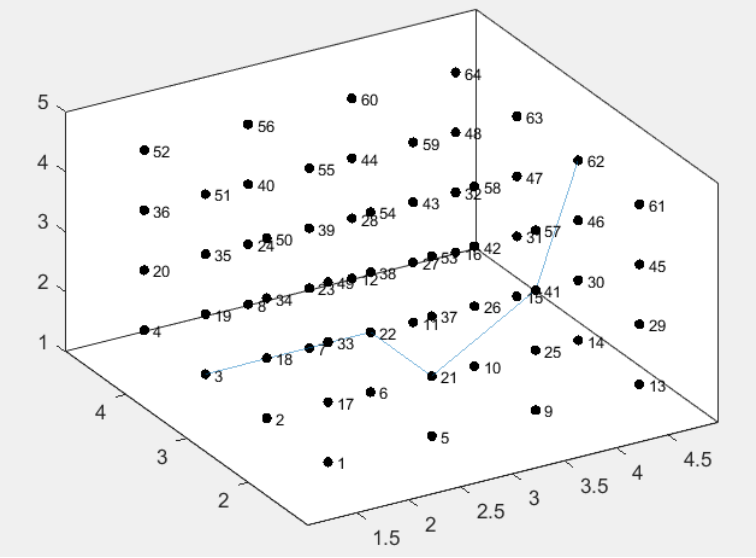


Figura 7: Trayectoria generada por Ant Colony Optimization

CAPÍTULO 8

Algoritmo de verificación de costos mínimos

En este capitulo se habla acerca del uso del 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 donde se quiere que llegue a converger el dron sobre un punto solución de mínimos.

8.1. Metodología

Se definen parámetros iniciales del algoritmo donde cual es la inercia de las partículas, el factor de corrección, la cantidad de iteraciones y el numero de partículas.

Se define la ecuación a la cual se le encontrara los mínimos, para este caso se hará un paraboloide elíptico. Esto con el fin de poder definir el mínima como el punto donde se quiere que llegue el dron como meta y donde eventualmente todas las partículas convergerán.

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.

8.2. Parametros

$PARMETROS$	$VALOR$
$iteraciones$	100
$inercia$	1.0
$factor - correccin$	2.0
$partculas$	20

Cuadro 2: Esta tabla muestra los valores ingresados en el algoritmo para su correcto funcionamiento dando resultados aceptables en encontrar la solución de la ecuación de costo.

8.3. Validación del algoritmo Particle Swarm Optimization

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

8.3.1. minimo u = 10, v = 5, W = 15

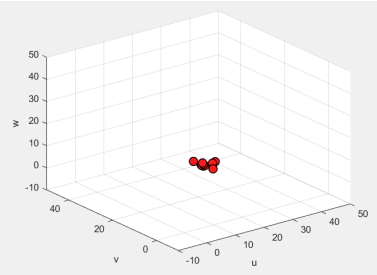


Figura 8: Código Particle Swarm Optimization ejecutándose

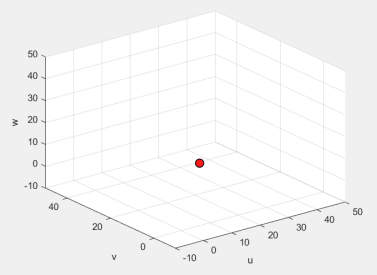


Figura 9: Solución encontrada con PSO

8.3.2. mínimo u = 40, v = 10, W = 30

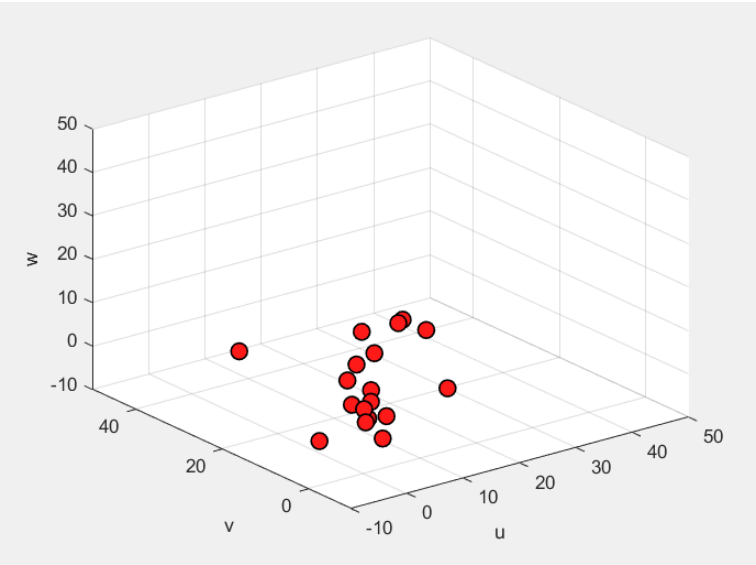


Figura 10: Código Particle Swarm Optimization ejecutándose

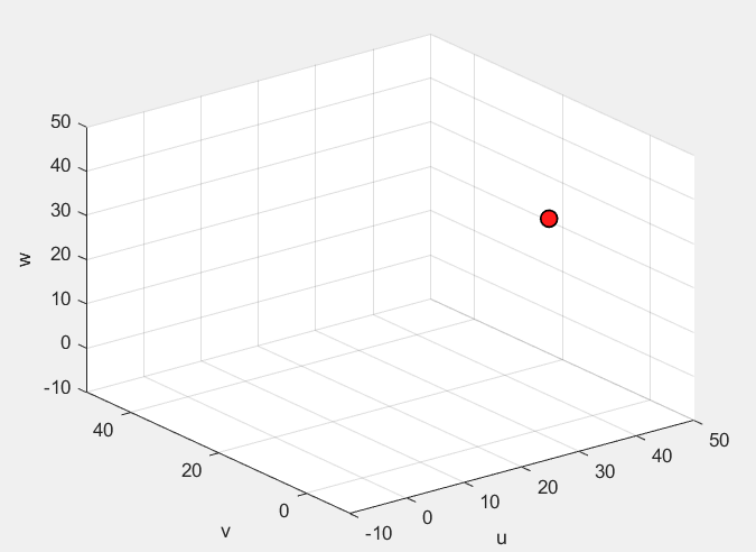


Figura 11: Solución encontrada con PSO

8.3.3. mínimo u = 1, v = 40, W = 40

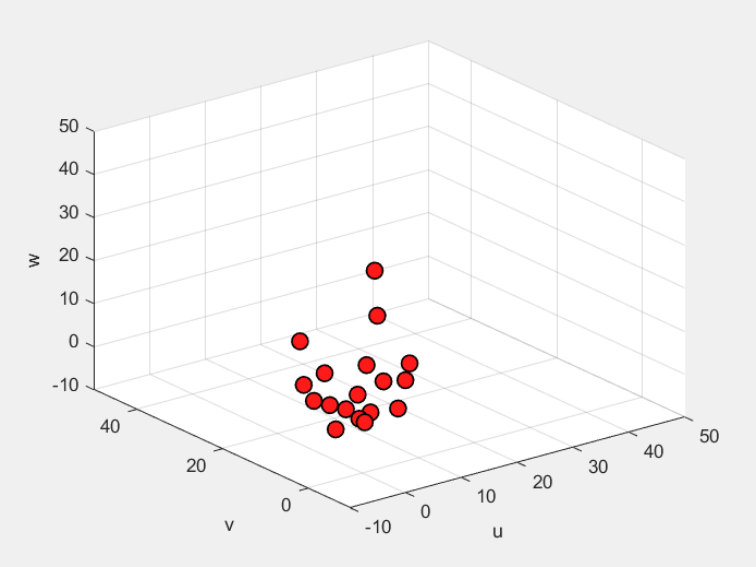


Figura 12: Código Particle Swarm Optimization ejecutándose

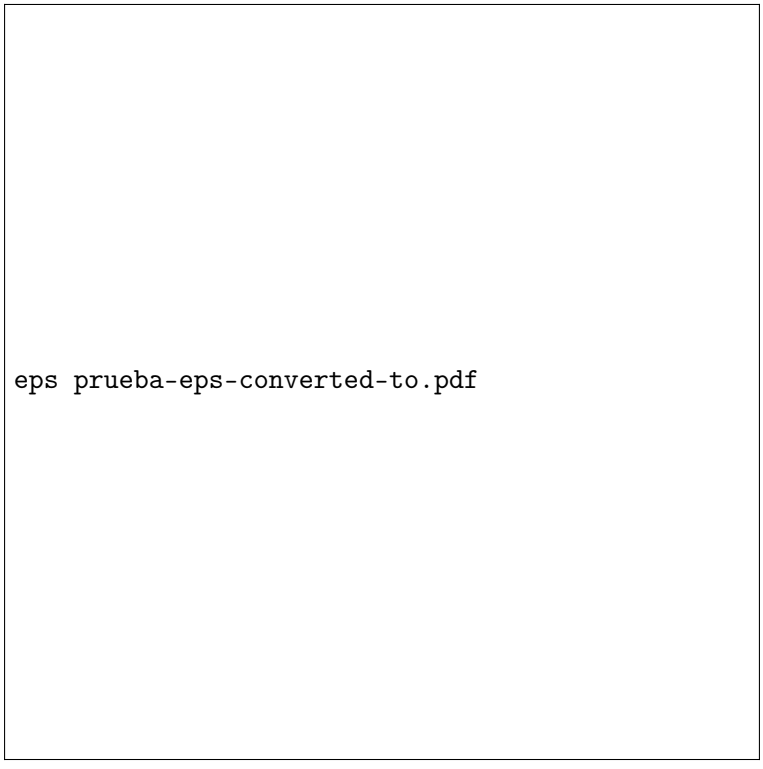


Figura 13: Solución encontrada con PSO

CAPÍTULO 9

Conclusiones

- Es posible cambiar los parámetros de los algoritmos para ajustarlos manualmente.
- Encontrar rutas optimas para el dron *CrazyFlie2.0* es caro computacionalmente.
- El algoritmo Particle Swarm Optimization mostro ser un algoritmo mas rápido que el Ant Colony Optimization.
- Se debe de acondicionar las dimensiones del sistema robotat con las de los algoritmos para encajar los sistemas.
- El OptiTrack funciona mejor en horas de la mañana-tarde.
- El dron *CrazyFlie2.0* sigue las trayectorias satisfactoriamente en complemente con el proyecto de Steve Chex.

CAPÍTULO 10

Recomendaciones

- Probar los algoritmos en una computadora con bastantes recursos para tener resultados rápidos
- Buscar que el dron *CrazyFlie2.0* tenga una batería no muy pesada de alimentación.
- El sistema OptiTrack no debe de usarse muy noche para tener mejores resultados
- Evitar la presencia de muchas personas en el aula para evitar accidentes cuando el dron se encuentre volando.
- Tener bastantes repuestos de hélices para el dron en caso de romper algunas en pruebas.
- probar otros lenguajes de programación para verificar eficiencia contra MATLAB.

CAPÍTULO 11

Bibliografía

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

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

[3] 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.

[4] 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.

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

[6] 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.

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

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

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

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

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

CAPÍTULO 12

Anexos

12.1. Repositorio Github

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