Desarrollo de un método para una asignación eficiente de recursos implementando algoritmos de robótica de enjambre y de agrupamiento

Ronal Andrés Berganza Torres

UVG | UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

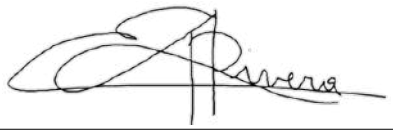
Desarrollo de un método para una asignación eficiente de recursos implementando algoritmos de robótica de enjambre y de agrupamiento

Trabajo de graduación presentado por Ronal Andrés Berganza Torres
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2025

Vo.Bo.:

(f) 

Dr. Luis Alberto Rivera Estrada

(f) 

M.Sc. Carlos Esquit Hernández

Fecha de aprobación: Guatemala, 20 de noviembre de 2025.

Prefacio

Este proyecto surge del interés sobre cómo realizar tareas no solo con un agente individual, sino utilizando diversos agentes robóticos para que en conjunto puedan realizar tareas dinámicas y abarcar mayor espacio y complejidad de una manera eficiente. Esta investigación concluye mi programa en la Universidad del Valle de Guatemala, donde, durante los últimos cinco años, he aprendido muchas cosas, tanto académicas como personales.

Agradezco primeramente a Dios por guiarme y sostenerme en todo momento; y a mis padres y a mi hermana por darme la oportunidad de estudiar en una de las mejores universidades de la región y por siempre apoyarme en mi desarrollo profesional y personal. También, agradezco a las personas y amistades que estuvieron desde el inicio, así como a las que se agregaron en el camino, por su apoyo para conseguir nuestra y por los momentos y recuerdos que me acompañarán toda la vida. Además, agradezco al Dr. Luis Alberto Rivera por su paciencia, dedicación, y disposición en apoyarme con retroalimentación, consejos y resolución de dudas, siempre mostrando su interés en ayudar. Por último, estoy agradecido con el tiempo que duró esta experiencia que sin duda atesoraré de aquí en adelante.

Índice general

Índice de figuras

Índice de cuadros

Resumen

La robótica de enjambre ofrece una alternativa para la asignación y la coordinación de múltiples agentes robóticos en entornos dinámicos, especialmente en aplicaciones como búsqueda, rescate y respuesta ante desastres naturales. Este trabajo presenta el desarrollo y validación de un método que combina un algoritmo de agrupamiento basado en densidad (DBSCAN) con el algoritmo de optimización de enjambre de partículas (PSO), con el objetivo de asignar recursos robóticos de manera proporcional y dirigirlos de forma eficiente hacia regiones de interés.

La metodología inicia con la generación de escenarios aleatorios en Matlab, donde DBSCAN identifica grupos de datos relevantes y elimina el ruido, permitiendo calcular centroides que sirven como objetivos. Los agentes se asignan proporcionalmente al tamaño de cada grupo y se desplazan hacia dichos puntos mediante trayectorias optimizadas con PSO, incorporando además un mecanismo local de evasión de obstáculos basado en campos potenciales y el uso de control PID para suavizar la navegación. Luego, los algoritmos se trasladan al entorno tridimensional Webots, donde se evalúan bajo condiciones físicas con mayor realismo.

Los resultados muestran un comportamiento consistente: el algoritmo combinado reduce el error cuadrático medio, mejora el tiempo de convergencia y mantiene trayectorias eficientes tanto en 2D como en 3D. Las simulaciones tridimensionales permitieron evaluar el algoritmo en entornos más realistas de una manera estable y escalable a distintos entornos.

Palabras clave: robótica de enjambre, agrupamiento, DBSCAN, optimización por enjambre de partículas, control de movimiento, *swarm robotics, data clustering, particle swarm optimization, robotic simulation, motion control.*

Abstract

Swarm robotics offers an efficient alternative for the allocation and coordination of multiple robotic agents in dynamic environments, particularly in applications such as search, rescue, and disaster response. This work presents the development and validation of a method that integrates a density-based clustering algorithm (DBSCAN) with the particle swarm optimization (PSO) algorithm, with the aim of proportionally assigning robotic resources and directing them efficiently toward regions of interest.

The methodology begins with the generation of random scenarios in Matlab, where DBSCAN identifies relevant data groups and filters out noise, enabling the computation of centroids that serve as navigation targets. Agents are assigned proportionally to the size of each group and move toward these points using PSO-optimized trajectories. A local obstacle-avoidance mechanism based on potential fields, together with a PID-based motion smoothing strategy, further enhances navigation. The algorithms are then transferred to the three-dimensional Webots simulation environment, where they are evaluated under more realistic physical conditions.

The results show consistent behavior: the combined approach reduces mean squared error, improves convergence time, and maintains efficient trajectories in both 2D and 3D simulations. The three-dimensional experiments demonstrate that the algorithm maintains stable and scalable performance across varied and realistic environments.

Keywords: swarm robotics, data clustering, particle swarm optimization, robotic simulation, motion control.

CAPÍTULO 1

Introducción

La premisa principal de la robótica de enjambre es la coordinación de múltiples agentes robóticos sencillos que, al trabajar colectivamente, resuelven problemas de mayor complejidad que los que podría abordar un único robot. La asignación eficiente de recursos robóticos en entornos dinámicos se presenta como un desafío relevante para aplicaciones como búsqueda y rescate, y logística inteligente de monitorización y mitigación de desastres naturales.

En este contexto, los algoritmos de agrupamiento permiten identificar patrones y concentraciones de datos en entornos dinámicos mientras que los algoritmos de optimización, como el *particle swarm optimization* (PSO), facilitan el direccionamiento de agentes hacia objetivos de manera eficiente y robusta. La combinación de ambas técnicas abre la posibilidad de desarrollar metodologías capaces de adaptarse a escenarios complejos, como los que surgen tras un desastre natural.

El objetivo de este proyecto es desarrollar un método para la asignación eficiente de recursos robóticos en entornos simulados de desastres naturales mediante la integración de un algoritmo de agrupamiento y el algoritmo PSO. El enfoque busca no solo mejorar la precisión y rapidez con que los agentes alcanzan las zonas de interés, sino también garantizar que el sistema mantenga estabilidad aun cuando se introduzcan obstáculos o se traslade a entornos tridimensionales más realistas.

El tema se delimita al uso de simulaciones en Matlab y Webots como plataformas principales de experimentación. Matlab se emplea para la validación inicial de algoritmos de agrupamiento y optimización en un entorno de dos dimensiones simplificado mientras que Webots permite trasladar dichos resultados a un entorno de tres dimensiones con dinámica física más cercana a la realidad. Asimismo, se consideran como referencia entornos físicos con robots diferenciales pequeños, como el E-puck y el Pololu 3Pi+, hacia los cuales podrían extenderse los resultados en futuras fases de investigación.

La metodología general implementada consistió en lo siguiente: (1) generar datos aleatorios representativos de focos de interés en un entorno; (2) aplicar algoritmos de agrupamiento para identificar grupos de datos y calcular sus centroides; (3) asignar proporcionalmente

agentes robóticos a cada grupo; (4) emplear PSO para optimizar la trayectoria de los agentes hacia dichos objetivos, y (5) validar el desempeño en presencia de obstáculos y condiciones de simulación tridimensional. Este procedimiento permitió evaluar la viabilidad del método en diferentes niveles de complejidad sin necesidad de hardware en esta primera fase.

CAPÍTULO 2

Antecedentes

2.1. Robótica de enjambre (*swarm robotics*)

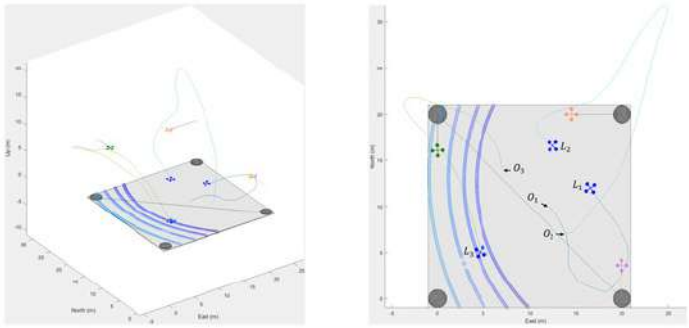
La robótica de enjambre surge como una forma de organizar distintos agentes robóticos para trabajar en conjunto con un mismo fin. Fue inspirada en el comportamiento colectivo de organismos como hormigas, abejas o bancos de peces en la realización de distintas tareas. Desde entonces ha evolucionado hacia sistemas descentralizados donde múltiples robots simples colaboran para resolver tareas complejas. El proyecto *Swarm-bots* desarrollado por Mondada et al. [1] ayudó a demostrar cómo robots pequeños podían autoensamblarse para lograr transportar objetos pesados, lo cual muestra la posibilidad de aplicación para la remoción de escombros. En su aplicación para atención de desastres, Yan et al. [2] utilizaron robótica de enjambre para búsqueda cooperativa en simulaciones de entornos desconocidos, logrando tiempos de respuesta un 20 % menores que robots individuales y un 35 % más rápidos que equipos humanos en áreas extensas. También el proyecto *Kilobot* [3] mostró cómo 1,000 microrrobots lograban analizar, crear, y formar patrones complejos, que marca una pauta para el mapeo de zonas afectadas.

2.2. *Particle swarm optimization* (PSO)

Este algoritmo, fue propuesto por Kennedy y Eberhart en 1995, como una técnica de optimización que fue bioinspirada por la emulación de la coordinación del movimiento de bandadas de aves o cardúmenes [4]. En PSO, las distintas "partículas" se utilizan para explorar un espacio de posible soluciones ajustando sus posiciones según el mejor resultado individual y el global del grupo. En [1] se utilizó un algoritmo PSO y se comparó con algoritmos de tipo genético, logrando superarlos con una convergencia a solución de un 15 % más veloz en las simulaciones de logística de emergencia. En complemento con la robótica de enjambre, Lalwani et al. [5] desarrollaron una variante del PSO llamado (*Multi-target PSO*),

como se observa en la Figura 1, para mapeo y búsqueda aérea con UAVs, logrando reducir el tiempo de localización de objetivos múltiples en un 25 % en comparación a métodos estáticos y un 10 % con respecto a un PSO estándar.

Figura 1. Multitarget-PSO.

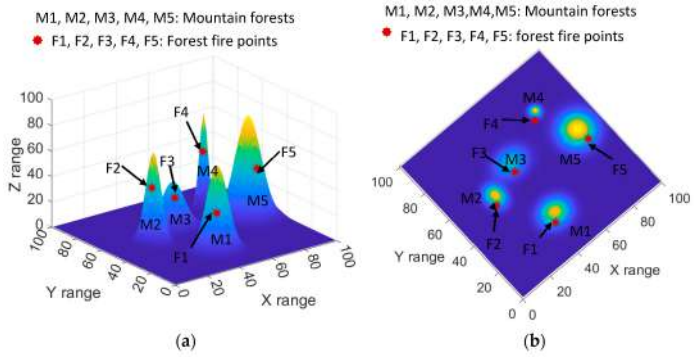


Nota. La imagen muestra las trayectorias de un enjambre de drones hacia distintos puntos objetivo. Esta imagen fue obtenida de [6].

2.3. Algoritmos de agrupamiento *(Clustering)*

Los algoritmos de agrupamiento o *clustering* (como *K-Means*, *DBSCAN* o *clustering* jerárquico) son formas de aprendizaje autónomo que consisten en organizar datos en grupos según similitudes entre ellos. Es muy utilizado en robótica para la recolección y agrupación de datos en entornos complejos. Gong et al. [7] combinaron *clustering* multiobjetivo con un algoritmo PSO para identificar agrupaciones de datos en redes biológicas complejas, para lograr mejorar la precisión de agrupamiento en un 10 % respecto a otros métodos.

Figura 2. Fire points clustering.



Nota. La imagen muestra el uso de agrupamiento DBSCAN en un escenario de incendio forestal. Esta imagen fue obtenida de [2].

Dentro de sus aplicaciones en atención de desastres, Ye et al. [8] integraron *clustering* a un modelo de respuesta a derrames de petróleo, logrando agrupar zonas por nivel de riesgo y optimizando la distribución de recursos para la atención del problema, lo que redujo el tiempo de respuesta en un 18 % y también los costos operativos hasta en un 12 %. También Yan et al. [2], aplicaron el algoritmo *DBSCAN* en enjambres de drones con el fin de detectar zonas críticas en simulaciones de incendios forestales, como se observa en la Figura 2, logran un incremento en la precisión de identificación en un 15 % frente a los métodos manuales.

2.4. Investigaciones en Universidad del Valle de Guatemala

En la Universidad del Valle de Guatemala (UVG), ya se han desarrollado múltiples investigaciones con orientación en la robótica de enjambre. En 2019, Aguilar [9] implementó un algoritmo de *modified particle swarm optimization* (MPSO) adaptado a las restricciones físicas de los robots Bitbots, enfocándose en la planificación de trayectorias, y Cahueque [10] aplicó PSO a escenarios de búsqueda y rescate. También en 2020, Iriarte [11] exploró el algoritmo *ant colony optimization* (ACO) para planificar trayectorias, que consistía en replicar el comportamiento de las hormigas que buscaban el camino óptimo entre un alimento y su colmena, y además dejan rastros para que las compañeras puedan conocer este camino, y lo compara con el MPSO para evaluar su rendimiento en robots diferenciales, y Menéndez [12] evaluó de manera física el algoritmo para identificar puntos de optimización en su funcionamiento en situaciones reales. Durante el 2024, Ana Barrientos [13] realizó una investigación para optimizar el algoritmo PSO en la detección de obstáculos en tiempo real estableciendo las posiciones iniciales, analizando las mejores posiciones que se pueden tomar según sea el entorno, y el cálculo de estas posiciones junto con la velocidad con la que se ejecutará la instrucción. También Ixcayau [14] utilizó técnicas de agrupamiento como K-means para categorizar segmentos en señales EEG, y así lograr diferenciar entre patrones epilépticos y no epilépticos.

CAPÍTULO 3

Justificación

En los últimos años se han presentado con mayor frecuencia e intensidad los desastres naturales, influenciada por factores como el cambio climático y el crecimiento demográfico, lo que ha puesto en evidencia la necesidad urgente de desarrollar tecnologías que optimicen las respuestas ante estas emergencias. Según el informe de *Emergency Events Database* (EM-DAT) [15], en 2023 se registraron 399 desastres naturales que afectaron a 94 millones de personas y generaron pérdidas económicas de aproximadamente 202.5 mil millones de dólares. La respuesta humana es necesaria, pero puede optimizarse en aspectos como velocidad, seguridad y resistencia, apoyándose con agentes robóticos. La robótica de enjambre es una solución eficiente y autónoma, al permitir que múltiples agentes robóticos colaboren de manera coordinada en la realización de tareas de forma rápida y efectiva.

En la Universidad del Valle de Guatemala (UVG), ya se han realizado investigaciones previas con esta orientación, con implementación de algoritmos de *particle swarm optimization* (PSO) y *ant colony optimization* (ACO) en robots Bitbots y Pololu 3Pi+ para tareas de búsqueda y planificación de trayectorias. Estas investigaciones ayudaron a identificar problemas a resolver, como el que los sistemas centralizados incrementan los tiempos de ejecución al asignarles más agentes, haciendo que la integración de obstáculos o análisis en tiempo real pueda mejorarse. Por ejemplo, el trabajo de Menéndez [12] mostró que el PSO, al escalar a más de diez agentes, reduce su eficiencia.

Por medio de estas investigaciones se ha mejorado la adaptabilidad a entornos dinámicos en tiempo real, no obstante, todavía carecen de la posibilidad de optimizar los recursos con base en la información recopilada. Por lo que, para superar estas limitaciones, se realizó la integración de *clustering* al algoritmo PSO ya desarrollado, junto con su validación en escenarios simulados de desastres naturales. Esto se hizo con el fin de optimizar la asignación de recursos en enjambres robóticos en entornos dinámicos, como los desastres naturales.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar un método para la asignación eficiente de recursos robóticos para el análisis de entornos simulados de desastres naturales, combinando algoritmos de agrupamiento y el algoritmo PSO.

4.2. Objetivos específicos

- Investigar y evaluar distintos algoritmos de agrupamiento para seleccionar el que mejor se adapte a entornos dinámicos de desastres naturales, y que sea apto para su combinación con el PSO.
- Implementar la combinación del algoritmo de agrupamiento seleccionado con el algoritmo PSO.
- Evaluar el rendimiento del algoritmo combinado utilizando simulaciones simples y escenarios de prueba básicos.
- Desarrollar escenarios de prueba con mayor complejidad que simulen situaciones de desastres naturales para evaluar el rendimiento del algoritmo combinado en dichos escenarios.

CAPÍTULO 5

Definición del problema

El alcance de esta investigación fue plantear la forma adecuada de combinar algoritmos de agrupamiento con optimización de partículas de enjambre para la generación de trayectorias y la asignación de agentes en un enjambre robótico. Este trabajo no buscó realizar validaciones físicas con agentes robóticos, sino centrarse en simulaciones realistas que permitieran analizar el desempeño de los algoritmos. Se llevó a cabo un estudio teórico de los algoritmos de agrupamiento y del algoritmo PSO, trasladando su aplicación desde un nivel de simulaciones simples en un entorno de dos dimensiones utilizando Matlab, hasta un entorno de tres dimensiones con mayor realismo simulado en WEBOTS. Se buscó evaluar cómo la identificación automática de grupos de objetivos mediante DBSCAN y la posterior asignación de robots con PSO influyen en la eficiencia del trabajo y el alcance del enjambre.

En Matlab se realizó la validación inicial de los parámetros de DBSCAN y PSO, analizando su comportamiento en el área de trabajo con obstáculos y generación de datos aleatorios. A partir de estos experimentos, se seleccionaron valores para la distribución de robots en el espacio de trabajo y una generación de trayectorias más rápidas a los objetivos definidos.

En WEBOTS, se implementaron los parámetros ajustados de DBSCAN y PSO para guiar a múltiples robots diferenciales en escenarios simulados. Se modelaron las ecuaciones de movimiento y leyes de control necesarias para el desplazamiento autónomo de los agentes, observándose una mejora significativa en la organización del enjambre frente a métodos de asignación aleatoria. Asimismo, se tomó en cuenta la importancia de evitar sobreasignaciones de robots a un mismo grupo, garantizando la robustez del sistema en un tiempo finito.

CAPÍTULO 6

Marco teórico

6.1. Robótica de enjambre

La robótica de enjambre, como dice la investigación realizada por Doe. et al. [16] tiene su origen en los comportamientos colectivos observados en sistemas biológicos como las colonias de hormigas, enjambres de abejas, o bandadas de aves. Estos sistemas se caracterizan por la cooperación no centralizada (pero sí ordenada), la organización y la capacidad que tienen de adaptarse a cambios en el entorno.

Para su aplicación en la ingeniería, la robótica de enjambre se define como el estudio, diseño e implementación de múltiples robots que cooperan entre sí mediante reglas simples para resolver tareas complejas. Brambilla et al. [17] muestra que la robótica de enjambre presenta ventajas como la escalabilidad, permitiendo incrementar el número de robots en determinadas situaciones para una respuesta mejorada. También destacan la robustez y la flexibilidad, las cuales permiten que el enjambre siga funcionando aunque varíe la cantidad de robots, redistribuyéndose y organizándose según lo requiera la situación. En sus aplicaciones prácticas, la robótica de enjambre ha sido utilizada en áreas como el monitoreo ambiental, logística en almacenes, e incluso en exploración espacial.

6.2. Comportamientos colectivos en enjambres robóticos

Los comportamientos colectivos son el resultado de la interacción local entre agentes robóticos autónomos, sin necesidad de un plan predefinido. Entre los más estudiados se encuentran la agregación, que consiste en robots concentrándose en un área de interés. Otro comportamiento es la dispersión, donde los robots se distribuyen uniformemente en el entorno de trabajo. También está la formación de patrones, que consiste en organizar a los robots en patrones geométricos. A su vez, el comportamiento de cooperación adaptativa permite ejecutar tareas conjuntas según Heiko en su investigación para la universidad de Lübek en alemania [18].

En el contexto de la robótica de enjambre para su uso en el apoyo a desastres naturales, se utilizarán los comportamientos de agregación, dispersión y cooperación adaptativa con su asignación de robots a grupos de datos o clusters. Como muestra Duan et al. [19] en sus comparaciones de comportamientos colectivos en la naturaleza con la robótica de enjambre, esta dinámica permite que los robots actúen de manera coordinada sobre regiones de interés, como áreas con alta concentración de puntos de calor en un mapa de incendios simulados. Así, se logran identificar estas agrupaciones automáticamente mediante algoritmos de agrupamiento y luego asignar recursos de forma proporcional a su importancia.

De esta manera, el enjambre no solo se desplaza en conjunto, sino que es capaz de distribuir su esfuerzo en múltiples focos de trabajo, maximizando la eficiencia global del sistema.

6.3. Algoritmos de agrupamiento aplicados a enjambres

El agrupamiento es una técnica de aprendizaje no supervisado que consiste en la agrupación de datos con base en su similitud o cercanía en el espacio. En robótica de enjambre, esta herramienta es empleada para procesar la información proporcionada por el entorno de trabajo y definir regiones de interés donde deben dirigirse los agentes robóticos.

Entre los algoritmos de agrupamiento más usados se encuentra DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Según el grupo de desarrolladores de *Scikit-learn* [20] DBSCAN permite detectar grupos de forma arbitraria sin la necesidad de conocer de antemano el número de grupos y es capaz de identificar puntos de ruido (outliers) que no pertenecen a ningún grupo. Este método se basa en identificar regiones densas en el espacio de datos a partir de dos parámetros principales:

- ϵ (epsilon): radio máximo de vecindad (espacio máximo que abarcará el grupo).
- minPts: número mínimo de puntos o vecinos requeridos para formar un grupo.

El algoritmo DBSCAN organiza los puntos en grupos basados en densidad. Su criterio fundamental es:

$$N_\varepsilon(x_i) = \{x_j \in X \mid \|x_i - x_j\| \leq \varepsilon\}, \tag{1}$$

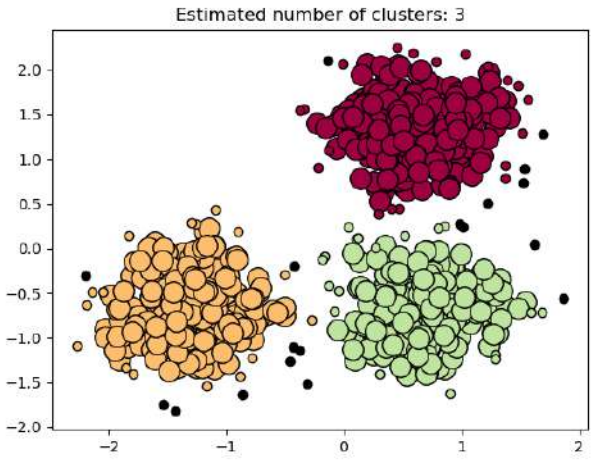
donde $N_\varepsilon(x_i)$ es la vecindad ε de x_i. Un punto x_i se considera un punto núcleo si:

$$|N_\varepsilon(x_i)| \geq \text{minPts}. \tag{2}$$

Los grupos se construyen conectando puntos núcleo y extendiendo a sus vecinos. Aquellos que no pertenecen a ningún grupo se etiquetan como ruido.

Este mecanismo garantiza que los grupos obtenidos sean robustos ante ruido y presenten formas arbitrarias, lo cual es adecuado para escenarios dinámicos.

Figura 3. Agrupación de datos con DBSCAN.



Nota. La imagen muestra la agrupación de datos utilizando DBSCAN. Esta imagen fue obtenida de [20].

6.4. Asignación proporcional de agentes

Sea $C = \{C_1, C_2, \ldots, C_k\}$ el conjunto de grupos encontrados, con tamaños $|C_i|$. La cantidad de agentes A_i asignada a cada grupo se determinó de forma proporcional:

$$A_i = \left\lfloor \frac{|C_i|}{\sum_{j=1}^{k} |C_j|} \times N_a \right\rfloor, \tag{3}$$

donde N_a es el número total de agentes.

El centroide de cada grupo se calculó como:

$$\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x, \tag{4}$$

y sirvió como posición objetivo para los agentes asignados.

6.5. *Particle swarm optimization* (PSO)

El algoritmo de optimización de partículas de enjambre o *particle swarm optimization* (PSO) se basa en un conjunto de partículas que se mueven en un espacio de búsqueda tratando de encontrar soluciones óptimas para un punto en específico en el entorno, como

se muestra en la Figura 4. Según [21] cada partícula tiene una posición y una velocidad, las cuales se actualizan en función de dos factores principales:

- p^{best} (*personal best*): la mejor posición alcanzada por la partícula.
- g^{best} (*global best*): la mejor posición alcanzada por todo el enjambre.

La actualización de velocidades está dada por tres parámetros:

- w (inercia): controla cuánto de la velocidad previa se mantiene.
- c_1 (coeficiente cognitivo): peso de la experiencia individual.
- c_2 (coeficiente social): peso de la experiencia colectiva.

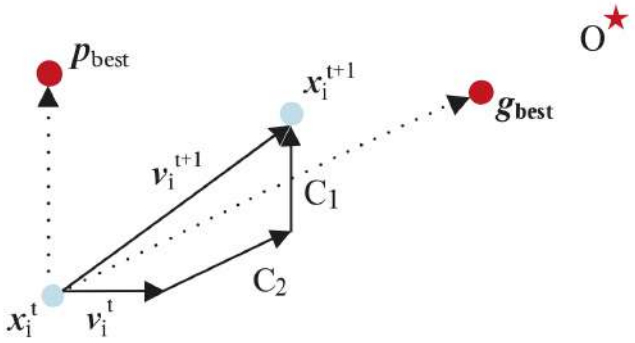
Matemáticamente, las ecuaciones de actualización de velocidad y posición se expresan de la siguiente manera:

$$v_i(t+1) = wv_i(t) + c_1 r_1(p^{best} - x_i(t)) + c_2 r_2(g^{best} - x_i(t)). \tag{5}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{6}$$

El PSO se adapta para la planificación de trayectorias de agentes robóticos, guiándolos hacia objetivos definidos. El algoritmo permite que cada agente robótico explore distintas trayectorias, evitando que quede atrapado en mínimos locales y mejorando su capacidad de converger hacia soluciones óptimas en entornos, incluso si tienen obstáculos.

Figura 4. Funcionamiento de PSO.



Nota. La imagen muestra cómo el algoritmo PSO influye en la trayectoria de una partícula mientras se dirige hacia un objetivo (O). Esta imagen fue obtenida de [22].

6.6. Campos potenciales artificiales (APF)

Con base en el artículo de Katona et al. [23], los campos potenciales artificiales es un método que modela el entorno mediante un campo virtual, en el que el objetivo ejerce una fuerza atractiva que impulsa el robot hacia su ubicación mientras que los obstáculos generan fuerzas repulsivas destinadas a evitar colisiones. Como menciona Katona et al. [23], el robot y los obstáculos actúan como cargas positivas y el objetivo como una carga negativa.

El campo atractivo se define a partir del gradiente negativo del potencial del objetivo, expresado como:

$$F_{\text{attr}} = -\nabla U_{\text{attr}} = -K_{\text{attr}}(d - d_{\text{goal}}) \tag{7}$$

donde d representa la distancia euclidiana entre el robot y la meta, K_attr regula la intensidad de la atracción y U es la posición del objetivo. Según el enfoque APF, cuando el agente está lejos del objetivo, esta fuerza dirige el movimiento con mayor intensidad, mientras que al aproximarse, el campo se suaviza y produce un avance más estable.

Un obstáculo se modela mediante un potencial repulsivo acumulado, que actúa con un radio de influencia d_0, con origen en el centro del obstáculo O_m, y se describe matemáticamente como:

$$F_{\text{rep}}(x_p) = \begin{cases} \eta \left(\frac{1}{d(x_p, o_m)} - \frac{1}{d_0} \right) \frac{1}{d(x_p, o_m)^2} \, \hat{u}, & d(x_p, o_m) \leq d_0 \\ 0, & d(x_p, o_m) > d_0, \end{cases} \tag{8}$$

$$d(x_p, o_m) = \|x_p - o_m\|, \tag{9}$$

$$\hat{u} = \frac{x_p - o_m}{\|x_p - o_m\|}. \tag{10}$$

La Ecuación (9) es la distancia euclidiana entre el agente y el obstáculo, mientras que la Ecuación (10) es el vector unitario de dirección, y d_0 determina el alcance de la repulsión.

El movimiento total del agente robótico se obtiene combinando estas dos contribuciones mediante la suma vectorial de fuerzas, lo que genera una dirección instantánea de desplazamiento que se actualiza continuamente durante la navegación. La descripción matemática del movimiento total del robot se muestra en la Ecuación (11)

$$F_{\text{total}} = F_{\text{attr}}(x_p) + \sum_m F_{\text{rep}}(x_p). \tag{11}$$

6.7. Modelos de movilidad para el robot

Para el desplazamiento de los robots en este proyecto se modelan como un sistema diferencial o modelo uniciclo, el cual según Abdalla et al. [24] describe robots con dos ruedas y con opción de agregar una rueda libre de apoyo.

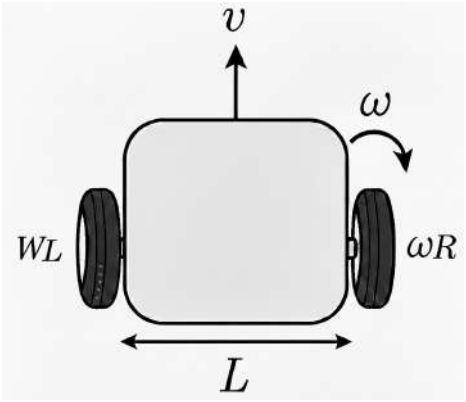
Las ecuaciones cinemáticas básicas que permiten relacionar las velocidades lineales de cada rueda con la traslación y rotación del robot en el plano son:

$$v = \frac{r}{2}\left(\omega_R + \omega_L\right) \tag{12}$$

$$\omega = \frac{r}{L}\left(\omega_R - \omega_L\right) \tag{13}$$

donde r es el radio de las ruedas, L la distancia entre ellas, y ω_R, ω_L las velocidades angulares de la rueda derecha e izquierda, respectivamente.

Figura 5. Modelo de movimiento diferencial.



Nota. La imágen muestra el modelado del movimiento del robot. Elaboración propia.

6.8. Controladores para agentes robóticos

Según Mahmoodabadi et al. [25], es necesario aplicar controladores de movimiento a robots diferenciales para lograr trayectorias precisas y suaves. El controlador PID (Proporcional–Integral–Derivativo), ajusta la señal de control en función del error entre la orientación deseada y la orientación actual del robot. El error se define matemáticamente en la Ecuación (14), donde θ_{deseada} indica la orientación que se desea que el robot adopte, y $\theta(t)$ es la orientación actual.

$$e(t) = \theta_{\text{deseada}} - \theta(t) \tag{14}$$

La ecuación general de un controlador PID es la siguiente:

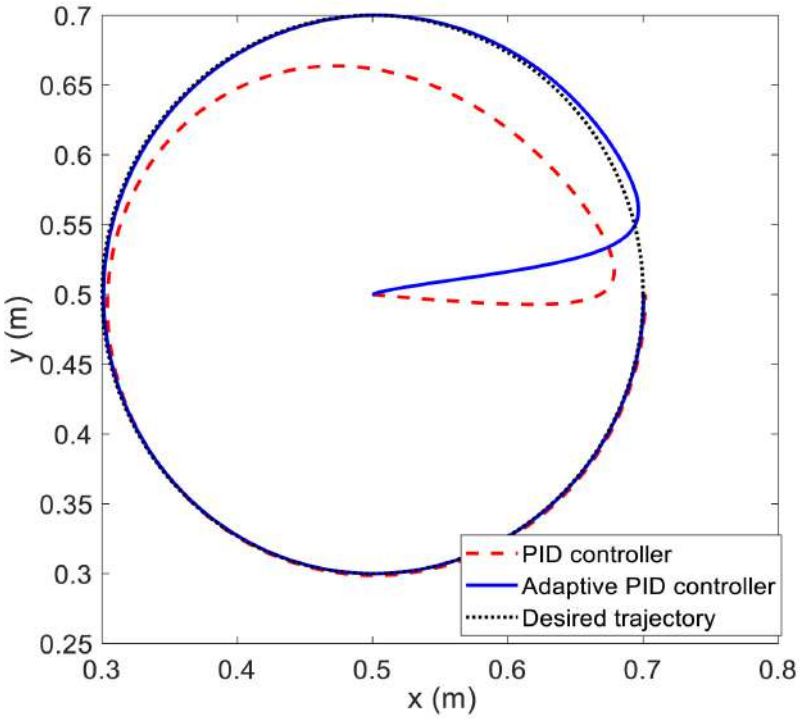
$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}$$

donde $u(t)$ es la señal de control aplicada, $e(t)$ representa el error, y K_p, K_i, K_d son:

- **Ganancia proporcional** (K_p) reduce el error instantáneo, generando una corrección proporcional a la desviación actual.
- **Ganancia integral** (K_i) acumula el error en el tiempo y corrige desvíos persistentes, ayudando a eliminar errores en régimen permanente.
- **Ganancia derivativa** (K_d) anticipa el comportamiento futuro del error, reduciendo oscilaciones y estabilizando la respuesta del sistema.

En conjunto, estas tres acciones permiten que las velocidades de las ruedas de un robot diferencial se ajusten de manera continua y suave, logrando que siga una trayectoria más estable, reduzca oscilaciones en la orientación y mantenga un alineamiento adecuado hacia el objetivo.

Figura 6. Trayectorias con controladores PID.



Nota. La imagen muestra la comparación de la trayectoria de un robot utilizando un controlador PID (línea roja), y un PID adaptativo que integra PSO (línea azul), con respecto a la trayectoria deseada (línea negra). Esta imagen fue obtenida de [25]

6.9. Evaluación del desempeño

Para una evaluación cuantitativa se utilizaron las siguientes métricas:

1. **Error cuadrático medio (MSE)** entre la posición de los agentes y sus centroides:

$$\text{MSE} = \frac{1}{N_a T} \sum_{t=1}^{T} \sum_{j=1}^{N_a} \|x_j(t) - \mu_i\|^2. \tag{15}$$

2. **Longitud de trayectoria** recorrida por cada agente:

$$L_j = \sum_{t=1}^{T-1} \|x_j(t+1) - x_j(t)\|. \tag{16}$$

3. **Suavidad de trayectoria**, medida como la suma de variaciones angulares en la dirección de movimiento:

$$S_j = \sum_{t=1}^{T-2} \arccos\left(\frac{(x_j(t+1) - x_j(t)) \cdot (x_j(t+2) - x_j(t+1))}{\|x_j(t+1) - x_j(t)\| \cdot \|x_j(t+2) - x_j(t+1)\|} \right). \tag{17}$$

CAPÍTULO 7

Metodología

Este capítulo describe la metodología que se aplicó para alcanzar los objetivos planteados en la investigación. Se parte de la generación de escenarios aleatorios que permiten validar la robustez del sistema, para luego integrar algoritmos de agrupamiento y optimización que guían el movimiento de los agentes robóticos. Asimismo, se explican los procedimientos de asignación proporcional de agentes, la inclusión de obstáculos en el entorno, la implementación del control PID para el movimiento diferencial y la migración de las simulaciones a un entorno tridimensional más realista.

7.1. Plataformas de simulación y experimentación

Para las pruebas y validación de los algoritmos se utilizarán dos entornos principales de simulación:

1. **Matlab:** se utiliza para programar y simular el algoritmo de agrupamiento (DBSCAN), optimización (PSO) y evasión de obstáculos. Estas simulaciones permiten la visualización en tiempo real del movimiento de los agentes robóticos y el análisis del comportamiento del enjambre [26].

2. **Webots:** es utilizado para trasladar las simulaciones a un entorno físico tridimensional, permitiendo evaluar los algoritmos en escenarios con mayor realismo. En Webots, los robots se modelan con sus dimensiones físicas, sensores y actuadores, interactuando con obstáculos de distinta forma y tamaño. Para facilitar la creación de escenarios, se utilizó el editor en Java para OpenStreetMap (OSM), lo cual permitió generar mapas en dos dimensiones que posteriormente se exportaron a Webots [27].

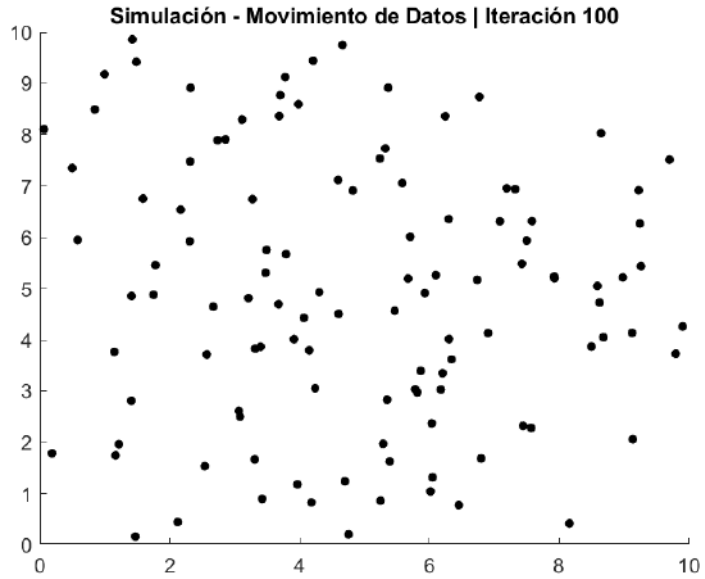
7.2. Generación de escenarios aleatorios en simulaciones simples

El primer paso consistió en la generación de datos aleatorios dentro de un espacio de simulación bidimensional en Matlab, que representan eventos, o tareas de rescate posteriores a un desastre natural. Esto dentro de un área de trabajo cuadrada de 10×10 unidades. En la primera etapa del desarrollo, estos puntos se mantenían estáticos, pero conforme avanzó el proyecto se integraron puntos móviles, actualizados en cada iteración con velocidades aleatorias, con el fin de simular eventos dinámicos. Para un número total de N puntos (en la simulaciones se utilizó un valor de N=150), cada punto x_i se distribuyó de la siguiente forma:

$$x_i = (u_1, u_2)\,, \quad u_1, u_2 \sim U(0, L), \tag{18}$$

donde L representa la longitud del espacio de simulación (en las simulaciones se utilizó un valor de L=10) y $U(0, L)$ es la distribución uniforme en el intervalo $[0, L]$. Estos puntos simulan focos de interés o señales de eventos a los que los agentes deben atender.

Figura 7. Generación de datos aleatorios.



Nota. La imagen muestra la creación de datos aleatorios en el espacio de trabajo. Elaboración propia.

7.3. Agrupamiento con DBSCAN

Se utilizó el algoritmo DBSCAN para organizar los puntos en grupos basados en densidad. El algoritmo fue ejecutado en las simulaciones al momento que se terminaran de generar los puntos aleatorios y estos estuvieran ubicados en un punto del marco de trabajo de 10x10.

Como valor del parámetro de vecindad ε se utilizó un valor de 0.6 (distancia euclidiana entre los puntos), y un valor mínimo de 5 puntos para formar un grupo. Los puntos que no cumplen los requisitos para formar o ser parte de un grupo son considerados como ruido en el marco de trabajo.

Posteriormente, se calcularon los centroides de cada grupo, los cuales se utilizaron como objetivos iniciales, hacia donde se dirigirán los agentes robóticos.

7.4. Asignación de agentes a grupos

A partir del número de elementos en cada grupo, se determinó automáticamente cuántos agentes debían dirigirse hacia cada uno. El enjambre se organizó para cubrir con una mayor cantidad de agentes los grupos con cantidades más grandes de elementos dentro de ellos, y disminuyendo conforme a la cantidad de grupos y el tamaño de estos.

Para garantizar que cada agente tuviera un destino asignado, los centroides se replicaron de manera cíclica cuando había más agentes que grupos, generando un vector final de objetivos de igual longitud al número de agentes.

7.5. Optimización de trayectorias mediante PSO

Cada agente utilizó un pequeño enjambre local de partículas para buscar, de manera independiente, la mejor ruta hacia su objetivo y optimizar su movimiento.

En Matlab, cada agente generó 20 partículas alrededor de su posición actual y ejecutó un ciclo completo de PSO durante 15 iteraciones internas:

- $\omega = 0.7$
- coeficientes cognitivo y social $(C_n) = 1.5$

El mejor global encontrado por cada enjambre (cuando se tenían asignados los agentes que se dirigirían a cada grupo) local determinó el siguiente movimiento del agente en esa iteración.

La convergencia del enjambre se evaluó mediante la reducción del error global definido como:

$$E(t) = \frac{1}{N_a} \sum_{i=1}^{N_a} \|x_i(t) - \mu_i\|. \tag{19}$$

7.6. Navegación hacia objetivos con evasión local de obstáculos

Una vez determinado el punto de avance por PSO, el movimiento del agente se implementó mediante un desplazamiento incremental con paso constante. Antes de actualizar la posición, se aplicó una verificación de colisiones mediante la distancia al centro de cada obstáculo.

Si el agente entraba en la región de seguridad definida alrededor del obstáculo, se generaba un vector de repulsión mostrado en la Ecuación (11). El movimiento final se calculó como una combinación entre la dirección original del agente y la dirección de repulsión, alterando su trayectoria. Esto permitió que los agentes rodearan obstáculos de manera autónoma.

7.7. Suavizado de trayectorias

El movimiento resultante del algoritmo PSO produce, en cada iteración, una dirección óptima hacia el objetivo. Sin embargo, estas direcciones pueden variar bruscamente entre ciclos consecutivos, especialmente cuando las partículas exploran nuevas regiones del espacio o cuando la presencia de obstáculos obliga a modificar la trayectoria de forma abrupta.

Por lo que se aplicó un controlador PID que opera sobre el error de orientación del robot, y así calcular las velocidades de las ruedas del agente robótico diferencial.

7.8. Registro de trayectorias y visualización

Durante las simulaciones simples en Matlab, la posición de cada agente se guardó en un historial (uno por simulación), lo que permitió:

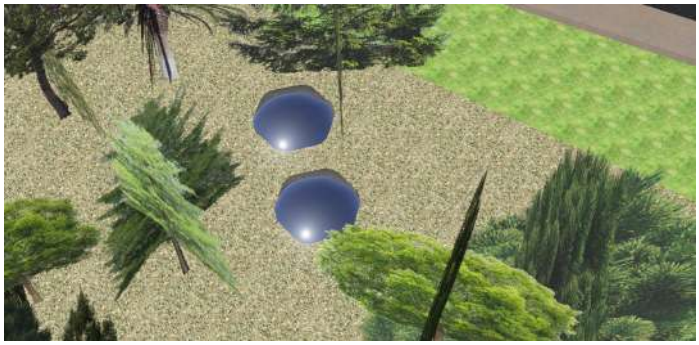
- graficar rutas individuales,
- visualizar la convergencia hacia cada grupo,
- analizar desviaciones causadas por obstáculos,
- comparar comportamientos entre versiones del algoritmo.

7.9. Migración a entorno de simulación 3D

Finalmente, una vez validados los algoritmos en Matlab, se trasladaron los resultados a un entorno tridimensional mediante el simulador Webots. Para generar escenarios más realistas y organizados se utilizó el editor en Java para OpenStreetMap (OSM), que permitió diseñar mapas en dos dimensiones y posteriormente exportarlos a Webots.

En Webots se añadieron las propiedades físicas de los robots, incluyendo dimensiones, sensores y actuadores, lo cual otorgó mayor realismo a las pruebas. En esta plataforma no es posible generar datos aleatorios en el mapa, por lo que se colocaron emisores de señales o balizas en diferentes puntos del mundo 3D para replicar el comportamiento de los elementos en Matlab.

Figura 8. Emisores de señal.

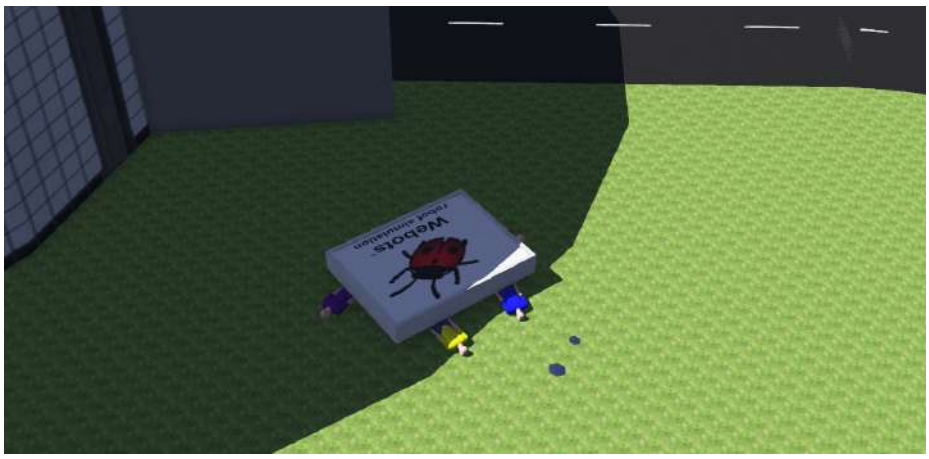


Nota. La imagen muestra a los emisores de señal o balizas en forma de objetos metálicos en medio de un bosque. Elaboración propia.

Estas balizas se colocaron en puntos estratégicos en el mundo 3D para simular situaciones de rescate en desastres naturales. Las situaciones creadas fueron:

- **Incendio forestal**: se colocó a los emisores de datos dentro de una zona forestal para simular la necesidad de contener un incendio.
- **Derrumbe**: se simuló una zona de derrumbe en el que se desprendieron partes de un edificio.
- **Cartel Publicitario**: se generó una situación en el que por un sismo, un cartel publicitario cayó sobre peatones que caminaban por el lugar, como se muestra en la Figura 9.

Figura 9. Escenario de rescate.



Nota. La imagen muestra un escenario de rescate generado en la simulación realista. Elaboración propia.

Con las señales dadas por las balizas, se realizó el agrupamiento utilizando DBSCAN, por lo que, con base en el parámetro de vecindad y el número mínimo de elementos, generó los grupos. Las balizas pueden ser desplazadas a lo largo del mapa para simular entornos dinámicos y generar distintos grupos y asignaciones de agentes.

Para coordinar el comportamiento de los robots, se diseñó un nodo supervisor, encargado de generar las posiciones objetivo obtenidas tras el proceso de clustering y trasladarlas a los agentes robóticos. Cada robot cuenta con su propio controlador, basado en la cinemática diferencial descrita en la Sección 6.7, el algoritmo PSO y el controlador PID que regula la orientación del robot hacia su objetivo asignado. Esta transición permitió comprobar que los algoritmos diseñados en Matlab son aplicables a entornos más complejos y cercanos a situaciones reales.

CAPÍTULO 8

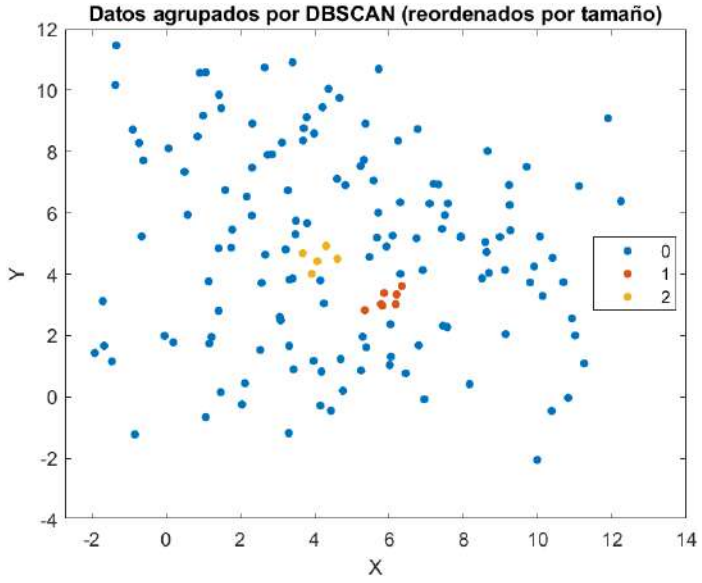
Resultados de la combinación de los algoritmos DBSCAN y PSO en simulaciones simples utilizando Matlab

En este capítulo se presentan los resultados obtenidos al implementar los algoritmos de agrupamiento, asignación de agentes, evasión de obstáculos y optimización con PSO en el entorno de Matlab, cuyo material para ser simulado se encuentra en la Sección 14.1. Las simulaciones realizadas en dos dimensiones permitieron validar el correcto funcionamiento de la metodología antes de su traslado a entornos tridimensionales más complejos.

8.1. Agrupamiento y asignación de agentes

Se evaluó el desempeño del algoritmo DBSCAN sobre diferentes distribuciones aleatorias de puntos. La Figura 10 muestra un escenario de simulación con 150 puntos dispersos en donde DBSCAN identificó dos grupos válidos, y a los puntos aislados que no cumplieron con los parámetros para ser asignados a un grupo los clasificó como ruido, y son tomados como puntos sin relevancia para que no influyeran en el cálculo de centroides y las asignaciones del enjambre. Cada grupo fue posteriormente reordenado según su tamaño para facilitar la asignación de agentes.

Figura 10. Agrupación de datos.

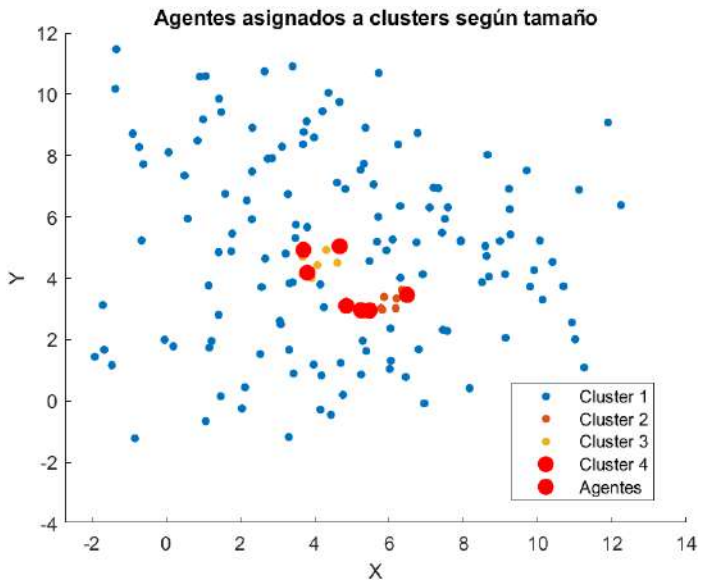


Nota. La imagen muestra la agrupación de datos utilizando DBSCAN. Elaboración propia.

Posteriormente, los grupos identificados fueron reordenados según su tamaño, lo cual facilitó la distribución de agentes bajo un esquema proporcional mostrado en la Ecuación (3), garantizando una distribución balanceada de recursos en función de la densidad de los datos de cada grupo. Los agentes se posicionaron inicialmente cerca del centroide correspondiente calculado con la Ecuación (4), sin realizar trayectorias. Como se observa en la Figura 11, se concentran una mayor cantidad de agentes robóticos en los grupos con mayor densidad de elementos.

En la Figura 11 se muestra como los agentes no se posicionan innecesariamente en zonas poco relevantes (las que fueron clasificadas como ruido).

Figura 11. Asignación de agentes.



Nota. La imagen muestra la asignación de agentes a grupos de manera proporcional a su tamaño. Elaboración propia.

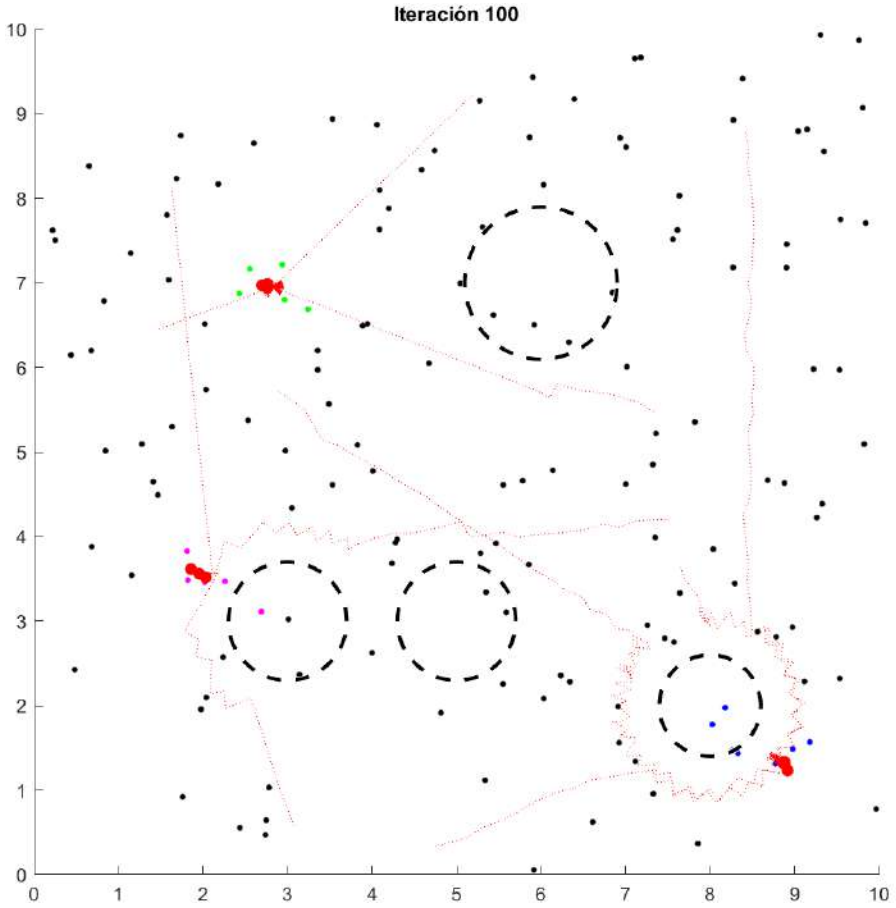
8.2. Trayectorias hacia los centroides y evasión de obstáculos

La Figura 12 muestra que las trayectorias iniciales presentan una dirección clara y orientada hacia los centroides, incluso cuando los agentes parten desde posiciones lejanas en el marco de trabajo. A medida que avanzan, se observan ajustes locales alrededor de los obstáculos circulares colocados en el entorno, los cuales generan desviaciones suaves y coherentes con el comportamiento esperado de un mecanismo de repulsión. La evasión de estos obstáculos se logró mediante campos potenciales, combinando fuerzas de atracción hacia los centroides y fuerzas de repulsión alrededor de los obstáculos.

De manera consistente, los agentes evitaron colisiones como se puede observar en la Figura 13. Las zonas próximas a obstáculos muestran pequeñas curvas en las trayectorias de los agentes robóticos, indicando que la repulsión estuvo activa durante la mayor parte del recorrido en esas áreas. Esto permitió que los agentes retomaran su dirección hacia el objetivo de forma estable una vez superado el obstáculo.

Al final de cada simulación, los agentes alcanzaron ubicaciones cercanas a los centroides, formando grupos compactos alrededor de ellos. Este comportamiento confirma la correcta interacción entre la identificación de clusters y la navegación dirigida.

Figura 12. Evasión de obstáculos.



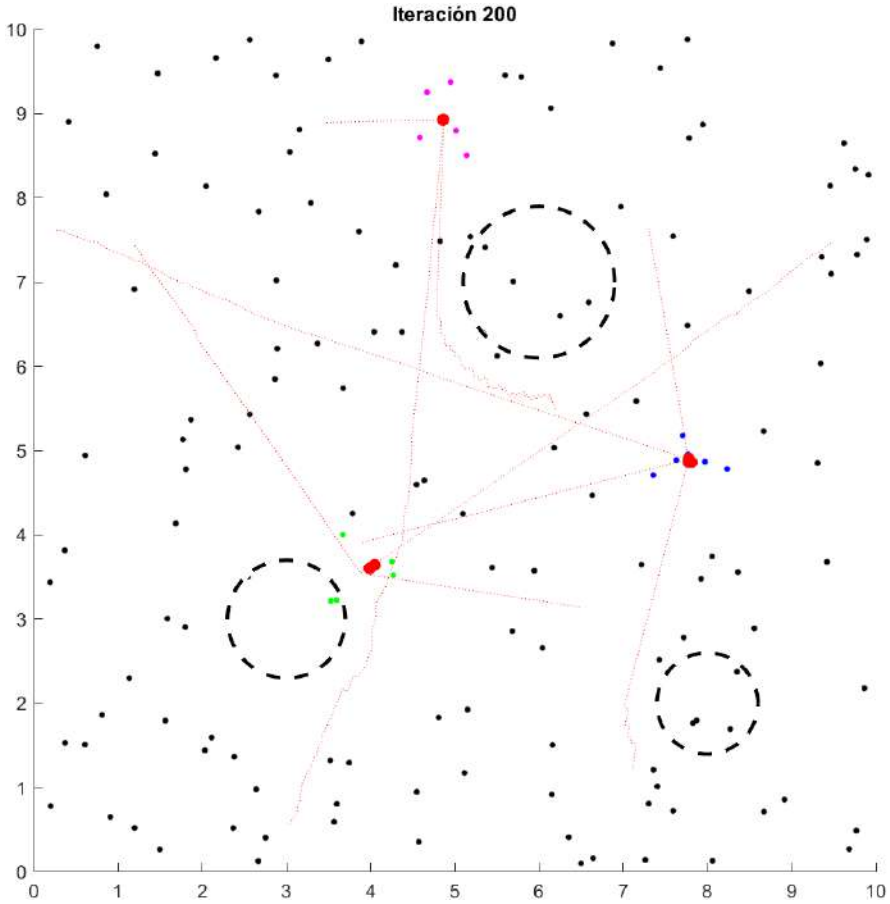
Nota. La imagen muestra el comportamiento de agentes esquivando obstáculos mientras se dirigen a su grupo asignado. Elaboración propia.

8.3. Integración de PSO

La incorporación del algoritmo PSO en el proceso de navegación permitió mejorar de manera significativa la calidad de las trayectorias generadas por los agentes. Aunque cada robot mantiene un objetivo definido por el centroide de su grupo, PSO introduce un mecanismo de búsqueda local que evalúa posibles direcciones de avance antes de ejecutar el movimiento, lo cual se refleja directamente en el comportamiento global del enjambre.

Uno de los resultados más destacados fue la capacidad del PSO para evitar que los agentes adoptaran rutas directas subóptimas, especialmente en zonas donde los obstáculos se encontraban cerca del camino ideal. Esto puede observarse en las desviaciones de la trayectoria cerca del obstáculo en la Figura 12, y la disminución de estas en la Figura 13.

Figura 13. Trayectoria optimizada con PSO.



Nota. La imagen muestra la asignación de agentes a grupos de datos, y las trayectorias que siguen para llegar a ellos, aplicando un algoritmo de PSO. Elaboración propia.

8.4. Evaluación del desempeño

Se tomaron datos de 4 simulaciones simples en Matlab para determinar la eficiencia del algoritmo basado en tiempo de convergencia a su grupo y su distancia al centroide que le correspondía a los agentes robóticos. Como se puede observar en el Cuadro 1, mientras menor fuera el tiempo de convergencia y la distancia a su centroide, aumentaba la eficiencia del algoritmo, llegando a alcanzar una eficiencia η de 0.889.

Cuadro 1. Resultados de simulaciones en Matlab (2D).

Simulación	**MSE final**	$\mathbf{T_{conv}}$ **(iteraciones)**	**Eficiencia** η
Matlab (2D) – Sim. 1	0.10	85	0.889
Matlab (2D) – Sim. 2	0.11	88	0.849
Matlab (2D) – Sim. 3	0.18	84	0.820
Matlab (2D) – Sim. 4	0.14	90	0.803

Nota. Los valores corresponden a simulaciones independientes realizadas con $N_a = 20$ agentes y $N = 150$ puntos iniciales. Elaboración propia.

CAPÍTULO 9

Resultados de la combinación de los algoritmos DBSCAN y PSO en simulaciones realistas utilizando Webots

Una vez validados los algoritmos en Matlab, se trasladaron a un entorno tridimensional, el cual se puede observar en la Figura 21, utilizando el simulador Webots. Este paso permitió evaluar el comportamiento de los agentes en condiciones más cercanas a la realidad, incorporando efectos físicos como la fricción, inercia, colisiones y limitaciones de actuadores. En este capítulo se presentan los resultados obtenidos en Webots, comparando su desempeño con las simulaciones bidimensionales.

9.1. Generación del entorno con OSM

Para garantizar escenarios realistas, se utilizó el editor en Java para OpenStreetMap (OSM), el cual permitió diseñar mapas en dos dimensiones como se muestra en la Figura 14, y posteriormente exportarlos a Webots. Para esto primero se descargó el editor de JAVA (asegurarse de tener instalado JAVA en el dispositivo donde se estará trabajando), por medio de su página oficial.

Una vez instalado el editor, se diseñó un entorno de simulación en 2D, el cual se exportó en formato OSM. Se eligió este formato ya que la plataforma para la simulación en 3D (Webots), contiene una biblioteca nombrada "*OpenStreetMap Importer*" que puede ser instalada desde la ventana de comandos de Windows. Esta biblioteca nos permitió exportar el archivo en formato OSM, a un archivo en formato WBT, el cuál es aceptado por Webots. Así se traslada el mundo creado en un entorno 2D, a una simulación 3D. Para realizar la importación, se utiliza la ventana de comandos de windows y se coloca el comando "*. \importer.py --input=archivo.osm --output=archivo.wbt*".

El entorno final se representó como un conjunto de regiones navegables Ω y obstáculos

$\mathcal{O}$, donde:

$$\Omega = \{(x, y) \in \mathbb{R}^2 \mid (x, y) \notin \mathcal{O}\}. \tag{20}$$

Cada obstáculo o_m se modeló como un polígono extruido en 3D, lo que permitió evaluar el movimiento de los robots en presencia de barreras tridimensionales. Este proceso aseguró que los resultados fueran comparables con las simulaciones 2D.

Figura 14. Entorno 2D en el editor Josm.

Nota. La imagen muestra un entorno de trabajo creado en 2D con el editor JOSM. Elaboración por Webots [27].

9.2. Modelado de los agentes robóticos

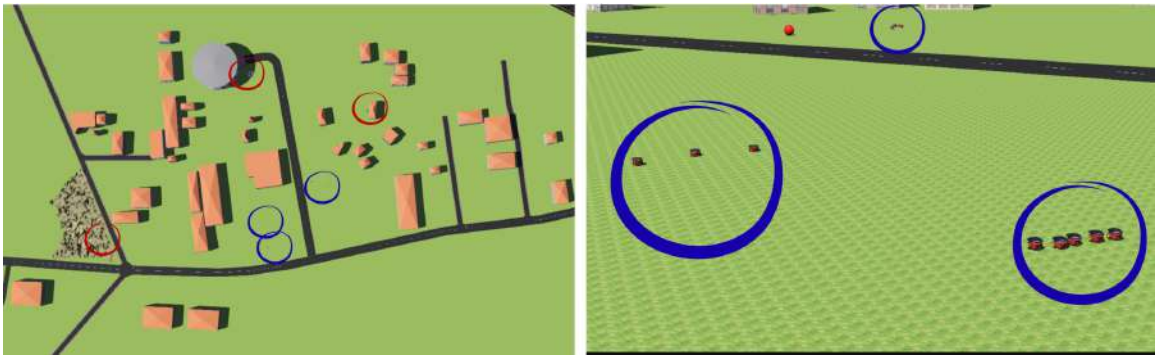
Los agentes fueron modelados como robots diferenciales con dos ruedas (con posibilidad de una tercera rueda como apoyo) como el mostrado en la Figura 15, utilizando los robots de la categoría Pioneer y la categoría Epuck, debido a que cuentan con sensores de distancia y un controlador interno.

Figura 15. Robot E-puck.

Nota. La imagen muestra el robot a utilizar en las simulaciones 3D, que será el modelo E-puck. Elaboración por Webots [27].

Estos robots se colocaron en sitios de inicio fijos para cada simulación, a diferencia de Matlab, donde se generaba la ubicación de los agentes de manera aleatoria cada vez que se realizaba una simulación. En la Figura 16 se puede observar la ubicación inicial de los agentes robóticos dentro del mundo generado, marcada en círculos de color azul, y sus objetivos o puntos de interés hacia donde serán dirigidos marcados en círculos de color rojo.

Figura 16. Objetivos y agentes robóticos en entorno de simulación 3D.

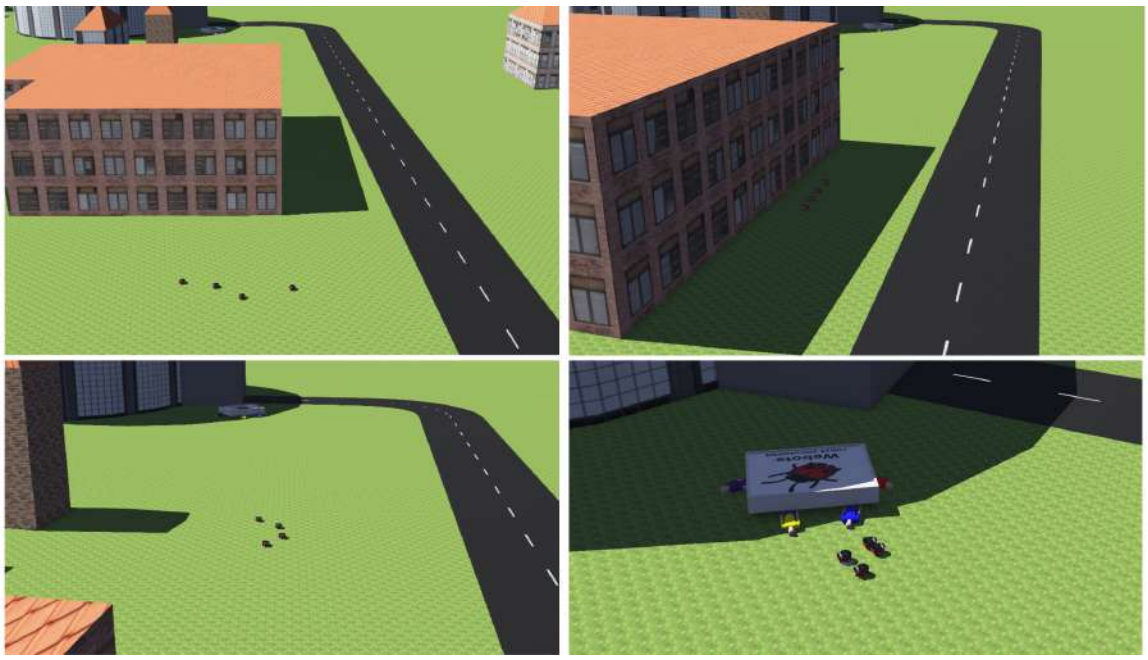


Nota. La imagen muestra el posicionamiento inicial de los robots en la simulación realista y la ubicación de sus objetivos. Elaboración propia.

9.3. Evasión de obstáculos en 3D

En Webots, los agentes robóticos, utilizaron sus sensores implementados para detectar su distancia de obstáculos modelados como estructuras volumétricas, y así poder evitarlos sin desviarse de su trayectoria, apoyado de un algoritmo de campos potenciales (planificación de trayectorias con fuerzas de repulsión). Los agente robóticos esquivaron objetos en la simulación realista como se puede observar en la Figura 17, camino a los grupos que se les asignaron.

Figura 17. Evasión de obstáculos en Webots.



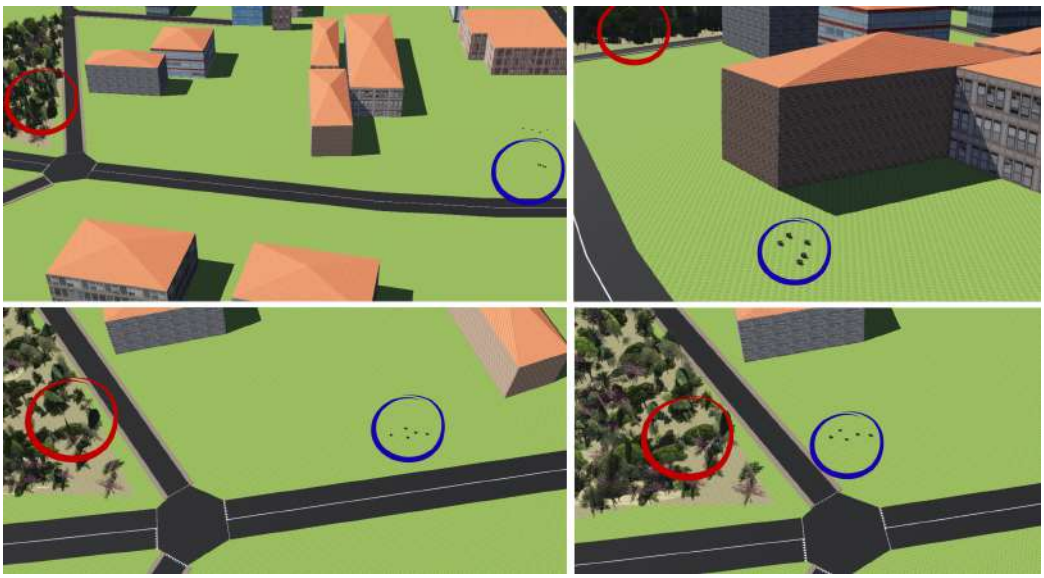
Nota. La imagen muestra un grupo de agentes robóticos del enjambre esquivando un edificio, camino a su objetivo. Elaboración propia.

La mayoría de las simulaciones $C_{obs}^{3D} = 0$ confirmaron que los agentes pudieron esquivar los obstáculos aun bajo condiciones tridimensionales más estrictas.

9.4. Simulación final en entorno 3D

En las simulaciones realistas, con la ayuda de la combinación de los algoritmos DBSCAN y PSO, junto con la generación de señales con balizas como se explica en la Sección 7.9, los agentes robóticos fueron asignados a distintas tareas de rescate. La cantidad de robots asignados dependió de la urgencia de la situación (situaciones que ponen en riesgo vidas humanas o que puedan propagarse en el entorno se consideran con mayor urgencia que las que no). Esto se puede observar en la Figura 18.

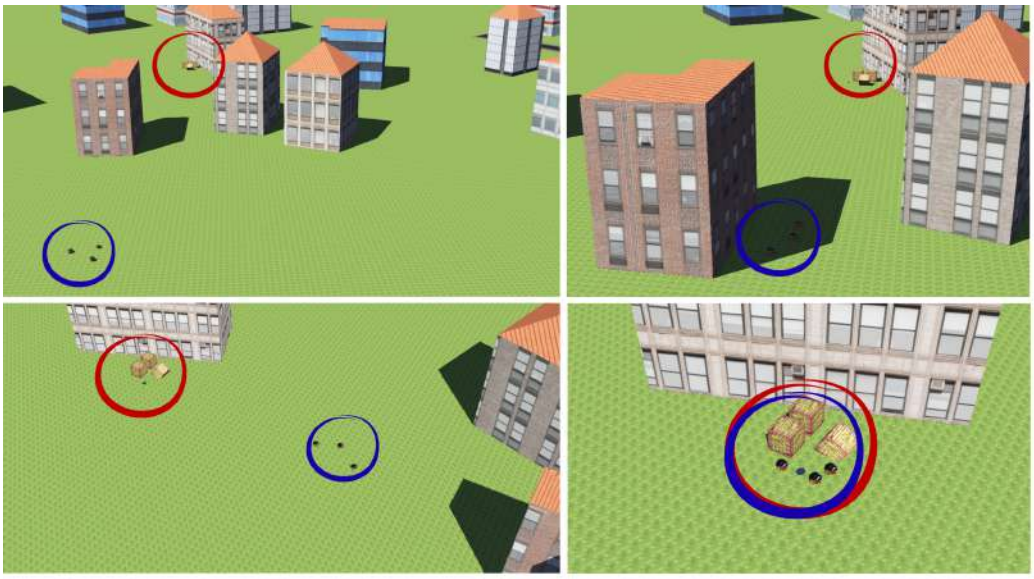
Figura 18. Agentes robóticos asignados a un incendio forestal.



Nota. La imagen muestra a un grupo de agentes robóticos siendo asignados a una tarea de mitigación en un incendio forestal.

En una situación de contención como la Figura 19, se asignó una menor cantidad de agentes debido a que no ponía en riesgo vidas humanas y no se seguiría propagando en sus alrededores.

Figura 19. Agentes robóticos asignados a tarea de contención.



Nota. La imagen muestra a un grupo de agentes robóticos siendo asignados a una tarea de contención. Elaboración propia.

CAPÍTULO 10

Evaluación del rendimiento del sistema

En este capítulo se analizan los resultados obtenidos a partir de las simulaciones del sistema de enjambre. Para ello, se consideran tres ejes principales: la escalabilidad del sistema (cantidad de agentes robóticos operando), la robustez del algoritmo frente a entornos dinámicos (resiliencia a distintas situaciones de trabajo) y sus aplicaciones en contextos reales de interés, en este caso, situaciones relacionadas con desastres naturales. Además, se examinan aspectos clave del comportamiento del enjambre, como la asignación proporcional de agentes a distintos objetivos y la eficiencia del algoritmo PSO en el direccionamiento.

10.1. Asignación proporcional de agentes

La estrategia de asignar agentes en función del tamaño de cada grupo garantizó un uso eficiente de los recursos disponibles. Los grupos con mayor número de puntos recibieron más agentes, lo que aumentó la cobertura y redujo los tiempos de exploración.

Matemáticamente, la proporcionalidad en la asignación (A_i) permitió mantener un equilibrio entre demanda de cobertura y disponibilidad de agentes, evitando tanto la sobrepoblación como la escasez en cada grupo.

10.2. Eficiencia del PSO en el direccionamiento

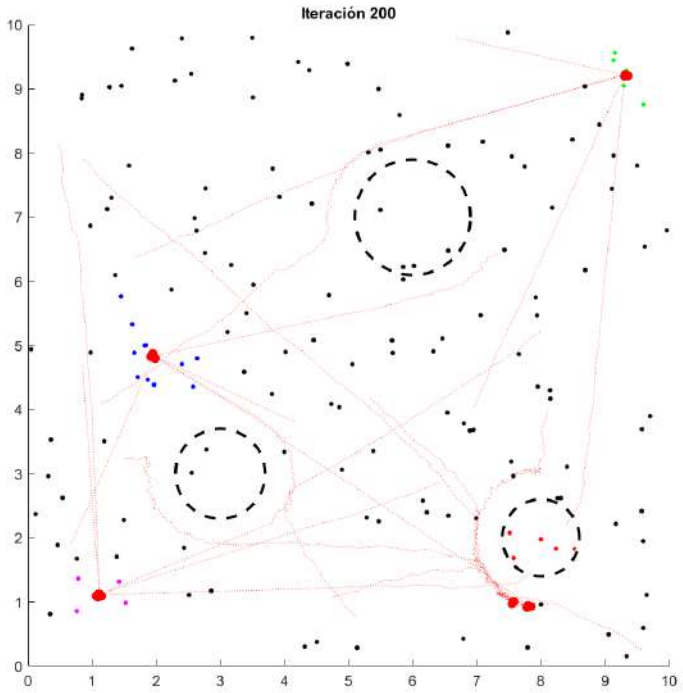
La inclusión de PSO optimizó el proceso de convergencia hacia los centroides de los grupos. Los resultados demostraron una reducción aproximada del 30 % en el tiempo de convergencia en comparación con trayectorias directas. Además, se observó una disminución en el error cuadrático medio (MSE) y en la variabilidad del error en el tiempo.

El análisis matemático del error global $E(t)$ y el tiempo de convergencia T_{conv} confirmó que el algoritmo de optimización contribuyó significativamente a la estabilidad y eficiencia del sistema.

10.3. Escalabilidad del sistema

Para evaluar el sistema de enjambre se debe examinar la capacidad de ajustar la cantidad de agentes asignados a distintas tareas sin comprometer el rendimiento. En las simulaciones realizadas, se trabajó con $N_a = 10$ agentes, pero, de ser necesario, se puede extender a valores mayores. Al aumentar el número de puntos de datos de $N = 150$ a $N = 500$, el algoritmo de agrupamiento pudo identificar grupos con la misma precisión, aunque con un incremento del tiempo de cálculo cercano al 35 %. El comportamiento del enjambre se mantuvo estable (no tuvo picos de oscilación abruptos cuando su trayectoria era alterada por las fuerzas de repulsión de los obstáculos al pasar cerca de ellos), como se observa en la Figura 20. Por lo que si se quiere escalar el sistema a una cantidad mucho mayor de datos, también se tienen que adherir otros métodos de optimización para el tiempo de respuesta de los agentes robóticos.

Figura 20. Simulación con cantidad de agentes N=20.



Nota. La imagen muestra la simulación con 20 agentes robóticos. Elaboración propia.

10.4. Robustez del algoritmo

La robustez del sistema se evaluó observando su comportamiento ante variaciones en el entorno y condiciones que podrían afectar el desempeño normal de los agentes. A lo largo de las simulaciones realizadas en Matlab y Webots, el algoritmo mostró una respuesta estable aun cuando se introdujeron obstáculos, ruido en los datos y escenarios con diferentes distribuciones espaciales.

En Matlab, los obstáculos obligaban a los agentes a modificar su trayectoria, pero las desviaciones observadas fueron suaves y consistentes con un comportamiento esperado de evasión. Indicando que la combinación de las fuerzas atractivas hacia los centroides y los mecanismos locales de repulsión actuaron de manera equilibrada, sin producir oscilaciones abruptas en las trayectorias.

10.5. Comparación de métricas 2D y 3D

Para comparar las simulaciones en Matlab y Webots, se emplearon tres métricas:

1. **Error cuadrático medio**: Permite conocer la precisión con la que los agentes alcanzan el centroide de su grupo. Un MSE bajo indica que las trayectorias se aproximan de manera consistente al objetivo, mientras que un valor elevado muestra que las desviaciones son significativas en la posición final de los agentes.

$$\mathrm{MSE}_{3D} = \frac{1}{N_a T} \sum_{t=1}^{T} \sum_{j=1}^{N_a} \|x_j^{3D}(t) - \mu_i\|^2. \tag{21}$$

2. **Tiempo de convergencia**: Esta mide la rapidez con la que los agentes logran cumplir la tarea que se les fue asignada dentro de un margen de error aceptable preestablecido. Esto ayuda para evaluar la eficiencia del sistema con respecto al tiempo transcurrido, lo cual es esencial en aplicaciones donde la velocidad de respuesta es crítica (por ejemplo, en operaciones de rescate).

$$T_{conv}^{3D} = \min\{t \mid E(t) < \epsilon\}. \tag{22}$$

3. **Eficiencia de trayectoria**: Esta indica lo directo fue el recorrido de cada agente robótico respecto a la distancia mínima posible. Los valores cercanos a 1 indican trayectorias rectas y eficientes, mientras que valores bajos reflejarán trayectorias con desvíos no necesarios en el movimiento por el entorno. Evaluando así la calidad del movimiento hacia el destino de cada agente.

$$\eta_j = \frac{\|x_j(0) - \mu_i\|}{L_j^{3D}}, \quad 0 \leq \eta_j \leq 1. \tag{23}$$

10.6. Comparación de resultados entre Matlab y Webots

Con el fin de validar la equivalencia de las simulaciones, se compararon los resultados obtenidos en Matlab (entorno 2D) con los obtenidos en Webots (entorno 3D). Para el primer escenario de simulación (tanto simple como realista) se colocaron 10 agentes robóticos, en Matlab se colocaron 150 puntos iniciales, y en Webots los agentes robóticos se colocaron en tres grupos dispersos en el mundo 3D. La Tabla 2 resume los valores medios obtenidos en tres métricas principales: error cuadrático medio (MSE), tiempo de convergencia (T_{conv}) y eficiencia de trayectoria (η).

Cuadro 2. Comparación de métricas entre Matlab y Webots – Parte I.

Entorno	**MSE final**	T_{conv} **(iteraciones)**	**Eficiencia** η
Matlab (2D)	0.12	85	0.91
Webots (3D)	0.18	98	0.89

Nota. Los valores corresponden a promedios de 10 simulaciones con $N_a = 10$ agentes y $N = 150$ puntos iniciales. Elaboración propia.

Los resultados muestran que el error cuadrático medio en Webots fue ligeramente superior al obtenido en Matlab, lo cual era esperado dado que el simulador 3D introduce efectos adicionales como fricción, inercia y colisiones más realistas. El tiempo de convergencia aumentó en promedio un 15 %, consistente con la necesidad de los agentes de ajustar su movimiento bajo restricciones físicas. Por otro lado, la eficiencia de trayectoria η se mantuvo estable en ambos entornos, con valores cercanos a 0.9, lo que indica que los agentes siguieron trayectorias relativamente directas hacia sus objetivos.

Para un segundo escenario de simulación, en Matlab se conservó el número de los puntos iniciales en 150 , y se cambió la cantidad de agentes a 20. En Webots se colocó a los agentes robóticos en medio de dos estructuras u obstáculos. La Tabla 3 nos muestra sus valores medios después de 10 simulaciones.

Cuadro 3. Comparación de métricas entre Matlab y Webots – Parte II.

Entorno	**MSE final**	T_{conv} **(iteraciones)**	**Eficiencia** η
Matlab (2D)	0.13	87	0.84
Webots (3D)	0.18	104	0.70

Nota. Los valores corresponden a promedios de 10 simulaciones con $N_a = 20$ agentes y $N = 150$ puntos iniciales en Matlab, y agentes robóticos colocados entre obstáculos en Webots. Elaboración propia.

En Matlab, al trabajar con 20 agentes y los mismos 150 puntos iniciales, el MSE se mantuvo bajo (0.13), lo que indica que los robots continuaron llegando a posiciones cercanas a sus objetivos. Sin embargo, tardaron 2 iteraciones más en hacerlo, lo que provocó una disminución aproximada del 7 % en la eficiencia. Esto se debe a que el PSO tuvo un mayor número de partículas (agentes) con las que trabajar, lo que aumentó ligeramente el tiempo necesario para ajustar las trayectorias.

En Webots, el MSE permaneció en 0.18, por lo que la precisión con la que los agentes alcanzaron sus objetivos no cambió de manera notable. No obstante, al colocarse los robots en zonas rodeadas de obstáculos cercanos, la repulsión generada hacia ambos obstáculos hizo que necesitaran más iteraciones para completar sus trayectorias. Como resultado, la eficiencia disminuyó aproximadamente un 19 %, reflejando el efecto del entorno tridimensional y las maniobras adicionales que los agentes debieron realizar para evitar colisiones.

En conjunto, ambos escenarios muestran que el algoritmo es funcional tanto en simulaciones en dos dimensiones como en tres dimensiones. Aunque en la segunda prueba el tiempo de convergencia y la eficiencia variaron debido al aumento de agentes en la simulación simple y a la presencia de obstáculos más cercanos en la simulación realista, el sistema logró completar el trabajo asignado. Esto demuestra que las estrategias de optimización y control implementadas continúan siendo confiables y efectivas, incluso al aumentar la complejidad del entorno.

CAPÍTULO 11

Conclusiones

- Se diseñó y validó un método que mejora la asignación de recursos robóticos en entornos simulados de desastres naturales mediante la combinación de un algoritmo de agrupamiento con PSO.
- DBSCAN fue el algoritmo de agrupamiento elegido para la aplicación de entornos dinámicos, debido a su capacidad de eliminar ruido y generar grupos sin la necesidad de predefinirlos.
- Se implementó la combinación del algoritmo de agrupamiento DBSCAN con PSO, lo que permitió que los agentes robóticos fueran asignados a grupos proporcionales a la cantidad de elementos en ellos, y optimizó las trayectorias mediante el ajuste dinámico de las velocidades de los agentes.
- El rendimiento del algoritmo combinado fue evaluado en Matlab mediante simulaciones en entornos básicos. Las métricas analizadas (error cuadrático medio, tiempo de convergencia y eficiencia de trayectoria) confirmaron que la combinación de DBSCAN y PSO incrementó la precisión y redujo el tiempo de convergencia.
- El algoritmo funcionó en las simulaciones 3D de Webots, aunque con una convergencia más lenta debido a las condiciones físicas del entorno de simulación realista.
- El algoritmo aporta una estrategia adaptable y escalable, capaz de coordinar múltiples agentes robóticos para tareas de búsqueda y exploración dentro de contextos dinámicos.

CAPÍTULO 12

Recomendaciones

Con los objetivos cumplidos, se identificaron futuros desafíos durante la investigación, por lo que se recomienda lo siguiente tanto para mejorar los algoritmos como para futuras líneas de investigación orientadas a entornos físicos con robots reales:

- Ajustar de manera adaptativa los parámetros de DBSCAN, incorporando mecanismos automáticos de selección que permitan adaptar el algoritmo a diferentes densidades de datos sin necesidad de calibración manual.
- Implementar variantes avanzadas de PSO, tales como PSO adaptativo o versiones híbridas con algoritmos genéticos, con el fin de mejorar la capacidad de exploración global del enjambre.
- Profundizar en la optimización del control en entornos tridimensionales, considerando los efectos físicos adicionales presentes en Webots (inercia, fricción y retrasos en los actuadores). Se recomienda explorar técnicas de control robusto o predictivo que permitan reducir el tiempo de convergencia y mejorar el rendimiento en escenarios más realistas.
- Diseñar experimentos con robots físicos como el E-puck o el Pololu 3Pi+, de modo que los algoritmos implementados en simulación puedan validarse en condiciones reales. Esto incluye la calibración de sensores (infrarrojos, ultrasónicos o cámaras) y la adaptación del modelo cinemático a las restricciones de hardware.
- Incorporar sistemas de comunicación inalámbrica entre robots (por ejemplo, mediante wifi o bluetooth) para validar la coordinación multiagente en entornos reales. Esto permitirá el análisis y resolución de problemas prácticos que podrían surgir como la latencia, la pérdida de lectura de datos y el consumo energético de los agentes robóticos.
- Realizar pruebas en escenarios físicos que simulen condiciones de desastre a pequeña escala, utilizando obstáculos reales y distribuciones de objetivos, para observar la robustez del algoritmo en entornos no controlados.

- Explorar estrategias de integración con plataformas o sistemas de captura de movimiento (como el sistema Robotat) o localización basada en sensores embarcados, con el fin de evaluar la precisión y la escalabilidad del sistema en aplicaciones reales.

CAPÍTULO 13

Referencias

[1] F. Mondada, L. Gambardella, D. Floreano, S. Nolfi, J. Deneubourg y M. Dorigo, «The cooperation of swarm-bots - Physical interactions in collective robotics,» *IEEE Robotics Automation Magazine*, vol. 12, págs. 21-28, 2 jun. de 2005, ISSN: 1070-9932. DOI: 10.1109/MRA.2005.1458313.

[2] X. Yan y R. Chen, «Application Strategy of Unmanned Aerial Vehicle Swarms in Forest Fire Detection Based on the Fusion of Particle Swarm Optimization and Artificial Bee Colony Algorithm,» *Applied Sciences (Switzerland)*, vol. 14, 11 jun. de 2024, ISSN: 20763417. DOI: 10.3390/app14114937.

[3] M. Rubenstein, A. Cornejo y R. N. B.-S. Joshi, «Programmable self-assembly in a thousand-robot swarm,» *Computing Culture and Society*, 2015.

[4] J. Kennedy y R. Eberhart, «Particle Swarm Optimization,» *Purdue School of Engineering and Technology*, 1995.

[5] S. Lalwani, S. Singhal, R. Kumar y N. Gupta, *A Comprehensive Survey: Applications of Multi-Objective Particle Swarm Optimization (MOPSO) Algorithm*, 2013. dirección: www.ui.ac.ir.

[6] B. Xia, I. Mantegh y W. Xie, «Decentralized UAV Swarm Control: A Multi-Layered Architecture for Integrated Flight Mode Management and Dynamic Target Interception,» *Drones*, vol. 8, 8 ago. de 2024, ISSN: 2504446X. DOI: 10.3390/drones8080350.

[7] M. Gong, Q. Cai, X. Chen y L. Ma, «Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition,» *IEEE Transactions on Evolutionary Computation*, vol. 18, págs. 82-97, 1 feb. de 2014, ISSN: 1089778X. DOI: 10.1109/TEVC.2013.2260862.

[8] X. Ye, B. Chen, K. Lee, R. Storesund, P. Li y Q. Kang, «An emergency response system by dynamic simulation and enhanced particle swarm optimization and application for a marine oil spill accident,» *Journal of Cleaner Production*, vol. 297, mayo de 2021, ISSN: 09596526. DOI: 10.1016/j.jclepro.2021.126591.

[9] A. Aguilar, «Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO),» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2019.

[10] J. Cahueque, «Implementacion de Teoria de Enjambres,» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2019.

[11] G. Iriarte, «Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica,» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2020.

[12] J. Menéndez, «Validación de los algoritmos de robótica de enjambre Particle Swarm Optimization y Ant Colony Optimization con sistemas robóticos físicos en el ecosistema Robotat,» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2023.

[13] A. Barrientos, «Optimización del algoritmo de robótica de enjambre Particle Swarm Optimization para su implementación con agentes robóticos físicos en escenarios con obstáculos en el ecosistema Robotat,» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2024.

[14] D. Ixcayau, «Aplicación de algoritmos de aprendizaje automático, con énfasis en aprendizaje no supervisado, para la identificación y categorización de segmentos de interés en señales bioeléctricas para el estudio de la epilepsia Fase V,» Trabajo de graduación de Licenciatura, Universidad del Valle de Guatemala, Ciudad de Guatemala, Guatemala, 2024.

[15] D. Delforge, R. Below, V. Wathelet, N. Speybroeck y J. V. Loenhout, «2023 Disaster in Numbers,» *Emergency Events Database*, pág. 5, 7 jul. de 2023, ISSN: 1546170X.

[16] J. Doe y A. Smith, «Swarm Robotics: A Survey from a Multi-Tasking Perspective,» *ACM Computing Surveys*, 2023. DOI: 10.1145/3611652.

[17] M. Brambilla, E. Ferrante, M. Birattari y M. Dorigo, «Swarm robotics: a review from the swarm engineering perspective,» *Swarm Intelligence*, vol. 7, n.º 1, págs. 1-41, 2013. DOI: 10.1007/s11721-012-0075-2.

[18] H. Hamann, «Swarm robotics: A formal approach,» *Springer International Publishing*, 2018. DOI: 10.1007/978-3-319-74528-2. dirección: https://link.springer.com/book/10.1007/978-3-319-74528-2.

[19] H. Duan, X. Chen, K. Zhang y W. Zhang, «From animal collective behaviors to swarm robotic cooperation,» *National Science Review*, vol. 10, n.º 5, nwad040, 2023. DOI: 10.1093/nsr/nwad040. dirección: https://academic.oup.com/nsr/article/10/5/nwad040/7043485.

[20] Scikit-learn developers, *DBSCAN Clustering*, Último acceso: 26 de septiembre de 2025, Scikit-learn, 2023. dirección: https://scikit-learn.org/stable/modules/clustering.html#dbscan.

[21] E. Atif, M. Awad y M. Shehata, «Performance evaluation of PSO-PID and PSO-FLC for trajectory tracking,» *Scientific Reports*, 2024. DOI: 10.1038/s41598-023-50551-0.

[22] X. Zhang, D. Zou y X. Shen, «A Novel Simple Particle Swarm Optimization Algorithm for Global Optimization,» *MDPI*, vol. 1, pág. 34, 2018. dirección: https://www.mdpi.com/2227-7390/6/12/287#metrics.

[23] K. Katona, H. Neamah y P. Korondi, «Obstacle Avoidance and Path Planning Methods for Autonomous Navigation of Mobile Robot,» *MDPI*, vol. 1, pág. 47, 2024. DOI: 10.1016/j.engappai.2022.105044. dirección: https://www.sciencedirect.com/science/article/pii/S2352914822001356.

[24] T. Abdalla y A. Abdulkareem, «Design of PID controller based on particle swarm optimization (PSO) for a mobile robot trajectory tracking,» *International Journal of Computer Applications*, vol. 47, n.º 23, 2012. DOI: 10.5120/7497-0601. dirección: https://www.ijcaonline.org/archives/volume47/number23/7497-0601.

[25] M. Mahmoodabadi, Z. Wang, L. Wang y Q. Zhu, «Adaptive proportional-integral-derivative control for surgical plane cable-driven robots,» *Engineering Applications of Artificial Intelligence*, vol. 114, pág. 105 044, 2022. DOI: 10.1016/j.engappai.2022.105044. dirección: https://www.sciencedirect.com/science/article/pii/S2352914822001356.

[26] MathWorks, *MATLAB Documentation*, The MathWorks, Inc., 2023. dirección: https://www.mathworks.com/help/matlab/.

[27] Cyberbotics Ltd., *Webots User Guide*, Cyberbotics, 2023. dirección: https://cyberbotics.com/doc/guide/index.

CAPÍTULO 14

Anexos

14.1. Repositorio

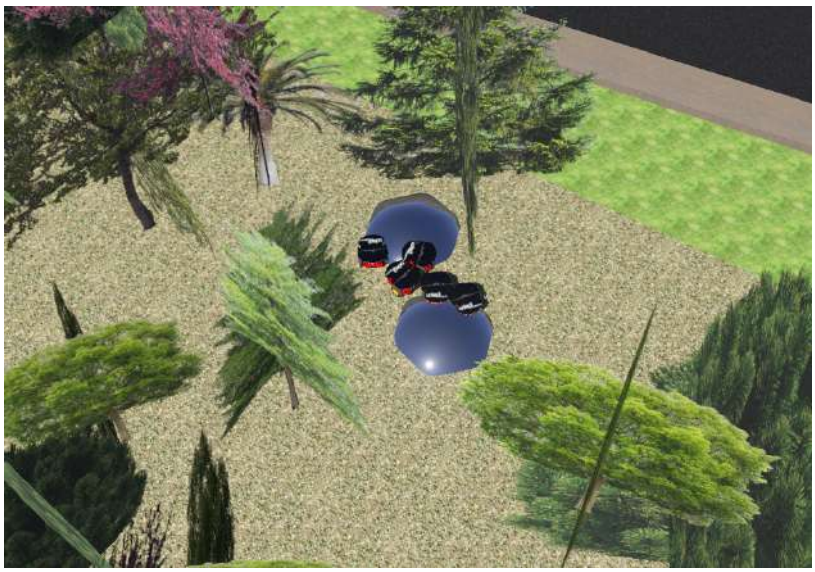
Los archivos de código utilizados para las simulaciones simples y realistas, junto con el entorno tridimensional generado en Webots, se encuentran en un repositorio de GitHub. El enlace para estos recursos es el siguiente: https://github.com/RonalBT/C-digo-Tesis

14.2. Entorno 3D

Figura 21. Mundo tridimensional generado en Webots.

Nota. La imagen muestra una vista del mundo tridimensional generado en Webots para las simulaciones realistas. Elaboración propia.

Figura 22. Asignación de agentes robóticos a incendio forestal.



Nota. La imagen muestra la asignación de agentes robóticos en una situación de incendio forestal. Elaboración propia.