UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Validación de los algoritmos de robótica de enjambre *Particle Swarm Optimization* y *Ant Colony Optimization* con sistemas robóticos físicos en el ecosistema Robotat

Protocolo de trabajo de graduación presentado por Jonathan Menéndez Cardona, estudiante de Ingeniería Mecatrónica

Guatemala,

2023

Resumen

El siguiente protocolo presenta una propuesta de trabajo de graduación enfocado en la robótica de enjambre, en especial en las validación de los algoritmos de Particle Swarm Optimization (PSO) y Ant Colony Optimization (ACO) con robots móviles. Estos algoritmos fueron desarrollados y probados a nivel de simulación durante fases anteriores.

Para poder realizar pruebas en físico se deberá llevar a cabo la migración de los algoritmos para usar los robots móviles Pololu 3pi+ en la plataforma de Robotat. Esta migración permitirá la implementación de los algoritmos en un entorno real, lo que a su vez facilitará la comparación de los resultados obtenidos con las simulaciones previas. La comparación entre los resultados de las pruebas físicas y los resultados simulados será de gran importancia para validar y verificar los avances logrados en el desarrollo de los algoritmos.

Además, durante este proceso se llevarán a cabo nuevas pruebas y se establecerá un criterio de comparación que permitirá realizar un análisis de los algoritmos. Se examinará en qué casos cada algoritmo muestra un mejor desempeño, y se identificarán los parámetros que ofrecen un funcionamiento óptimo en diferentes situaciones. Este análisis y comparación de los algoritmos en un entorno físico proporcionará una comprensión más profunda sobre su rendimiento, lo que permitirá realizar ajustes y mejoras para optimizar su funcionamiento en robots móviles reales.

Antecedentes

Georgia Tech: Robotarium

El Instituto Tecnológico de Georgia desarrolló un laboratorio que permite la investigación y pruebas de robótica de forma remota [1]. Este proyecto provee una plataforma de libre acceso para realizar la verificación de algoritmos y trayectorias en la robótica de enjambre con robots reales, en lugar de quedarse puramente en simulaciones. Para poder realizar los experimentos deseados en los robots del Robotarium es necesario descargar un simulador, ya sea en Matlab o en Python, y verificar que el código a nivel de simulación en forma de un prototipo. Luego, se debe registrar con una cuenta en la página de Robotarium y esperar a que uno de los administradores apruebe la solicitud para poder acceder a la interfaz web, donde se podrá subir el código para su ejecución.

Avances en investigación de robótica de enjambres

Desde que el área de estudio de inteligencia de enjambre atrajo interés ha ido evolucionando hasta ser un proceso interdisciplinario, incluyendo inteligencia artificial, economía, sociología, biología, etc. La robótica de enjambre cuenta con características favorables al compararlos con otros sistemas, como que su control que puede ser descentralizado y autónomo, que es altamente flexible, que tiene una alta escalabilidad para seguir ejecutando acciones sin importar un cambio en la cantidad de agentes, entre otros. La robótica de enjambre puede ser aplicada en problemas que de otra forma sería difíciles de abordar, típi-

camente en la ayuda ante desastres naturales que impidan la intervención humana de forma directa, en la minería y hasta en el control de vehículos aéreos no tripulados [2].

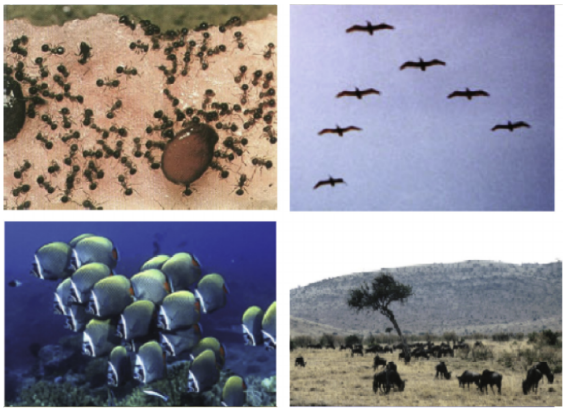

Figura 1: Enjambres en la naturaleza que han inspirado la robótica de enjambre [2].

Megaproyecto Robotat

El departamento de Ingeniería Electrónica, Mecatrónica y Biomédica de la Universidad del Valle de Guatemala realizó un proyecto, desarrollado en varias fases, para promover el aprendizaje y realizar pruebas físicas de robótica de enjambre. La primera fase constó del diseño y posterior elaboración de una plataforma que implementa un algoritmo de visión de computadora para obtener datos de robots en una superficie plana que permitiera la comprobación futura de los algoritmos de enjambre.

Para tener una retroalimentación actual de cada agente dentro de la mesa de pruebas se utiliza el sistema de captura de movimiento OptiTrack. El software que se necesita para trabajar con OptiTrack es Motive, que nos permite obtener las coordenadas y orientación de unos marcadores que se colocan en cada uno de los agentes. Todo el sistema funciona por medio de un servidor elaborado por el Msc. Miguel Zea, que permite tanto la obtención de las coordenadas proporcionadas por las cámaras como el enviar coordenadas a la dirección IP que haga la petición de las mismas [3].

Particle Swarm Optimization (PSO)

En la segunda fase del Megaproyecto Robotat, Aldo Aguilar [4] implementó el algoritmo PSO clásico con modificaciones en una serie de parámetros que permita un mejor comportamiento del enjambre y que se adapta a las dimensiones físicas y cinemáticas de robots Bibots. El algoritmo incluye un planeador de trayectorias que ajusta el parámetro de inercia, el parámetro de constricción, dos parámetros de escalamiento y dos parámetros de uniformidad, optimizados para robots diferenciales.

Durante la investigación se pudo observar que era necesario el uso de controladores para poder seguir las trayectorias computadas por el planeador, generando velocidades suaves y continuas. Se realizó la evaluación de hasta siete diferentes controladores, por medio de

simulaciones en el software Webots, donde se obtuvo que el controlador Transformación de Uniciclo con LQI (TUC-LQI) brinda los mejores resultados en cuanto a la genera trayectorias, manteniendo sin cambio la velocidad pico debido a un amortiguador de control proporcional y evitando oscilaciones gracias a un amortiguador de control integral [5].

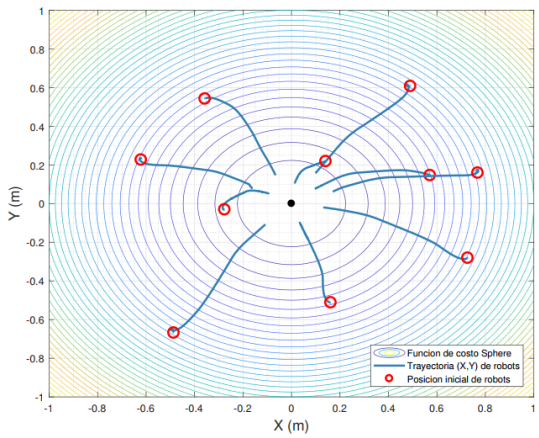

Figura 2: Trayectorias de enjambre en simulación con controlador TUC-LQI [5].

En la fase posterior, llevada a cabo por Alex Maas [6], se inició la implementación en físico del algoritmo Modified Particle Swarm Optimization, evaluando distintas opciones de microcontroladores, sistemas embebidos, lenguajes de programación, entornos de desarrollo y robots móviles. La investigación necesitó el desarrollo de técnicas para la validación del algoritmo en un ambiente controlado debido a que no se contaban con plataformas móviles operativas en el momento. Durante esta etapa fueron desarrollados dos sistemas de comunicación, el primero para la obtención de la pose de cada agente dentro de la mesa de pruebas en tiempo real y el segundo que permite el intercambio de información útil entre los agentes para apoyar el algoritmo MPSO. Ambos sistemas de comunicación utilizan una programación multihilos para permitir la recepción de información de los agentes sin alterar la ejecución en paralelo del algoritmo de PSO modificado.

Por último, Rubén Lima [3] realizó las primeras pruebas en la plataforma de captura de movimiento de Robotat con plataformas móviles, que incluían un Bytebot y un Bytebot3B. Se encontró que los resultados obtenidos tenían ciertas diferencias en comparación de los logrados en simulación, tomando en cuenta que los robots tenían características diferentes entre sí y con la limitante de solo contar con dos robots para la validación.

Ant Colony Optimization (ACO)

Durante la tercera fase del proyecto Robotat, Gabriela Iriarte [7] implementó un algoritmo basado en Ant Colony que trata de replicar el comportamiento de la hormigas para encontrar un camino óptimo para buscar alimento y llevarlo al hormiguero. Uno de los objetivos principales de la investigación era compararlo con el algoritmo de MPSO [4] y comprobar si este podría ser una alternativa para la aplicación en pruebas físicas. Este algoritmo incluye parámetros como una feromona de la colonia para permitir al resto de agentes encuentren el camino entre el nodo inicial y el final deseado. Se evaluaron métodos

de planificación de trayectorias con grafos y sin grafos, todo por medio de simulaciones, donde los que si cuentan con grafos son más computacionalmente demandantes y los que no los incluyen tienen trayectorias no tan ideales entre el punto inicial y el punto final. De igual forma, una de las conclusiones obtenidas por la investigación es que si se desea controlar enjambres de robots de forma simultánea es mejor utilizar el algoritmo de MPSO debido a que puede llevar a varios agentes directamente al punto final deseado debido a su naturaleza vectorial.

En la siguiente fase Walter Sierra [8] tomo estos avances y el algoritmo planteado en [7] para poder migrarlos a una plataforma Raspberry Pi con la que se pudieran realizar pruebas en físico. Al no contar con una plataforma móvil funcional fue necesario realizar pruebas simples en ambientes controlados donde poder validar el funcionamiento correcto del algoritmo. Se implementó una programación multihilos y transmisión de datos mediante un protocolo UPD que permite la ejecución de varias tareas en simultáneo. Además, se implementó el controlador de pose y el controlador de pose de Lyapunov que en trabajos anteriores presentaron la respuesta más suave de velocidad [7]. Para la validación se realizaron dos escenarios, uno con un mapa sin obstáculos y otro con un mapa con varios obstáculos, donde se obtuvo que el funcionamiento del algoritmo al ser capaz de cambiar de ruta al detectar un obstáculo nuevo.

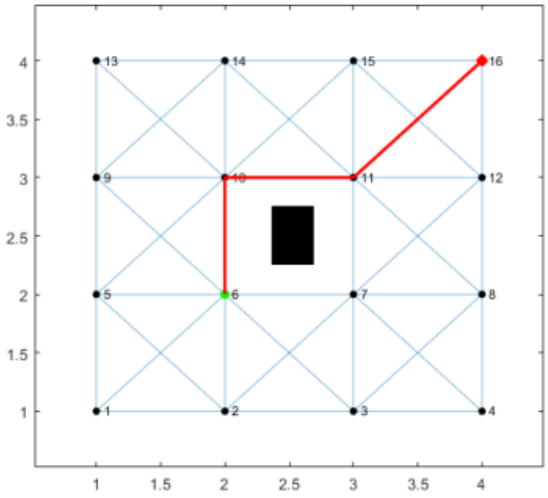

Figura 3: Ruta óptima generada por algoritmo ACO con un obstáculo [8].

Justificación

La robótica de enjambre se inspira en los ejemplos de enjambres que encontramos en la naturaleza, como colonias de bacterias, colonia de hormigas, bancos de peces, bandadas de pájaros, entre otros. La idea es poder replicar estos comportamiento con robots que nos permitan utilizarlos en aplicaciones prácticas.

En el departamento de Ingeniería Macatrónica, Electrónica y Biomédica de la Universidad del Valle de Guatemala se llevan trabajando distintos proyectos de robótica de enjambre

durante varios años. Se han llevado a cabo distintas fases en las que se incluyen avances en la selección de distintos algoritmos de enjambre, el desarrollo de algoritmos para pruebas con plataformas móviles, el desarrollo de una plataforma de pruebas por medio de visión por computadora, la validación de algoritmos y selección de controladores en simulación, la evaluación de distintas opciones de microcontroladores, sistemas embebidos, lenguajes de programación, entornos de desarrollo y robots móviles. En la fase más reciente se iniciaron pruebas en físico para la validación de los avances en los algoritmos de *Particle Swarm Optimization* y *Ant Colony* con la restricción de no contar con las suficientes plataformas móviles capacitadas para realizar todas las pruebas necesarias.

Con este trabajo de graduación se busca avanzar en la línea de investigación de robótica de enjambre, aprovechando el ecosistema Robotat y la mayor cantidad de plataformas móviles con las que se cuenta. Se realizará una nueva validación de los algoritmos para encontrar trayectorias de agentes móviles. El proceso se continúa con la migración y desarrolló de los algoritmos necesarios que puedan ser aplicados en las robots móviles Pololu 3Pi+ que están disponibles, para luego hacer que los agentes ejecuten las trayectorias resultantes.

Objetivos

Objetivo General

Implementar y validar los algoritmos de robótica de enjambre *Particle Swarm Optimization* (PSO) y *Ant Colony Optimization* (ACO) en el ecosistema Robotat, utilizando las plataformas móviles Pololu 3Pi+.

Objetivos Específicos

- Migrar los algoritmos de PSO y ACO desarrollados con anterioridad para poder ser utilizados en las plataformas móviles Pololu 3Pi+.
- Replicar pruebas y experimentos realizados en fases anteriores a nivel de simulación y con plataformas móviles y no móviles, ahora con las nuevas plataformas móviles disponibles para la mesa de pruebas del Robotat.
- Evaluar y comparar el funcionamiento de los algoritmos y las plataformas móviles en distintos escenarios y verificar bajo qué condiciones y parámetros se logran los mejores desempeños.

Marco teórico

Robótica de Enjambre

La robótica de enjambre es la coordinación de múltiples robots para llevar a cabo una tarea de forma colectiva de forma más eficiente que un único robot. Comúnmente se desarrolla por medio de la comunicación autónoma entre los agentes del enjambre para poder interactuar entre sí y con el ambiente [9].

Taxonomía

Un enjambre debe de contar con una serie de comportamientos colectivos principales para que los agentes puedan afrontar cualquier aplicación compleja en la vida real. Estos comportamientos colectivos son clasificados en cuatro categorías principales [10]:

- Organización espacial: Son los comportamientos que permiten a los enjambres moverse de manera coordinada en su entorno, logrando organizarse y distribuirse tanto entre sí como en relación a los objetos presentes en el área.
  - Agregación: La meta es reunir a todos los robots en una región específica del entorno. Es fundamental para permitir que los robots estén suficientemente cerca para interactuar entre sí.
  - Formación de patrones: Tiene como objetivo desplegar robots de manera regular y repetitiva. Un caso especial es la formación de cadena, donde la idea es posicionar a los robots de manera que se conecten dos puntos en el espacio. La cadena que forman luego puede utilizarse como guía para la navegación.
  - Agrupación y ensamblaje de objetos: Permite que los robots puedan manipular objetos distribuidos espacialmente y es esencial para procesos de construcción.

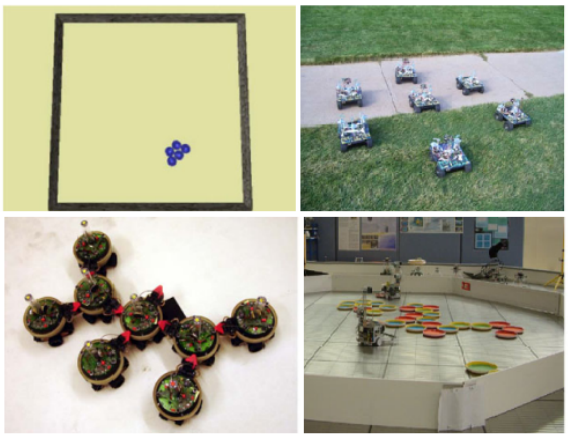

Figura 4: Ejemplos de comportamientos de navegación [11].

- Navegación: Son los comportamientos que permiten afrontar el problema de la coordinación de los movimientos para un enjambre de robots.
  - Exploración colectiva: El enjambre de robots navega de manera cooperativa a través del entorno con el objetivo de explorarlo. Es utilizado para obtener una visión general de una situación, buscar objetos, supervisar el entorno o establecer una red de comunicación.
  - Movimiento coordinado: Desplaza al enjambre de robots en una formación, donde se puede tener una forma bien definida, como una línea, o ser arbitraria.
  - Transporte colectivo: Permite al enjambre de robots mover colectivamente objetos que son demasiado pesados o grandes para un solo robot.

- Localización colectiva: Permite al enjambre que encuentren su posición y orientación relativa entre sí mediante el establecimiento de un sistema de coordenadas local.

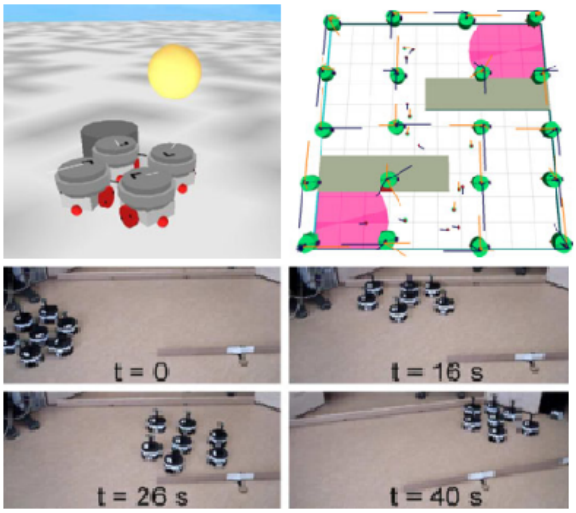

Figura 5: Ejemplos de comportamientos de organización espacial [11].

- Toma de decisiones Colectiva: Se ocupa de como los robots se influyen entre sí para la toma de una decisión, donde se utiliza para responder las necesidades de ponerse de acuerdo y con la especialización.

  - Consenso: Permite que los robots individuales en el enjambre lleguen a un acuerdo o converjan hacia una única elección común entre varias alternativas.
  - Asignación de tareas: Comportamiento en donde los robots se distribuyen dinámicamente entre sí para realizar distintas tareas. El objetivo es maximizar el rendimiento de todo el sistema del enjambre.
  - Detección colectiva de fallas: Determinación autónoma por parte del enjambre de las deficiencias de robots individuales. Permite identificar robots que se desvían del comportamiento deseado del enjambre basado en la emisión de una señal síncrona.
  - Percepción colectiva: Se combinan todos los datos detectados localmente por los robots en el enjambre en una imagen global. Permite al enjambre tomar decisiones colectivas de manera informada, como puede ser la clasificación de objetos de manera confiable.
  - Regulación de tamaño del grupo: Permite que los robots en el enjambre se organicen en grupos del tamaño deseado, donde si se excede el tamaño, se dividen en múltiples grupos.

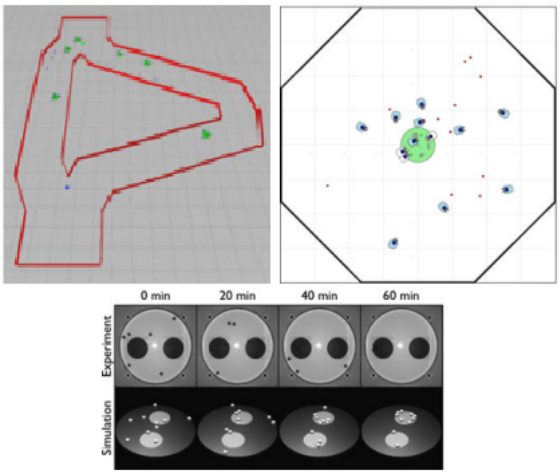

Figura 6: Ejemplos de comportamientos de toma de decisiones colectiva [11].

- Otros Comportamientos Colectivos
  - Auto-reparación: Permite que el enjambre se recupere de fallas causadas por deficiencias en algún robot. El objetivo es minimizar el impacto de las fallas de los robots en el resto del enjambre.
  - Auto-reproducción: Permite que un enjambre de robots cree nuevos robots o replique un patrón creado a partir de varios individuos. La principal idea es aumentar la autonomía del enjambre al eliminar la necesidad de un ingeniero humano para crear nuevos robots.
  - Interacción humano-enjambre: Permite que los humanos controlen los robots en el enjambre o reciban información de ellos. La interacción puede ser remota o de forma próxima en un entorno compartido.

Particle Swarm Optimization (PSO)

El algoritmo original fue introducido en 1995 por James Kennedy y Russell Eberhart [12]. Esta técnica utiliza un mecanismo simple que imita el comportamiento de enjambre de algunos grupos de animales, como lo son las parvadas y los cardúmenes, para guiar a las partículas a buscar soluciones óptimas globales [13].

En PSO, las partículas se colocan en un espacio de búsqueda de un problema o función, y cada entidad evalúa una función objetivo en su ubicación actual. Luego, cada partícula determina su posición en el espacio de búsqueda combinando algún aspecto de su historial de ubicaciones actuales y las mejores ubicaciones con las de uno o más miembros del enjambre, incluyendo algunas perturbaciones aleatorias. La siguiente iteración se produce después de que se hayan movido todas las partículas. En su última instancia, es probable que el enjambre se acerque al valor óptimo de la función ajustada [14].

Estructura del algoritmo PSO

Primero se inicializa la población deseada, donde cada partícula tiene una solución actual particular dada por un vector

$$P_i = [x_1, x_2, \ldots, x_d]\,,$$

siendo d la dimensión del espacio de trabajo. Para cada partícula se evalúa la función de costo, *f*, o *fitness function*

$$f(P).$$

Después, cada partícula guarda en su memoria la solución con el menor costo que ha encontrado durante un número específico de iteraciones, conocida como el *local best* de cada partícula. Además, cada partícula transmite su mejor solución encontrada y se determina cuál de todas las soluciones es la mejor en el conjunto de todas las partículas, llamada *global best*. Utilizando esta información, cada partícula calcula los dos primeros factores de la ecuación de PSO:

Factor cognitivo: Es la resta vectorial entre la solución de *local best* y la solución actual.

$$P_{local} - P_i$$

Factor social: Es la resta vectorial entre la solución de *global best* y la solución actual.

$$P_{global} - P_i$$

Ya contando con ambos factores, puede estructurarse la ecuación para la actualización de la velocidad de las partículas:

$$V_{i+1} = V_i + (P_{local} - P_i) + (P_{global} - P_i)\,,$$

que se compone de los factores previamente mencionados, sumados a la velocidad actual de cada partícula. El algoritmo tiene en cuenta una variedad de factores, incluido el factor de escalamiento, que determina si las partículas tienen un área de búsqueda más amplia o se concentran en un área más estrecha, representado por los valores *C1* y *C2*. El factor de uniformidad se refiere a dos números aleatorios generados en el rango de 0 a 1, denotados como *r1* y *r2*.

Por otro lado, el factor de constricción φ controla la longitud de los pasos que cada partícula puede dar en cada iteración del algoritmo [15]. Este parámetro se calcula utilizando la fórmula específica:

$$\varphi = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}; \quad \phi = C_1 + C_2; \quad \phi > 4 \tag{1}$$

El factor de inercia ω es una ponderación que se le da al factor de memoria de la ecuación de velocidad PSO. Según Eberhart y Shi , en su modificación para el algoritmo, existen varias ecuaciones para calcular este parámetro [16], como pueden ser:

Inercia constante:

$$0.8 < w < 1.2 \tag{2}$$

Inercia Linear Decreciente:

$$w = w_{\text{máx}} - (w_{\text{máx}} - w_{\text{mín}}) \frac{iter}{MAX_{iter}} \tag{3}$$

Inercia Caótica:

$$\begin{gathered} cZ_i \in [0, \ldots 1] \\ Z_{i+1} = 4Z_i (1 - Z_i) \\ w = (w_{\text{máx}} - w_{\text{mín}}) \frac{MAX_{iter} - iter}{MAX_{iter}} w_{\text{mín}} Z_{i+1} \end{gathered} \tag{4}$$

Inercia Aleatoria:

$$\begin{gathered} \text{rand}() \in [0, \ldots 1] \\ w = 0.5 + \frac{\text{rand}()}{2} \end{gathered} \tag{5}$$

Inercia Exponencial:

$$w = w_{\text{mín}} + (w_{\text{máx}} - w_{\text{mín}}) \, e^{\frac{MA-t}{10}} \tag{6}$$

Ya contando con todos los parámetros de ponderación a utilizar, la ecuación para la velocidad PSO resulta:

$$V_{i+1} = \varphi \left[\omega V_i + C_1 \rho_1 \left(P_{local} - P_i \right) + C_2 \rho_2 \left(P_{global} - P_i \right) \right], \tag{7}$$

donde en la ecuación, V_i representa la velocidad actual de la partícula y V_{i+1} representa la nueva velocidad calculada para la partícula. Después de actualizar las velocidades de todas las partículas, se procede a calcular las posiciones de cada una de ellas:

$$X_{i+1} = X_i + V_{i+1} \Delta t, \tag{8}$$

donde X_i representa la posición actual de la partícula y X_{i+1} representa la nueva posición calculada para la partícula. El término Δt representa el tiempo que lleva al algoritmo realizar cada iteración.

Ant Colony Optimization (ACO)

Este algoritmo está basado en el comportamiento natural de una colonia de hormigas, donde se observa como las hormigas tienen la capacidad de encontrar un camino óptimo entre el hormiguero y una fuente de alimento. A partir de este comportamiento, Marco Dorigo desarrolló un algoritmo denominado *Ant System* (SA), que luego de basarse en un método metaheurístico, en donde se ven a las hormigas como base del algoritmo ACO, que permite la solución de problemas discretos de optimización [17].

Funcionamiento ACO

Las colonias de hormigas proporcionan un ejemplo de estigmergia, donde el comportamiento de las hormigas individuales conduce a una inteligencia colectiva. En muchas especies de hormigas, cuando las hormigas se desplazan hacia y desde una fuente de alimento, liberan una sustancia llamada feromona en el suelo. Otras hormigas detectan la presencia de feromona y tienden a seguir los caminos con una mayor la concentración. Este mecanismo permite que las hormigas transporten eficientemente el alimento de regreso al hormiguero [17].

Deneubourg investigó exhaustivamente el comportamiento de las hormigas en cuanto a la colocación y seguimiento de feromonas, en un experimento conocido como el experimento del doble puente [18]. Inicialmente, cada hormiga elige aleatoriamente uno de los dos puentes, pero luego, debido a fluctuaciones aleatorias, uno de los puentes presenta una concentración más alta de feromonas que el otro y, por lo tanto, atrae a más hormigas. Esto a su vez lleva a una mayor cantidad de feromonas en ese puente, lo que lo hace que después de algún tiempo, toda la colonia converge en el uso del mismo puente. A partir de este experimento se desarrollo un modelo dado por:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h} \tag{9}$$

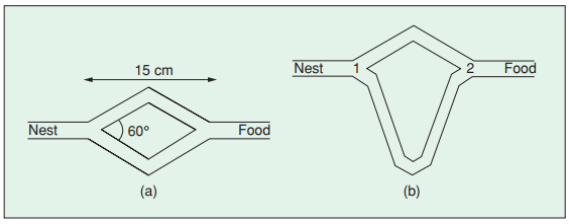

Figura 7: Configuración para el experimento del doble puente [17].

Simple Ant Colony Optimization (SACO)

Este algoritmo fue utilizado en las pruebas del experimento del doble puente, donde una hormiga artificial se encarga de navegar un grafo de construcción completamente conectado $G_C(\mathbf{V},\mathbf{E})$, donde $\mathbf{V}$ es un conjunto de vértices, nodos, y $\mathbf{E}$ es un conjunto de aristas. Las

hormigas se mueven entre nodos, a lo largo de las aristas del grafo, construyendo incrementalmente una solución parcial. Además, las hormigas depositan una cierta cantidad de feromonas en los componentes, ya sea en los vértices o en las aristas que atraviesan [17].

Su principal característica es que, en cada iteración, los valores de feromona son actualizados por todas las hormigas, m que han construido una solución en la propia iteración. La feromona τ_{ij}, asociada con la arista que une los nodos i y j, se actualiza de la siguiente manera:

$$\tau_i j = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{10}$$

Donde, ρ es la taza de evaporación de la feromona, m es el número de hormigas y $\Delta\tau_{ij}{}^{k}$ es la cantidad de feromona que hay en las aristas.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{si } k \in \text{aristas}(i,j) \\ 0 & \text{en otro caso} \end{cases}, \tag{11}$$

Donde, Q es una constante y L_k es la longitud del recorrido construido por la hormiga k.

$$(p_{(i,j)}^{k}(t) = \begin{cases} \frac{(\tau_y(t))^{\alpha}}{\sum_{k\in J_k}(\tau_i(t))^{\alpha}} & \text{si } j \in N_i^k \\ 0 & \text{si } j \notin N_i^k \end{cases} \tag{12}$$

Donde, $\mathbf{N}(s^p)$, es el conjunto de componentes factibles, es decir las aristas $(i; l)$ donde l son los nodos que no han sido visitado por la hormiga k. Los parámetros α y β controlan la importancia de la feromona [17].

Robots Móviles

Son una clase de robots capaces de moverse a través del entorno. Existen robots capaces de moverse en el suelo, sobre el agua, bajo el agua y a través del aire. Una de las funciones más importantes de estos robots es llegar a un punto de destino, sin importar cuál sea, tomando algún camino donde se enfrentara ante desafíos como obstáculos que puede bloquear su camino [19].

Pololu 3Pi+ 32U4

El Pololu 3Pi+ es un una plataforma móvil versátil, de alto rendimiento y programable por un usuario. El robot cuenta con un microcontrolador ATmega32UA AVR de Microchip, una interfaz de USB y viene precargado con un arranque compatible con Arduino. El 3pi+ incluye dos drivers puentes H y una variedad de sensores integrados, incluyendo un par de codificadores de cuadratura para control de motor de bucle cerrado, una unidad de medida inercial completa, cinco sensores de reflectancia orientados hacia abajo para seguimiento de línea o detección de bordes, y sensores de impacto izquierdo y derecho a lo largo de la cara frontal del robot. Tres botones integrados ofrecen una interfaz conveniente para la entrada del usuario y una pantalla OLED gráfica de 128 × 64, un zumbador y LEDs indicadores que permiten que el robot proporcione retroalimentación [20].

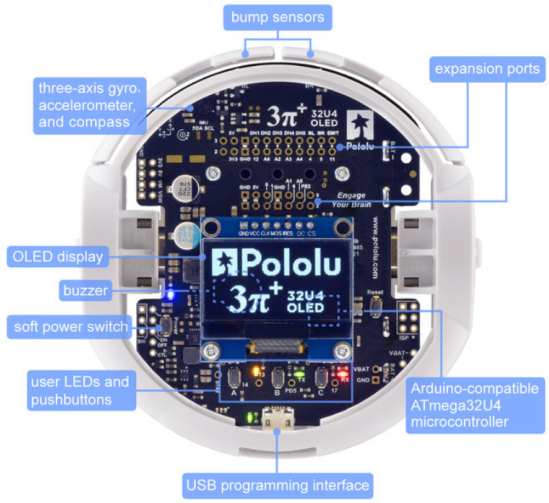

Figura 8: Estructura y ubicación de los componentes en Pololu 3pi+ [20].

OptiTrack

La plataforma de pruebas del Robotat utiliza seis cámaras Prime^{x} 41 de OptiTrack. Este es un sistema de captura de movimiento óptico altamente preciso con una resolución de captura de hasta 4.1 megapíxeles y una velocidad de captura de hasta 180 cuadros por segundo. La cámara Prime^{x} 41 ofrece una calidad excepcional en la captura de movimiento. debido al sensor CMOS y diseño óptico que le permiten una gran precisión para la captura de movimientos.

El sistema de seguimiento óptico de OptiTrack utiliza marcadores activos o pasivos que se colocan en los objetos o sujetos que se desean rastrear. La cámara Prime^{x} 41 detecta y registra en tiempo real la posición y la orientación de estos marcadores, lo que permite un análisis detallado y preciso del movimiento.

Motive es el software de captura de movimiento de OptiTrack, diseñado para aprovechar al máximo los datos obtenidos de los sistemas de seguimiento óptico. El software permite la configuración y calibración precisa de los sistemas de captura de movimiento de OptiTrack, asegurando una captura de datos precisa y confiable. Permite la sincronización de múltiples cámaras y otros dispositivos de captura para lograr un seguimiento y una grabación de movimiento precisos y coordinados. Motive procesa los datos de las cámaras de OptiTrack para proporcionar posiciones globales en tres dimensiones, identificadores de marcadores e información de rotación con respecto al sistema de referencia local del objeto.

Figura 9: Cámara Prime^{x} 41 [21].

Protocolos de Comunicación

El Protocolo de Datagramas de Usuario (UDP) y el Protocolo de Control de Transmisión (TCP) son protocolos de enrutamiento de la capa de transporte que se consideran parte de los protocolos principales del conjunto de protocolos de Internet. Estos protocolos manejan los datos de forma diferente.

TCP utiliza un enfoque orientado a la conexión, lo que significa que establece una conexión entre el emisor y el receptor antes de transmitir datos. Se busca proporcionar una forma confiable de enviar mensajes o información, ya que garantiza la entrega de los paquetes. Si se produce algún error durante la transmisión, el protocolo reenvía automáticamente los paquetes perdidos o dañados a través de la red.

Por otro lado, UDP utiliza un modelo de transporte más simple y directo. No establece una conexión previa entre el emisor y el receptor, lo que lo hace más liviano. Las aplicaciones informáticas que utilizan UDP pueden transmitir mensajes en forma de datagramas, que son paquetes independientes. Esto hace que UDP sea adecuado para aplicaciones que requieren una transmisión rápida de datos, como transmisiones en tiempo real de voz y video [22].

Metodología

Verificación de algoritmos previos

Primero se realizará una verificación de los algoritmos en simulación que fueron desarrollados durante fases previas del proyecto. Para esto se iniciará con una validación de las simulaciones en Matlab para tener un entendimiento del funcionamiento de cada algoritmo, así como de los parámetros utilizados para los casos estudiados. Luego, se revisarán las simulaciones realizadas en Webots, de nuevo para poder comprender los avances al utilizar un simulador con plataformas móviles.

Actualización de software: Webots2023a

Se llevará a cabo la migración de los algoritmos desarrollados en Webots hacia la versión 2023a, la más reciente del software. Aquí se busca obtener información valiosa sobre el funcionamiento de los algoritmos en plataformas móviles, lo que resultará en una comprensión más sólida y facilitará la migración hacia robots móviles físicos más adelante.

Pruebas con los robots Pololu 3Pi+

Se efectuarán pruebas generales con los robots, para poder contar con un mayor entendimiento de su funcionamiento, en conjunto con la plataforma de pruebas del Robotat. La idea es poder comprender la comunicación que puede establecerse y facilitar la toma de decisiones en cuanto a la arquitectura de control entre los robots y la plataforma.

Migración de los algoritmos

Luego de comprender a detalle los algoritmos de PSO y ACO, se iniciará con la migración y adaptación de estos para poder aplicarlos con los robots móviles en la plataforma del Robotat. Se tomarán en cuenta que detalles se deberán cambiar dentro del código para que el funcionamiento sea el ideal, ajustando los valores y funciones que sean necesarias. Se tendrá que verificar el lenguaje de programación que debe utilizarse para la migración.

Validación de la migración de algoritmos

Se deberán realizar pruebas para verificar que los algoritmos funciones correctamente, empezando con verificar que el código no de ningún error inicial y que los robots estén recibiendo los datos que deseamos que utilicen para la comunicación de enjambre. Para ambos algoritmos se iniciará corriendo los códigos sin efectuar acciones en los robots para poder analizar que se esta transmitiendo en todo el sistema, para evitar problemas de mayor gravedad si algún robot realiza una acción no deseada. Ya con todo comprobado, se realizarán pruebas con los algoritmos, tomando primero algunos casos estudiados en las fases previas con los cuáles tener un punto de comparación, para luego desarrollar nuevos casos y configuraciones.

Desarrollo de pruebas y comparación para los algoritmos

Se creará una lista de escenarios con la finalidad de poder comprobar bajo que condiciones funciona mejor cada algoritmo y que brindé una idea de en que casos es mejor utilizar los enjambres con PSO y en que casos con ACO. Se planteará un criterio de comparación para poder identificar las fortalezas y debilidades de los algoritmos ante cada escenario. Con dichas pruebas realizadas, se analizarán los resultados para ver si es posible concluir sobre los mejores parámetros y el mejor algoritmo para cada escenario con las plataformas móviles.

Cronograma de actividades

ACTIVIDADES	FECHA INICIO DE SEMANA: INICIO	FECHA INICIO DE SEMANA: FINAL
1. Verificación de algoritmos previos		
1.1 Investigación sobre los parámetros de cada algoritmo	1	2
1.2 Validación de simulaciones en Matlab.	1	2
1.3 Validación de simulaciones en Webots.	1	2
2. Actualización de simulaciones en Webots 2023a		
2.1 Comparación entre versiones del software.	2	3
2.2 Creación de los mundos y controladores en Webots 2023a.	2	4
2.3 Recreación de las pruebas hechas en versiones anteriores y revisión de los parámetros importantes en una migración.	2	5
3. Pruebas con los Pololu 3pi+		
3.1 Realizar movimientos sencillos con librerías de robótica	4	5
3.2 Revisar la comunicación con el sistema de captura de movimiento para ver qué información se necesita enviar por parte el algoritmo.	4	6
3.3 Pruebas para decidir la arquitectura de control.	4	6
4. Migración de Algoritmos		
4.1 Verificación del lenguaje de programación.	5	6
4.2 Migración de algoritmo PSO.	6	9
4.3 Migración de algoritmo ACO.	6	9

	MES		JUNIO				JULIO					AGOSTO				SEPTIEMBRE			
	FECHA INICIO DE SEMANA		5	12	19	26	3	10	17	24	31	7	14	21	28	4	11	18	25
ACTIVIDADES	INICIO	FINAL	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
5. Validación de migración de algoritmos																			
5.1 Verificación de código libre de errores.	9	15																	
5.2 Comprobación de comunicación con plataforma robótica.	9	11																	
5.3 Pruebas de algoritmo con ejemplos de fases previas para su comparación.	10	12																	
6. Desarrollo de pruebas y comparación de algoritmos																			
6.1 Creación de escenarios para pruebas.	10	12																	
6.2 Desarrollo de criterio para comparación.	11	12																	
6.3 Pruebas nuevas para los escenarios.	11	14																	
6.4 Análisis de los resultados.	12	15																	
6.5 Evaluar si hay conclusiones que puedan obtenerse.	13	16																	
7. Generales																			
7.1 Redacción de Tesis.	1	17																	

Índice preliminar

Referencias

[1] D. Pickem, P. Glotfelter, L. Wang et al., «The Robotarium: A remotely accessible swarm robotics research testbed,» en *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, págs. 1699-1706. DOI: 10.1109/ICRA.2017.7989200.

[2] Y. Tan y Z.-y. Zheng, «Research Advance in Swarm Robotics,» *Defence Technology*, vol. 239, mar. de 2013. DOI: 10.1016/j.dt.2013.03.001.

[3] R. A. Lima, «Implementación y validación de algoritmos de robótica de enjambre en plataformas móviles en la nueva mesa de pruebas del laboratorio de robótica de la UVG,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2022.

[4] A. S. Aguilar, «Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO),» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2019.

[5] A. Aguilar, M. Zea y L. Rivera, «PSO Trajectory Planner Using Kinematic Controllers that Ensure Smooth Differential Robot Velocities,» dic. de 2020, págs. 481-488. DOI: 10.1109/SSCI47803.2020.9308498.

[6] A. D. 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.

[7] G. Iriarte, «Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica,» Tesis de licenciatura, Universidad Del Valle de Guatemala, 2021.

[8] W. A. 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.

[9] M. M. Shahzad, Z. Saeed, A. Akhtar et al., «A Review of Swarm Robotics in a NutShell,» *Drones*, vol. 7, n.º 4, 2023, ISSN: 2504-446X. dirección: https://www.mdpi.com/2504-446X/7/4/269.

[10] M. Schranz, M. Umlauft, M. Sende y W. Elmenreich, «Swarm Robotic Behaviors and Current Applications,» *Frontiers in Robotics and AI*, vol. 7, 2020, ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00036. dirección: https://www.frontiersin.org/articles/10.3389/frobt.2020.00036.

[11] M. Brambilla, E. Ferrante, M. Birattari y M. Dorigo, «Swarm Robotics: A Review from the Swarm Engineering Perspective,» *Swarm Intelligence*, vol. 7, págs. 1-41, mar. de 2013. DOI: 10.1007/s11721-012-0075-2.

[12] R. Eberhart y J. Kennedy, «A new optimizer using particle swarm theory,» en *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, págs. 39-43. DOI: 10.1109/MHS.1995.494215.

[13] M. N. Ab Wahab, S. Nefti-Meziani y A. Atyabi, «A Comprehensive Review of Swarm Optimization Algorithms,» *PLOS ONE*, vol. 10, n.º 5, págs. 1-36, mayo de 2015. DOI: 10.1371/journal.pone.0122827. dirección: https://doi.org/10.1371/journal.pone.0122827.

[14] R. Poli, J. Kennedy y T. Blackwell, «Particle Swarm Optimization: An Overview,» *Swarm Intelligence*, vol. 1, oct. de 2007. DOI: 10.1007/s11721-007-0002-0.

[15] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon y A. Abraham, «Inertia Weight strategies in Particle Swarm Optimization,» en *2011 Third World Congress on Nature and Biologically Inspired Computing*, 2011, págs. 633-640. DOI: 10.1109/NaBIC.2011.6089659.

[16] B. Obayyanahatti e Y. Shi, «Comparing inertial weights and Constriction factor in particle swarm optimization,» vol. 1, feb. de 2000, 84-88 vol.1, ISBN: 0-7803-6375-2. DOI: 10.1109/CEC.2000.870279.

[17] M. Dorigo, M. Birattari y T. Stutzle, «Ant colony optimization,» *IEEE Computational Intelligence Magazine*, vol. 1, n.º 4, págs. 28-39, 2006. DOI: 10.1109/MCI.2006.329691.

[18] J.-L. Deneubourg, S. Aron, S. Goss y J. Pasteels, «The Self-Organizing Exploratory Pattern of the Argentine Ant,» *J. Insect Behav.*, vol. 3, pág. 159, mar. de 1990. DOI: 10.1007/BF01417909.

[19] P. Corke, *Robotics, Vision and Control - Fundamental Algorithms in MATLAB®* (Springer Tracts in Advanced Robotics). Springer, 2011, vol. 73, págs. 1-495, ISBN: 978-3-642-20143-1. dirección: http://dblp.uni-trier.de/db/series/star/index.html#Corke11.

[20] *Pololu 3pi+ 32U4 User's Guide*, English, Pololu Corporation, 2022, 86 págs. dirección: https://www.pololu.com/docs/pdf/0J83/3pi_plus_32u4.pdf, 2022.

[21] I. NaturalPoint, *PrimeX 41*, 2023. dirección: https://optitrack.com/cameras/primex-41/.

[22] F. Al-Dhief, N. Sabri, N. M. Abdul Latiff et al., «Performance comparison between TCP and UDP protocols in different simulation scenarios,» *International Journal of Engineering & Technology*, vol. 7, n.º 4, págs. 172-176, 2018. DOI: 10.14419/ijet.v7i4.12832.