Simposio CEA de Robótica, Bioingeniería, Visión Artificial y Automática Marina 2025

RBVM
Simposio CEA de Robótica, Bioingeniería, Visión por Computador y Automática Marina

Simulador para el control de flotas de Kilobots con Python en CoppeliaSim

Aporta Costela, J.[a], Caparroz, M.[a], Castilla, M.[a], Moreno, J.C.[a,*]

[a]*Departamento de Informática, Universidad de Almería, ceiA3, CIESOL, Ctra. Sacramento, s/n, 04120 Almería, España.*

Resumen

Este trabajo aborda el control de flotas de robots con capacidades sensoriales limitadas, centrándose en la implementación de algoritmos de robótica de enjambre utilizando CoppeliaSim y Python. En particular, se desarrollan y prueban, por un lado, un método de localización distribuida y resiliente, que permite a los minirobots estimar su posición sin sistemas centralizados, y, por otro, el algoritmo "Wave" para la propagación coordinada de información. Las simulaciones validan la efectividad de los algoritmos bajo diferentes condiciones, analizando errores de localización y el impacto del ruido en la comunicación. Los experimentos reales con una flota basada en Kilobots evidencian los desafíos derivados de las limitaciones de hardware y de la variabilidad en la comunicación, que afectan la precisión y la coordinación grupal. Los resultados muestran el potencial de estos enfoques para el control de enjambres robóticos de bajo coste y capacidades limitadas, aunque se identifican áreas de mejora para reducir la brecha entre la simulación y la implementación física.

Palabras clave: Tecnología de enjambre, control de flotas, Kilobots, CoppeliaSim, sistemas multirobot.

Kilobot fleet control simulator with Python in CoppeliaSim

Abstract

This paper addresses the control of robot fleets with limited sensory capabilities, focusing on the implementation of swarm robotics algorithms using CoppeliaSim and Python. In particular, a distributed and resilient localisation method, which allows minirobots to estimate their position without centralised systems, and the "Wave" algorithm for coordinated information propagation are developed and tested. Simulations validate the effectiveness of the algorithms under different conditions, analysing localisation errors and the impact of noise on communication. Real experiments with a Kilobot-based fleet highlight the challenges arising from hardware limitations and communication variability, which affect accuracy and group coordination. The results show the potential of these approaches for the control of low-cost, limited-capacity robotic swarms, although areas for improvement are identified to reduce the gap between simulation and physical implementation.

Keywords: Swarm technology, swarm robotics, fleet control, Kilobots, CoppeliaSim, multirobot systems.

1. Introducción

La robótica de enjambre, inspirada en los comportamientos sociales de los animales como hormigas, abejas, pájaros y peces, representa un paradigma emergente en la robótica distribuida, con un creciente interés en la comunidad científica (Şahin, 2004). Este campo se centra en la coordinación de un gran número de robots simples para lograr tareas complejas a través de interacciones locales y sin una unidad de control centralizada. Se trata de un novedoso enfoque para la coordinación de sistemas robotizados que cuentan con un gran número de robots y se centra en cómo diseñar un número de agentes simples de modo que un comportamiento colectivo deseado emerja de la interacción entre los propios agentes y el medio que les rodea, ofreciendo robustez, flexibilidad y escalabilidad (Brambilla et al., 2013; Hamann, 2018).

Los sistemas de enjambre aprovechan principios de la inteligencia de enjambre, donde la flota de robots puede concebirse como una red neuronal distribuida, capaz de resolver tareas colectivamente mediante el intercambio de información local sobre el entorno y las acciones (Otte, 2016; Rubenstein et al., 2014b). Si bien la aplicación de esta tecnología a la robótica aún se encuentra en una fase de maduración, su potencial para abordar problemas complejos de manera cooperativa, como

*Autor para correspondencia: jcmoreno@ual.es

la optimización de rutas en entornos dinámicos, la exploración autónoma sin información previa y la ejecución de tareas en entornos desconocidos o peligrosos, es significativo.

En este contexto, el presente trabajo se centra en la investigación y la implementación de algoritmos de control para flotas de robots con limitaciones sensoriales, específicamente los Kilobots (Rubenstein et al., 2014b). El trabajo explora la aplicación de algoritmos clave en la robótica de enjambre, como un método de localización distribuida y resiliente que permite a los robots estimar su posición sin depender de un sistema centralizado, y el algoritmo "Wave" para la propagación de información y la coordinación de movimientos dentro del enjambre.

La metodología adoptada involucra una fase de simulación exhaustiva utilizando el software CoppeliaSim, un entorno que permite la modelización y programación individual de los robots, en este caso, utilizando el lenguaje Python. Posteriormente, se traslada la implementación y validación de los algoritmos a experimentos con Kilobots reales, confrontando los desafíos inherentes a la transferencia de la simulación al mundo físico, incluyendo las limitaciones de hardware y el ruido en las interacciones.

Esta investigación busca contribuir al avance en la comprensión y el control de sistemas robóticos de enjambre, particularmente en escenarios con robots de bajo coste y capacidades limitadas, en el ámbito de la asignatura Sistemas Robotizados del Máster en Ingeniería Informática de la Universidad de Almería.

2. Materiales y métodos

2.1. Kilobots

Los Kilobots son pequeños robots de bajo coste diseñados específicamente para la experimentación con algoritmos de robótica de enjambre (Rubenstein et al., 2014a). Sus características principales incluyen capacidad de movimiento mediante vibración, detección de luz ambiental, detección de la distancia a Kilobots vecinos y comunicación por infrarrojos en un radio de hasta 7 cm. El motor vibratorio, encargado del movimiento del dispositivo, permite movimientos de rotación en sentido horario o antihorario y movimientos de avance en línea recta, según su configuración.

Cada Kilobot cuenta con un microcontrolador ATMega 328p de 8 bits y una frecuencia de reloj de 8 MHz. Cuenta con una memoria flash de 32 KBytes, memoria EEPROM para ajustes de calibración y memoria SRAM para procesamiento de datos. La programación de los Kilobots reales se realiza en lenguaje C utilizando la librería *Kilolib*, que proporciona las funciones necesarias para interactuar con sus sensores y actuadores. Para este fin es necesario además el uso del módulo *Overhead Controller* (OHC), que permite comunicar los Kilobots con el ordenador, ya sea a través de infrarrojos o de forma física a través de un puerto serial o el puerto de programación. Para la calibración y programación de los Kilobots reales se utilizó la aplicación KiloGUI. Se cuenta con un total de cinco Kilobots.

2.2. CoppeliaSim y Python

El simulador CoppeliaSim es uno de los más utilizados con fines pedagógicos en la actualidad. Este simulador proporciona un marco versátil y escalable para crear simulaciones 3D en un periodo de tiempo relativamente corto. Dispone de un entorno de desarrollo integrado (IDE) que se basa en una arquitectura distribuida y de scripts: cada objeto de escena puede tener un script incorporado, todos ellos funcionando al mismo tiempo, en forma de hilos. Además, es multiplataforma y soporta diversos lenguajes de programación, incluyendo Python (Rohmer et al., 2013; Calderón-Arce et al., 2022).

Para la programación en Python se ha utilizado Anaconda, una plataforma para la gestión de librerías de Python que incluye Jupyter, un entorno de desarrollo interactivo basado en celdas de código. Se preparó una escena en CoppeliaSim con un área de trabajo delimitada, robots Kilobots (incluyendo la adición de un sensor de luz simulado) y un foco de luz para los ensayos de guiado. Se implementó una arquitectura de Programación Orientada a Objetos (POO) en Python para modelar el comportamiento de cada Kilobot.

2.3. Algoritmos implementados

La localización de los robots es un factor clave para tener en cuenta en el mundo de la robótica móvil, ya que se debe tener conciencia de donde se encuentran los robots en todo momento para tener una idea del comportamiento que está teniendo el propio robot y para que no se produzca ningún comportamiento indeseado y poder actuar correctamente en caso de fallas o accidentes. Para este fin, se ha hecho uso del algoritmo *Distributed and resilient localization* (de Sá et al., 2017), que permite a un robot con limitaciones sensoriales determinar su posición relativa utilizando la información de la distancia a otros robots ancla (con posición conocida) y nodos vecinos. Para el uso de este algoritmo, el enjambre debe ser dividido entre robots denominados "anclas" (aquellos que conocen su propia posición) y robots "nodos" (que obtienen su posición en base a los robots anclas).

El algoritmo se divide en varias etapas. En primer lugar, se hace uso el algoritmo *Sum-Dist*, que utiliza la capacidad de un robot de enviar y recibir mensajes con sus vecinos para determinar la distancia que han recorrido los mensajes enviados por el enjambre. En este proceso, los robots anclas transmiten un mensaje con su identificación, su posición y la distancia que ha recorrido el mensaje. Éste es recibido por los robots nodos, que almacenan la información y replican el mensaje a sus vecinos, incluyendo además la distancia a la que se encontraba el remitente. Este proceso se ejecuta durante el tiempo necesario para que todos los robots nodos puedan comunicarse con todos los robots anclas. A continuación, se realiza una estimación inicial de la posición mediante el uso del método *Min-Max*, donde cada robot necesita la información de un mínimo de 3 robots anclas para que el método funcione correctamente. En caso de no disponer de dicha información, se utiliza el algoritmo *Multi-hop Collaborative Min-Max*, en el que se hace uso de la información de los nodos adyacentes como si se tratara de nodos anclas. Por último, se hace uso del algoritmo *Backtracking Search* para optimizar la función de posición de los robots, con el objetivo de minimizar el error de las distancias a las que se encuentra un robot nodo de sus vecinos.

Por otro lado, para la comunicación entre robots se ha hecho uso del algoritmo *Wave* (Junior and Nedjah, 2017), el cual permite difundir los mensajes entre todos los miembros del enjambre y tomar decisiones de forma coordinada. Un robot "líder"

envía un mensaje con la orden al resto de robots y recibe realimentación del enjambre para tomar decisiones en base a estos.

3. Resultados y discusiones

3.1. Resultados en simulación

La fase de simulación, implementada en CoppeliaSim, permitió la evaluación de los algoritmos de localización distribuida y los métodos de control de movimiento de la flota de robots hacia una fuente lumínica.

3.1.1. Implementación del algoritmo "Distributed and Resilient Localization"

Se realizaron diversos ensayos con el algoritmo de localización distribuida para evaluar su desempeño en la estimación de la posición de un robot nodo en relación con robots ancla, variando tanto la posición del robot nodo como el tiempo de ejecución del algoritmo *Sum-Dist* (T_s) para analizar las fluctuaciones en los resultados. En todos los ensayos, se utilizaron 7 Kilobots, con números de identificación del 0 al 6. Los Kilobots 0, 2 y 4 actuaron como robots ancla, con el Kilobot 0 establecido como el origen de coordenadas (0,0). Los Kilobots restantes (1, 3, 5 y 6) se colocaron en las coordenadas $(0, 70)$, $(70, 0)$, $(30, 30)$ y $(65, 65)$, respectivamente.

En primer lugar, se realizaron dos ensayos con la disposición de cuatro Kilobots nodo según se ilustra en la Figura 1, utilizando tiempos de ejecución del algoritmo de 5 segundos para el ensayo 1 y 6 segundos para el ensayo 2. Al aumentar T_s en el segundo ensayo, se consigue recopilar información de todos los robots ancla del enjambre y evaluar si se produce una mejora en la precisión de la localización.

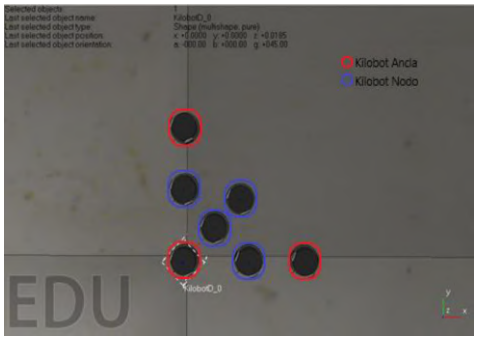


Figura 1: Disposición inicial de los ensayos 1 y 2

En las Tablas 1 y 2 se observan los errores cometidos en la estimación de la posición en cada uno de los ensayos. En la primer columna se muestra el número de identificación de cada Kilobot y en la segunda el número de anclas utilizados para la estimación de su posición. En las siguientes columnas, se pueden observar los errores en las coordenadas X e Y cometidos en cada etapa de la estimación: *Min-Max*, *Multi-hop Collabotative Min-Max* (MCMM) y *Backtracking Search Algorithm* (BSA).

Los resultados sugieren una mejora considerable en el error para los Kilobots 1 y 3 al disponer de información de tres anclas, conseguido al aumentar T_s. En el segundo ensayo, la tercera etapa (MCMM) ha sido omitida al no ser necesaria con la información de tres anclas.

Tabla 1: Error (en mm) cometido en cada etapa del ensayo 1(Simulación)

ID	Nº anclas	Min-Max	MCMM	BSA
1	2	(0,5)	(0,5)	(0.5,5)
3	2	(5,0)	(5,0)	(5,0)
5	3	(1,1)	(1,1)	(5.3,4)
6	3	(16.5,16.5)	(16.5,16.5)	(10.6,6)

Tabla 2: Error (en mm) cometido en cada etapa del ensayo 2 (Simulación)

ID	Nº anclas	Min-Max	BSA
1	3	(16,0)	(3.2,0)
3	3	(0,16)	(0,0.5)
5	3	(1,1)	(5.7,3.7)
6	3	(16.5,16.5)	(3.8,8.7)

En el tercer ensayo realizado, se modificó la distribución de los Kilobots 5 y 6 a las coordenadas $(10, 35)$ y $(35, 10)$, respectivamente (Figura 2), manteniendo el mismo número de nodos y aumentando el tiempo a 7 segundos. Esta configuración se realizó para comparar los resultados con los obtenidos en los ensayos reales mostrados en la sección 3.2.1. La Tabla 3 se resumen los errores cometidos en la estimación de este ensayo.

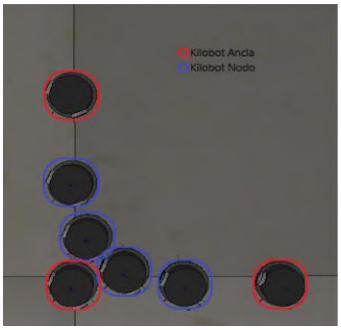


Figura 2: Disposición del ensayo 3

Al comparar los tres resultados, puede observarse que la variación en la disposición afectó el error de localización, especialmente en los Kilobots cuya posición fue modificada (5 y 6).

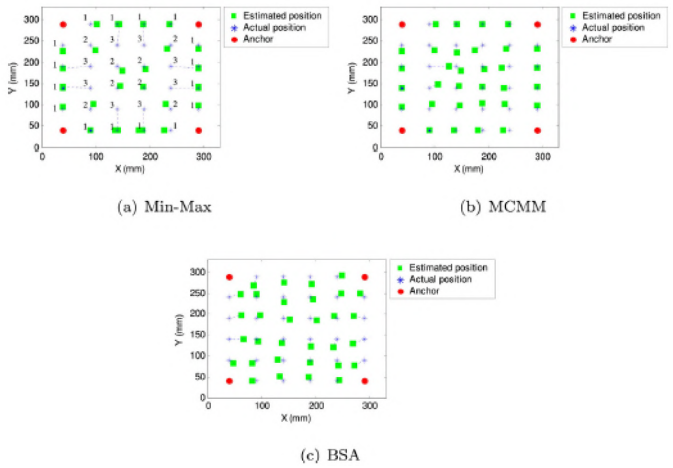
Tabla 3: Error (en mm) cometido en cada etapa del ensayo 3 (Simulación)

ID	Nº anclas	Min-Max	MCMM	BSA
1	3	(17,5)	(17,5)	(6,5)
3	3	(5,17)	(5,17)	(5,17)
5	3	(8,0)	(8,0)	(9.6,0.7)
6	3	(0,8)	(0,8)	(0,8)

Por último, se realizó un ensayo con 32 Kilobots nodo y 4 anclas, cuyos resultados se presentan en la Figura 3. Este ensayo ilustra la escalabilidad del algoritmo en un entorno simulado.

3.1.2. Implementación del método de réplica de órdenes

En este caso, se evaluó el método de réplica de órdenes (algoritmo *Wave*) para guiar un enjambre hacia una fuente de luz, realizando tres ensayos. En dos de ellos se ha incluido ruido, simulando las variaciones en la velocidad de movimiento de los Kilobots reales.



(a) Min-Max (b) MCMM

(c) BSA

Figura 3: Resultados del ensayo realizado con 32 Kilobots nodo y 4 anclas

En el primer ensayo, realizado sin ruido, se utilizó la disposición de la Figura 1 para verificar la funcionalidad básica del método. Los resultados confirmaron la corrección del código, aunque se observó un ligero retardo entre la ejecución del código y la simulación en CoppeliaSim. En el segundo ensayo se realizó la misma configuración pero se introdujo ruido en la velocidad de giro de los Kilobots. La Figura 4 muestra los resultados de simulación, evidenciando cómo la falta de realimentación provoca la desorientación de los Kilobots más alejados del líder, mientras que los cercanos lo siguen de forma descoordinada.

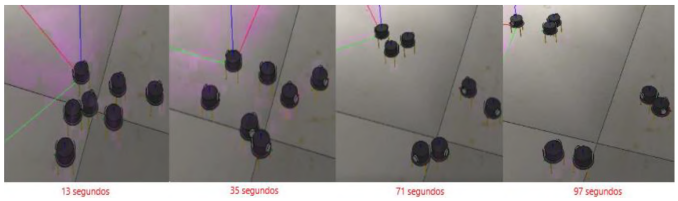


Figura 4: Simulación ensayo 2 método réplica de órdenes

El tercer ensayo se realizó disponiendo cinco Kilobots en fila, tal como se muestra en la Figura 5, con el objetivo de observar el efecto del ruido en la propagación de órdenes. Puede observarse en la secuencia que el segundo Kilobot perdió la comunicación con el líder, lo que desorientó al resto del enjambre al no recibir nuevas órdenes.

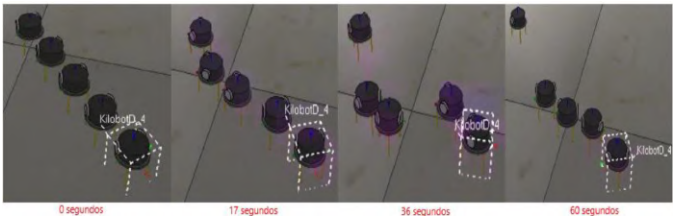


Figura 5: Simulación ensayo 3 método réplica de órdenes

3.1.3. *Implementación del método con realimentación*

En el siguiente método, un iniciador denominado "robot líder" es el encargado de guiar el enjambre hacia un foco de luz. Dicho robot envía mensajes al resto del enjambre con las órdenes que deben seguir y, posteriormente, espera una realimentación proveniente del enjambre en base a la cual tomará la siguiente decisión. En caso de no ser un robot líder, cada robot del enjambre debe tener un robot "padre". Al recibir un mensaje, si aún no cuenta con un robot padre, se configura al remitente como tal y, a partir de dicho momento, todos los mensajes que no provengan de este serán descartados.

Debido a la complejidad computacional y los retrasos en la ejecución del código, se limitaron los ensayos con el método basado en la distancia con realimentación a dos Kilobots seguidores y una velocidad de simulación reducida.

El primer ensayo en simulación (Figura 6) demostró el funcionamiento del método al esperar el avance de los Kilobots rezagados. Sin embargo, se observaron colisiones entre los robots hijos, dificultando el progreso hacia el líder.

Figura 6: Simulación ensayo 1 método realimentado

Posteriormente, se modificó la estructura de comunicación padre/hijo, utilizando un Kilobot intermedio como puente (Figura 7). Esta disposición redujo las colisiones y aceleró el movimiento del enjambre hacia la luz.

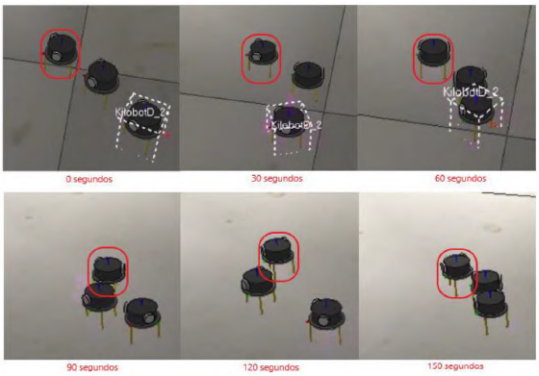


Figura 7: Simulación ensayo 2 método realimentado

3.2. *Ensayos Reales*

La experimentación con Kilobots reales se centró en validar el algoritmo de localización distribuida y el método de réplica de movimientos para la guía hacia una fuente lumínica. Las

limitaciones de memoria del Kilobot impidieron la implementación completa del algoritmo de localización y el método de control con realimentación.

3.2.1. Implementación de las dos primeras etapas del algoritmo "Distributed and Resilient Localization"

Se realizaron tres ensayos variando la posición de un Kilobot nodo (conectado mediante cable serie) con respecto a tres Kilobots ancla ubicados en las posiciones $(0, 0)$,$(0, 70)$ y $(70, 0)$ mm (Figura 8).

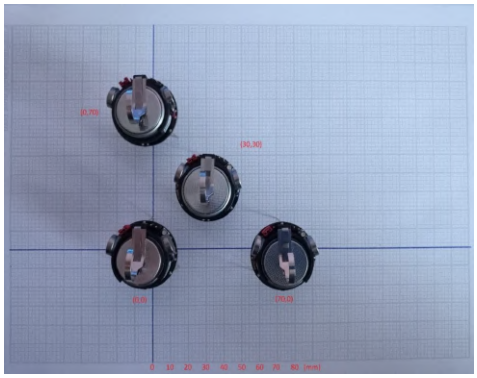


Figura 8: Disposición de los Kilobots - Ensayo 1

Al ubicar el Kilobot nodo en la posición real $(30, 30)$ mm, los datos obtenidos por cable serie arrojaron un error de $(3, 3)$ mm en las coordenadas estimadas en la Etapa II. Este resultado es comparable al error de $(1, 1)$ mm obtenido en simulación para la misma etapa. Las discrepancias se atribuyen a las diferencias en la colocación de los robots ancla y la precisión en la medición de distancias entre robots.

En el segundo ensayo, se colocó el Kilobot nodo cerca del límite de la región definida por los anclas, en la posición $(10, 30)$ mm (Figura 9). Las coordenadas estimadas fueron $(25, 35)$ mm, obteniendo un error mayor que en el ensayo anterior, lo que concuerda con la tendencia observada en la simulación de un incremento del error cuando el Kilobot nodo se encuentra cerca de los límites del área de trabajo delimitado por los Kilobots ancla.

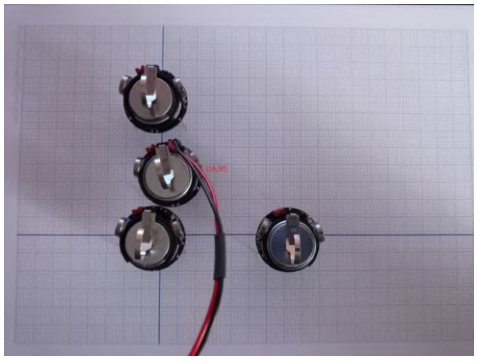


Figura 9: Ensayo 2 - Min-Max

Por último, se realizó un tercer ensayo, similar al segundo, en el cual el Kilobot nodo se ubicó cerca del límite, en la posición $(40, 10)$ mm (Figura 10). Los resultados mostraron un error considerable, similar al del ensayo 2, con coordenadas estimadas $(35, 24)$ mm.

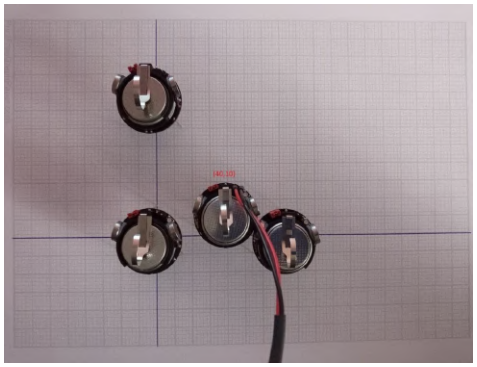


Figura 10: Ensayo 3 - Min-Max

En general, los errores en los ensayos reales fueron mayores que en las simulaciones, lo cual es esperable debido a las diversas fuentes de ruido inherentes al entorno físico.

3.2.2. Ensayos con método de réplica de movimientos

Con el método de réplica de movimientos se realizaron cuatro ensayos distintos en los que se buscaba guiar un enjambre hacia una fuente de luz. En el primer ensayo, se utilizó un líder y un seguidor (Figura 11). En este caso, las diferencias de movimiento entre ambos Kilobots provocó una pérdida de la conexión, quedando el robot seguidor incomunicado y sin poder recibir nuevas órdenes del líder.

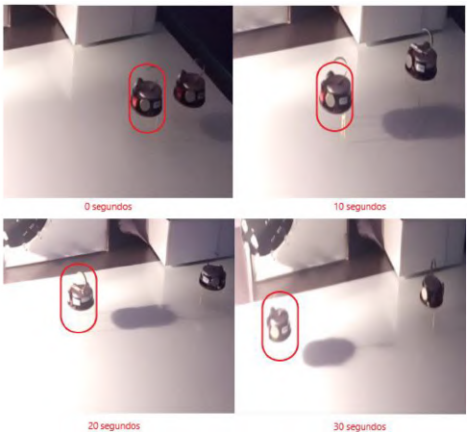


Figura 11: Resultados método réplica de movimientos - Ensayo 1

A continuación, se incrementó el número de seguidores a dos, intentando una calibración más precisa del movimiento (Figura 12). Si bien inicialmente un seguidor respondió correctamente, ambos se desorientaron con el tiempo por las mismas razones que en el ensayo anterior.

Posteriormente, se añadieron dos seguidores más (Figura 13), observándose que un único seguidor logró alcanzar la fuente de luz, aunque con dificultades. El resto de seguidores se desorientó debido a las diferencias en el movimiento y la pérdida de sincronización con el líder.

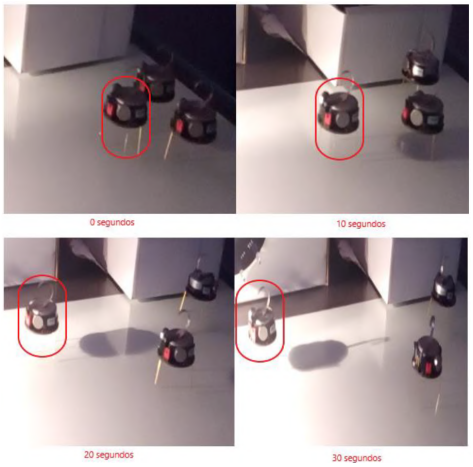


Figura 12: Resultados método réplica de movimientos - Ensayo 2

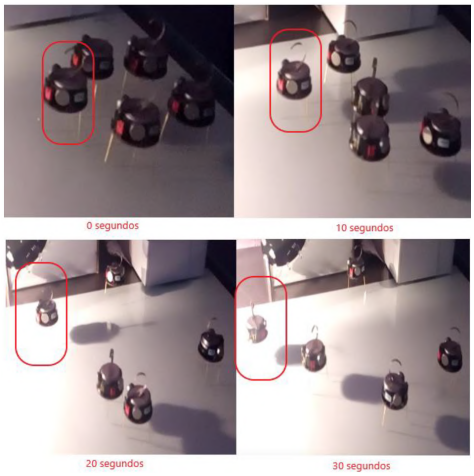


Figura 13: Resultados método réplica de movimientos - Ensayo 3

Por último, se implementó una cadena de comunicación padre/hijo (Figura 14). Los últimos Kilobots siguieron las órdenes del líder a través de la cadena, pero el ruido en el movimiento condujo a la desorientación final, similar a lo observado en la simulación.

Figura 14: Resultados método réplica de movimientos - Ensayo 4

Los resultados de los ensayos reales con el método de réplica de movimientos reflejan los desafíos de transferir algoritmos simulados a robots físicos con limitaciones inherentes en su actuación y comunicación. Las pequeñas variaciones en el movimiento individual de cada Kilobot se amplifican en el enjambre, dificultando el mantenimiento de la coherencia grupal.

4. Conclusiones

Este trabajo demostró la viabilidad de implementar algoritmos de robótica de enjambre para el control de flotas de minirobots Kilobots tanto en un entorno de simulación utilizando CoppeliaSim y Python, como en ensayos con robots reales. El algoritmo de localización distribuida y resiliente ("distributed and resilient localization") mostró ser una aproximación interesante para la auto-localización de robots con limitaciones sensoriales, aunque su precisión depende de la distribución de los robots ancla y puede beneficiarse de la fusión sensorial con otros métodos de localización. El algoritmo "Wave" se reveló como un método robusto para la gestión de la comunicación en el enjambre y para la implementación de estrategias de control. Sin embargo, los ensayos reales pusieron de manifiesto las limitaciones de memoria y las variaciones en el comportamiento de los robots individuales, lo que afectó principalmente al método de réplica de movimientos para el guiado del enjambre. El trabajo con el simulador facilitará al estudiante del Máster en Ingeniería Informática la comprensión de los fundamentos del control de flotas de robots, y su aplicación al sistema real le permitirá vislumbrar las diferencias entre trabajar con un modelo del sistema y un sistema real, justificando así la necesidad de una adecuada etapa de validación.

Agradecimientos

La autora Malena Caparroz es beneficiaria de un contrato de Formación de Profesorado Universitario del Ministerio de Ciencia, Innovación y Universidades (FPU23/02235).

Referencias

Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M., 2013. Swarm robotics: a review from the swarm engineering perspective. Swarm Intelligence 7, 1–41.

Calderón-Arce, C., Brenes-Torres, J. C., Solis-Ortega, R., 2022. Swarm robotics: Simulators, platforms and applications review. Computation 10 (6), 80, https://doi.org/10.3390/computation10060080.

de Sá, A. O., Nedjah, N., de Macedo Mourelle, L., 2017. Distributed and resilient localization algorithm for swarm robotic systems. Applied Soft Computing 57, 738–750, https://doi.org/10.1016/j.asoc.2016.07.049.

Hamann, H., 2018. Swarm robotics: A formal approach. Vol. 221. Springer.

Junior, L. S., Nedjah, N., 2017. Wave algorithm applied to collective navigation of robotic swarms. Applied Soft Computing 57, 698–707, https://doi.org/10.1016/j.asoc.2016.06.004.

Otte, M., 2016. Collective cognition and sensing in robotic swarms via an emergent group-mind. In: International Symposium on Experimental Robotics. Springer, pp. 829–840.

Rohmer, E., Singh, S. P., Freese, M., 2013. V-rep: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. IEEE, pp. 1321–1326, https://doi.org/10.1109/IROS.2013.6696520.

Rubenstein, M., Ahler, C., Hoff, N., Cabrera, A., Nagpal, R., 2014a. Kilobot: A low cost robot with scalable operations designed for collective behaviors. Robotics and Autonomous Systems 62 (7), 966–975, https://doi.org/10.1016/j.robot.2013.08.006.

Rubenstein, M., Cornejo, A., Nagpal, R., 2014b. Programmable self-assembly in a thousand-robot swarm. Science 345 (6198), 795–799.

Şahin, E., 2004. Swarm robotics: From sources of inspiration to domains of application. In: International workshop on swarm robotics. Springer, pp. 10–20.