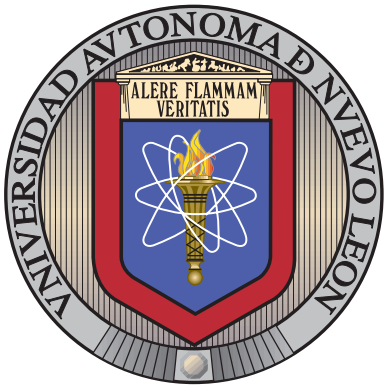
Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado



Colaboración emergente en enjambres de robots, con reglas inspiradas en el comportamiento de animales sociales

por

M.C. Erick de Jesús Ordaz Rivas

como requisito parcial para obtener el grado de

DOCTORADO EN INGENIERÍA ELÉCTRICA

Septiembre 2020

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Colaboración emergente en enjambres de robots, con reglas inspiradas en el comportamiento de animales sociales

por

M.C. Erick de Jesús Ordaz Rivas

como requisito parcial para obtener el grado de

DOCTORADO EN INGENIERÍA ELÉCTRICA

Septiembre 2020

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis "Colaboración emergente en enjambres de robots, con reglas inspiradas en el comportamiento de animales sociales", realizada por el alumno M.C. Erick de Jesús Ordaz Rivas, con número de matrícula 1411496, sea aceptada para su defensa como requisito para obtener el grado de Doctorado en Ingeniería Eléctrica.

El Comité de Tesis

Dr. Luis Martín Torres Treviño
Director

Dr. Juan Ángel Rodríguez Liñán
Revisor

Dr. Romeo Sánchez Nigenda
Revisor

Dr. Joaquín Gutiérrez Jagüey
Revisor

Dr. Jonatán Peña Ramírez
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO FIME

San Nicolás de los Garza, Nuevo León, septiembre de 2020

Ciudad Universitaria Pedro de Alba s/n, C.P. 66455. A.P. 076 Suc. "F"
San Nicolás de los Garza, Nuevo León, México. Tels: (81) 8332 0903 /
Conm.: 8329 4020 / Fax: (81) 8332 0904

A mi madre.

Espero que este nuevo logro en mi vida sea un motivo por el cual te sientas orgullosa de mí.

Índice general

Nomenclaturas xviii

Agradecimientos xix

Resumen xx

1. Introducción 1

1.1. Motivación 1

1.2. Antecedentes 4

1.3. Descripción del problema 8

1.4. Hipótesis 10

1.5. Objetivos 10

1.5.1. Objetivo general 10

1.5.2. Objetivos particulares 10

1.6. Contribuciones 11

1.7. Organización de la tesis 12

2. Robótica de enjambres **13**

2.1. Modelo matemático de robots 13

2.1.1. Modelo cinemático 14

2.1.2. Modelo dinámico 16

2.1.3. Modelo del actuador 17

2.2. Arquitectura de los robots 19

2.2.1. Terran 20

2.2.2. BugBot 20

2.2.3. Limitaciones sensoriales de los robots 22

2.3. Reglas de comportamiento 23

2.3.1. Implementación de reglas en un sistema embebido 28

2.3.2. Implementación de reglas en una plataforma de simulación . . 39

3. Descripción experimental **45**

3.1. Desarrollo de experimentos físicos 45

3.1.1. Especificaciones de robots en experimentos físicos 46

3.2. Desarrollo de simulaciones 48

3.3. Métricas de evaluación 49

3.3.1. Sistema de procesamiento de imágenes 49

3.4. Descripción de casos experimentales 51

3.4.1. Depredador-presa 51

3.4.2. Transporte de objetos 52

4. Resultados **53**

4.1. Primer caso experimental (Tarea de relaciones tipo depredador-presa) 53

4.1.1. Simulaciones 53

4.1.2. Experimentos por implementación física 55

4.1.3. Análisis de datos y modelado de la emergencia de colaboración en el enjambre, a partir de los resultados obtenidos en la tarea de depredador-presa 76

4.1.4. Discusión 80

4.2. Segundo caso experimental (Tarea de transporte de objetos) 83

4.2.1. Simulaciones 83

4.2.2. Experimentos por implementación física 84

4.2.3. Análisis de datos y modelado de la emergencia de colaboración en el enjambre, a partir de los resultados obtenidos en la tarea de transporte de objetos 100

4.2.4. Discusión 107

5. Conclusiones y trabajo futuro 110

5.1. Conclusiones 110

5.2. Trabajo futuro 112

5.3. Divulgación científica 113

5.3.1. Artículos de revista 113

5.3.2. Artículos de congreso 113

A. Reporte técnico de BugBots 114

A.1. Diseño 114

A.2. Hardware 114

A.2.1. Microcontrolador 114

A.2.2. Sensores y actuadores 115

A.2.3. Alimentación 117

A.3. Software 117

A.4. Funcionamiento 118

B. Códigos en Arduino - Reglas de comportamiento 119

B.1. Tarea de relación tipo depredador-presa 119

B.2. Tarea de transporte de objetos 129

C. Código en Scilab - Reglas de comportamiento 142

C.1. Tarea de relación tipo depredador-presa 142

C.2. Tarea de transporte de objetos 158

Índice de figuras

2.1. Robot móvil con configuración diferencial. 14

2.2. Ubicación de sensores y dispositivos en la arquitectura del robot *Terran*. 20

2.3. Ubicación de sensores y dispositivos en la arquitectura del robot *BugBot*. 21

2.4. Zonas de percepción correspondientes a los sensores incorporados. . . 22

2.5. Limitaciones de los sensores de proximidad. 23

2.6. Zonas propuestas para el modelo de comportamiento de enjambre. . . 24

2.7. Vectores de dirección respecto a zona detectada. 25

2.8. Comportamiento emergente en el enjambre. 28

2.9. Vectores de dirección de cada sensor. 29

2.10. Zonas *RAOI* por robot. 30

2.11. Funciones de movimiento del robot en relación al voltaje aplicado. . . 34

2.12. Estado de tenzas de *BugBot*. 36

2.13. Estados de comportamiento para tarea de transporte de objetos. . . . 38

2.14. Diagrama de flujo de las reglas de comportamiento de enjambre. . . . 43

2.15. Diagrama de flujo de las reglas de comportamiento de robot-presa. . . 44

2.16. Diagrama de flujo de las reglas de comportamiento para tarea de transporte de objetos. 44

3.1. Etapas de desarrollo de experimentos físicos. 46

3.2. Etapas de desarrollo de simulaciones. 48

3.3. Diagrama de flujo de herramienta de procesamiento de imagen. 50

4.1. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.2$ para los robot-depredadores. 56

4.2. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.2$ para los robot-depredadores. 57

4.3. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.2$ para los robot-depredadores. 58

4.4. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.6$ para los robot-depredadores. 59

4.5. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.6$ para los robot-depredadores. 60

4.6. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.6$ para los robot-depredadores. 61

4.7. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 1$ para los robot-depredadores. 62

4.8. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores. . 63

4.9. Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 1$ para los robot-depredadores. 64

4.10. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.2$ para los robot-depredadores. 66

4.11. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.2$ para los robot-depredadores. 67

4.12. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.2$ para los robot-depredadores. 68

4.13. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.6$ para los robot-depredadores. 69

4.14. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.6$ para los robot-depredadores. 70

4.15. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.6$ para los robot-depredadores. 71

4.16. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 1$ para los robot-depredadores. 72

4.17. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores. 73

4.18. Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores. 74

4.19. Efecto de repulsión y atracción en el área de cobertura para tarea depredador-presa. 77

4.20. Efecto de repulsión y atracción en el tiempo de captura para tarea depredador-presa. 77

4.21. Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$. 85

4.22. Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$. 86

4.23. Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$. 87

4.24. Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$. 88

4.25. Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$. 89

4.26. Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$. 90

4.27. Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$. 91

4.28. Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$. 92

4.29. Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$. 93

4.30. Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$. 94

4.31. Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$. 95

4.32. Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$. 96

4.33. Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.05$ y $r_a = 0.2$. 101

4.34. Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$. 101

4.35. Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.05$ y $r_a = 1$. 102

4.36. Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$. 102

4.37. Efecto de repulsión y atracción en el tiempo de ejecución para tarea de transporte de objetos, basado en simulaciones. 104

4.38. Efecto de repulsión y atracción en el tiempo de ejecución para tarea de transporte de objetos, basado en implementación física. 105

A.1. Vista explosionada de la estructura del robot. 115

A.2. Sensor ultrasónico HC-SR04. 115

A.3. Sensor infrarrojo TCRT5000. 116

A.4. Resistencia dependiente de luz (LDR). 116

A.5. Alimentación y encendido del robot. 118

Índice de tablas

2.1. Parámetros del robot. 40

3.1. Especificaciones de las zonas de repulsión, orientación, atracción e influencia en función a los sensores empleados. 47

3.2. Radios permitidos en las zonas de percepción. 47

3.3. Velocidades en zonas de percepción en experimentos físicos. 47

4.1. Resultados de tarea depredador-presa, basados en simulaciones. . . . 65

4.2. Resultados de tarea depredador-presa, basados en implementación física. 75

4.3. Estadísticas de la regresión para modelos de tarea depredador-presa. . 78

4.4. Relación de coeficientes para modelos de tarea depredador-presa. . . . 79

4.5. Correlación de parámetros para modelos de tarea depredador-presa. . 80

4.6. Resultados de tarea de transporte de objetos con 5 *BugBot*, basados en simulación. 97

4.7. Resultados de tarea de transporte de objetos con 10 *BugBot*, basados en simulación. 98

4.8. Resultados de tarea de transporte de objetos con 20 *BugBot*, basados en simulación. 99

4.10. Estadísticas de la regresión para tarea de transporte de objetos. . . . 100

4.9. Resultados de tarea de transporte de objetos con 3 *BugBot*, basados en implementación física. 103

4.11. Estadísticas de regresión para la superficie de comportamiento de la Figura 4.37c. 104

4.13. Correlación de parámetros de tarea de transporte de objetos. 105

4.12. Relación de coeficientes para modelos de tarea de transporte de objetos. 106

A.1. Características de módulo ESP32. 116

A.2. Características de sensor ultrasónico HC-SR04. 117

A.3. Características de sensor infrarrojo TCRT5000. 117

A.4. Características de LDR. 118

A.5. Características de motor CC. 118

Nomenclaturas

SMR	Sistemas Multi-Robot.
RE	Robótica de enjambre.
IE	Inteligencia de enjambre.
RAOI	Repulsión, Atracción, Orientación e Influencia.
ZOR	Zona de repulsión.
ZOO	Zona de orientación.
ZOA	Zona de atracción.
ZOI	Zona de influencia.
CC	Corriente continua.
LDR	Resistencia dependiente de la luz.

Agradecimientos

Agradezco al Dr. Luis Martín Torres Treviño, mi Director de Tesis, por su tiempo, apoyo y orientación en el desarrollo de esta Tesis.

Al Comité de Tesis: Dr. Juan Ángel Rodríguez Liñán, Dr. Romeo Sánchez Nigenda, Dr. Joaquín Gutiérrez Jagüey y Dr. Jonatán Peña Ramírez, por sus comentarios y observaciones que ayudaron a mejorar el contenido de esta Tesis. A los profesores del Posgrado en Ingeniería Eléctrica, por sus enseñanzas durante mis estudios.

Al Consejo Nacional de Ciencia y Tecnología (CONACYT), la Universidad Autónoma de Nuevo León (UANL) y la Facultad de Ingeniería Mecánica y Eléctrica (FIME), por otorgarme la beca y los medios para realizar esta Tesis.

A mis compañeros de posgrado, por su apoyo en el transcurso de mis estudios.

A mis amigos, por su apoyo, consejos y motivación en el transcurso de mi vida.

A mi familia, por su apoyo constante y motivación para perseverar con mis estudios y, en especial, a mi madre Lic. María Francisca Ordaz Rivas, mi abuela Sra. Ma. de Jesús Rivas Ávila y mi abuelo Sr. Miguel Ordaz Hernández por sus enseñanzas, guía y amor a lo largo de mi vida.

Gracias a Dios, por escuchar mis oraciones, darme fuerza y bendecirme a mí y todas las personas que me rodean.

Resumen

M.C. Erick de Jesús Ordaz Rivas.

Candidato para obtener el grado de Doctorado en Ingeniería Eléctrica.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Colaboración emergente en enjambres de robots, con reglas inspiradas en el comportamiento de animales sociales.

Número de páginas: 193.

Objetivos y método de estudio: La Robótica de Enjambre, es un enfoque para la coordinación de múltiples robots, que toma inspiración de los comportamientos complejos observados en los sistemas naturales de enjambre, donde la funcionalidad del sistema surge de la interacción local de cada agente con sus vecinos y el entorno, y, como consecuencia, se espera que surja un comportamiento colectivo sin un líder, un control jerárquico o información global. Sin embargo, a pesar de que existen una gran cantidad de propuestas de algoritmos de enjambre con aproximaciones a comportamientos colectivos, relativamente pocos estudios basados en simulaciones consideran las limitaciones de los sensores reales de los robots, y sólo algunos algoritmos de comportamiento se transfieren a sistemas robóticos reales. Además, la implementación de un esquema de enjambres en robots físicos es un tema de investigación abierto.

En este trabajo de tesis, se propone un algoritmo de navegación de cada individuo, que sólo depende de cuatro parámetros relacionados con un modelo general del comportamiento de los animales sociales llamados: repulsión, atracción, orientación e influencia; los cuales tienen una mayor influencia en el

comportamiento de enjambres de robots, cuando se realizan tareas colaborativas. Para esto, se desarrolló una plataforma de simulación de comportamientos de enjambre y se establecieron métricas de evaluación, para probar los efectos de los parámetros en el comportamiento del enjambre. Además, se construyeron prototipos de robots móviles, donde se implementaron las reglas de comportamiento para la ejecución de tareas de depredador-presa y transporte de objetos.

CONTRIBUCIONES Y CONCLUSIONES: El modelo propuesto se inspira por reglas simples propuestas por Couzin (repulsión, atracción y orientación) [1], las cuales establecen un comportamiento colectivo específico, en adición, se incluye una cuarta regla llamada "influencia", que consiste en la detección de estímulos específicos para relacionarlos con alguna tarea. Un aspecto importante, es que la colaboración surge de la interacción de los robots como enjambre, sin control centralizado y en ausencia de líderes. En general, sólo se implementan tres reglas: evitar colisiones (repulsión), seguir a otros robots (atracción) y seguir a la influencia al cambiar el estímulo. La principal contribución consiste en la implementación de reglas de comportamiento basadas en parámetros *RAOI* a un enjambre de robots físico, sin intervención humana o métodos centralizados. Además, se obtienen modelos de índice para cuantificar el impacto de los parámetros en el enjambre. Las reglas de comportamiento local se diseñaron inspiradas en el comportamiento de las formas de vida social observadas en animales, por lo que se genera un comportamiento similar en nuestra manada de robots.

Firma del asesor: 

Dr. Luis Martín Torres Treviño

Capítulo 1

Introducción

1.1 Motivación

Los Sistemas Multi-Robot (*SMR*), son una extensión en la investigación de robots móviles y surgen para superar la falta de capacidad de los robots individuales para realizar tareas [2, 3]; que, para completarse de manera eficiente, necesitan colaboración entre grupos de robots [4].

Existen dos enfoques principales para el diseño de *SMR* [5]. El primero es un sistema centralizado, donde un componente central coordina todos los robots y sus tareas [6]. La ventaja principal es su facilidad de implementación; sin embargo, hay que considerar ciertas desventajas. Por ejemplo, si el componente principal deja de funcionar, todo el sistema falla; además, al agregar más robots al sistema, la carga de procesamiento en la unidad central aumenta. En consecuencia, se implica un alto costo computacional, porque se requiere una arquitectura sofisticada de hardware y software, así como conocer en cada momento la posición de cada robot.

Por otro lado, la Robótica de Enjambre (*RE*), es un enfoque para la coordinación de *SMR*, que toma inspiración de los comportamientos complejos observados en los sistemas naturales de enjambre (por ejemplo, enjambres de

abejas, colonias de hormigas, bancos de peces y bandadas de aves, entre otros) [7, 8, 9, 10]. Estos seres tienen habilidades individuales muy simples, pero al vivir en grupos grandes, surgen comportamientos colectivos que promueven la realización de tareas de manera eficiente. A este tipo de comportamientos se les conoce como Inteligencia de Enjambre (*IE*) [11].

Los algoritmos basados en *IE* tienen muchas similitudes con los comportamientos en la *RE*, ya que ambos toman inspiración de sistemas de enjambre de la naturaleza [12]. Estas técnicas también se utilizan en el diseño de robots y controladores, algunas metaheurísticas generalizadas son: la computación evolutiva [13], algoritmos genéticos [14], robótica evolutiva [15], optimización por enjambre de partículas (*PSO*) [16], optimización por colonia de hormigas [17], entre otras [18, 19, 20, 21].

A diferencia de los *SMR* con esquemas de control centralizado, los sistemas de *RE* tienen como objetivo construir sistemas descentralizados totalmente distribuidos con agentes simples, en los que la funcionalidad del sistema surge de la interacción local de cada agente con sus vecinos y el entorno, y, como consecuencia, se espera que surja un comportamiento colectivo sin un líder, un control jerárquico o información global [22]. Por lo cual, el comportamiento colectivo es un aspecto importante en un sistema de enjambre en general y la *RE* en particular. En la *RE*, los robots individuales exhiben un comportamiento que se basa en un conjunto de reglas locales que puede variar desde un mapeo reactivo simple entre las entradas del sensor y las salidas del actuador para elaborar algoritmos locales. En Schranz et al. [23] se discuten los comportamientos fundamentales observados en el campo de la *RE*.

Otro aspecto importante de la *RE* es el papel de la auto-organización [24, 25], el cual es un proceso presente en los sistemas de enjambres de la naturaleza [26]. Por medio de la auto-organización se presentan comportamientos emergentes de coordinación, cooperación y colaboración [27] De manera que, la emergencia de colaboración ocurre cuando los agentes de un sistema se organizan

espontáneamente y sin instrucciones explicitas para la consecución de objetivos deseados; remplazando el control jerárquico centralizado como mecanismo de colaboración tradicional [28]. Además de la auto-organización, algunas ventajas de la *RE* son robustez, escalabilidad y flexibilidad; que son propiedades deseadas en enjambres de robots [29, 30]. Se dice que un sistema es robusto si puede continuar funcionando correctamente, incluso si uno o más agentes dejan de funcionar, debido a fallas internas o condiciones en el entorno. Es escalable si puede adaptarse a cambios en el número de agentes. Por último, es flexible si los parámetros internos de los agentes pueden ajustarse con la aparición de cambios en el entorno.

Debido a las características presentes en los enjambres de robots, se han propuesto numerosas estrategias para el desarrollo de una gran variedad de tareas, por ejemplo, agregación, ensamble y agrupamiento de objetos, exploración, navegación, manipulación colaborativa, entre otras [31, 32, 33, 34]. De este modo, por medio de la *RE*, es posible resolver una amplia gama de problemas, que involucran grandes cantidades de tiempo y espacio, que pueden ser peligrosas o inaccesibles para los humanos e incluso para los robots [35].

Hoy en día, relativamente pocos estudios basados en simulaciones consideran las limitaciones de los sensores reales de los robots y la mayor parte de los algoritmos de comportamiento, no se transfieren a ningún sistema robótico real [36, 37]. Además, las aplicaciones del mundo real de la *RE* son pocas en el mercado y la mayoría de ellas se basan en herramientas de simulación, por lo que el gran logro de las aplicaciones de robótica móvil se espera en los próximos años [38]. Algunos ejemplos de aplicaciones potenciales se encuentran en la minería, búsqueda de sobrevivientes en deslizamientos de tierra, detección de derrames de petróleo en el mar, monitoreo de recursos, vigilancia, aplicaciones militares, exploración de otros planetas, entre otros [39, 40].

Para la aplicación a equipos de robots móviles, el comportamiento de agentes está restringido por ciertas limitaciones (por ejemplo, velocidades, energía

disponible, estimado de posiciones de agentes, capacidades de cómputo y comunicación limitadas) y por la necesidad de preservar la unidad en el grupo. Para los robots móviles, esta tarea generalmente se considera un problema de congregación (*flocking*), que consiste en preservar un grupo geométricamente distinguido formado por los agentes en el espacio Euclidiano [6]. Por otra parte, los sistemas de *RE* están diseñados para ser autónomos y tomar decisiones de manera distribuida. En general, estas características se consideran positivas, pero restringe el grado de control sobre el sistema. Por lo que se han propuesto varias alternativas para interactuar con el enjambre y gobernar en su comportamiento, sin modificar su autonomía y auto-organización [41, 42].

En este trabajo de tesis, se presenta un modelo que incluye la cinemática y dinámica de cada robot en el enjambre, una configuración de los sensores para detectar robots vecinos y un algoritmo único, para que cada robot decida sus acciones en el enjambre. El algoritmo sólo depende de cuatro parámetros relacionados con un modelo general del comportamiento de los animales sociales llamados: repulsión, atracción, orientación e influencia (*RAOI*); los cuales permiten gobernar el comportamiento de enjambres de robots cuando se realizan tareas colaborativas.

1.2 Antecedentes

El comportamiento de bandada o *flocking*, puede definirse como el comportamiento colectivo de un gran número de agentes, que interactúan con un objetivo grupal común. Por lo general, la dirección de movimiento surge de las interacciones entre los agentes en el enjambre. Estas interacciones se basan en la llamada "*regla del vecino más cercano*", donde los agentes ajustan su movimiento basándose únicamente en sus vecinos más cercanos [43]. La definición de *flocking* se inspira en la observación de especies biológicas como parvadas de aves, bancos de

peces o enjambres de insectos [44].

En el trabajo de Reynolds [45], se propuso uno de los primeros modelos de movimiento en grupos de animales, que exhiben un comportamiento de *flocking*. Reynolds, define una bandada de aves como un grupo de objetos *"boids"* (bird-oid objects) que presentan un movimiento basado en tres reglas básicas de comportamiento:

- Separación (evitar colisiones), cada agente preserva distancias mínimas entre los agentes.
- Alineación (coincidencia de velocidad), cada agente conserva su dirección y velocidad lo más similar posible a las velocidades de los agentes más cercanos.
- Cohesión (agrupamiento), cada agente se mantiene lo más cerca posible de los agentes cercanos.

A pesar de la simplicidad de estas reglas, proporcionan al sistema las condiciones necesarias para preservar el enjambre. En este modelo, los individuos simulados tienen acceso directo a la posición, orientación y velocidad exactas de todos los objetos a su alrededor. Por lo que este tipo de enfoque, representa una percepción alejada de las capacidades reales de los animales sociales en la naturaleza, que conducen a la emergencia de un comportamiento colectivo.

Desde entonces, distintos trabajos han tratado de reproducir el comportamiento de enjambre (*flocking*) [46]. Por ejemplo, Vicsek et al. [47], modela un enjambre de partículas puntuales, que se mueven a velocidad constante en la dirección promedio de los vecinos locales, incluyendo perturbaciones aleatorias. Hartman y Benes [48] presentan otra variante del modelo original, al agregar una fuerza complementaria a la regla de alineación, que llaman un cambio de liderazgo. Además, se han realizado otros enfoques con agentes informados o líderes fijos [49, 50, 51]. En Gazi y Passino [43] se presenta una lista de algunos problemas

referentes a este marco, se detalla la consideración de los objetivos de los enjambres y las posibles soluciones teóricas.

Por otro lado, Couzin et al. [1], proponen un modelo que imita de manera más realista el movimiento de bandadas de aves y simula el comportamiento de los individuos con tendencias de repulsión, orientación y atracción, basadas en las características de las especies biológicas. Estas características son representadas por zonas que rodean a cada individuo. Para este modelo se establecen dos reglas de comportamiento: (1) Cada individuo intenta mantener una distancia mínima con sus vecinos en todo momento. (2) Si los individuos no están evadiendo a otros, entonces tienden a atraerse hacia otros y alinearse con sus vecinos.

De esta manera, las reglas de separación, alineación y cohesión se han utilizado para simular enjambres de robots, ejecutando una gran cantidad de tareas como: exploración, navegación, cobertura, transporte de objetos, pastoreo, entre otros [52]. De igual modo, en trabajos como Turgut et al. [53], se han logrado obtener comportamientos de *flocking*, en enjambres de robots con capacidades de detección simples.

En otros trabajos, se han buscado formas de control para influir en el comportamiento del enjambre. En Pac et al. [54] y Pimenta et al. [55], se propone un marco basado en la hidrodinámica de partículas, en enjambres modelados como fluidos para el control de comportamientos. Kira y Potter [56], utilizan un aprendizaje basado en casos para producir un modelo generalizado de los parámetros y propiedades globales, para situaciones específicas. Por otra parte, en trabajos como Goodrich et al. [57] y Jung et al. [58], desarrollaron un control que responde a la presencia de líderes o mediadores con interacción humana, manteniendo la conectividad del enjambre.

En estudios como Torney et al. [59], se mostraron las ventajas de la señalización para escalar gradientes ambientales. En Bashyal y Venayagamoorthy

[60], se estudió el comportamiento del enjambre con la interacción humana, considerando la localización de una fuente de radiación. Martinez et al. [61], proponen una estrategia basada en el movimiento Browniano, en el que cada robot se modela como una partícula cuyo movimiento está influenciado por señales del medio ambiente. Sin embargo, en la mayoría de los casos, el enfoque de control implementado para influir en el comportamiento de enjambre, requiere interacción humana, información global o conocer la posición de los miembros del enjambre, por lo que se pierde la propiedad de descentralización en el enjambre.

Aunque la *RE* es un campo de investigación relativamente joven y no ha sido ampliamente aceptado en la industria, se han diseñado y desarrollado distintas plataformas para probar y analizar algoritmos de enjambre, que son clasificadas de acuerdo con el entorno en el que se utilizan: terrestre, aéreo, acuático o espacial [23]. En relación a las aplicaciones de enjambre en la industria, se han desarrollado trabajos de agricultura [62, 63, 64], auxilio y rescate [65, 66, 67, 68], exploración y monitoreo [69, 70] y navegación [71]. Sin embargo, aún no se usan de forma íntegra algoritmos de enjambre, en cambio, se prefiere utilizar partes de los algoritmos y adaptarlos a la aplicación, por lo que muchos proyectos industriales todavía dependen del control centralizado.

Un desafío importante es la aplicación de la *RE* al campo de la ingeniería. Brambilla et al. [72], analizan la literatura y proponen definiciones y conceptos que contribuyen al uso de la *RE* al campo de la ingeniería. Del mismo modo, Garattoni y Birattari [73], presentan una investigación sobre trabajos existentes en *RE* desde una perspectiva de ingeniería. Otro desafió, es el modelado del sistema para poder simular enjambres de robots de manera efectiva. Hamman y Schmickl [74], promueven el modelado matemático formal de sistemas distribuidos y auto-organizados; mientras que en Correll y Hamann [75], contribuyen con otro trabajo sobre el modelado en la *RE*, donde proporcionan una visión generalizada de diferentes aspectos en los sistemas de enjambre (dinámica de población, colaboración, distribución de los miembros del enjambre, decisión colectiva y

optimización).

Hoy en día, el uso de enjambres de robots ha aumentado y se ha convertido en un área de investigación muy interesante, debido a sus propiedades de auto-organización, robustez, escalabilidad y flexibilidad, además de su capacidad de resolver gran cantidad de tareas; por lo cual se han propuesto muchas metodologías para el control de enjambres de robots. Sin embargo, en la actualidad, las aplicaciones de la *RE* todavía son pocas, aunque la investigación se ha llevado a cabo durante varias décadas, todavía no se ha producido un gran avance en este campo.

1.3 Descripción del problema

A pesar de que existen una gran cantidad de propuestas de algoritmos de enjambre con aproximaciones a comportamientos colectivos, las configuraciones presentadas en la literatura a menudo son complejas con entornos muy específicos, donde los agentes están diseñados para ser más sensibles a entradas particulares. Si bien se están aportando resultados valiosos a la comunidad, es necesario desarrollar modelos con un diseño más general y simple.

En la actualidad, relativamente pocos estudios basados en simulaciones consideran las limitaciones de los sensores reales de los robots, y la mayor parte de los algoritmos de comportamiento, no se transfieren a ningún sistema robótico real [36, 37]. Las aplicaciones del mundo real de la *RE* son pocas en el mercado y la mayoría de ellas son en realidad herramientas de simulación [38]. Además, la implementación de un esquema de enjambres en robots físicos es un tema de investigación abierto. A menudo, en la industria se utiliza el término "enjambre" únicamente para implicar el alto número de agentes, pero generalmente no implementan algoritmos basados en la *RE*. Las implementaciones descuidan la idea

principal de la *RE*, que es la toma de decisiones distribuida que conduce a un comportamiento emergente auto-organizado. Con el fin de facilitar la implementación física, es necesaria la creación de esquemas que se basen en la información local percibida y no dependan de la posición de los demás robots, es decir, sin el uso de métodos centralizados.

Basados en Couzin [1], se propone un algoritmo de navegación de cada individuo con tendencias de repulsión, orientación y atracción. Añadiendo un cuarto parámetro llamado influencia, el cual permite estimular al enjambre para enfatizar en una tarea específica. Por medio de cambios en los parámetros de repulsión, atracción, orientación e influencia, es posible gobernar en el comportamiento de un enjambre de robots. Los parámetros *RAOI*, explican el comportamiento de un enjambre en la naturaleza y son incluidos en ecuaciones matemáticas. Estas ecuaciones son transformadas en políticas de comportamientos que son programadas en cada uno de los robots del enjambre. El modelo propuesto incluye la cinemática y la dinámica de cada robot en el enjambre, una configuración de los sensores para detectar robots vecinos y un algoritmo único para que cada robot decida sus acciones en el enjambre. El desarrollo de nuevas reglas de comportamiento consiste en modificar los parámetros *RAOI*; por lo que existe una relación entre la configuración de parámetros, que puede adoptar un enjambre, con el comportamiento colectivo emergente al ejecutar una tarea. El enfoque propuesto espera promover en el enjambre la aparición de comportamientos colectivos.

Se proponen dos tareas con el fin de analizar el comportamiento del enjambre con el modelo propuesto. En primer lugar, se aborda una tarea de tipo depredador-presa, donde una bandada de robots depredadores colabora para capturar a un robot presa. En segundo lugar, se aborda una tarea de transporte de objetos, donde el enjambre de robots manipula objetos distribuidos espacialmente y los agrupa en una zona de descarga. El objetivo es estimular las propiedades del enjambre mediante los parámetros *RAOI* y cuantificar las características del

comportamiento colaborativo emergente. Mediante simulaciones computacionales, se prueba el modelo propuesto para conocer el impacto de los parámetros; además, el algoritmo se implementa y se valida experimentalmente en una mandada de robots móviles con limitaciones de hardware y software.

1.4 Hipótesis

El cambio en los parámetros *RAOI* en un enjambre de robots, mantiene las características descentralizadas del sistema y provoca emergencia de colaboración entre los robots, que se refleja en una variación en el tiempo de ejecución de una tarea específica.

1.5 Objetivos

1.5.1 Objetivo general

Por medio de indicadores de eficiencia, analizar el desempeño en un enjambre de robots en la ejecución de tareas, al aplicar reglas de comportamiento basadas en los parámetros *RAOI*, asociados a señales sensoriales locales.

1.5.2 Objetivos particulares

- Generar reglas de comportamiento locales, inspiradas en computación de enjambres basado en los parámetros *RAOI*.
- Desarrollar una plataforma de simulación para tareas colaborativas, incluyendo

las reglas de comportamiento de enjambre.

- Probar tareas de depredador-presa y transporte de objetos, en el simulador de enjambres y establecer indicadores de eficiencia.
- Explorar en un enjambre de robots físicos el efecto de los parámetros *RAOI*, en tareas de depredador-presa y transporte de objetos, así como su eficiencia.

1.6 Contribuciones

Este trabajo propone mantener las propiedades del enjambre utilizando reglas de comportamiento simples, basadas en parámetros *RAOI* e información local. Por lo tanto, algunas contribuciones del enfoque propuesto se pueden resumir de la siguiente manera:

- El esquema de navegación de cada individuo permite la ejecución de tareas colaborativas en un enjambre en ausencia de líderes, control centralizado e información global.
- Se obtienen modelos de índice para cuantificar el impacto de los parámetros *RAOI* en el enjambre.
- El enfoque se implementa en un enjambre experimental de robots físicos con capacidades de hardware y software limitadas.

El modelo propuesto se inspira por reglas simples propuestas por Couzin [1], cuyas ecuaciones establecen un comportamiento colectivo específico y se regulan con tres parámetros (repulsión, atracción, orientación). En adición se incluye un cuarto parámetro llamado “influencia”, que consiste en la detección de estímulos específicos. De modo que, algunas tareas sólo pueden incluirse mediante la

inclusión de ciertos estímulos de influencia y configuraciones paramétricas de comportamiento, considerando los parámetros de repulsión, atracción y orientación. Un aspecto importante presentado en este trabajo de tesis, es que la colaboración surge de la interacción de los robots como enjambre. Debido a que en la implementación física no es posible conocer la orientación de los vecinos, en general, sólo se implementan tres reglas: (1) evitar colisiones (repulsión), (2) seguir a otros robots (atracción) y (3) seguir a la influencia al cambiar el estímulo. Todos los robots exhiben cooperación para realizar tareas colaborativas.

1.7 Organización de la tesis

El Capítulo 2, presenta a los miembros del enjambre conformado por robots móviles de tipo diferencial, cuyo comportamiento se describe mediante sus modelos cinemático y dinámico. Posteriormente, se muestra la arquitectura de cada robot y las limitaciones sensoriales de cada uno, se explican las reglas de comportamiento y cómo opera cada robot al interactuar con sus vecinos y su entorno. En el Capítulo 3, se describe el desarrollo de las simulaciones y experimentos. Además, se describen los casos experimentales planteados para el análisis del comportamiento del enjambre. En el Capítulo 4, se muestran los resultados obtenidos de las simulaciones y experimentos. En el Capítulo 5, se señalan las conclusiones y propuestas de trabajo futuro.

En el Apéndice A, se añade un reporte técnico de los robot “BugBot”, que muestra sus características y funcionamiento. Finalmente, en los Apéndices B y C, se presentan los códigos de las reglas de comportamiento de cada individuo implementados en *Arduino*^{TM} y *Scilab*, respectivamente.

Capítulo 2

Robótica de enjambres

En este capítulo, se describen las características del enjambre, así como las reglas de comportamiento implementadas. Al inicio, se presenta el modelo matemático de un robot móvil con configuración diferencial para representar a cada miembro del enjambre. Posteriormente, se describe la arquitectura de los robots, capacidad sensorial y las limitaciones de cada uno de ellos. Después, se desarrollan reglas de comportamiento sencillas, tomando como base el modelo de Couzin [1] y adaptándolo a robots simples con movimiento en dos dimensiones. Estas reglas son implementadas en cada robot para cambiar su dirección en base a los parámetros de repulsión y atracción, y con la interacción de factores de influencia. Por último, se describe el proceso para la validación numérica y experimental de las reglas propuestas.

2.1 Modelo matemático de robots

Para probar algunos conceptos sobre el rendimiento del enjambre bajo las reglas de comportamiento locales propuestas, se requieren simulaciones de un enjambre de robots lo más cercano posible a la realidad. Cada miembro del enjambre está representado por un robot móvil con configuración diferencial, como se muestra en

la Figura 2.1. Tomando como base el trabajo de Bara y Dale [76], se obtienen los modelos cinemático y dinámico de esta configuración, así como el modelo de los motores de CC incorporados en las ruedas [77, 78], lo cual también se ilustra en [79]. Existen otros modelos que consideran interacciones dinámicas complejas como Liu et al. [80, 81]; sin embargo, en este trabajo se considera un modelo simplificado.

2.1.1 Modelo cinemático

En la Figura 2.1, el vector de coordenadas generalizadas $\mathbf{q}_g = [x_g\ y_g\ \theta]^T$ especifica la posición y orientación del centro de masa G del robot móvil con respecto al marco cartesiano inercial $\{I\}$.

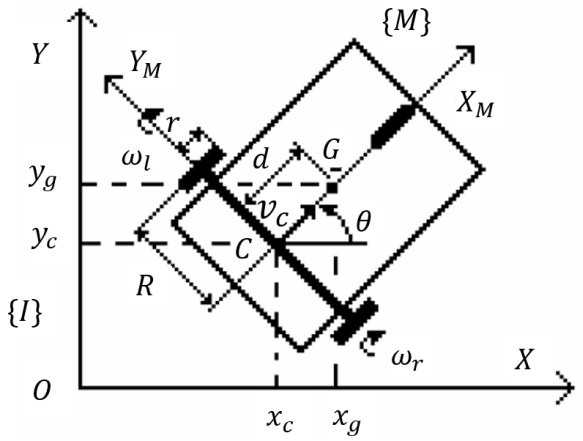


Figura 2.1: Robot móvil con configuración diferencial.

Para esta configuración, las velocidades angulares ω_r y ω_l aplicadas a las ruedas derecha e izquierda, respectivamente, pueden ser diferentes y por lo tanto el movimiento del robot en el plano horizontal es compuesto.

$$\begin{bmatrix} \upsilon_c \\ \omega_c \end{bmatrix} = A \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}, \tag{2.1}$$

con

$$A = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{2R} & -\frac{r}{2R} \end{bmatrix}.$$

La Ecuación (2.1) es válida siempre y cuando el robot satisfaga las condiciones de rodadura pura; donde υ_c y ω_c son las magnitudes de las velocidades lineal y angular, respectivamente, del marco conectado al robot $\{M\}$, R es la distancia entre las ruedas y el eje central del robot C y r es el radio de las ruedas.

Por otro lado, el vector de velocidades $\mathbf{v} = [\upsilon_c \; \omega_c]^T$ del marco móvil $\{M\}$, puede ser transformado a velocidades cartesianas $\dot{\mathbf{q}}_g = \begin{bmatrix} \dot{x}_g & \dot{y}_g & \dot{\theta} \end{bmatrix}^T$ del marco inercial $\{I\}$, mediante la Ecuación (2.2) con matriz Jacobiana $\mathbf{S}(\mathbf{q}_g)$ (Ec. 2.3); donde d es la distancia entre el centro de masas G y el origen del marco móvil $\{M\}$.

$$\dot{\mathbf{q}}_g = \mathbf{S}(\mathbf{q}_g)\mathbf{v}, \tag{2.2}$$

$$\mathbf{S}(\mathbf{q}_g) = \begin{bmatrix} \cos(\theta) & -d\sin(\theta) \\ \sin(\theta) & d\cos(\theta) \\ 0 & 1 \end{bmatrix}. \tag{2.3}$$

Finalmente, de acuerdo con las Ecuaciones (2.1) y (2.2), el modelo cinemático del robot puede ser escrito de forma explícita:

$$\begin{bmatrix} \dot{x}_g \\ \dot{y}_g \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\theta) - \frac{rd}{2R}\sin(\theta) & \frac{r}{2}\cos(\theta) + \frac{rd}{2R}\sin(\theta) \\ \frac{r}{2}\sin(\theta) + \frac{rd}{2R}\cos(\theta) & \frac{r}{2}\sin(\theta) - \frac{rd}{2R}\cos(\theta) \\ \frac{r}{2R} & -\frac{r}{2R} \end{bmatrix} \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix}. \tag{2.4}$$

2.1.2 Modelo dinámico

Dado que la posición $\overrightarrow{P}_g$ del centro de masas G se escribe en forma polar como:

$$\overrightarrow{P}_g = \overrightarrow{P}_c + d\ e^{j\theta}, \tag{2.5}$$

donde $\overrightarrow{P}_c$ es la posición del origen del marco móvil; la aceleración lineal $\overrightarrow{a}_g$ del centro de masas G, es obtenida por la derivada de segundo orden de la Ecuación (2.5):

$$\overrightarrow{a}_g = (\dot{v}_c - d\dot{\theta}^2)e^{j\theta} + j(d\ddot{\theta} + v_c\dot{\theta})e^{j\theta}. \tag{2.6}$$

El primer término es el componente radial que tiene la misma dirección que el vector de desplazamiento y el segundo es el componente tangencial; de modo que, el movimiento hacia adelante del centro de masa G del robot, se describe por la Segunda Ley de Newton:

$$m\dot{v}_c - md\dot{\theta}^2 = F_c, \tag{2.7}$$

y el movimiento de rotación por:

$$(I + md^2)\ddot{\theta} + mdv_c\dot{\theta} = \tau_c, \tag{2.8}$$

donde I es el momento de inercia y m es la masa del robot.

Por otra parte, la fuerza F_c y par τ_c son producidos por los pares τ_r y τ_l aplicados en la rueda derecha e izquierda, respectivamente, es decir:

$$\begin{aligned} F_c &= \tfrac{1}{r}\left(\tau_r + \tau_l\right), \\ \tau_c &= \tfrac{R}{r}\left(\tau_r - \tau_l\right). \end{aligned} \tag{2.9}$$

Tomando las Ecuaciones (2.7), (2.8) y (2.9), y considerando fricción viscosa en el modelo dinámico del robot, este se representa en forma compacta como:

$$M\dot{\mathbf{v}} + H(\mathbf{v}) + Z\mathbf{v} = B\tau, \tag{2.10}$$

con

$$M = \begin{bmatrix} m & 0 \\ 0 & I + md^2 \end{bmatrix},\ H(\mathbf{v}) = \begin{bmatrix} -md\dot{\theta}^2 \\ mdv_c\dot{\theta} \end{bmatrix},$$

$$Z = \begin{bmatrix} \zeta_{11} & \zeta_{12} \\ \zeta_{21} & \zeta_{22} \end{bmatrix},\ B = \begin{bmatrix} \frac{1}{r} & \frac{1}{r} \\ \frac{R}{r} & -\frac{R}{r} \end{bmatrix},$$

donde M es la matriz de inercia, simétrica y definida positiva, $H(\mathbf{v})$ es el vector de Coriolis y fuerzas centrífugas, Z es una matriz con coeficientes de fricción viscosa ($\zeta_{11}, \zeta_{12}, \zeta_{21}$ y ζ_{22}), B es la matriz de entrada, y $\tau = [\tau_r\ \tau_l]^T$ es el vector de entrada de los pares de las ruedas.

2.1.3 Modelo del actuador

Por otro lado, se considera que los actuadores de las ruedas están dados por motores de CC; por tanto, basados en la teoría de perturbación singular, el modelo se obtiene bajo el supuesto de "pequeños"valores de inductancia de armadura ($L_{a,i}\frac{di_{a,i}}{dt} \approx 0$) para ambas ruedas con motor de CC [77, 78], es decir:

$$J_{m,i}\dot{\omega}_i + b_i\omega_i + \frac{k_{\tau,i}k_{fem,i}}{R_{a,i}}\omega_i + \frac{1}{N_i^2}\tau_i = \frac{k_{\tau,i}}{N_i R_{a,i}}u_i, \tag{2.11}$$

donde $J_{m,i}$ es la inercia del rotor y rueda, ω_i y $\dot{\omega}_i$ es la velocidad y aceleración angular de la rueda, respectivamente, b_i es una constante de fricción viscosa, $L_{a,i}$ es la inductancia de armadura, τ_i es el par de carga aplicado sobre la caja de cambios en la rueda, $R_{a,i}$ es la resistencia de armadura, $i_{a,i}$ es la corriente de armadura, $k_{\tau,i}$ es la constante de par-motor, $k_{fem,i}$ es la constante de fuerza contraelectromotriz, N_i es la relación de reducción de engranes, u_i es el voltaje de la armadura como entrada e $i = \{r, l\}$ es el subíndice para la rueda derecha e izquierda, respectivamente.

Para simplificar la dependencia de los parámetros del motor, la Ecuación (2.11) puede reescribirse como:

$$\tau_{s,i}\dot{\omega}_i + \omega_i + k_{L,i}\tau_i = k_{s,i}u_i, \tag{2.12}$$

donde

$$\tau_{s,i} = \frac{J_{m,i}}{b_i + \frac{k_{\tau,i}k_{emf,i}}{R_{a,i}}},\ k_{L,i} = \frac{\frac{1}{N_i^2}}{b_i + \frac{k_{\tau,i}k_{emf,i}}{R_{a,i}}},\ k_{s,i} = \frac{\frac{k_{\tau,i}}{N_i R_{a,i}}}{b_i + \frac{k_{\tau,i}k_{emf,i}}{R_{a,i}}},$$

para cada rueda $i = \{r, l\}$.

De esta forma, el modelo de las dos ruedas con actuadores se representa en forma compacta

$$T\begin{bmatrix} \dot{\omega}_r \\ \dot{\omega}_l \end{bmatrix} + \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} + K_L\begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} = K_s\begin{bmatrix} u_r \\ u_l \end{bmatrix}, \tag{2.13}$$

con

$$T = \begin{bmatrix} \tau_{s,r} & 0 \\ 0 & \tau_{s,l} \end{bmatrix},\ K_L = \begin{bmatrix} k_{L,r} & 0 \\ 0 & k_{L,l} \end{bmatrix},\ K_s = \begin{bmatrix} k_{s,r} & 0 \\ 0 & k_{s,l} \end{bmatrix}.$$

De modo que, la Ecuación (2.13) se puede ajustar en términos del vector de velocidades del robot $\mathbf{v} = [\upsilon_c\ \omega_c]^T$, al considerar la Ecuación (2.1)

$$TA^{-1}\dot{\mathbf{v}} + A^{-1}\mathbf{v} + K_L \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix} = K_s \begin{bmatrix} u_r \\ u_l \end{bmatrix}. \tag{2.14}$$

Por último, manipulando las Ecuaciones (2.10) y (2.14), se obtiene el modelo dinámico completo del robot con actuadores, dado por:

$$\left(M + BK_L^{-1}TA^{-1}\right)\dot{\mathbf{v}} + \left(H(\mathbf{v}) + Z\mathbf{v} + BK_L^{-1}A^{-1}\mathbf{v}\right) = BK_L^{-1}K_s u, \tag{2.15}$$

donde $u = [u_r\ u_l]^T$ es el vector de las entradas de voltaje para los motores de CC derecho e izquierdo.

2.2 Arquitectura de los robots

Debido a que se abordan dos casos de estudio, se construyeron dos modelos de robot con las capacidades mínimas requeridas en función a la tarea a ejecutar (depredador-presa o transporte de objetos). Ambos modelos tienen arquitectura homogénea (respectiva a cada enjambre) con configuración diferencial, donde la cinemática y dinámica se describen por las Ecuaciones (2.4) y (2.15). Cada robot tiene sensores de proximidad para obtener la distancia y detectar otros robots u obstáculos (sensores ultrasónicos e infrarrojos), además *LDR*s (*resistencia dependiente de la luz*) que permiten obtener la información de la luz percibida en el entorno. Incluyen también un magnetómetro para conocer la orientación del robot.

2.2.1 Terran

Los robots *Terrans* cuentan con un dispositivo $Arduino^{TM} micro$ y un módulo *Bluetooth*®, que se ha utilizado para tener comunicación entre el usuario y el *Terran*, y establecer una configuración de valores paramétricos. La Figura 2.2, ilustra la ubicación de los sensores y dispositivos en el robot.

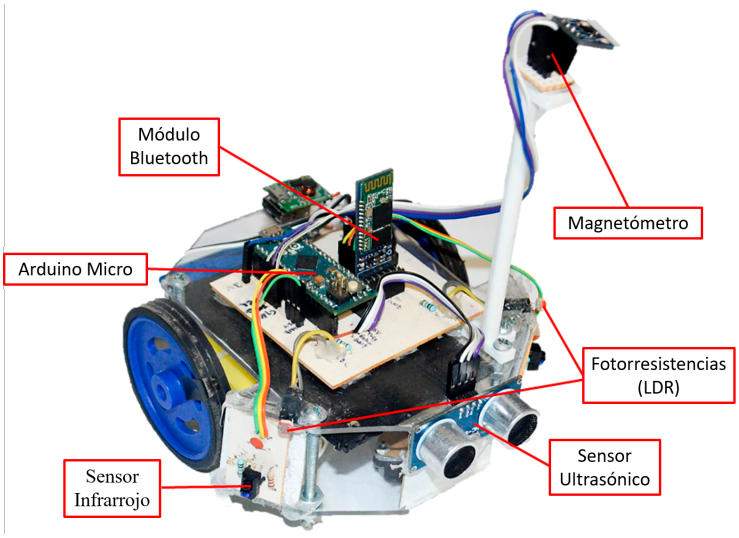
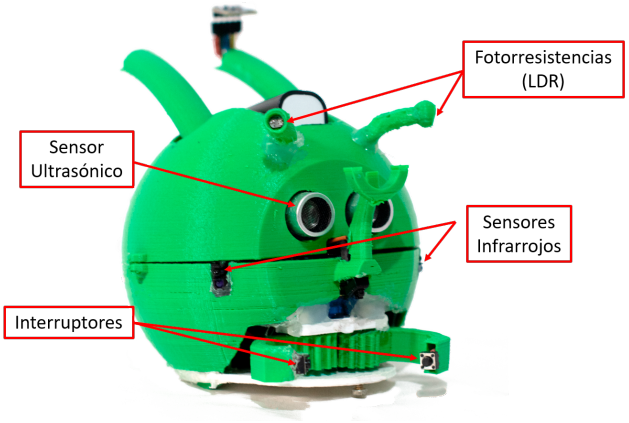


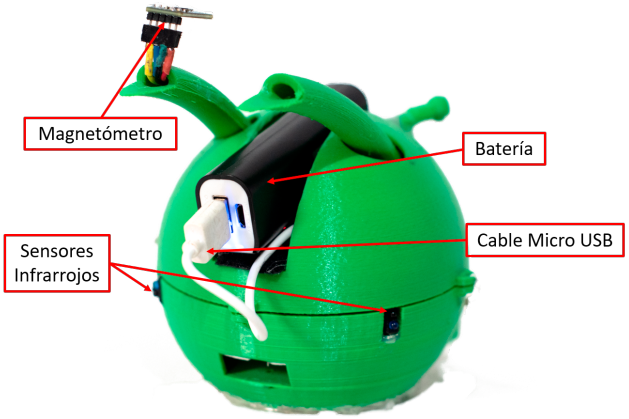
Figura 2.2: Ubicación de sensores y dispositivos en la arquitectura del robot *Terran*.

2.2.2 BugBot

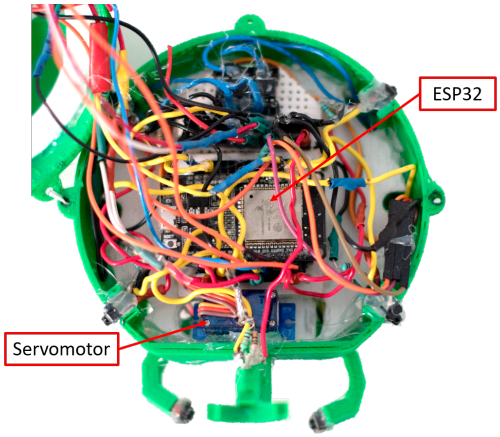
Los robots *BugBot* cuentan con un dispositivo *ESP32* con tecnología *Wi-Fi* y *Bluetooth*® de modo dual integrada, que se ha utilizado para tener comunicación entre el usuario y el *BugBot*, y establecer una configuración de valores paramétricos. Además, incorpora un mecanismo de sujeción para tomar objetos pequeños, el cual es activado por un servomotor. Por último, el diseño incluye dos interruptores tipo *push-button* para encender y apagar el *BugBot*. La Figura 2.3, ilustra la ubicación de los sensores y dispositivos en el robot.



(a) Vista frontal.



(b) Vista posterior.



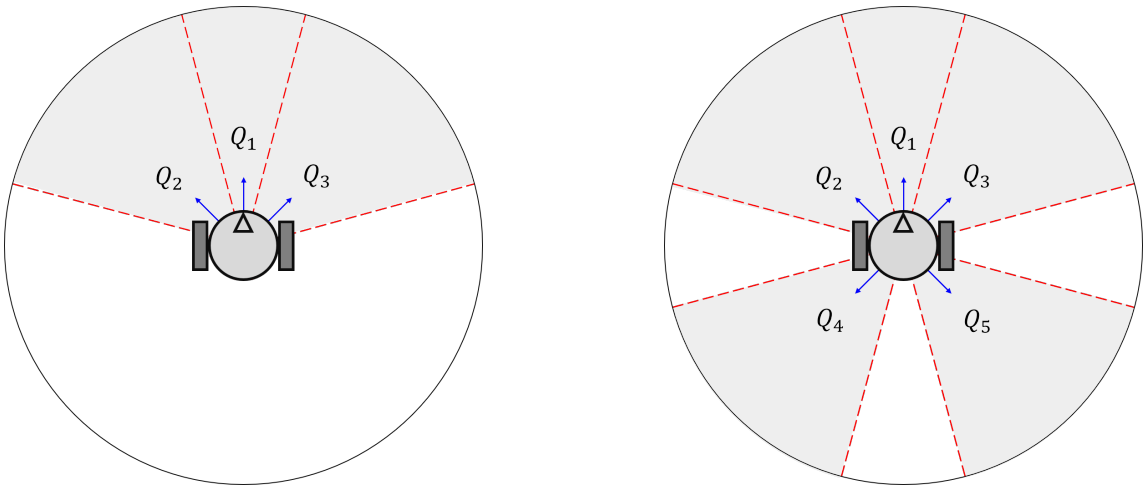
(c) Vista superior.

Figura 2.3: Ubicación de sensores y dispositivos en la arquitectura del robot *BugBot*.

2.2.3 Limitaciones sensoriales de los robots

Los movimientos permitidos en los robots son giros sobre su propio eje y avance; esta limitación se asemeja con las capacidades sensoriales de los seres vivos, ya que cuentan con un determinado rango de visión y no cubre la totalidad del espacio a su alrededor.

El número de objetos detectados, depende de la cantidad de sensores disponibles en los robots. Sin importar el número de robots que se encuentren dentro de la zona de percepción del sensor, este los considera como uno, ya que no puede determinar la cantidad de vecinos frente a él. La Figura 2.4, muestra las zonas de percepción Q_1, Q_2, Q_3, Q_4 y Q_5, que corresponden a los sensores de proximidad de cada robot.



(a) *Terran.* (b) *BugBot.*

Figura 2.4: Zonas de percepción correspondientes a los sensores incorporados.

El uso de sensores de proximidad limita la capacidad sensorial de los robots; en consecuencia, si algún robot se encuentra detrás de los vecinos más cercanos, estos no pueden ser detectados aunque se encuentren dentro del área de percepción, como se muestra en la Figura 2.5.

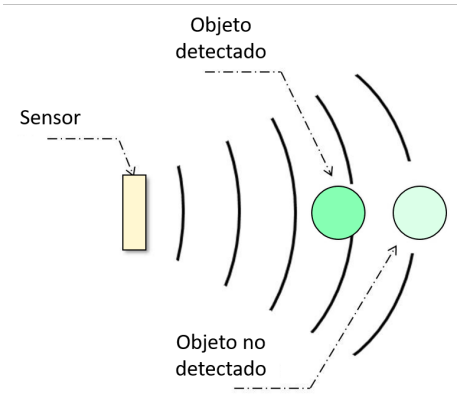


Figura 2.5: Limitaciones de los sensores de proximidad.

Debido a estas consideraciones, los robots sólo tienen información sobre una distancia aproximada a sus vecinos u obstáculos en la dirección del sensor de proximidad correspondiente; en consecuencia, no pueden percibir el movimiento que se genera en su entorno. Además, no pueden comunicarse entre sí, existen puntos ciegos de percepción y no existe información sobre la orientación o el número exacto de los vecinos.

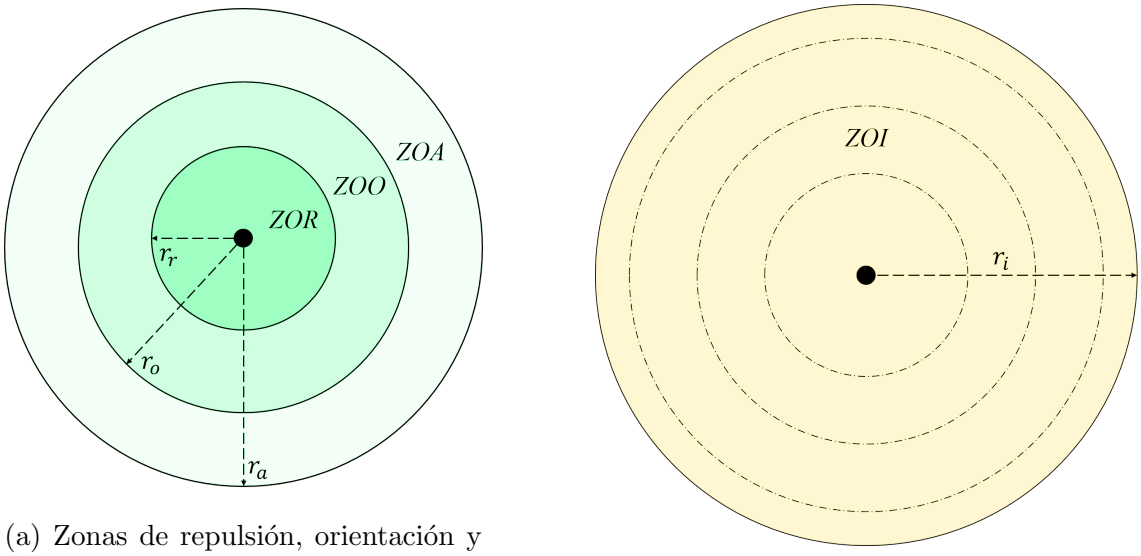
2.3 Reglas de comportamiento

El comportamiento de los miembros del enjambre se basa en el modelo propuesto por Couzin et al. [1], que utiliza las reglas de congregación (*flocking*) propuestas por Reynolds [45]. Este modelo que imita de manera más realista el movimiento de bandadas de aves y simula el comportamiento de los individuos, basado en tres características de las especies biológicas (repulsión, orientación y atracción). Estas características son representadas por zonas que rodean a cada individuo (Figura 2.6a).

La primera zona es la zona de repulsión (ZOR) de radio r_r es la más cercana al robot, cuando se detectan vecinos dentro de esta zona, el individuo intenta mantenerse alejado de ellos. La segunda zona es la zona de orientación (ZOO) de

radio r_o, cuando se detectan vecinos dentro de esta zona, el individuo intenta alinearse de la misma forma que ellos. La tercera zona es la zona de atracción (ZOA) de radio r_a, cuando se detectan vecinos dentro de esta zona, el individuo intenta acercarse a ellos. Cada zona se superpone una de la otra, teniendo como prioridad el orden en que fueron mencionadas. Al modificar los valores de los radios r_r, r_o y r_a surgen comportamientos colectivos con la interacción de sus vecinos, permitiendo cambiar la distribución o forma que puede adaptar el enjambre.

En este modelo se propone una cuarta zona llamada zona de influencia (ZOI) de radio r_i, que se superpone sobre las otras zonas (Figura 2.6b), cuando se percibe un estímulo de influencia en el entorno dentro de esta zona, el individuo tiende a acercarse a la fuente de influencia detectada. Esta zona estimula a los miembros del enjambre para cambiar su comportamiento y enfatizar sobre la tarea a realizar.



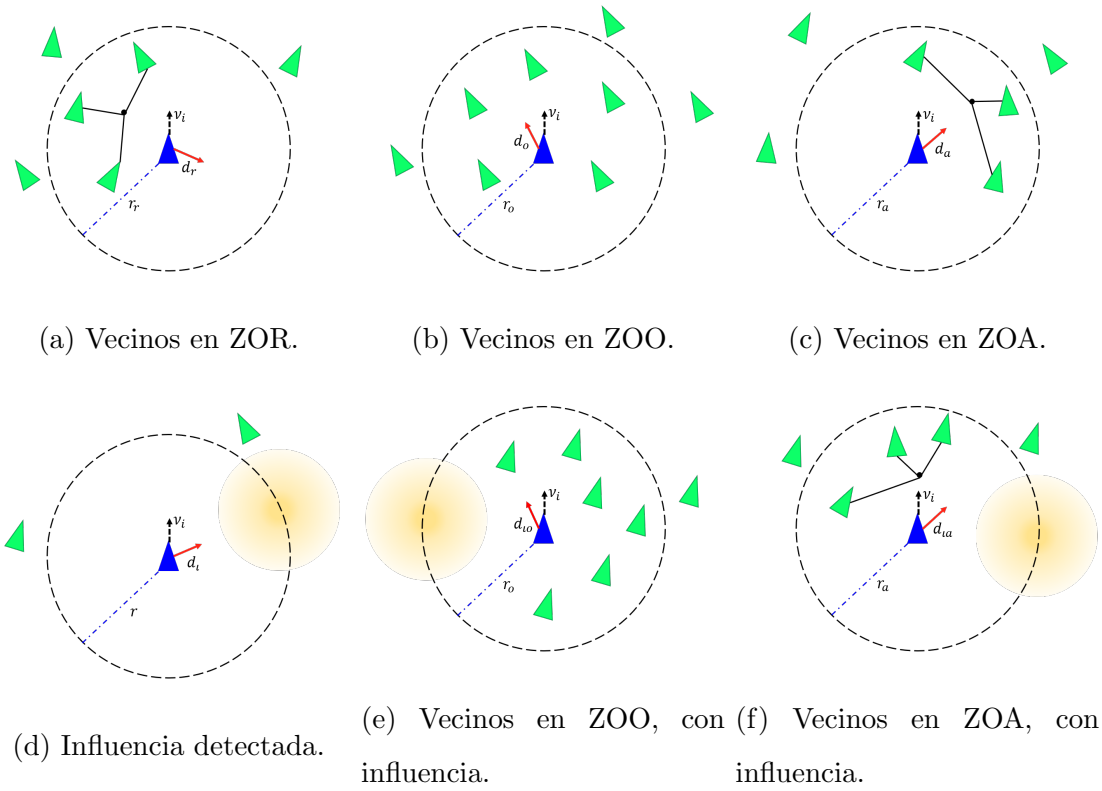
(a) Zonas de repulsión, orientación y atracción [1].

(b) Zona de influencia.

Figura 2.6: Zonas propuestas para el modelo de comportamiento de enjambre.

Cada miembro del enjambre tiene la posibilidad de realizar diferentes comportamientos (repulsión, orientación, atracción e influencia). La dirección de movimiento surge de las interacciones con los vecinos e influencia detectados en ZOR, ZOO, ZOA, y ZOI respectivamente. La Figura 2.7, muestra el vector de

dirección de cada individuo en relación a la zona detectada.



(a) Vecinos en ZOR. (b) Vecinos en ZOO. (c) Vecinos en ZOA.

(d) Influencia detectada. (e) Vecinos en ZOO, con influencia. (f) Vecinos en ZOA, con influencia.

Figura 2.7: Vectores de dirección respecto a zona detectada.

El modelo matemático se basa en dar a cada individuo una noción de repulsión, orientación, atracción hacia otros miembros del enjambre y estímulos por influencia con el entorno. El enjambre está compuesto de N individuos $(i = 1, 2, ..., N)$ con vectores de posición (ρ_i) y vectores de dirección unitaria (ν_i), que viajan a través del espacio de búsqueda a velocidad constante, en cada iteración los individuos determinan la nueva dirección de viaje deseada, mediante la detección de los vecinos dentro de las zonas.

Si un individuo (i) encuentra a vecinos (j) dentro de ZOR (Figura 2.7a), el vector de dirección deseado cambia por la siguiente ecuación:

$$d_r(t+1) = -\sum_{j \neq i}^{n_r} \frac{(\rho_j(t) - \rho_i(t))}{|\rho_j(t) - \rho_i(t)|} \tag{2.16}$$

Esta regla de comportamiento tiene mayor prioridad en el modelo, de modo que, si se encuentran vecinos en ZOR ($n_r > 0$) la dirección deseada $\nu_i(t+1) = d_r(t+1)$; por tanto, ZOR permite a los individuos mantener su espacio personal y evitar colisiones. Si no se encuentran vecinos en ZOR ($n_r = 0$), los individuos responden a otros dentro de las zonas ZOO y ZOA, y estímulos por influencia detectados en ZOI.

Si un individuo encuentra a vecinos dentro de ZOO (Figura 2.7b), intenta alinearse con ellos:

$$d_o(t+1) = \sum_{j=i}^{n_o} \frac{\nu_j(t)}{|\nu_j(t)|}, \tag{2.17}$$

y si son encontrados dentro de ZOA (Figura 2.7c), intenta acercarse a ellos:

$$d_a(t+1) = \sum_{j\neq i}^{n_a} \frac{(\rho_j(t) - \rho_i(t))}{|\rho_j(t) - \rho_i(t)|}. \tag{2.18}$$

La atracción es una tendencia de los organismos para unirse a grupos, mientras que la orientación permite el movimiento colectivo. De modo que, si vecinos son encontrados en ZOO ($n = n_o$), entonces $\nu_i(t+1) = d_o(t+1)$, de igual modo, si vecinos son encontrados en ZOA ($n = n_a$), entonces $\nu_i(t+1) = d_a(t+1)$ y si vecinos son encontrados en ambas zonas, entonces $\nu_i(t+1) = \varpi_o d_o(t+1) + \varpi_a d_a(t+1)$; donde ϖ_o y ϖ_a son la ponderación de peso de ZOR y ZOA, respectivamente.

Por otro lado, si se detecta un estímulo por influencia (s_ι), dentro de ZOI (Figura 2.7d), el individuo tiende hacia la posición del factor de influencia detectado (ρ_ι); por lo que el vector de dirección deseado cambia por la siguiente ecuación:

$$d_\iota(t+1) = \frac{(\rho_\iota - \rho_i)}{|\rho_\iota - \rho_i|}. \tag{2.19}$$

Si se detectan vecinos en ZOO y $s_\iota = 1$ (Figura 2.7e), entonces $\nu_i(t+1) = \varpi_\iota d_\iota(t+1) + \varpi_o d_o(t+1)$, de igual forma, si se detectan vecinos en ZOA y $s_\iota = 1$ (Figura 2.7f), entonces $\nu_i(t+1) = \varpi_\iota d_\iota(t+1) + \varpi_a d_a(t+1)$. Por último, si se detectan vecinos en ZOO y ZOA, y $s_\iota = 1$, entonces $\nu_i(t+1) = \varpi_\iota d_\iota(t+1) + \varpi_o d_o(t+1) + \varpi_a d_a(t+1)$; donde ϖ_ι es la ponderación de peso de ZOI.

Las reglas de comportamiento presentadas se resumen en la siguiente ecuación:

$$\nu_i(t+1) = \begin{cases} d_r(t+1), & \text{si} \quad n_r > 0 \\ d_o(t+1), & \text{si} \quad n_r, n_a, s_\iota = 0 \quad \& \quad n_o > 0 \\ d_a(t+1), & \text{si} \quad n_r, n_o, s_\iota = 0 \quad \& \quad n_a > 0 \\ d_\iota(t+1), & \text{si} \quad n_r, n_o, n_a = 0 \quad \& \quad s_\iota = 1 \\ \varpi_o d_o(t+1) + \varpi_a d_a(t+1), & \text{si} \quad n_r, s_\iota = 0 \quad \& \quad n_o, n_a > 0 \\ \varpi_\iota d_\iota(t+1) + \varpi_o d_o(t+1), & \text{si} \quad n_r, n_a = 0 \quad \& \quad s_\iota, n_o > 0 \\ \varpi_\iota d_\iota(t+1) + \varpi_a d_a(t+1), & \text{si} \quad n_r, n_o = 0 \quad \& \quad s_\iota, n_a > 0 \\ \varpi_\iota d_\iota(t+1) + \varpi_o d_o(t+1) + \varpi_a d_a(t+1), & \text{si} \quad n_r = 0 \quad \& \quad s_\iota, n_o, n_a > 0 \end{cases} \tag{2.20}$$

Finalmente cada individuo gira hacia el vector de dirección $\nu_i(t+1)$ por un ángulo de giro (ϑ). Siempre que el ángulo entre $\nu_i(t)$ y $\nu_i(t+1)$ sea menor que el ángulo de giro máximo (ϑ_{max}), el cual es determinado en función de las restricciones de movimiento por la arquitectura del robot, entonces la dirección de giro cambia a $\nu_i(t+1)$; sino, el individuo gira ϑ_{max} hasta la dirección deseada.

$$\vartheta = \begin{cases} \vartheta, & \text{si} \quad \vartheta < \vartheta_{max}, \\ \vartheta_{max}, & \text{si} \quad \vartheta \geq \vartheta_{max} \ . \end{cases} \tag{2.21}$$

De esta forma, cada robot interactúa por medio de estas reglas de comportamiento con sus vecinos y el entorno, y como consecuencia, surge un comportamiento global (Figura 2.8); el cual puede gobernarse mediante cambios en la configuración de valores paramétricos r_r, r_a, r_o y r_ι.

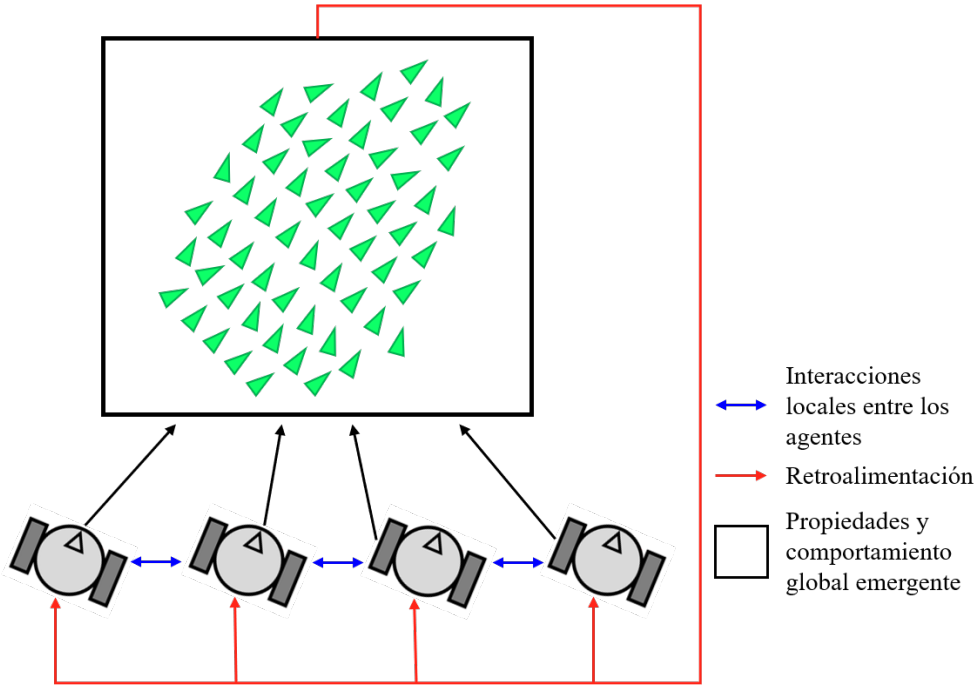


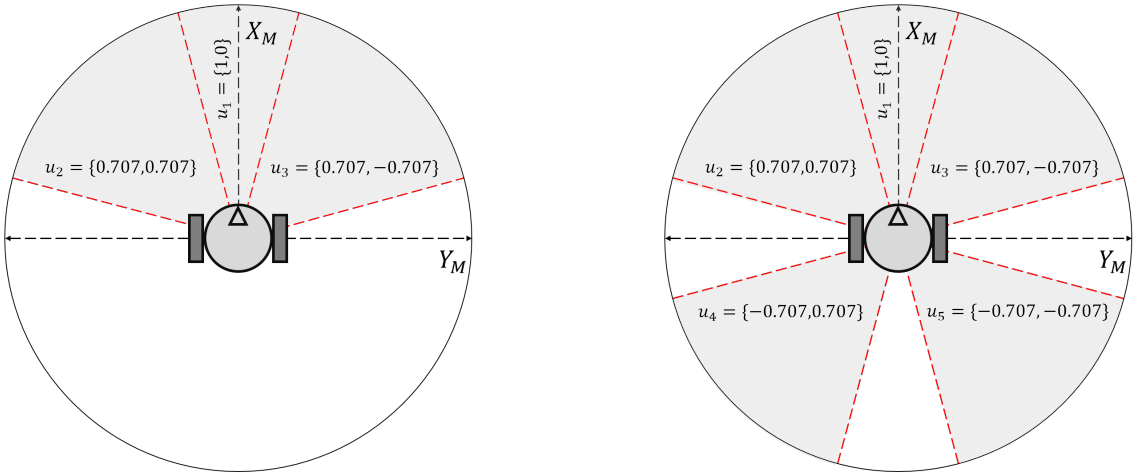
Figura 2.8: Comportamiento emergente en el enjambre.

2.3.1 Implementación de reglas en un sistema embebido

Para transportar las reglas de comportamiento de la Ecuación (2.20) en pequeños robots que no usan comunicación o mecanismos de detección complejos, el modelo propuesto fue simplificado en dos dimensiones; además se consideran las limitaciones de hardware y software de cada robot.

En la arquitectura de robot propuesta, cada miembro del enjambre tiene la capacidad de monitorear sectores de visión para detectar vecinos y obstáculos. Estos sectores, son representados por vectores unitarios (u_1, u_2, u_3, u_4 y u_5) en la dirección de los sensores con los que cuenta cada robot en un sistema de coordenadas local, como se muestra en la Figura 2.9. La ubicación de los sensores permite dividir las zonas de comportamiento en nuevas secciones (Figuras 2.10a y 2.10b). La zona Q_{1a} es la única que permite al robot ser atraído por sus vecinos, mientras que las zonas Q_{kr} para $k = 1, 2, 3, 4, 5$, se limitan sólo para repeler a sus vecinos y así evitar colisiones. Debido a que no existe información sobre la orientación, la zona Q_{1o} se considera

con un valor nulo para todos los casos.



(a) *Terran* (b) *BugBot*

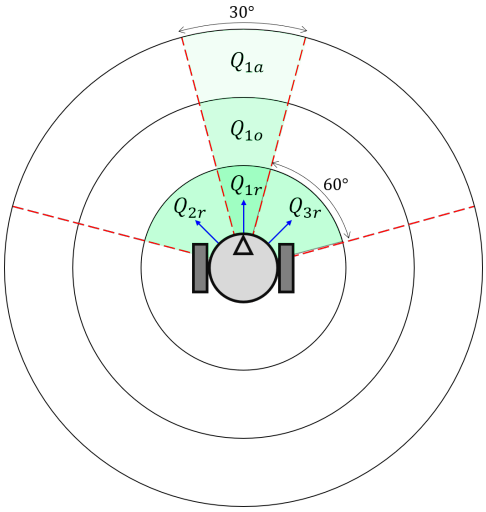
Figura 2.9: Vectores de dirección de cada sensor.

Como factor de influencia se propone la luz que hay en el entorno, para esto se consideran zonas de recepción de luz γ_l y γ_r (Figura 2.10c); que corresponden a los sensores de luz (LDR) incorporados en los robot. La zona donde se traslapan γ_l y γ_r en la Figura 2.10c, indica que ambos sensores detectan una cantidad de luz similar en la zona frontal del robot.

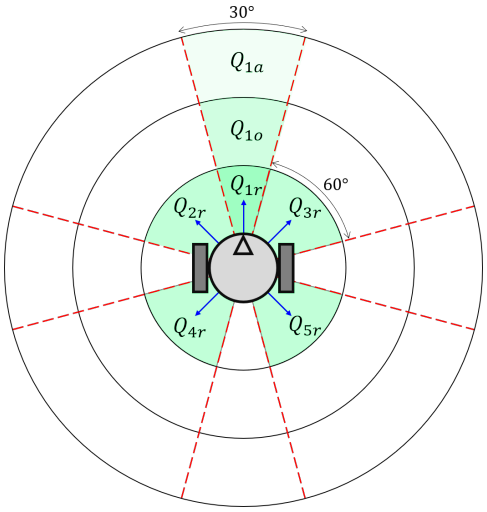
Una vez obtenidas las lecturas de los sensores, éstas se procesan para calcular la distancia aproximada hacia los vecinos dentro de las zonas de percepción locales y calcular los movimientos. Si un robot detecta a otros en sus zonas de repulsión Q_{kr}, la velocidad disminuye para prevenir colisiones y cambia su trayectoria hacia un vector que apunta en la dirección promedio opuesta a las secciones de repulsión en las que se detectaron sus vecinos, este vector es dado por:

$$d_r = \begin{cases} -[u_1 q_{1r} + u_2 q_{2r} + u_3 q_{3r}], & \text{para } \textit{Terran}, \\ -[u_1 q_{1r} + u_2 q_{2r} + u_3 q_{3r} + u_4 q_{4r} + u_5 q_{5r}], & \text{para } \textit{BugBot}. \end{cases} \tag{2.22}$$

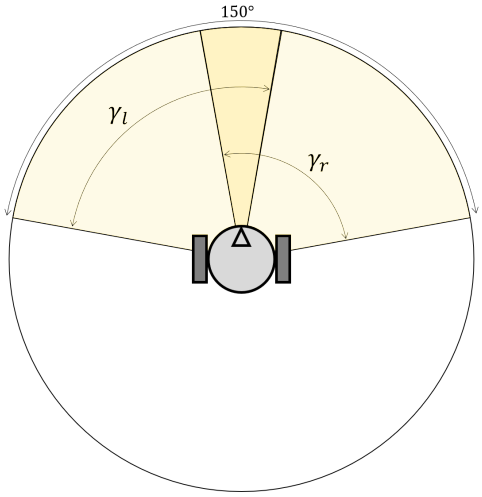
Para las secciones Q_{kr}, el valor de los sensores q_{kr} será cero si no se detecta



(a) Zonas de repulsión, orientación y atracción para *Terran*.



(b) Zonas de repulsión, orientación y atracción para *BugBot*.



(c) Zonas de influencia.

Figura 2.10: Zonas *RAOI* por robot.

ningún robot, o será de uno si se detecta al menos uno.

$$q_{kr} = \begin{cases} 1, & \text{si} \quad n_{Q_{kr}} > 0, \\ 0, & \text{si} \quad n_{Q_{kr}} = 0, \end{cases} \tag{2.23}$$

donde $n_{Q_{kr}}$ es la cantidad de vecinos en la zona Q_{kr} para $k = 1, 2, 3, 4, 5$.

Cuando $n_{Q_{kr}} = 0$ y se detectan vecinos en la zona de atracción Q_{1a}, surge un aumento de velocidad en la dirección actual del robot, para acercarse a los miembros del enjambre percibidos.

$$d_a = u_1 q_{1a}. \tag{2.24}$$

Para la sección Q_{1a} el valor del sensor q_{1a} será cero si no se detecta ningún robot, o será de uno si se detecta al menos uno.

$$q_{1a} = \begin{cases} 1, & \text{si} \quad n_{Q_{1a}} > 0, \\ 0, & \text{si} \quad n_{Q_{1a}} = 0. \end{cases} \tag{2.25}$$

donde $n_{Q_{1a}}$ es la cantidad de vecinos en la zona Q_{1a}.

Una vez obtenido el vector de dirección, el ángulo de giro de los robot en ZOR y ZOA se obtiene por:

$$\vartheta = \begin{cases} \frac{\pi}{2} sgn(d_{ry}) - arctan\frac{d_{rx}}{d_{ry}}, & \text{si } q_{kr} = 1, \\ \frac{\pi}{2} sgn(d_{ay}) - arctan\frac{d_{ax}}{d_{ay}}, & \text{si } q_{1a} = 1 \quad \& \quad q_{kr} = 0, \end{cases} \tag{2.26}$$

donde d_{rx} y d_{ry} son los componentes de dirección deseada en ZOR, mientras que d_{ax} y d_{ay} son los componentes de dirección deseada en ZOA.

Cuando el robot realiza un giro hacia la derecha, corresponde a un giro con un ángulo menor de 0 grados, mientras que un giro a la izquierda, a un ángulo mayor. Por tanto, la dirección de giro de los motores causada por ZOO y ZOA es representada por la Ecuación (2.27).

$$S = \begin{bmatrix} S_l & 0 \\ 0 & S_r \end{bmatrix}. \tag{2.27}$$

con

$$S_l = \begin{cases} 1, & \text{si} \quad \vartheta \leq 0, \\ -1, & \text{si} \quad \vartheta > 0, \end{cases} \qquad S_r = \begin{cases} 1, & \text{si} \quad \vartheta \geq 0, \\ -1, & \text{si} \quad \vartheta < 0, \end{cases}$$

donde S_l y S_r representan la dirección de giro de los motores izquierdo y derecho en ZOR y ZOA, respectivamente, y cambian en función del movimiento compuesto por las dos ruedas del robot.

Ya que no es posible conocer la orientación de los vecinos, la influencia toma prioridad en la dirección de navegación de cada robot. Cuando éstos detecten a vecinos dentro de su zona de orientación (Q_{1o}), la dirección es afectada por la luz percibida en el entorno, guiando a cada robot hacia la fuente de luz detectada con mayor intensidad. Cuando la intensidad de luz es similar en γ_l y γ_r, el robot mantiene su dirección hasta encontrarse con otros, en la búsqueda por una orientación surgen las reglas de repulsión, atracción e influencia. Para esto, el alcance de la zona de influencia se define por r_ι, que a su vez esta limitado por las restricciones de los sensores de luz empleados; por lo que el valor de r_ι se establece como constante con el valor máximo de alcance permitido por γ_l y γ_r para todos los casos. De este modo, la dirección de giro de los motores causada por la influencia es representada por:

$$S_\iota = \begin{bmatrix} S_{\iota,l} & 0 \\ 0 & S_{\iota,r} \end{bmatrix}, \tag{2.28}$$

con

$$S_{\iota,l} = \begin{cases} 1, & \text{si} \quad \gamma_l > \gamma_r, \\ -1, & \text{si} \quad \gamma_l < \gamma_r, \end{cases} \qquad S_{\iota,r} = \begin{cases} 1, & \text{si} \quad \gamma_r > \gamma_l, \\ -1, & \text{si} \quad \gamma_r < \gamma_l, \end{cases}$$

donde $S_{\iota,l}$ y $S_{\iota,r}$ representan la dirección de giro de los motores izquierdo y derecho causada por la influencia.

Por otro lado, las velocidades de los robots son generadas por el voltaje de referencia aplicado a los motores de cada rueda, que depende de la zona activa de los robots y se representa por un vector de la forma $v_i = [v_l \ v_r]^T$. Los voltajes de repulsión $v_{r,i}$ y orientación $v_{o,i}$ son valores constantes, que son definidos por un usuario en el código de programación, mientras que los voltajes de influencia y atracción están dados por:

$$v_{\iota,i} = \begin{cases} v_{max}\delta_\iota, & \text{si} \quad \gamma_l > \gamma_u \quad \& \quad \gamma_r > \gamma_u, \\ v_{min}, & \text{si} \quad \gamma_l \leq \gamma_u \quad || \quad \gamma_r \leq \gamma_u, \end{cases} \tag{2.29}$$

$$v_{a,i} = \frac{1}{2}\,[v_{\iota,i} + v_{max}], \tag{2.30}$$

donde δ_ι es la intensidad de luz percibida en el entorno, v_{min} y v_{max}, son el voltaje mínimo y máximo permitidos y γ_u es un umbral de influencia, el cual es definido por un usuario bajo las características del mecanismo de percepción, al modificar este valor cambia la sensibilidad de percepción de influencia.

Finalmente, la dirección de giro y el voltaje aplicado en cada motor es:

$$u_i = \begin{cases} Sv_r, & \text{si} \quad q_{kr} = 1, \\ S_\iota v_i, & \text{si} \quad q_{kr} = 0 \quad \& \quad s_\iota = 1 \ , \\ Sv_a, & \text{si} \quad s_\iota, q_{kr} = 0 \quad \& \quad q_{1a} = 1 \ , \\ S_\iota v_o, & \text{si} \quad s_\iota, q_{kr}, q_{1a} = 0 \ , \end{cases} \tag{2.31}$$

donde

$$s_\iota = \begin{cases} 1, & \text{si} \quad \gamma_l \neq \gamma_r, \\ 0, & \text{si} \quad \gamma_l \approx \gamma_r. \end{cases}$$

Para generar el movimiento del robot, la Figura 2.11 muestra un diagrama de las funciones utilizadas; donde u_i, para $i = \{r, l\}$, corresponde al voltaje aplicado en los motores izquierdo y derecho, respectivamente (Ec. 2.31).

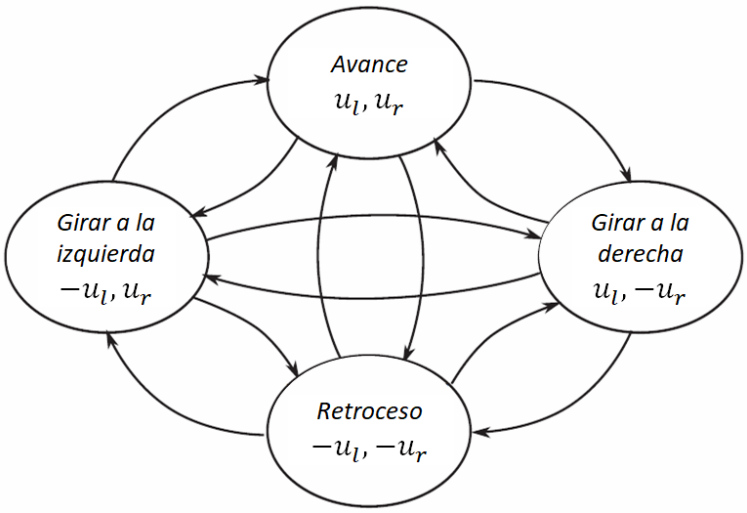


Figura 2.11: Funciones de movimiento del robot en relación al voltaje aplicado.

Cada robot cuenta con un microcontrolador en donde se implementaron las reglas de comportamiento que se muestran en el Apéndice B y son explicadas por el Algoritmo 1, que es representado por el diagrama de flujo de la Figura 2.14. La programación se realizó en el entorno de desarrollo integrado de $Arduino^{TM}$ (*Arduino IDE 1.8.12*). Debido a que se abordan dos tareas con arquitecturas de robot distintas,

se consideran breves modificaciones en las reglas de comportamiento, que se explican en las siguientes secciones.

2.3.1.1 Depredador-Presa

En la tarea de depredador-presa se utilizan los robot *Terran* (Figura 2.2), donde un robot toma el papel de presa mientras que los demás de depredadores. Los robot-depredadores cuentan con las reglas de comportamiento del Algoritmo 1; mientras que, al robot-presa se le incorpora una fuente de luz, para poder ser distinguido entre los depredadores y actuar como un factor de influencia. Debido a esto, el robot-presa no cuenta con la regla de influencia (se deshabilitan los sensores *LDR*) y sólo se mantienen las reglas de repulsión y atracción. Para este caso, el Algoritmo 2, representado por el diagrama de flujo de la Figura 2.15, muestra los cambios realizados en el robot-presa.

Si el robot-presa detecta otros robots depredadores en ZOR, cambia su dirección según la Ecuación (2.22). Si detecta un vecino en ZOA, el robot-presa aumenta su velocidad en su dirección actual, (Ec. 2.24). Cuando no se detecta otro robot en ZOR y ZOA, entonces el robot-presa mantiene la misma dirección hasta que estas condiciones cambien. Modificar los valores de los parámetros, hace que el robot-presa tenga un comportamiento diferente al de los robot-depredadores.

2.3.1.2 Transporte de objetos

En la tarea de transporte de objetos se utilizan los robot *BugBot* (Figura 2.3), donde cada uno, a diferencia del modelo *Terran*, cuenta con un mecanismo de sujeción para transportar objetos y un sensor de proximidad para detectar su presencia (Figura 2.12). Los *BugBot*, son capaces de realizar dos subtareas (búsqueda

y entrega de objetos), la subtarea que realiza cada *BugBot* se define por el estado de las tenazas (ϱ_i); donde $\varrho_i = 0$, si las tenazas se encuentran abiertas (estado de búsqueda/objeto no detectado) y $\varrho_i = 1$, si las tenazas se encuentran cerradas (estado de entrega/objeto detectado).

(a) Tenazas abiertas. (b) Tenazas cerradas.

Figura 2.12: Estado de tenzas de *BugBot*.

Por otro lado, se cuenta con un magnetómetro, que permite a cada *BugBot* estimar su orientación (ψ_i) y, de esta manera, identificar si se encuentra en dirección de búsqueda o entrega. En consecuencia, por medio de ψ_i, es posible definir un rango de navegación para cada subtarea y establecer valores mínimos y máximos para cada caso; de manera que, $\psi_{s,min}$ y $\psi_{s,max}$, son el valor mínimo y máximo del rango de dirección para la búsqueda de objetos y, $\psi_{d,min}$ y $\psi_{d,max}$, son el valor mínimo y máximo del rango de dirección para la entrega de objetos. De manera que, la dirección de giro de los motores, causado por ψ_i es representada por:

$$S_\psi = \begin{cases} S_{\psi s}, & \text{si} \quad \varrho_i = 0, \\ S_{\psi d}, & \text{si} \quad \varrho_i = 1, \end{cases} \tag{2.32}$$

donde $S_{\psi s}$ y $S_{\psi d}$, son matrices de dirección de giro para los procesos de búsqueda y entrega, respectivamente.

$$S_{\psi s} = \begin{bmatrix} S_{\psi s,l} & 0 \\ 0 & S_{\psi s,r} \end{bmatrix}, \tag{2.33}$$

$$S_{\psi d} = \begin{bmatrix} S_{\psi d,l} & 0 \\ 0 & S_{\psi d,r} \end{bmatrix}. \tag{2.34}$$

Asimismo, los componentes de las matrices $S_{\psi s}$ y $S_{\psi d}$, toman el valor de -1 o 1 en función de las condiciones de ψ_i:

$$S_{\psi s,l} = \begin{cases} 1, & \text{si} \quad \psi_{s,min} < \psi_i < \psi_{s,max}, \\ -1, & \text{si} \quad \psi_i > \psi_{s,max}, \end{cases} \quad S_{\psi s,r} = \begin{cases} 1, & \text{si} \quad \psi_{s,min} < \psi_i < \psi_{s,max}, \\ -1, & \text{si} \quad \psi_i < \psi_{s,min}, \end{cases}$$

$$S_{\psi d,l} = \begin{cases} 1, & \text{si} \quad \psi_{d,min} < \psi_i < \psi_{d,max}, \\ -1, & \text{si} \quad \psi_i > \psi_{d,max}, \end{cases} \quad S_{\psi d,r} = \begin{cases} 1, & \text{si} \quad \psi_{d,min} < \psi_i < \psi_{d,max}, \\ -1, & \text{si} \quad \psi_i < \psi_{d,min}. \end{cases}$$

De este modo, las matrices de giro S y S_ι de la Ecuación (2.31), actualizan sus valores por S_ψ si:

$$S = \begin{cases} S, & \text{si} \quad (\psi_{s,min} < \psi_i < \psi_{s,max}) \quad \& \quad \varrho_i = 0, \\ S, & \text{si} \quad (\psi_{d,min} < \psi_i < \psi_{d,max}) \quad \& \quad \varrho_i = 1, \\ S_\psi, & \text{si} \quad (\psi_{s,min} > \psi_i \,||\, \psi_i > \psi_{s,max}) \quad \& \quad \varrho_i = 0, \\ S_\psi, & \text{si} \quad (\psi_{d,min} > \psi_i \,||\, \psi_i > \psi_{d,max}) \quad \& \quad \varrho_i = 1. \end{cases}$$

$$S_\iota = \begin{cases} S_\iota, & \text{si} \quad (\psi_{s,min} < \psi_i < \psi_{s,max}) \quad \& \quad \varrho_i = 0, \\ S_\iota, & \text{si} \quad (\psi_{d,min} < \psi_i < \psi_{d,max}) \quad \& \quad \varrho_i = 1, \\ S_\psi, & \text{si} \quad (\psi_{s,min} > \psi_i \,||\, \psi_i > \psi_{s,max}) \quad \& \quad \varrho_i = 0, \\ S_\psi, & \text{si} \quad (\psi_{d,min} > \psi_i \,||\, \psi_i > \psi_{d,max}) \quad \& \quad \varrho_i = 1. \end{cases}$$

Las zonas de búsqueda y de entrega, son identificadas por un factor de influencia (fuente de luz), que permite a cada *BugBot* dirigirse hacia a la zona correspondiente en función del estado de las tenazas (ϱ_i) y guiado por su dirección

de navegación (ψ_i). En el proceso de ejecución de la tarea, cada *BugBot* inicia en el "nido", que corresponde a la zona de entrega, y se dirigen hacia la zona de búsqueda hasta encontrar un objeto; los cuales están colocados de forma aleatoria dentro de una zona iluminada, que da una aproximación de su ubicación. Una vez que un objeto es encontrado y tomado ($\varrho_i = 1$), el *BugBot* se dirige de vuelta al nido que, de igual forma, se encuentra iluminado para aproximar el punto de descarga. Ya que el objeto es colocado en el nido, el proceso de búsqueda vuelve a iniciar. Cada subtarea es realizada manteniendo las reglas de repulsión, atracción e influencia que permiten a cada *BugBot* interactuar con sus vecinos y el entorno. El proceso de la tarea de transporte de objetos es descrito por la Figura 2.13.

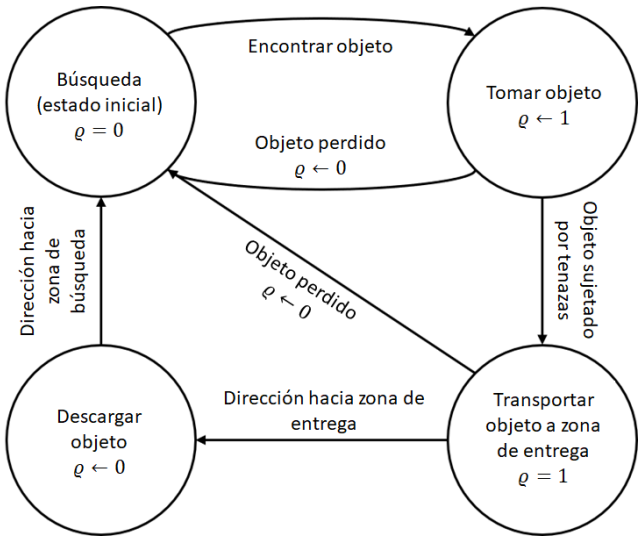


Figura 2.13: Estados de comportamiento para tarea de transporte de objetos.

Cuando un *BugBot* se encuentra en la zona de búsqueda y percibe una intensidad de influencia alta, se realiza un cambio de dirección aleatorio para continuar con el proceso de búsqueda y converger una vez más hacia la fuente de influencia desde otro ángulo; este cambio de dirección es representado por:

$$\nu_i(t+1) = \begin{cases} \nu_i(t), & \text{si} \quad \delta_\iota < \gamma_u, \\ d_\varkappa(t+1), & \text{si} \quad \delta_\iota \geq \gamma_u, \end{cases} \tag{2.35}$$

donde las componentes $d_{\varkappa x}$ y $d_{\varkappa y}$ del vector $d_{\varkappa}(t+1)$ son aleatorias dentro de un rango establecido.

De no contemplar esta consideración, la concentración del enjambre estaría en el punto de mayor intensidad de influencia y los vecinos impedirían que se continuara con el proceso de búsqueda al estar ZOR activa.

El estado de las pinzas y el sentido de orientación actúan como indicadores que permiten a cada *BugBot* conocer la subtarea que hay que realizar, y se complementan con los parámetros de repulsión, atracción e influencia, que siempre están presentes en cada proceso (búsqueda y entrega). Tomando como base el Algoritmo 1, se propone el Algoritmo 3, que es representado por el diagrama de flujo de la Figura 2.16 e incluye las consideraciones de las variables ϱ_i y ψ_i.

2.3.2 Implementación de reglas en una plataforma de simulación

La finalidad de tener una plataforma de simulación es poder realizar tantos experimentos como sean necesarios, sin importar la cantidad de individuos y el costo de fabricación que estos requieren. Esta plataforma permite explorar los parámetros del modelo propuesto en distintas condiciones y conocer sus efectos en el comportamiento del enjambre, olvidando las limitaciones que se tienen en el sistema físico. Las reglas del enjambre y la capacidad de percepción de cada robot se implementaron en la versión 6.1.0 de *Scilab*, un software de código abierto para análisis numérico, donde se tomó el modelo presentado en la Figura 2.1, del cual se toman los parámetros del robot de [79] y se genera la Tabla 2.1. El código implementado en *Scilab* se encuentra en el Apéndice C y se explica en el Algoritmo 4; donde N corresponde al total de miembros del enjambre, K es el total de pasos de tiempo y cada paso es equivalente a 1 segundo.

Tabla 2.1: Parámetros del robot.

Parámetro	**Valor**
m	$0.38\ kg$
I	$0.005\ kg \cdot m^2$
r	$0.03\ m$
R	$0.05\ m$
d	$0.02\ m$
$\tau_{s,r} = \tau_{s,l}$	$0.434\ s$
$k_{L,r} = k_{L,l}$	$2.745\ \frac{rad}{s \cdot N - m}$
$k_{s,r} = k_{s,l}$	$1460.2705\ \frac{rad}{s \cdot V}$

Debido a que se requieren simulaciones lo más cercanas a la realidad; la representación de los robots involucra ruido blanco Gaussiano [82], fricción, errores inherentes en las lecturas del sensor y valores atípicos[83, 84]. Los resultados exitosos en simulaciones abren la puerta a la implementación donde las reglas de comportamiento se adaptan, para implementarse en robots simples y reales.

Algoritmo 1: Reglas de comportamiento de enjambre.

Entradas: r_r, r_a;
Inicializar velocidades y posiciones de cada individuo;
repetir
  Calcular distancias hacia vecinos e influencia;
  Actualizar zonas activas (Ec. 2.23 y 2.25);
  si $\delta_k \leq r_r$ **entonces**
    Calcular dirección de repulsión d_r (Ec. 2.22);
    Acualizar velocidades y dirección actual. (Ec. 2.27);
    $n \leftarrow n_r$;
  fin
  si $n_r, s_\iota = 0$ & $\delta_k \leq r_a$ **entonces**
    Calcular dirección de atracción d_a (Ec. 2.24);
    Acualizar velocidades y dirección actual (Ec. 2.27);
    $n \leftarrow n_a$;
  en otro caso
    si $s_\iota = 1$ **entonces**
      Calcular dirección por influencia (Ec.2.28);
    fin
  fin
  si $n = 0$ **entonces**
    Mantener dirección actual;
  fin
  Conversión al sistema de referencia del robot;
  Escribir datos en actuadores (Ec. 2.31);
hasta que *detener proceso*;

Algoritmo 2: Reglas de comportamiento de robot-presa.

Entradas: r_r, r_a;
Inicializar velocidades y posiciones de cada individuo;
repetir
  Calcular distancias hacia vecinos;
  Actualizar zonas activas (Ec. 2.23 y 2.25);
  si $\delta_k \leq r_r$ **entonces**
    Calcular dirección de repulsión d_r (Ec. 2.22);
    Acualizar velocidades y dirección actual. (Ec. 2.27);
    $n \leftarrow n_r$;
  fin
  si $n_r = 0$ & $\delta_k \leq r_a$ **entonces**
    Calcular dirección de atracción d_a (Ec. 2.24);
    Acualizar velocidades y dirección actual (Ec. 2.27);
    $n \leftarrow n_a$;
  fin
  si $n = 0$ **entonces**
    Mantener dirección actual;
  fin
  Conversión al sistema de referencia del robot;
  Escribir datos en actuadores (Ec. 2.31);
hasta que *detener proceso*;

Algoritmo 3: Reglas de comportamiento de enjambre para tarea de transporte de objetos.

Entradas: r_r, r_a;
Inicializar velocidades y posiciones de cada individuo;
repetir
  Calcular distancias hacia vecinos e influencia;
  Actualizar zonas activas (Ec. 2.23 y 2.25);
  Comporbar estado de tenazas y orientación;
  si $\varrho_i = 0$ **entonces**
    si $(\psi_{s,min} < \psi_i < \psi_{s,max})$ **entonces**
      Ejecutar reglas de comportamiento de enjambre (Ec. 2.31);
    en otro caso
      Actualizar dirección de búsqueda (Ec. 2.32);
      Ejecutar reglas de comportamiento de enjambre (Ec. 2.31);
    fin
  en otro caso
    si $(\psi_{d,min} < \psi_i < \psi_{d,max})$ **entonces**
      Ejecutar reglas de comportamiento de enjambre (Ec. 2.31);
    en otro caso
      Actualizar dirección de entrega (Ec. 2.32);
      Ejecutar reglas de comportamiento de enjambre (Ec. 2.31);
    fin
  fin
  Conversión al sistema de referencia del robot;
  Escribir datos en actuadores (Ec. 2.31);
hasta que *detener proceso*;

Algoritmo 4: Simulador de enjambre.

Entradas: r_r, r_a;
Inicializar velocidades y posiciones de cada individuo;
para κ*=1:K* **hacer**
  para *i=1:N* **hacer**
    Calcular distancia y ángulo hacia influencia;
    para *j=1:N* **hacer**
      Calcular distancia y ángulo hacia individuo j;
    fin
    Calcular vector unitario d_r (Ec. 2.16);
    Calcular vector unitario d_a (Ec. 2.18);
    Calcular vector unitario d_o (Ec. 2.17);
    Calcular vector unitario d_i (Ec. 2.18);
    Actualizar vector de dirección ν_i (Ec. 2.20);
    Calcular posición y velocidad actual;
  fin
  si *Tarea completada* **entonces**
    break;
  fin
fin

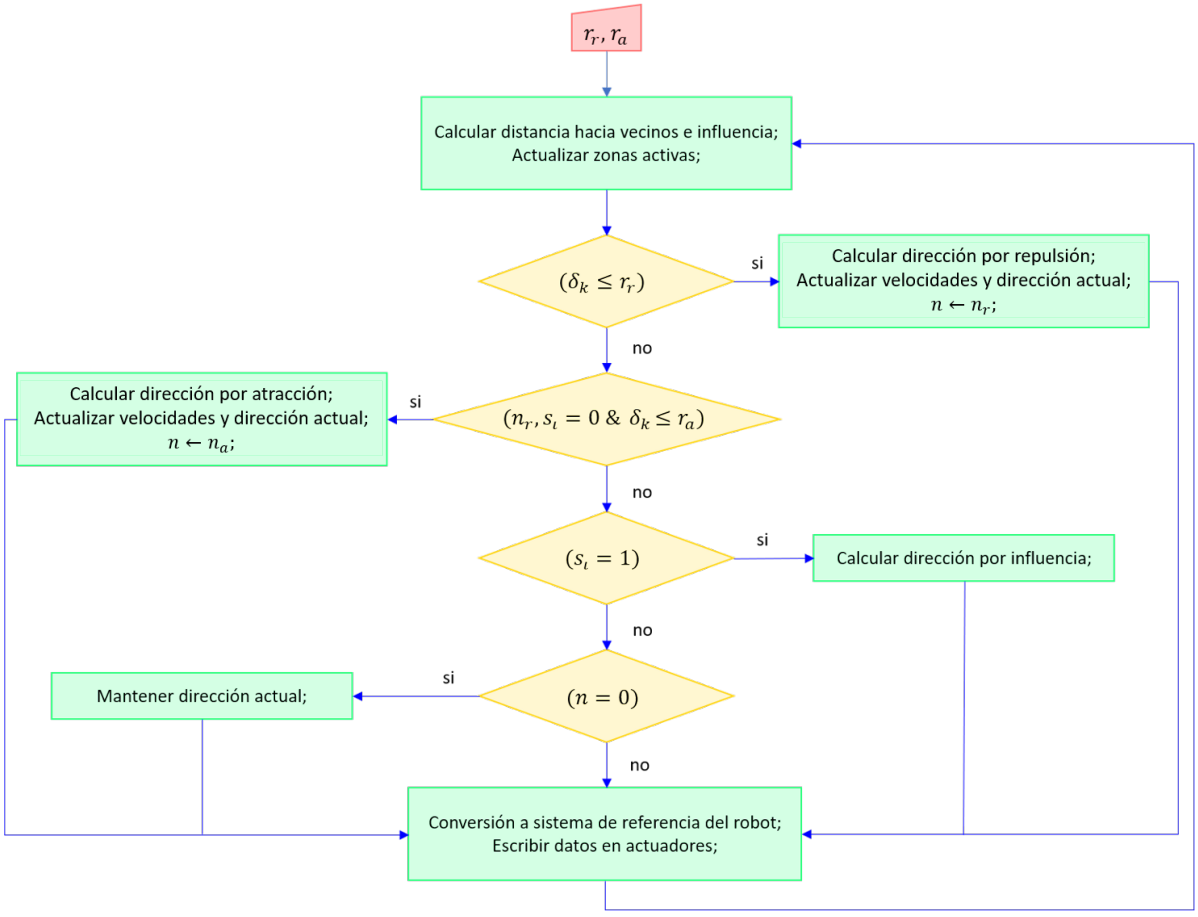


Figura 2.14: Diagrama de flujo de las reglas de comportamiento de enjambre.

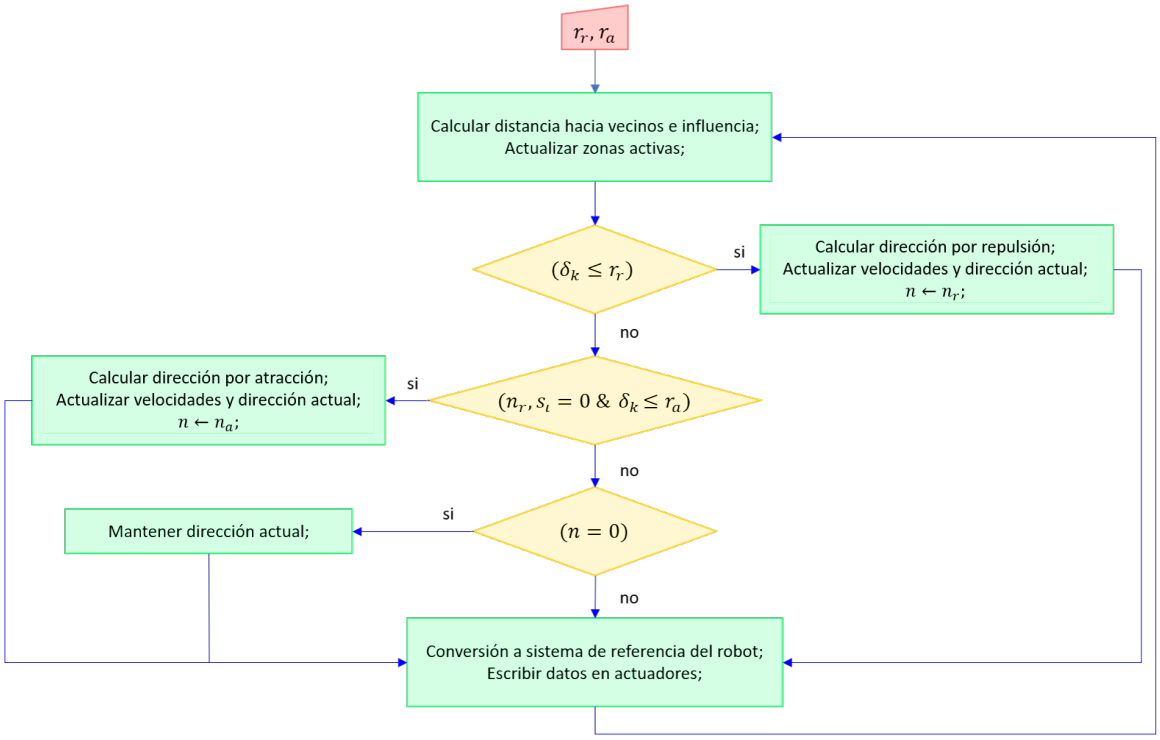


Figura 2.15: Diagrama de flujo de las reglas de comportamiento de robot-presa.

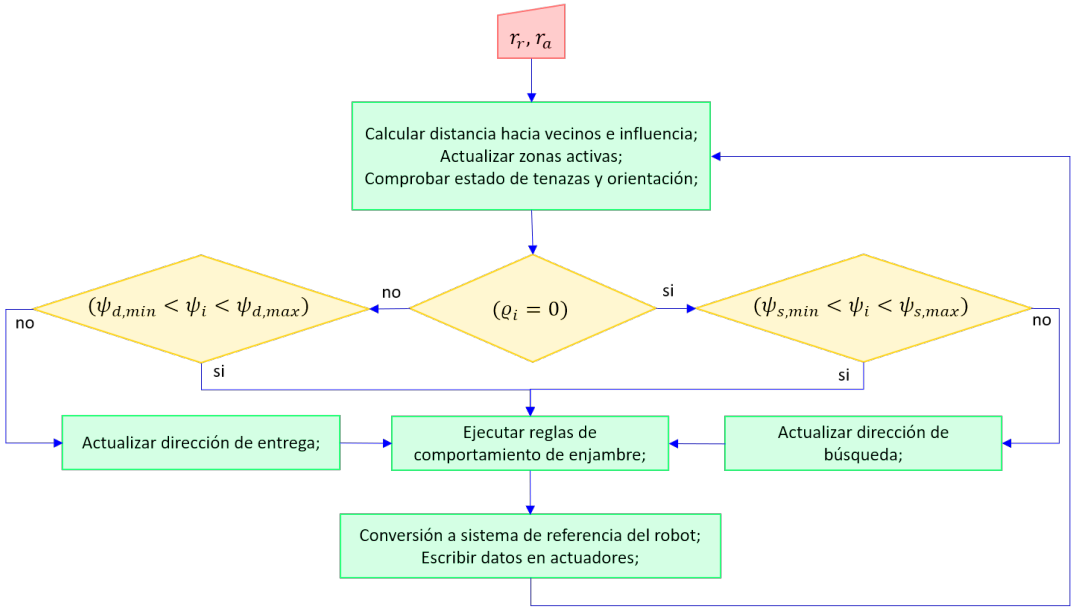


Figura 2.16: Diagrama de flujo de las reglas de comportamiento para tarea de transporte de objetos.

Capítulo 3

Descripción experimental

En este capítulo, se describen las etapas para el desarrollo de las simulaciones y experimentos, además del método utilizado para un análisis numérico, mediante una herramienta de cómputo con técnicas de visión. Por último, se exponen dos casos experimentales para el estudio del enjambre. El objetivo de estos experimentos es observar los cambios que surgen en el comportamiento ante variaciones en los parámetros y el efecto de la exposición a factores de influencia.

3.1 Desarrollo de experimentos físicos

Las etapas que se siguieron para la obtención de resultados en los experimentos físicos se muestran en la Figura 3.1. Cada experimento se repite cambiando el valor de los parámetros de repulsión y atracción. Estos se ajustan mediante una interfaz usuario-robot por medio de *Bluetooth*®. Los robots obtienen por medio de sus sensores la información percibida en el entorno y esta es procesada por el microcontrolador, convirtiéndola en movimiento conforme a las reglas de comportamiento. En el transcurso del experimento, se captura video y es procesado para obtener datos cuantitativos en el análisis del enjambre.

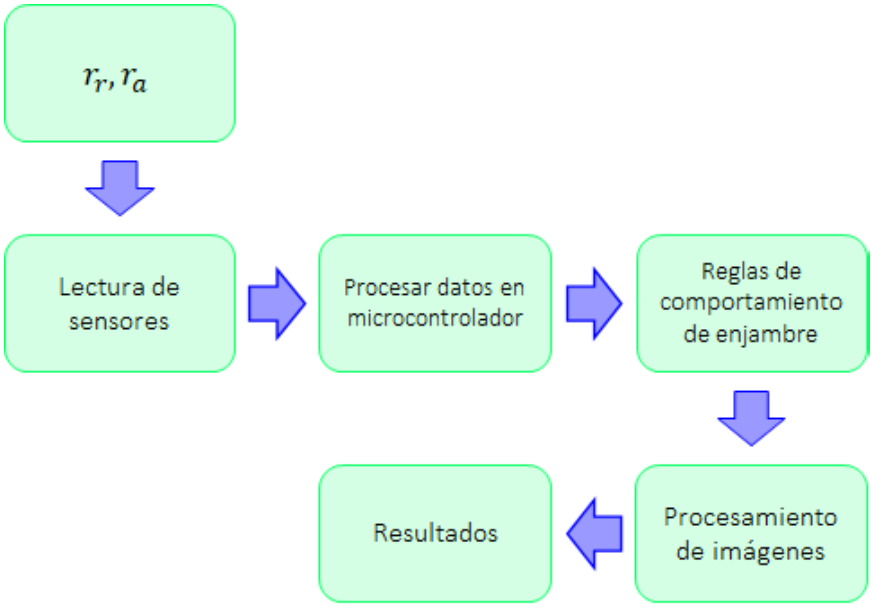


Figura 3.1: Etapas de desarrollo de experimentos físicos.

3.1.1 Especificaciones de robots en experimentos físicos

En la Tabla 3.1, se dan las especificaciones de percepción en las zonas de repulsión, orientación, atracción e influencia, en función a las características de funcionamiento de los sensores empleados en los robot y a su ubicación. Estos valores parten al tomar como origen el vector que apunta en la dirección del sensor (Figura 2.9). En los experimentos realizados, los radios r_r y r_a cambian según los parámetros establecidos, mientras que r_o y r_i permanecen constantes con valor nulo y con la distancia máxima de percepción por los fotorresistores, respectivamente. El alcance de las zonas por los robot puede verse en la Tabla 3.2.

Con respecto a las velocidades (aproximadas) de operación permitidas, éstas son en relación a un voltaje de referencia aplicado en los motores de CC, en correspondencia a la zona activa del robot (Tabla 3.3).

Tabla 3.1: Especificaciones de las zonas de repulsión, orientación, atracción e influencia en función a los sensores empleados.

Zona	**Radio de percepción (m)**	**Rango de percepción (grados)**
Q_{1a}	$0-4$	$\pm 15°$
Q_{1o}	$0-4$	$\pm 15°$
Q_{1r}	$0-4$	$\pm 15°$
Q_{2r}	$0-0.15$	$\pm 30°$
Q_{3r}	$0-0.15$	$\pm 30°$
Q_{4r}	$0-0.1$	$\pm 30°$
Q_{5r}	$0-0.1$	$\pm 30°$
γ_l	$0-2$	$\pm 45°$
γ_r	$0-2$	$\pm 45°$

Tabla 3.2: Radios permitidos en las zonas de percepción.

Zona	**Radio de percepción (m)**
Repulsión	$0-0.15$
Orientación	$N.A.$
Atracción	$0-4$
Influencia	2

Tabla 3.3: Velocidades en zonas de percepción en experimentos físicos.

Zona	**Velocidad (m/s)**	**Voltaje aplicado (v)**
Repulsión	0.08	2.8
Orientación	0.1	3
Atracción	0.2	4.2
Influencia	0.1 - 0.2	3 - 4.2
Fuera de rango	0.1	3

3.2 Desarrollo de simulaciones

Al igual que los experimentos físicos, cada simulación se repite cambiando los parámetros del enjambre para cada caso experimental. A partir del modelo cinemático (Ec. 2.4) y dinámico (Ec. 2.15), y considerando los parámetros propuestos para cada robot (Tabla 2.1), se obtienen las condiciones de los robots en la plataforma de simulación, donde se implementaron las reglas de comportamiento ajustando únicamente los valores r_r y r_a. Una vez finalizada la simulación, se calculan las métricas de rendimiento del enjambre y se generan tablas con los datos recolectados de cada simulación. Las etapas que se siguieron para la obtención de resultados en la plataforma de simulación se muestran en la Figura 3.2.

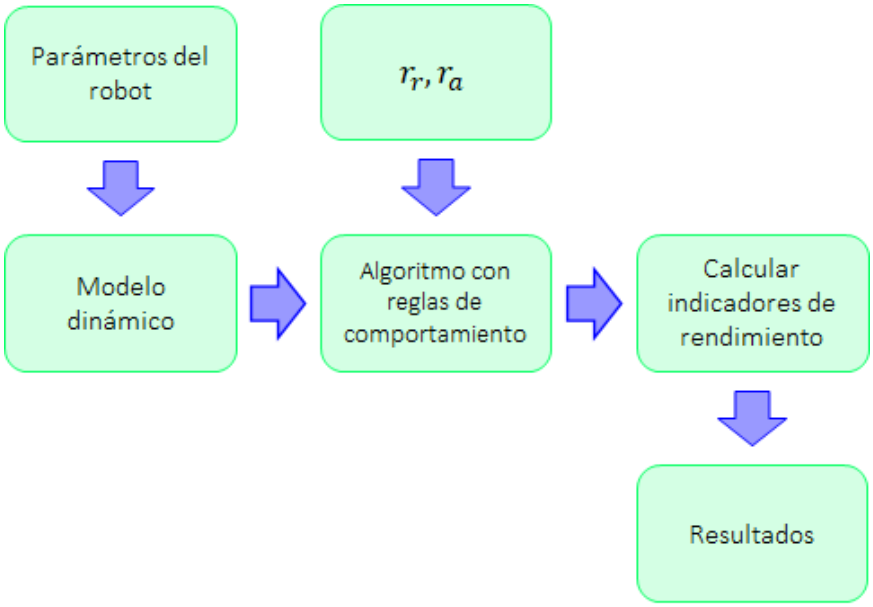


Figura 3.2: Etapas de desarrollo de simulaciones.

Los robots simulados consideran las limitaciones de los robot reales, las cuales se describen en la Sección 2.2.3, y se ajustan para tener el mismo rango de percepción que se mostró en la Tabla 3.2 y las velocidades de la Tabla 3.3. Las simulaciones fueron realizados en el software *Scilab 6.0.1*, con una máquina con procesador *Intel*® *Core*^{TM} i7-7500U CPU 2.9 GHz, RAM 8GB, y 64-bit.

3.3 Métricas de evaluación

En la literatura, se han definido diferentes métricas de rendimiento para evaluar un enjambre de robots al realizar una tarea [33]. En este trabajo, el rendimiento del enjambre se mide considerando el tiempo en completar la tarea y las medidas estadísticas propuestas: (1) el centro de masa del enjambre (C_m), y (2) una métrica de dispersión (ς). Representadas por las Ecuaciones (3.1) y (3.2), respectivamente.

$$C_m = \frac{1}{N}\sum_{i=1}^{N}\rho_i \tag{3.1}$$

$$\varsigma = \sqrt{\sum_{i=1}\frac{(\rho_i - C_m)^2}{N}} \tag{3.2}$$

donde N es el total de individuos y ρ_i es la posición de cada miembro del enjambre (x, y) respecto a una referencia preestablecida.

El centro de masa y la dispersión, son indicadores que nos muestran de manera cuantitativa, los cambios en el comportamiento que se genera en el enjambre bajo el factor de influencia y cambios en r_r y r_a. Se forma una elipse para visualizar cualquier cambio realizado en el enjambre, donde se toma el centro de masa y la dispersión como el origen y los semiejes de la elipse, respectivamente.

3.3.1 Sistema de procesamiento de imágenes

Se desarrolló una herramienta con técnicas de visión por computador en *Scilab*, para obtener la posición de los miembros del enjambre en un tiempo determinado. En la Figura 3.3, se describe el proceso de esta herramienta con un diagrama de flujo.

En el caso de la tarea de depredador-presa, el video capturado en los experimentos, se realiza desde una vista superior con una cámara *HD* con lente gran angular, mientras que la tarea de transporte de objetos, con un ángulo de inclinación aproximado de 30° con respecto al eje vertical, con una cámara *DSLR*. Una vez capturado el video, se extraen imágenes en tiempos específicos, las cuales son transformadas a escala de grises y pasan por un umbral que las convierte en imágenes binarias. Después, se buscan las regiones conectadas, asignando a cada región un número. Todos los píxeles que no pertenecen a una región son píxeles de fondo y se les da el número cero. Cada región conectada indica un miembro del enjambre, del cual se obtienen sus coordenadas. Teniendo las coordenadas del plano virtual (x, y) de cada robot, se obtiene la posición del centro de masa y la dispersión del enjambre aplicando las Ecuaciones (3.1) y (3.2).

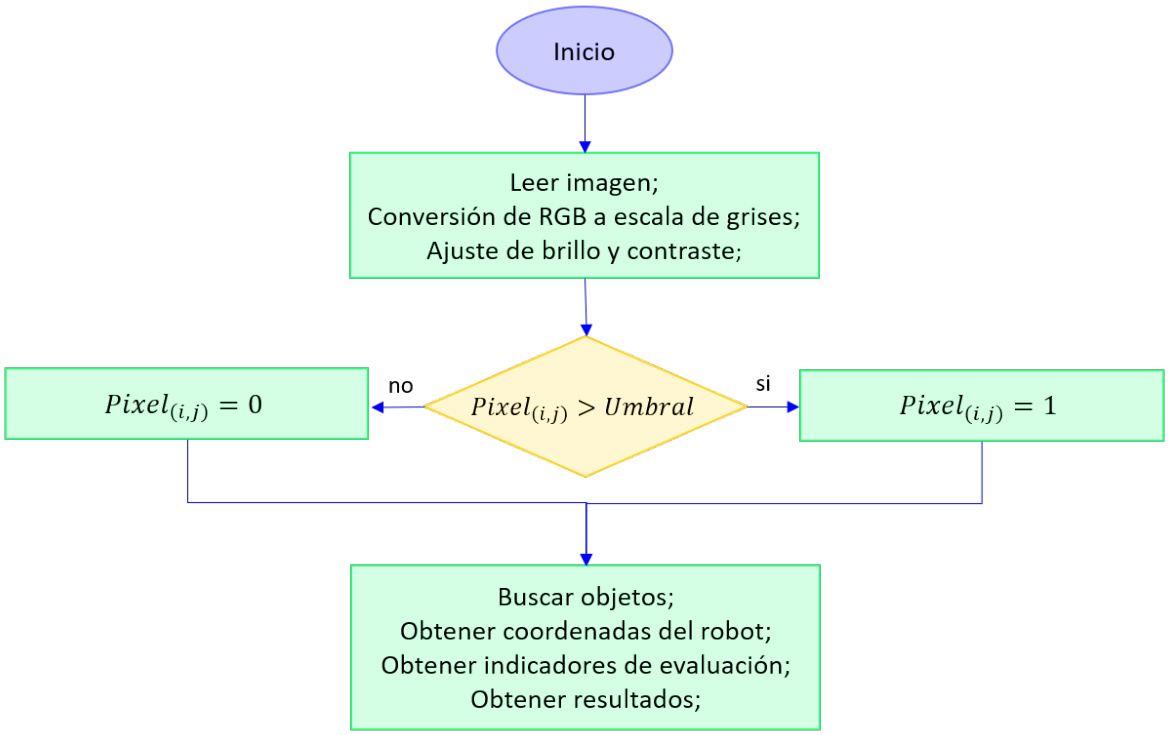


Figura 3.3: Diagrama de flujo de herramienta de procesamiento de imagen.

3.4 Descripción de casos experimentales

En esta sección se describen dos casos experimentales en los cuales se ajustan los valores r_r y r_a para gobernar en el comportamiento y que emerja colaboración entre los miembros del enjambre.

3.4.1 Depredador-presa

La tarea de depredador-presa, se inspira en especies que comparten un ambiente y pueden desarrollar interacciones entre sí, donde el comportamiento del sistema depende del tamaño de la densidad de ambas poblaciones. Esta tarea ha sido estudiada en *SMR* con múltiples variantes [85, 86, 87, 88, 89, 90, 91, 92, 93]. Algunas posibilidades de relaciones entre presas y depredadores: (1) único depredador - única presa, (2) múltiples depredadores - única presa, (3) único depredador - múltiples presas, (4) múltiples depredadores - múltiples presas, (5) uso de mediadores entre depredadores y presas. En este trabajo, nos enfocamos en el comportamiento colaborativo de múltiples depredadores a través de reglas locales para atrapar una sola presa.

Para este caso, se utilizó la arquitectura de robot *Terran* (Figura 2.2). Los robot-depredadores, se sitúan con una posición y orientación aleatoria dentro de un área delimitada, mientras que el robot-presa se encuentra en el centro del área de prueba. El robot-presa es otro *Terran*, que emite luz para poder ser distinguido de los robot-depredadores. El experimento termina cuando los robot-depredadores logran rodear al robot-presa.

3.4.2 Transporte de objetos

La tarea de transporte de objetos, puede observarse en el comportamiento de muchos insectos; por ejemplo, las hormigas agrupan sus crías [94] y las termitas depositan lodo para construir nidos complejos [95]. Los insectos generalmente explotan los gradientes naturales, como los gradientes de temperatura o los gradientes de feromonas para abordar estas tareas. En la *RE*, el transporte de objetos generalmente se aborda utilizando máquinas probabilísticas de estados finitos. Los robots exploran el entorno al azar y reaccionan de diferentes maneras al descubrimiento de objetos disponibles. Se han realizado múltiples estudios en *SMR* relacionados a este enfoque [96, 97, 98, 99, 100, 101].

Para este caso se utiliza la arquitectura de robot *BugBot* (Figura 2.3), el objetivo es recolectar objetos y agruparlos en la zona deseada. Los *BugBot* se sitúan con una posición y orientación aleatoria dentro de un área limitada que llamaremos "nido", que corresponde al lugar donde hay que depositar los objetos. La posición de los objetos es de forma aleatoria dentro de una zona iluminada (influencia) que ayuda a los *BugBot* a encontrarlos, dando una aproximación de su ubicación. Cuando un robot recoge un objeto, un nuevo estímulo lo mueve de regreso al nido, que también se encuentra iluminado, para dar una aproximación del punto de descarga. Esto permite que la tarea principal se divida en subtareas de búsqueda y entrega. Los estímulos son generados por una fuente de luz en el entorno y la tarea finaliza cuando los robots depositan todos los objetos en el nido.

Capítulo 4

Resultados

En este capítulo se describen los resultados obtenidos al implementar las reglas de comportamiento, en una plataforma de simulación y un sistema físico. Para cada experimento, se especifican las dimensiones del área de prueba, el tamaño de población del enjambre y el set de configuraciones paramétricas. Además, se presentan las gráficas y tablas con los resultados de cada experimento.

4.1 Primer caso experimental (Tarea de relaciones tipo depredador-presa)

4.1.1 Simulaciones

Se llevó a cabo una serie de simulaciones, para comprender los efectos de los parámetros de manera realista, antes de ser implementados en un sistema físico. Las simulaciones se realizaron con 25 robot-depredadores y un robot-presa, con las capacidades sensoriales de los robot *Terran*, en un área de prueba de 10 x 10 m^2, y por cada simulación se hicieron tres repeticiones. Todos los robot-depredadores

tienen la misma configuración paramétrica por cada experimento, mientras que al robot-presa se le da un valor alto de repulsión para mantenerse alejado de los robots depredadores y un valor bajo de atracción para no ser atraído por ellos, estos valores se establecieron en $r_r = 0.15$ y $r_a = 0.2$ para todas las pruebas. La Tabla 2.31, muestra el voltaje aplicado en los motores de CC, que corresponde a las zonas activas en los robots; mientras que el umbral e influencia se estableció en el 10 % de la capacidad total de las LDRs. Estos valores se seleccionaron en función de las restricciones físicas de los sensores. Las posiciones iniciales de los robot-depredadores y el robot-presa son aleatorias. La simulación termina cuando los robot-depredadores rodean al robot-presa, es decir, cuando gira sobre su eje.

Se realizó una simulación gráfica para ilustrar el rendimiento del enjambre, y se dibujó una elipse para visualizar cualquier cambio realizado en el enjambre, que representa el área de distribución del enjambre (área de cobertura). El factor de influencia está determinado por el estímulo de luz emitido por la presa-robot, que se representa como una flecha amarilla; mientras que los robot-depredadores con flechas negras. Las Figuras 4.1 - 4.9, ilustran algunas instantáneas de simulaciones, cuando los robot-depredadores están atrapando al robot-presa. Considerando todas las pruebas y cambios de parámetros, se generó la Tabla 4.1, que incluye el centro de masa del enjambre (C_m), la dispersión (ς), el área de la elipse (ε) y el tiempo de captura (t_c).

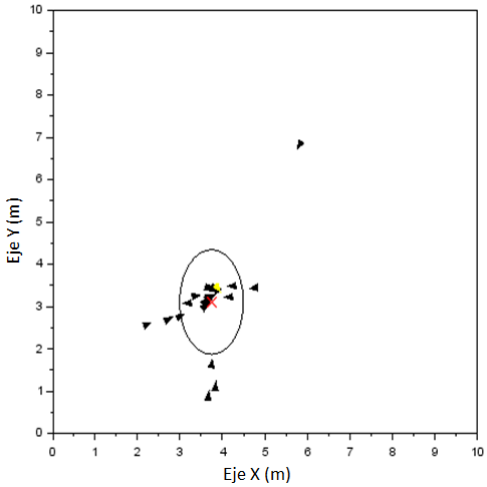
En base a los resultados obtenidos por simulaciones, se observó que la repulsión tiene una fuerte influencia en el comportamiento del enjambre. Con valores bajos (ver Figuras 4.1, 4.4 y 4.7), se genera una mayor concentración en el enjambre y la mayoría de los robot-depredadores logran rodear al robot-presa; mientras que, con valores altos (ver Figuras 4.3, 4.6 y 4.9), se genera una expansión en el área de cobertura y algunos robot-depredadores pierden el rastro del robot-presa o no logran localizarlo. Por otro lado, con valores bajos de atracción (ver Figuras 4.1, 4.2 y 4.3), los robot-depredadores son más independientes entre sí, provocando que sean más fieles a la trayectoria del robot-presa, pero pueden perderse fácilmente; mientras que,

con valores altos (ver Figuras 4.7, 4.8 y 4.9), los robot-depredadores se atraen por otros que son localizados en la zona frontal, formando cadenas de robots, además, los robot-depredadores extraviados vuelven a unirse al enjambre. El detalle de la discusión se comenta en la Sección 4.1.4.

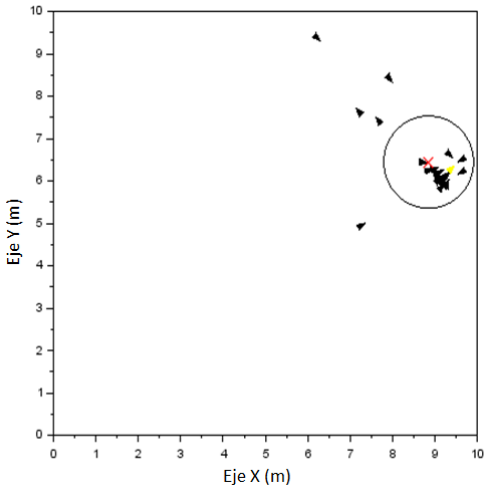
4.1.2 Experimentos por implementación física

Los resultados por simulación confirman que las ecuaciones basadas en repulsión, atracción, orientación e influencia, pueden transformarse en reglas que utilizan la percepción local y mantienen las propiedades del enjambre. Estas reglas se implementaron en un microcontrolador, considerando los sensores locales de cada robot con arquitectura *Terran*, para replicar las mismas pruebas que las simulaciones. Cada robot puede reprogramar sus parámetros comunicándose con una computadora a través de *Bluetooth*®.

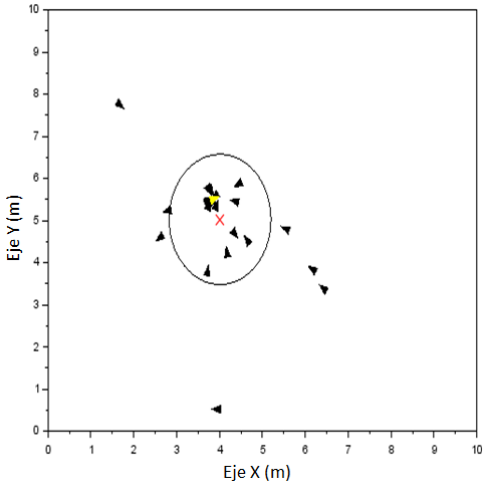
Se desarrolló una herramienta de visión por computadora para procesar instantáneas de video, considerando que el tamaño de la imagen es de 610 x 460 píxeles^2. Se desconocen las posiciones de cada miembro del enjambre robótico, por lo que estas instantáneas se tomaron en un momento específico y se procesaron para obtener las posiciones en un plano virtual (x, y) de cada miembro del enjambre en un momento específico. Estas posiciones son útiles para estimar el centro de masa del enjambre y su dispersión, utilizando las Ecuaciones (3.1) y (3.2), respectivamente. Los experimentos se realizaron con siete robot-depredadores y un robot-presa, en un área de prueba de 3 x 2.4 m^2. Se realizaron las mismas configuraciones paramétricas de las simulaciones por medio de la plataforma robótica, donde para cada configuración se realizaron tres pruebas.



(a) Prueba 1

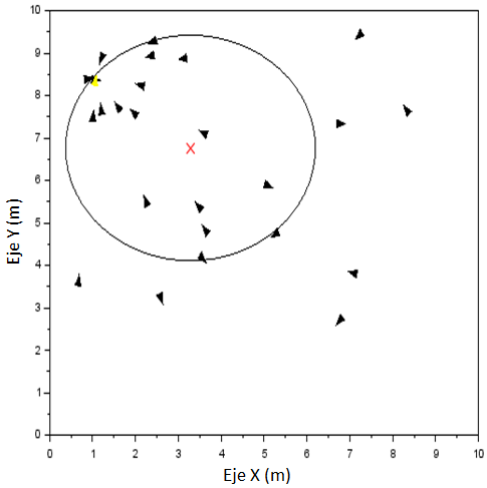


(b) Prueba 2

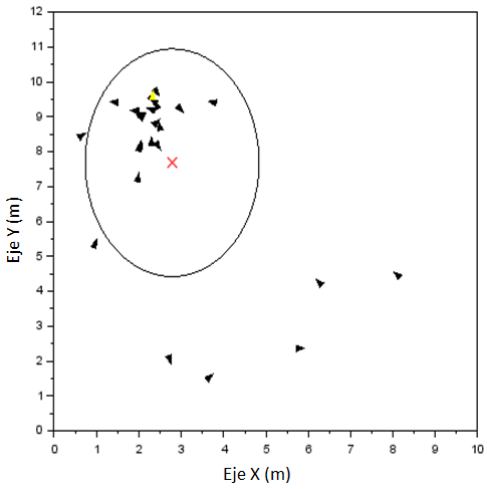


(c) Prueba 3

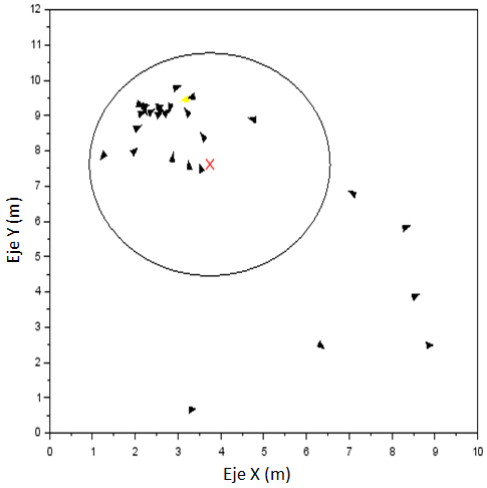
Figura 4.1: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

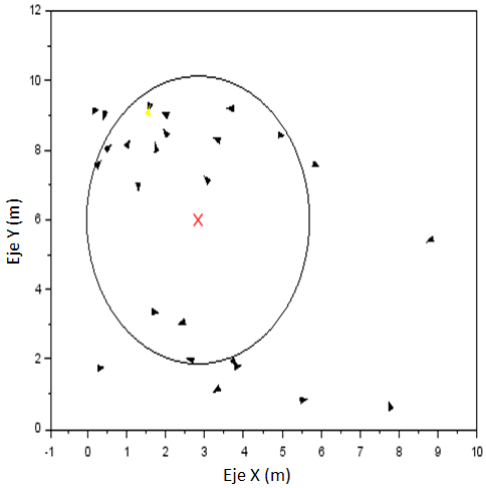


(b) Prueba 2

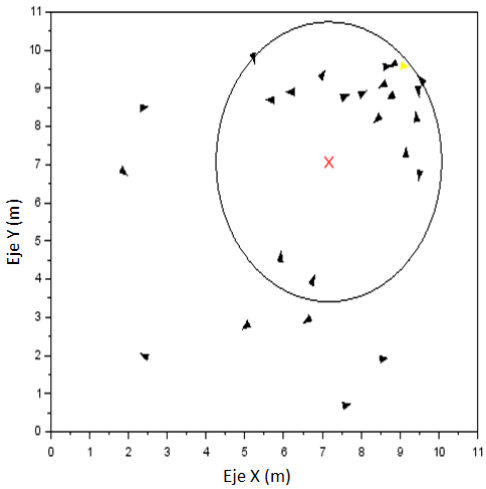


(c) Prueba 3

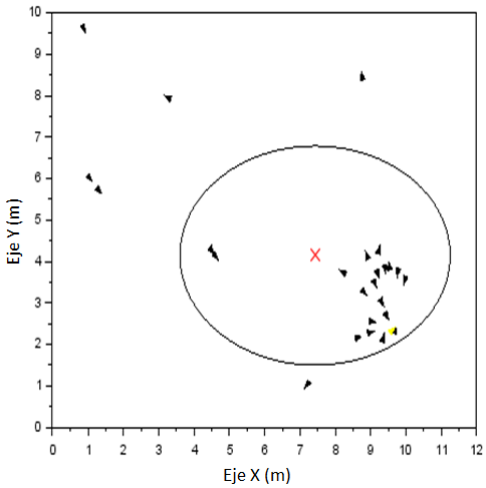
Figura 4.2: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

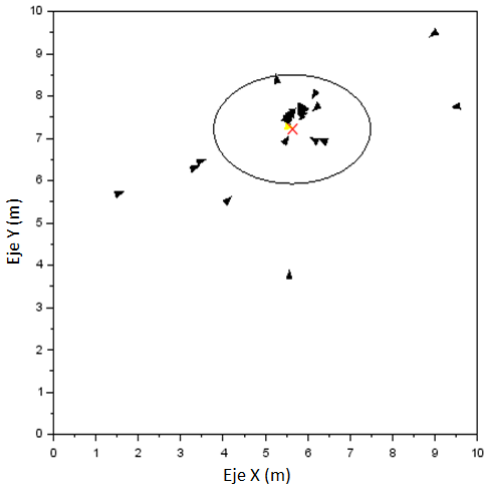


(b) Prueba 2

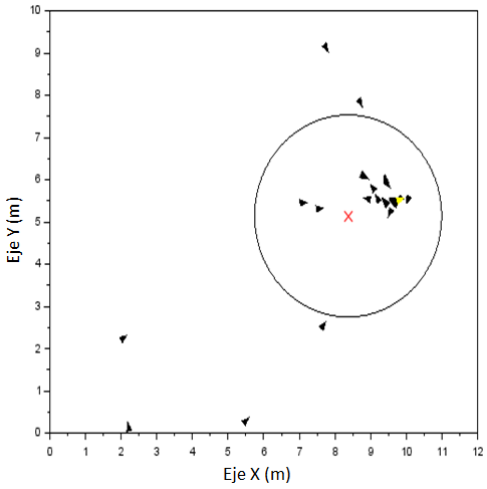


(c) Prueba 3

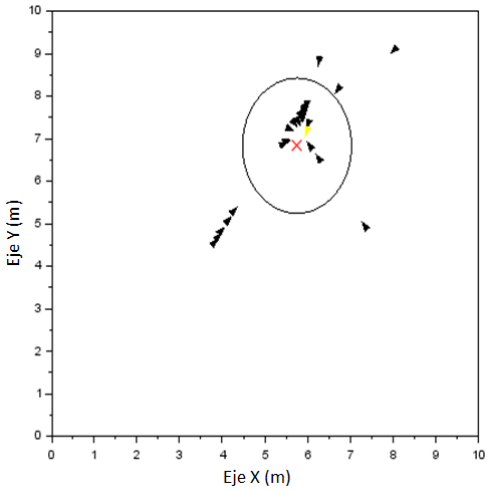
Figura 4.3: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

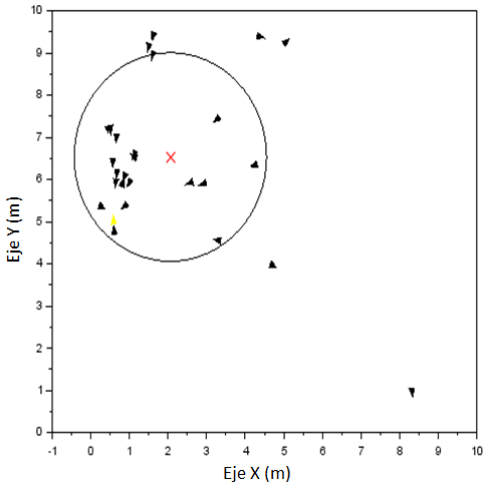


(b) Prueba 2

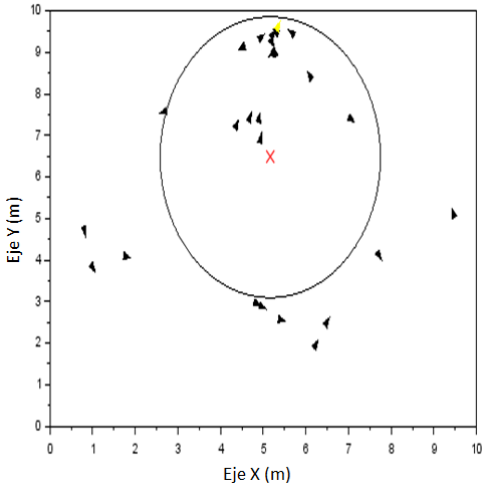


(c) Prueba 3

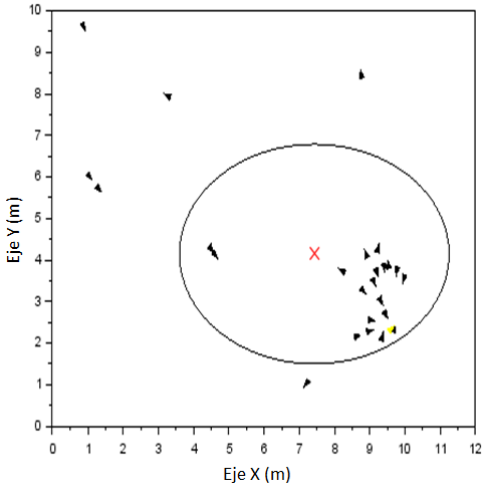
Figura 4.4: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

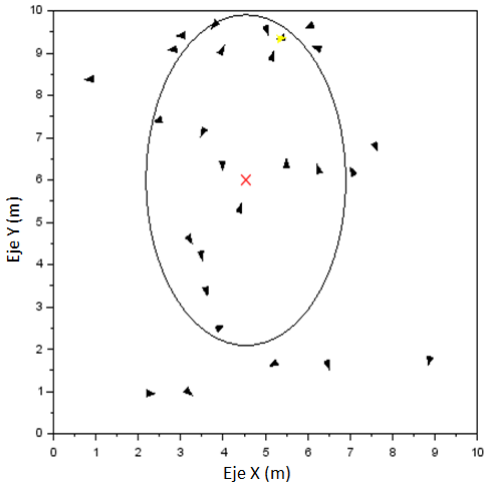


(b) Prueba 2

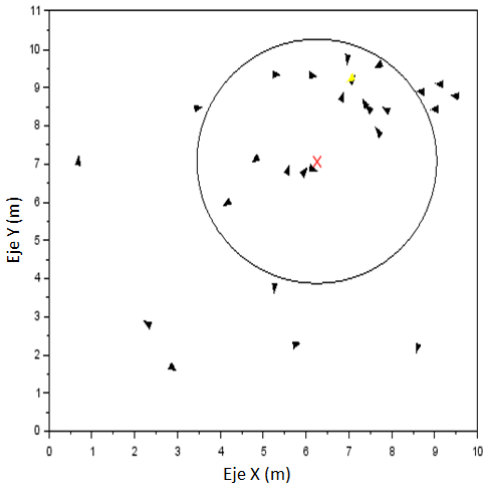


(c) Prueba 3

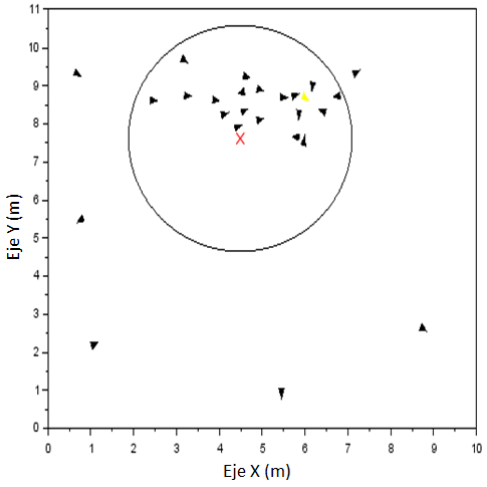
Figura 4.5: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

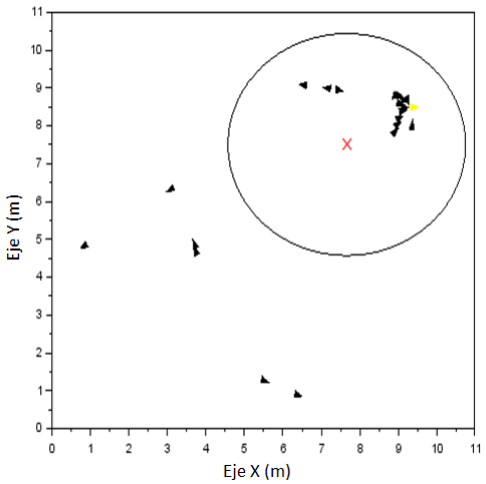


(b) Prueba 2

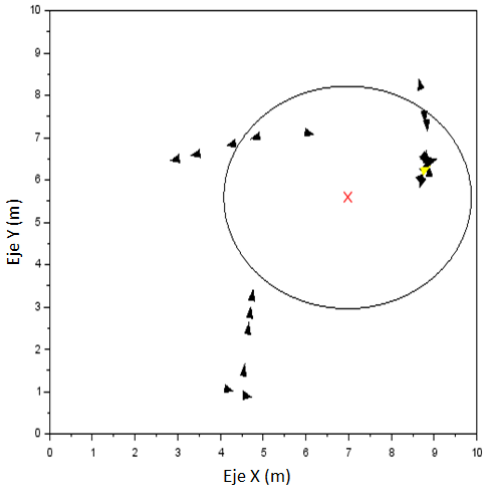


(c) Prueba 3

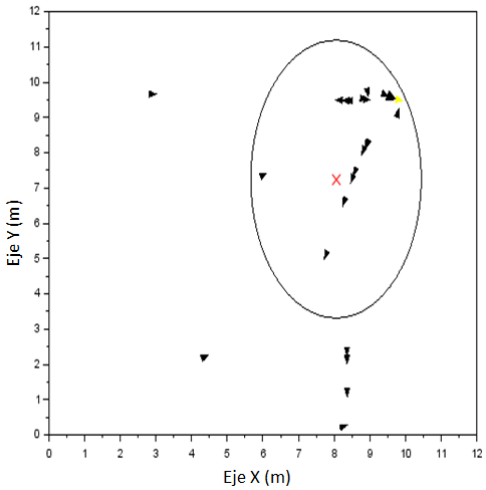
Figura 4.6: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

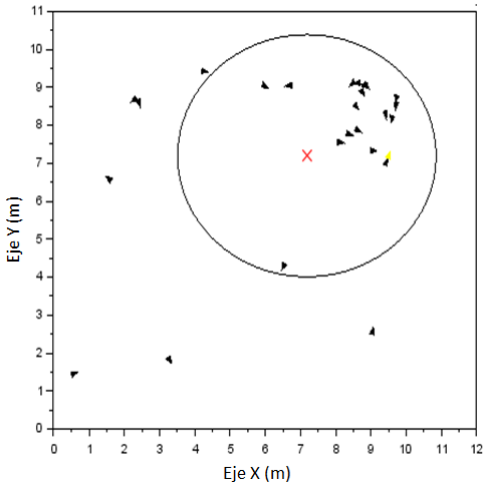


(b) Prueba 2

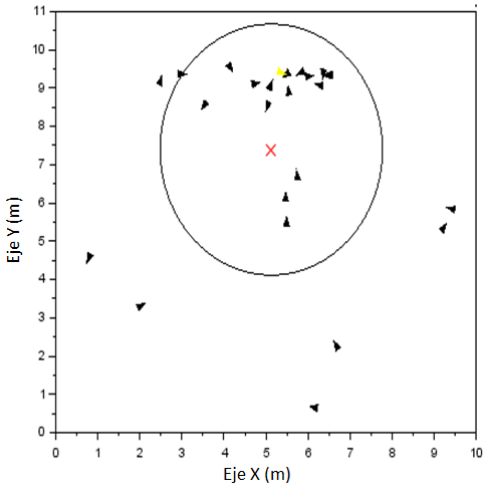


(c) Prueba 3

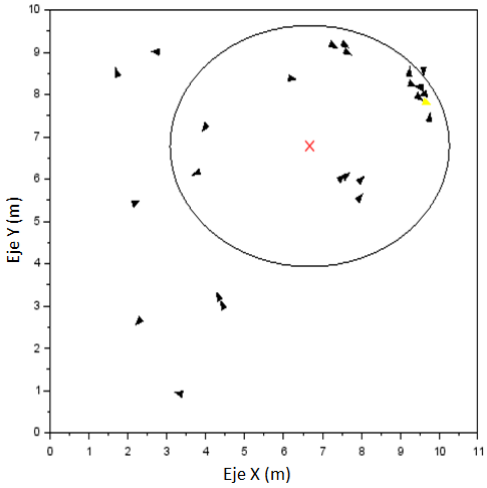
Figura 4.7: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 1$ para los robot-depredadores.



(a) Prueba 1

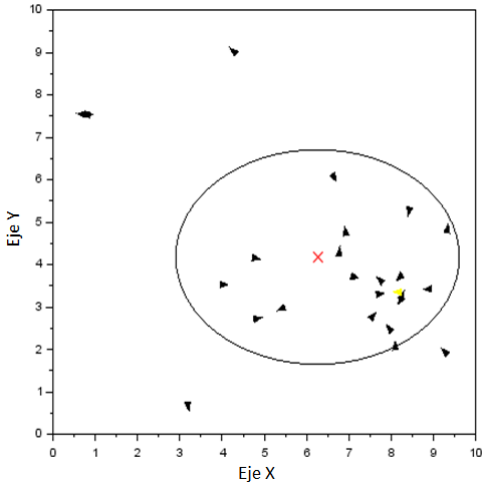


(b) Prueba 2

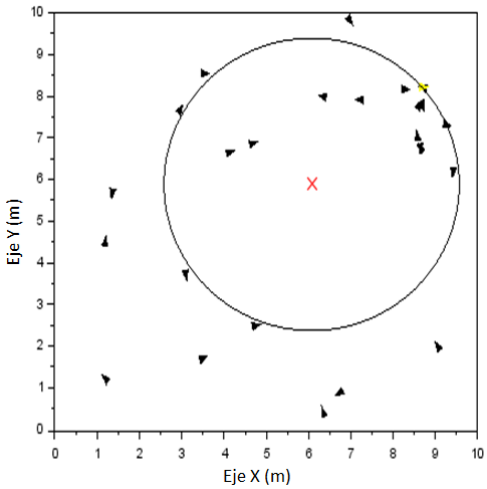


(c) Prueba 3

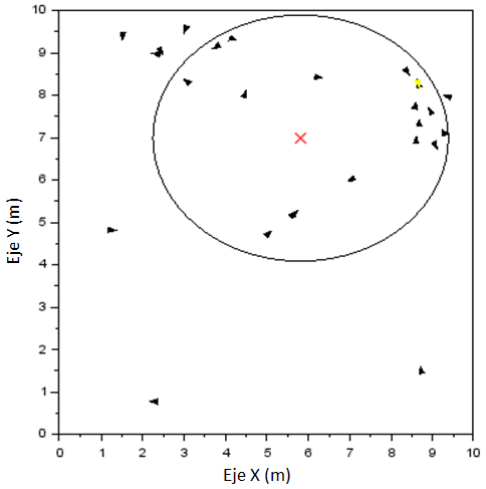
Figura 4.8: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores.



(a) Prueba 1



(b) Prueba 2

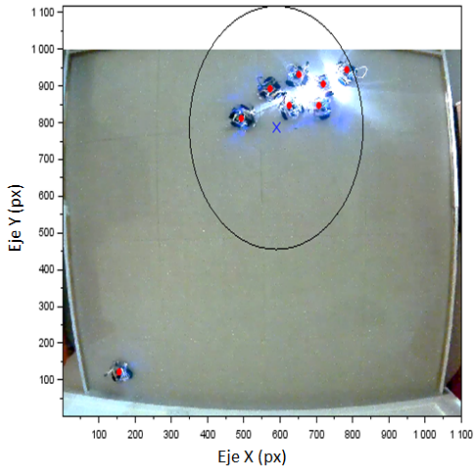


(c) Prueba 3

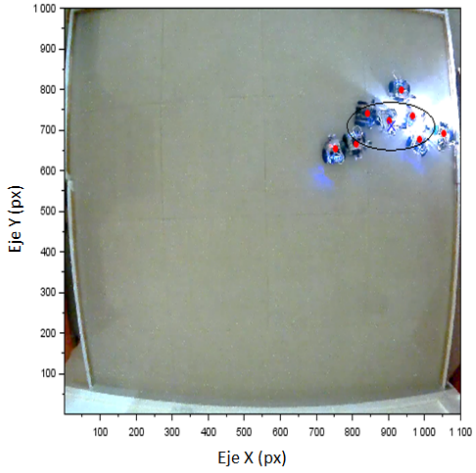
Figura 4.9: Simulación de enjambre de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 1$ para los robot-depredadores.

Tabla 4.1: Resultados de tarea depredador-presa, basados en simulaciones.

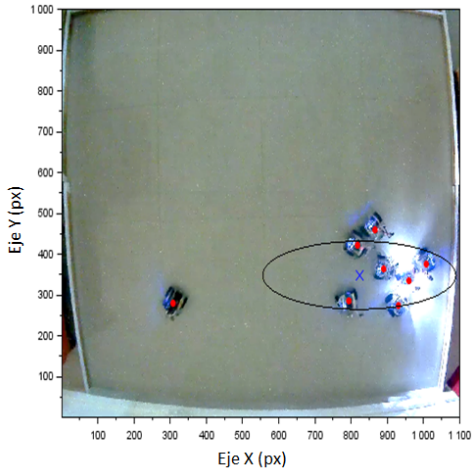
Test	r_r (m)	r_a (m)	No. de prueba	Cm_x	Cm_y	ς_x (m)	ς_y (m)	ε (m^2)	t_c (s)
1	0.05	0.2	1	3.744	3.104	0.950	0.577	1.721	123
			2	8.850	6.444	0.838	0.814	2.143	128
			3	4.011	5.024	1.189	0.913	3.413	132
			Media	**5.535**	**4.857**	**0.992**	**0.768**	**2.426**	**128**
2	0.1	0.2	1	3.284	6.763	2.043	2.240	14.379	136
			2	2.779	7.682	2.510	1.583	12.480	132
			3	3.741	7.614	2.429	2.164	16.509	123
			Media	**3.268**	**7.353**	**2.327**	**1.995**	**14.456**	**130**
3	0.15	0.2	1	2.836	5.992	3.181	2.203	22.021	157
			2	7.167	7.072	2.819	2.236	19.806	190
			3	6.395	6.497	2.626	2.521	20.799	158
			Media	**5.466**	**6.520**	**2.876**	**2.320**	**20.875**	**168**
4	0.05	0.6	1	5.628	7.214	0.991	1.422	4.425	162
			2	8.366	5.140	1.841	2.018	11.670	154
			3	5.752	6.831	1.224	0.982	3.774	153
			Media	**6.582**	**6.395**	**1.352**	**1.474**	**6.623**	**156**
5	0.1	0.6	1	2.062	6.532	1.903	1.916	11.454	176
			2	5.165	6.471	2.603	1.987	16.246	177
			3	7.427	4.141	2.034	2.948	18.840	184
			Media	**4.885**	**5.715**	**2.180**	**2.283**	**15.513**	**179**
6	0.15	0.6	1	4.539	5.994	3.007	1.812	17.117	169
			2	6.250	7.071	2.461	2.153	16.649	182
			3	4.488	7.617	2.285	2.007	14.410	202
			Media	**5.092**	**6.894**	**2.584**	**1.991**	**16.059**	**184**
7	0.05	1	1	7.658	7.502	2.256	2.374	16.822	203
			2	6.972	5.586	2.023	2.229	14.163	185
			3	8.051	7.251	3.030	1.838	17.501	176
			Media	**7.560**	**6.780**	**2.436**	**2.147**	**16.162**	**188**
8	0.1	1	1	7.182	7.196	2.453	2.826	21.783	194
			2	5.126	7.390	2.523	2.035	16.128	236
			3	6.670	6.781	2.194	2.756	19.003	196
			Media	**6.326**	**7.122**	**2.390**	**2.539**	**18.971**	**209**
9	0.15	1	1	6.257	4.176	1.947	2.579	15.776	327
			2	6.073	5.878	2.702	2.692	22.849	227
			3	5.813	6.987	2.235	2.746	19.282	264
			Media	**6.048**	**5.681**	**2.295**	**2.673**	**19.302**	**273**



(a) Prueba 1

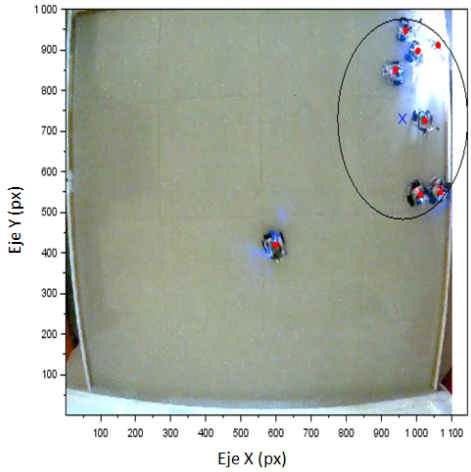


(b) Prueba 2

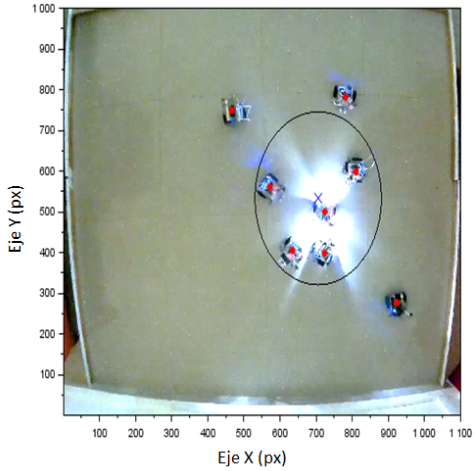


(c) Prueba 3

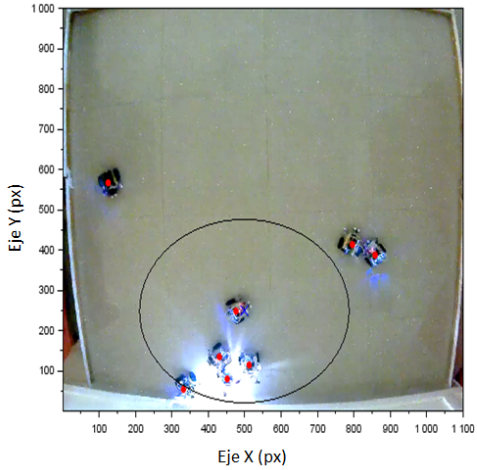
Figura 4.10: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

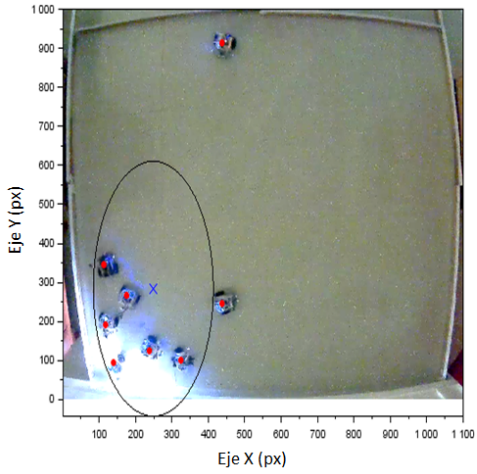


(b) Prueba 2

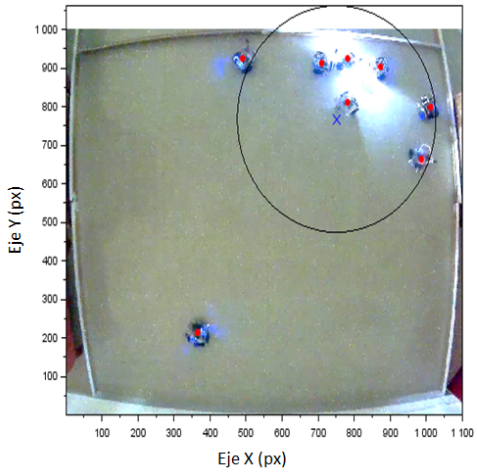


(c) Prueba 3

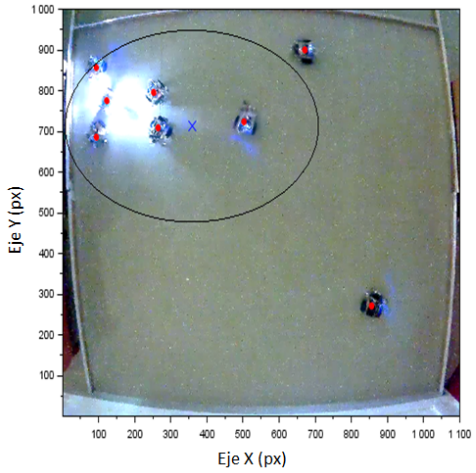
Figura 4.11: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

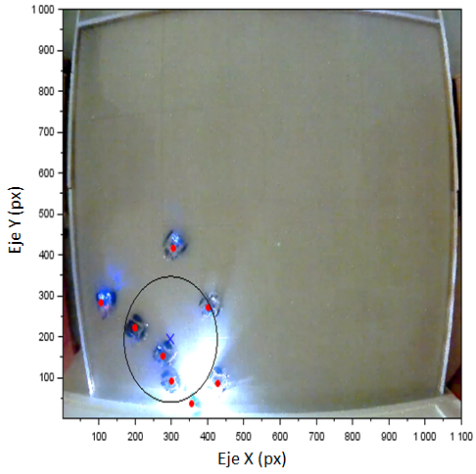


(b) Prueba 2

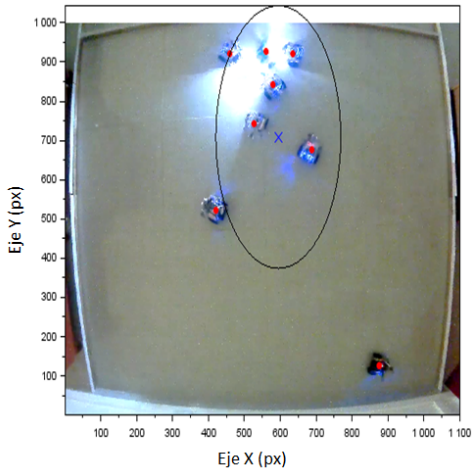


(c) Prueba 3

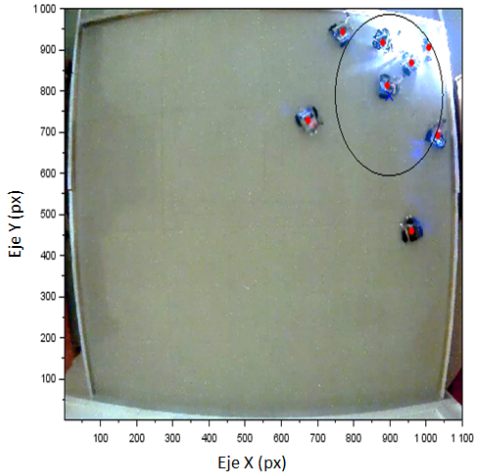
Figura 4.12: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.2$ para los robot-depredadores.



(a) Prueba 1

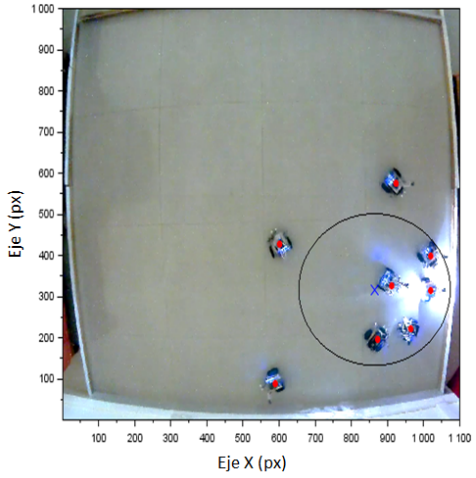


(b) Prueba 2

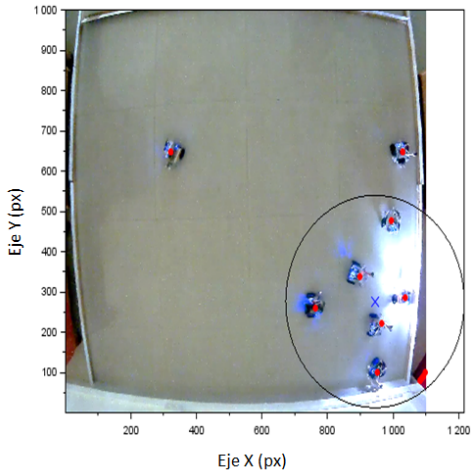


(c) Prueba 3

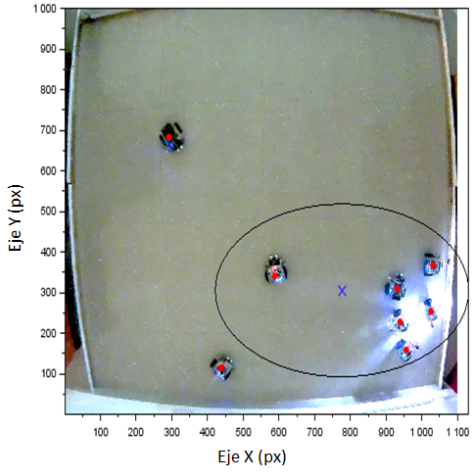
Figura 4.13: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

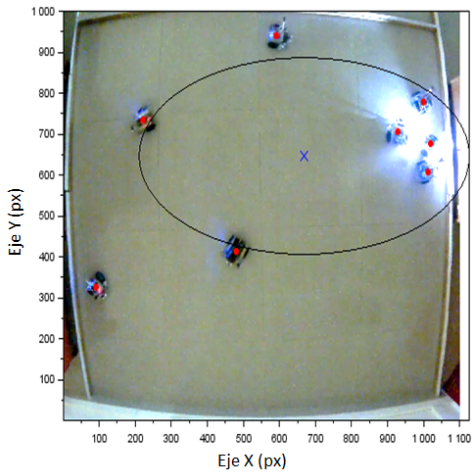


(b) Prueba 2

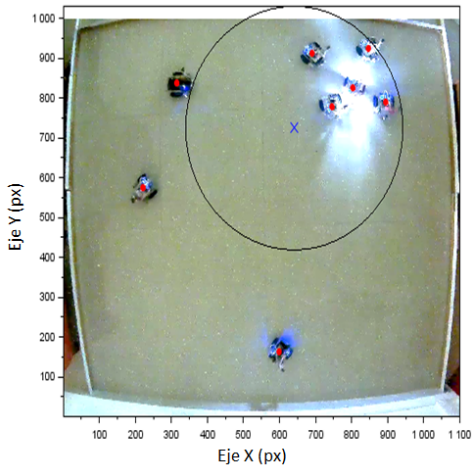


(c) Prueba 3

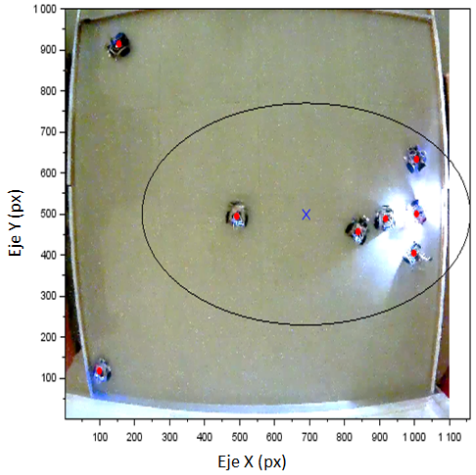
Figura 4.14: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

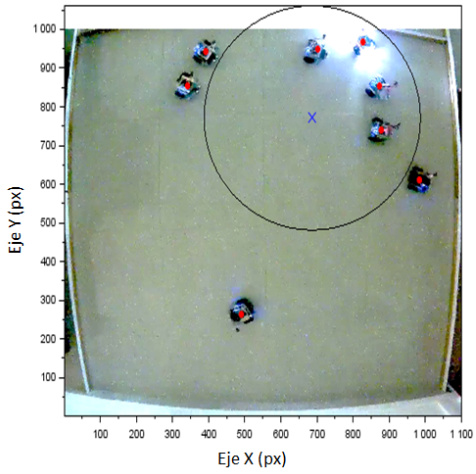


(b) Prueba 2

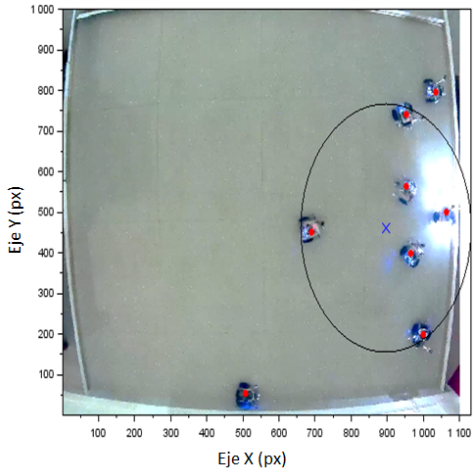


(c) Prueba 3

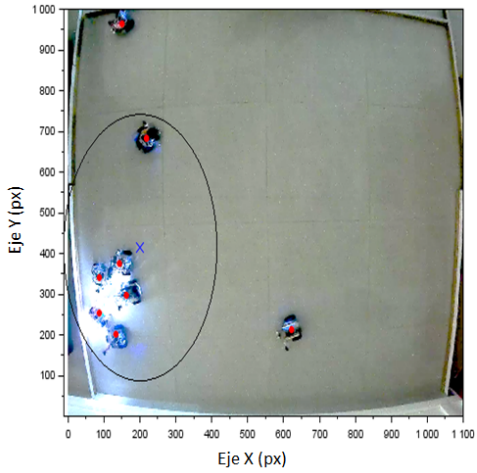
Figura 4.15: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.15$ y $r_a = 0.6$ para los robot-depredadores.



(a) Prueba 1

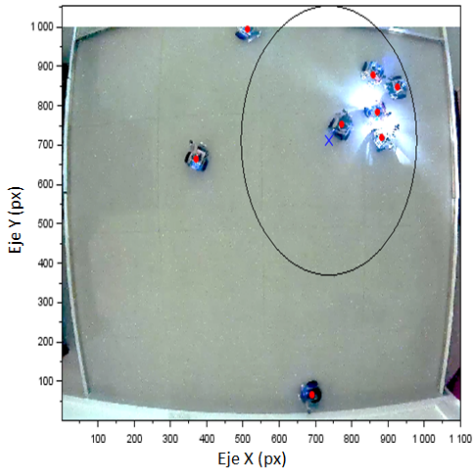


(b) Prueba 2

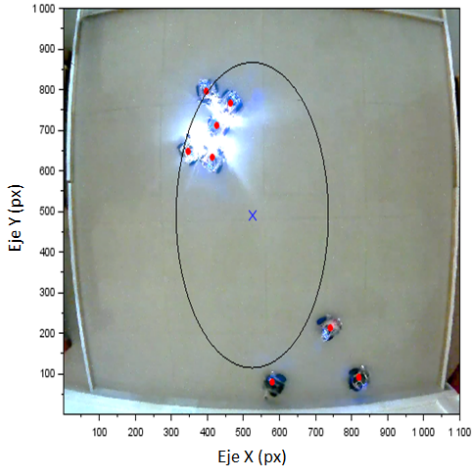


(c) Prueba 3

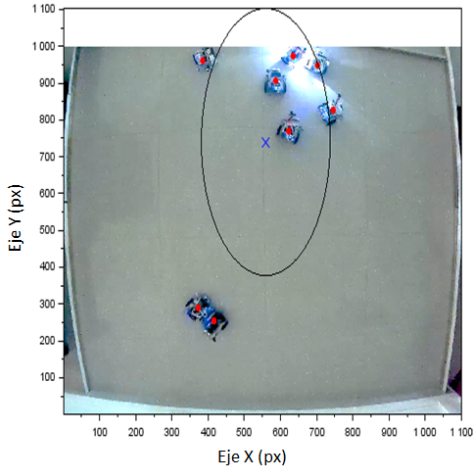
Figura 4.16: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.05$ y $r_a = 1$ para los robot-depredadores.



(a) Prueba 1

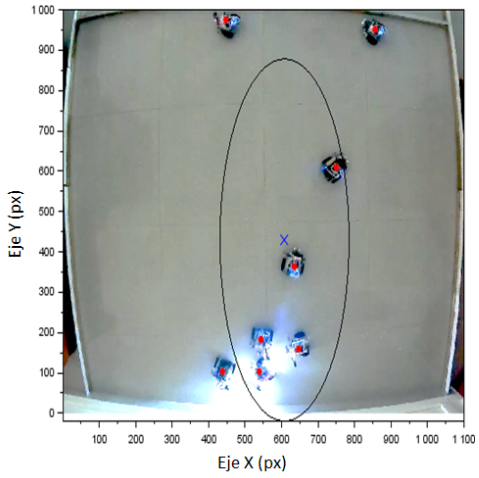


(b) Prueba 2

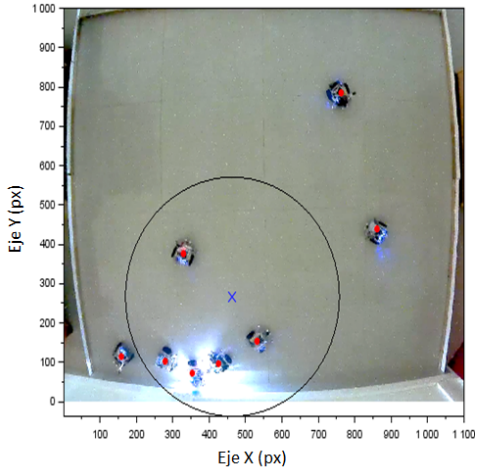


(c) Prueba 3

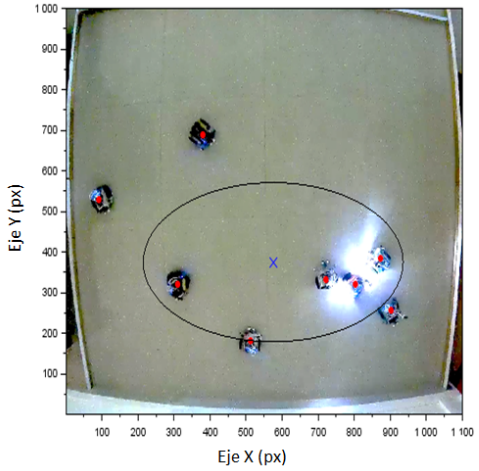
Figura 4.17: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores.



(a) Prueba 1



(b) Prueba 2



(c) Prueba 3

Figura 4.18: Manada de robots ejecutando tarea de depredador-presa con $r_r = 0.1$ y $r_a = 1$ para los robot-depredadores.

Tabla 4.2: Resultados de tarea depredador-presa, basados en implementación física.

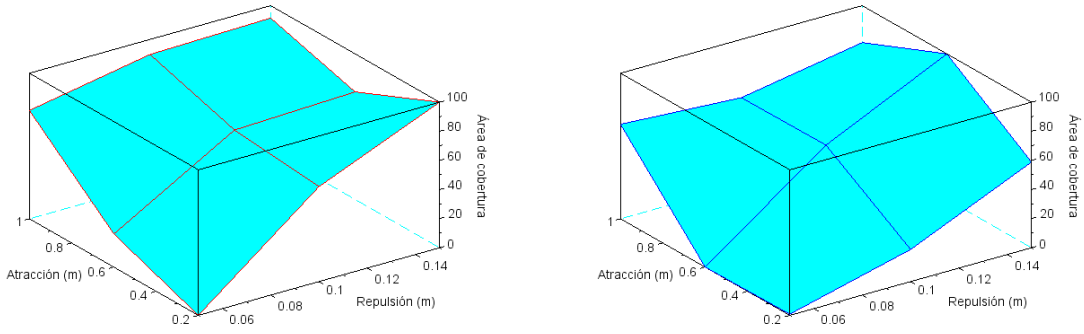
Test	r_r (m)	r_a (m)	No. de prueba	Cm_x	Cm_y	ς_x (px)	ς_y (px)	ε (px^2)	t_c (s)
1	0.05	0.2	1	906.625	709.500	93.513	45.144	13262	23
			2	589.125	786.375	184.077	254.752	147321	29
			3	822.375	348.625	205.200	64.110	41328	26
			Media	**772.708**	**614.833**	**160.930**	**121.335**	**67304**	**26**
2	0.1	0.2	1	960.375	729.625	142.360	189.759	84867	34
			2	705.625	533.375	134.612	163.629	69198	36
			3	498.875	248.625	221.393	175.128	121806	32
			Media	**721.625**	**503.875**	**166.122**	**176.172**	**91957**	**34**
3	0.15	0.2	1	249.000	283.875	126.884	251.440	100228	40
			2	750.750	768.000	211.870	226.060	150467	35
			3	358.375	713.125	269.503	181.153	153375	49
			Media	**452.708**	**588.333**	**202.752**	**219.551**	**134690**	**41**
4	0.05	0.6	1	298.000	193.625	98.774	118.246	36692	41
			2	594.375	708.625	133.840	257.571	108300	50
			3	897.375	790.250	114.262	150.715	54101	48
			Media	**596.583**	**564.167**	**115.625**	**175.510**	**66364**	**46**
5	0.1	0.6	1	863.875	317.250	161.319	141.918	71923	45
			2	945.042	276.458	209.543	202.874	133552	48
			3	668.875	646.750	352.396	184.788	204575	60
			Media	**825.931**	**413.486**	**241.086**	**176.527**	**136683**	**51**
6	0.15	0.6	1	640.750	724.750	231.701	235.215	171215	38
			2	776.375	305.750	272.843	163.725	140338	52
			3	690.125	500.000	361.117	207.970	235938	55
			Media	**702.417**	**510.167**	**288.553**	**202.303**	**182497**	**48**
7	0.05	1	1	607.750	429.500	136.113	345.585	147775	63
			2	524.375	491.000	162.486	289.447	147752	42
			3	201.500	414.875	164.050	251.812	129778	57
			Media	**444.542**	**445.125**	**154.216**	**295.614**	**141768**	**54**
8	0.1	1	1	736.875	712.000	186.244	263.095	153937	72
			2	896.500	461.500	181.537	235.187	134130	78
			3	559.125	740.250	136.939	279.097	120069	61
			Media	**730.833**	**637.917**	**168.240**	**259.126**	**136045**	**70**
9	0.15	1	1	686.750	771.250	230.473	223.175	161590	112
			2	463.750	266.750	226.034	233.924	166110	91
			3	575.000	375.125	277.264	150.827	131377	79
			Media	**575.167**	**471.042**	**244.590**	**202.642**	**153026**	**94**

Las Figuras 4.10 - 4.18, ilustran algunas instantáneas de los experimentos físicos y muestran la posición de cada miembro del enjambre, cuando los robot-depredadores atrapan al robot-presa. Considerando todos los experimentos, cambios de parámetros y pruebas por experimento, se generó la Tabla 4.2 que incluye el centro de masa del enjambre (C_m), la dispersión (ς), el área de la elipse (ε) y el tiempo de captura (t_c). Como en los resultados de la simulación, se formó una elipse en el enjambre considerando su área cubierta a través de la dispersión y el centro de masa calculado.

4.1.3 Análisis de datos y modelado de la emergencia de colaboración en el enjambre, a partir de los resultados obtenidos en la tarea de depredador-presa

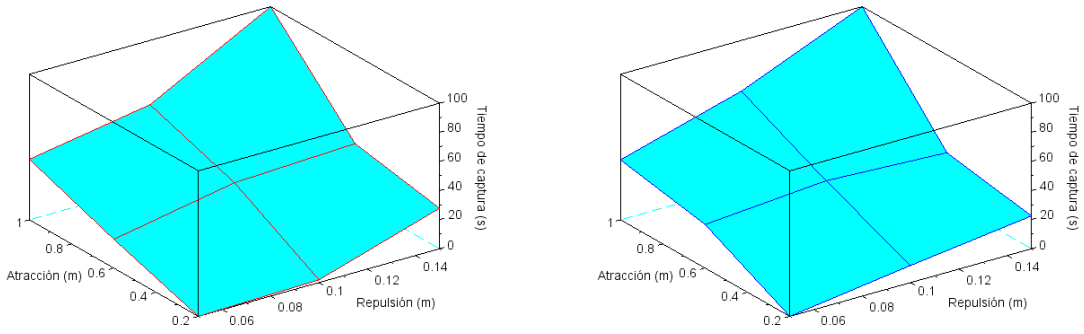
Las Figuras 4.19 y 4.20 ilustran el efecto de repulsión y atracción en el enjambre con respecto al área de cobertura (ε) y el tiempo de captura (t_c). Los datos fueron extraídos de las Tablas 4.1 y 4.2 con los valores promedio normalizados en un rango de 0 a 100. En las Figuras 4.19a y 4.19b, el parámetro de repulsión tiene mayor peso sobre el de atracción, donde el efecto en las simulaciones es más fuerte; además, se presenta un resultado semejante en el área de cobertura cuando el valor de los parámetros cambia. De la misma manera, las Figuras 4.20a y 4.20b, presentan grandes similitudes, a diferencia del efecto en el área de cobertura, el parámetro de atracción tiene mayor peso sobre el de repulsión; además, el efecto de los parámetros es muy parecido, otorgando resultados muy cercanos.

Con el fin de obtener una aproximación al modelo de comportamiento de las Figuras 4.19 y 4.20, y conocer el impacto que tienen los parámetros r_r y r_a, se realiza una regresión lineal múltiple que permite generar un modelo lineal con ecuación de tipo:



(a) Basado en simulaciones (b) Basado en implementación física

Figura 4.19: Efecto de repulsión y atracción en el área de cobertura para tarea depredador-presa.



(a) Basado en simulaciones (b) Basado en implementación física

Figura 4.20: Efecto de repulsión y atracción en el tiempo de captura para tarea depredador-presa.

$$Y = (b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n) \tag{4.1}$$

donde la variable dependiente Y se determina a partir de un conjunto de variables independientes llamadas predictoras $(X_1, X_2, ..., X_n)$, b_0 es la ordenada en el origen, $b_1, b_2, ..., b_n$ son los coeficientes de regresión que muestran el efecto de las variables predictoras sobre la variable independiente, y e es el error que muestra la diferencia entre el valor observado y el estimado por el modelo.

Para estimar la ecuación de regresión múltiple de cada superficie generada en las Figuras 4.19a, 4.19b, 4.20a y 4.20b se hizo uso del software *IBM® SPSS Statistics*, un programa estadístico informático. Los resultados de la salida para cada modelo son los siguientes:

Tabla 4.3: Estadísticas de la regresión para modelos de tarea depredador-presa.

Modelo	R	R^2	R^2 ajustado	Error típico
Figura 4.19a	0.833378	0.694519	0.592692	21.103805
Figura 4.19b	0.86723	0.75208	0.66944	19.61621
Figura 4.20a	0.94561	0.89418	0.85891	11.36563
Figura 4.20b	0.93109	0.86663	0.822567	12.51166

Los resultados de la Tabla 4.3 indican el coeficiente de relación múltiple R, es decir, la correlación entre el conjunto de variables predictoras (r_r y r_a) y la criterio (ε o t_c). El coeficiente de determinación R^2 que determina la calidad del modelo para replicar los resultados y su proporción de variación. El coeficiente de correlación múltiple al cuadrado R^2 ajustado que es una forma de minimizar la suma de todas las correlaciones que mantienen las variables predictoras con el criterio. Por último, el error típico de estimación que se refiere a la desviación típica de las puntuaciones de error.

Por otro lado, la Tabla 4.4 nos muestra información sobe la ecuación de regresión y sus coeficientes. Las primeras columnas contienen los coeficientes de

regresión parcial de las variables predictoras del modelo de regresión y su error típico. La última columna muestra el valor de probabilidad (p-valor), donde valores por debajo de 0.05 indican que la variable contribuye significativamente a mejorar la calidad del modelo de regresión.

Tabla 4.4: Relación de coeficientes para modelos de tarea depredador-presa.

Modelo		**Coeficiente**	**Error típico**	**P-valor**
Figura 4.19a	b_0	-13.278346	22.658625	0.579225
	b_1	560.5543	172.311843	0.0173977
	b_2	37.6669	21.53898	0.130911
Figura 4.19b	b_0	-36.287419	21.061431	0.135673
	b_1	559.063436	160.165679	0.012975
	b_2	49.113625	20.02071	0.049578
Figura 4.20a	b_0	-41.41797	12.20299	0.014602
	b_1	351.724136	92.799986	0.009072
	b_2	69.923563	11.56	0.000941
Figura 4.20b	b_0	-33.31977	13.433456	0.0477822
	b_1	281.04412	102.157282	0.033243
	b_2	71.691176	12.769660	0.001363

De modo que el modelo aproximado de las superficies en las Figuras 4.19a, 4.19b, 4.20a y 4.20b, corresponden a las Ecuaciones (4.2), (4.3), (4.4) y (4.5), con una certeza del 59.2 %, 66.9 %, 85.91 % y 82.2 %, respectivamente.

$$\varepsilon_s = -13.278346 + 560.5543 r_r + 37.6669 r_a \tag{4.2}$$

$$\varepsilon_i = -36.287419 + 559.063436 r_r + 49.113625 r_a \tag{4.3}$$

$$t_{cs} = -41.417967 + 351.724136 r_r + 69.923563 r_a \tag{4.4}$$

$$t_{ci} = -33.31977 + 281.04412r_r + 71.691176r_a \tag{4.5}$$

Por último, en la Tabla 4.5 se muestra el porcentaje de correlación de las variables r_r y r_a con la variable dependiente.

Tabla 4.5: Correlación de parámetros para modelos de tarea depredador-presa.

Modelo	**Repulsión**		**Atracción**	
	Correlación	**Porcentaje**	**Correlación**	**Porcentaje**
Ec. 4.2	0.734039	53.88 %	0.394594	18.57 %
Ec. 4.3	0.709526	50.34 %	0.498654	24.86 %
Ec. 4.4	0.503338	25.33 %	0.800519	64.08 %
Ec. 4.5	0.409710	16.78 %	0.836099	69.9 %

Las Ecuaciones (4.2) - (4.5) son sólo una aproximación para los modelos de comportamiento de las superficies generadas en las Figuras 4.19 y 4.20 para condiciones especificas de influencia, número de población, capacidad de percepción y área de prueba; además, están acotadas al rango de los valores en las configuraciones paramétricas utilizadas. Sin embargo, podemos ver una relación importante entre los parámetros con el comportamiento del enjambre y la significancia que tiene cada uno con el desempeño de la tarea.

4.1.4 Discusión

En los resultados se muestra el rendimiento de los robot-depredadores cuando intentan capturar el robot-presa para cada configuración paramétrica. El comportamiento cambia ante las variaciones en los parámetros r_r y r_a; con estos cambios, no solo mejora el rendimiento del enjambre, sino incluso la eficiencia

individual del robot. Según los resultados, surgen diferentes comportamientos o escenarios típicos de la *RE* [40].

El comportamiento observado en las simulaciones computacionales fue consistente con la implementación en una bandada de robots físicos, como se esperaba. La Tabla 4.3 muestra que los resultados por simulación no se encuentran alejados de la implementación física, obteniendo coeficientes de determinación con errores típicos muy similares para los comportamientos de las Figuras 4.19 y 4.20. Por otro lado, vemos que la relación entre los parámetros es significativa en función del caso (Tabla 4.5). Con respecto al área de cobertura, el parámetro de repulsión tiene un porcentaje de correlación mayor, ya que al aumentar, ocasiona que los agentes se eviten unos a otros para prevenir colisiones provocando un aumento en el área de dispersión. Mientras que, con respecto al tiempo de captura, el parámetro de atracción tiene un porcentaje de correlación mayor, permitiendo al enjambre moverse de forma más coordinada en el espacio por la formación de cadenas de robots (ver Figura 4.7).

Con valores bajos de repulsión, surge un comportamiento de agregación y pastoreo [102, 103, 79], donde los robot-depredadores se mantienen unidos y son más fieles a la trayectoria del robot-presa (ver Figuras 4.1 y 4.10). Los robot-depredadores intentan acercarse uno al otro minimizando las distancias entre ellos y en colaboración con otros tienden a seguir al robot-presa. Aunque el comportamiento de pastoreo se centra en robots heterogéneos o especializados [104], el robot-presa toma el papel de "pastor" por instantes estimulando a los robot-depredadores con un factor de influencia que los guía hasta que es capturado. Al aumentar los valores de repulsión, se genera un comportamiento de dispersión [105, 106], las distancias entre los robot-depredadores se maximizan y el área de cobertura aumenta, lo que provoca que dejen de detectar al robot-presa y tomen más tiempo para capturarlo (ver Figuras 4.3 y 4.15). El comportamiento causado por esta configuración de parámetros es útil para evitar colisiones y explorar grandes áreas; sin embargo, los robots deben permanecer dentro del área de

detección, de lo contrario, se perderán. Es decir, al tener un área de cobertura pequeña (ver Figuras 4.1 y 4.10) la interacción entre los robot-depredadores es más cercana, y la mayoría de ellos se encuentran alrededor del robot-presa; mientras que, al tener un área de cobertura grande (ver Figuras 4.3 y 4.12), hay menos interacción entre los robot-depredadores, por lo que algunos de ellos pueden perder el rastro del robot-presa.

Por otro lado, los valores bajos de atracción causa independencia entre los miembros del enjambre; mientras que, valores altos de atracción provoca que sea difícil para los robot-depredadores, decidir entre seguir al robot-presa o reagruparse con el enjambre (ver Figuras 4.9 y 4.17). Los robot-depredadores exploran su entorno y verifican si hay grupos más grandes a los que deben unirse, esto ayuda a mantener un comportamiento de agregación formando subgrupos, pero se pierde precisión en el seguimiento de la trayectoria del robot-presa, aumentando el área de cobertura. En algunos casos, se forman cadenas de robots (ver Figura 4.7), similar a un estudio reportado por S. Nouyan et al. [107].

En general, los valores bajos de repulsión y atracción hacen que los robots depredadores sigan al robot de presa como un guía, con esta configuración es más probable que se capture en las esquinas o bordes del área de prueba. Los valores altos de repulsión y atracción hacen que los robot-depredadores se dispersen y regresen a su objetivo desde diferentes ángulos, lo que permite capturar al robot-presa en el centro del área de prueba; sin embargo, con esta configuración el área de cobertura y tiempo de captura aumenta. Los robot-depredadores al evitar colisiones con sus vecinos crearán nuevos caminos que volverán a converger al robot-presa atrapándolo desde otra perspectiva.

4.2 Segundo caso experimental (Tarea de transporte de objetos)

4.2.1 Simulaciones

Se realizaron simulaciones gráficas con 20 objetos y para 5, 10 y 20 robots, los cuales cuentan con las capacidades sensoriales del robot *BugBot*, en un área de prueba de 10 x 10 m^2, y por cada simulación se hicieron tres repeticiones. Todos los robot tienen la misma configuración parámetrica por cada experimento, el voltaje aplicado en los motores corresponde a la Ecuación (2.31) y el umbral de influencia se estableció en el 10 % de la capacidad total de las LDRs. La posición inicial de los robots (flechas negras) es aleatoria dentro del "nido", el cual se marca con un cuadro rojo de dimensiones de 2 x 2 m^2; mientras que, los objetos (círculos verdes) son colocados de forma aleatoria dentro de la zona de búsqueda, marcada con un cuadro azul de dimensiones de 2 x 2 m^2. El origen de la zona de búsqueda es el punto $(7.5, 7.5)$ con radio de 1.5 m; mientras que, el origen de la zona de entrega (nido), es el punto $(1, 1)$ con radio de 4 m. Ambas zonas (búsqueda y entrega) representan el factor de influencia que estimula a los robots a encontrar o descargar los objetos en la zona correspondiente en función del estado de las tenazas.

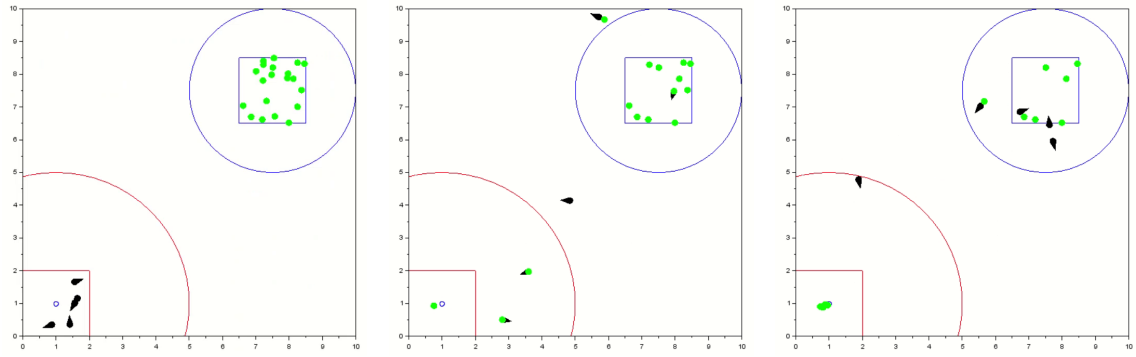
Las Figuras 4.21 - 4.32 ilustran algunas instantáneas de simulaciones con la evolución del desarrollo de la tarea, donde κ corresponde el número de iteración equivalente a 1 segundo. Las Tablas 4.6 - 4.8 muestran el resultado promedio de todas las pruebas para 5, 10 y 20 miembros del enjambre e incluye la distancia recorrida (λ), tiempo de entrega (t_d) y búsqueda por robot (t_s), y el tiempo de ejecución de la tarea por el enjambre (t_e).

En las simulaciones con 5 robots (ver Figuras 4.21 - 4.24), el efecto de los parámetros es débil, ya que la interacción entre los miembros del enjambre es muy

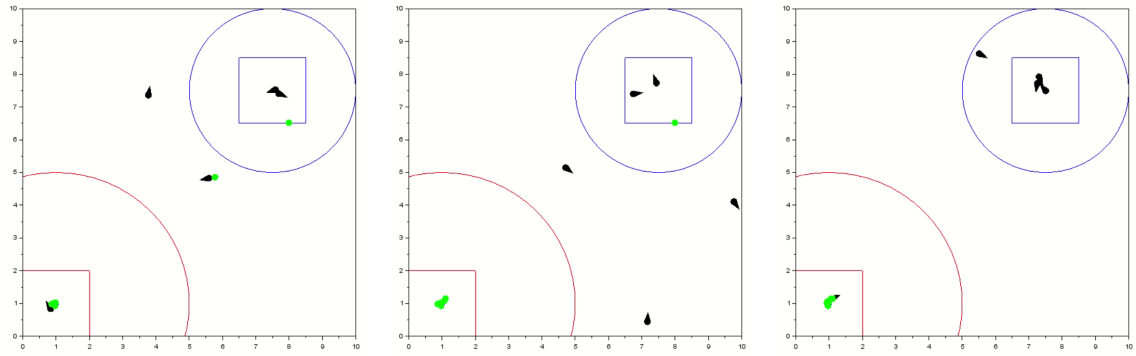
poca debido al gran espacio de trabajo. De modo que, la repulsión ayuda a que los miembros del enjambre se dispersen y cubran mayor territorio para localizar los objetos, mientras que la atracción no parece tener un efecto notable. En las simulaciones con 10 robots (ver Figuras 4.25 - 4.28), el efecto de los parámetros es el mismo que con 5 robots, sin embargo, el parámetro de atracción se hace más fuerte, provocando la formación de cadenas de robots. Finalmente, en las simulaciones con 20 robots (ver Figuras 4.29 - 4.32), surge un comportamiento distinto al observado con 5 y 10 robots, los mejores resultados son obtenidos con valores intermedios de repulsión, ya que cuando se tienen valores bajos, los robots tardan más en localizar los objetos, mientras que, con valor altos, los robots se estorban entre sí impidiendo tomar los objetos. Por otro lado, el efecto de la atracción se intensifica ayudando a que los robots se trasladen entre las áreas de entrega y búsqueda con la formación de las cadenas de robots. El detalle de la discusión se comenta en la Sección 4.2.4.

4.2.2 Experimentos por implementación física

Con el fin de validar experimentalmente las pruebas realizadas por simulación, se implementaron las reglas de comportamiento considerando los sensores locales de los robots tipo *BugBot*. Los experimentos se realizaron con tres *BugBot*, en un área de prueba de 1.8 x 1.1 m^2, donde se ejecutaron casos específicos de configuraciones paramétricas, y para cada una se hicieron cuatro réplicas. Como objetos, se consideran cubos pequeños de diámetro de 0.03 m. En cada experimento se colocan 20 objetos en la zona de búsqueda y la tarea termina cuando se entregan 10 objetos en el nido. Cada zona es influenciada por fuentes de luz colocadas en esquinas del área de prueba.

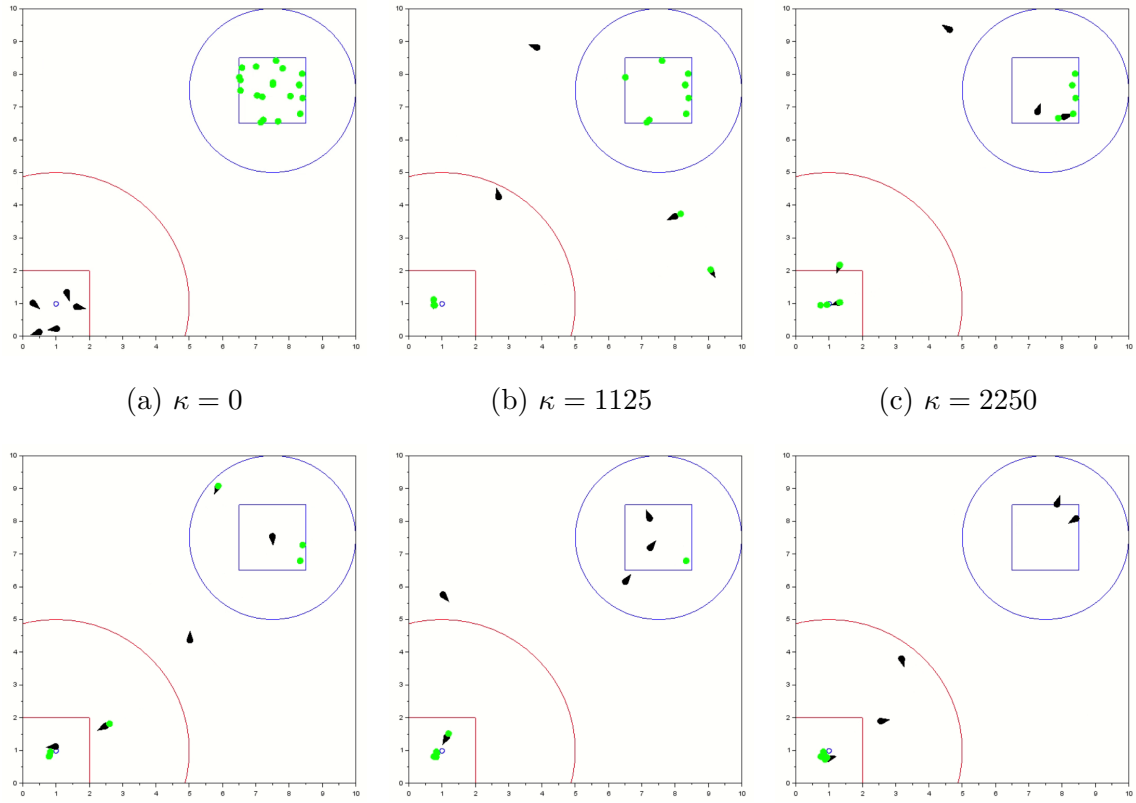


(a) $\kappa = 0$ (b) $\kappa = 840$ (c) $\kappa = 1680$



(d) $\kappa = 2520$ (e) $\kappa = 3360$ (f) $\kappa = 4197$

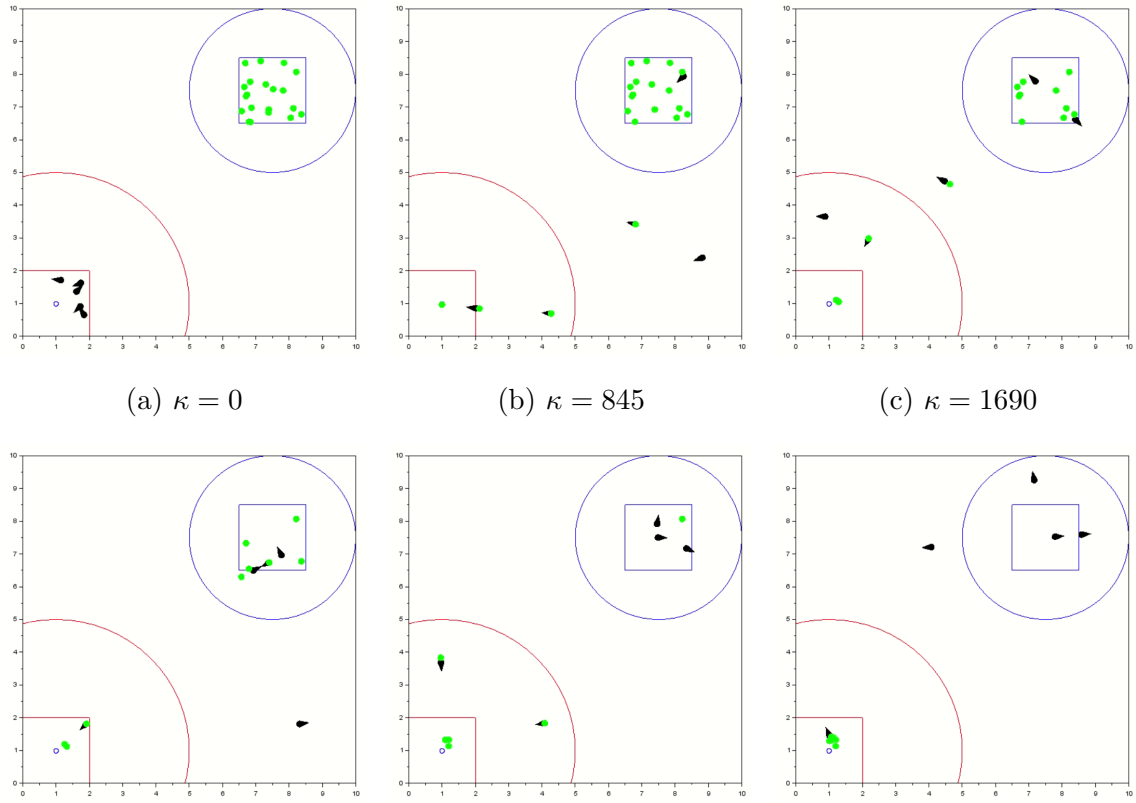
Figura 4.21: Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$.



(a) $\kappa = 0$ (b) $\kappa = 1125$ (c) $\kappa = 2250$

(d) $\kappa = 3375$ (e) $\kappa = 4500$ (f) $\kappa = 5634$

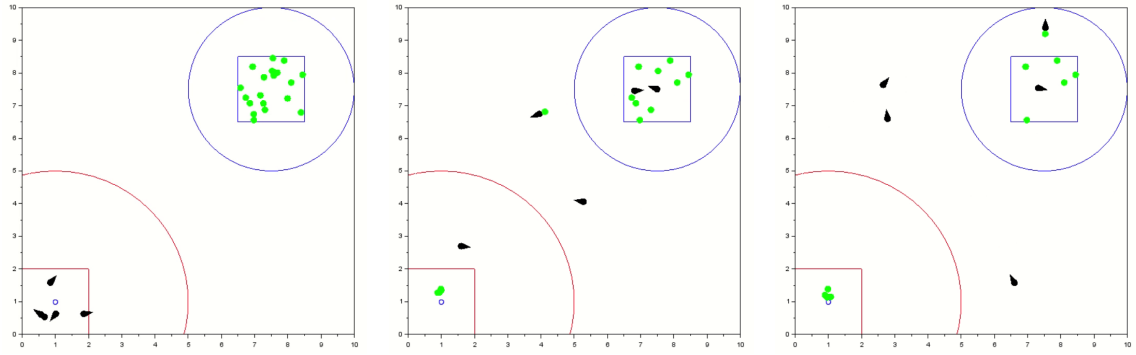
Figura 4.22: Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$.



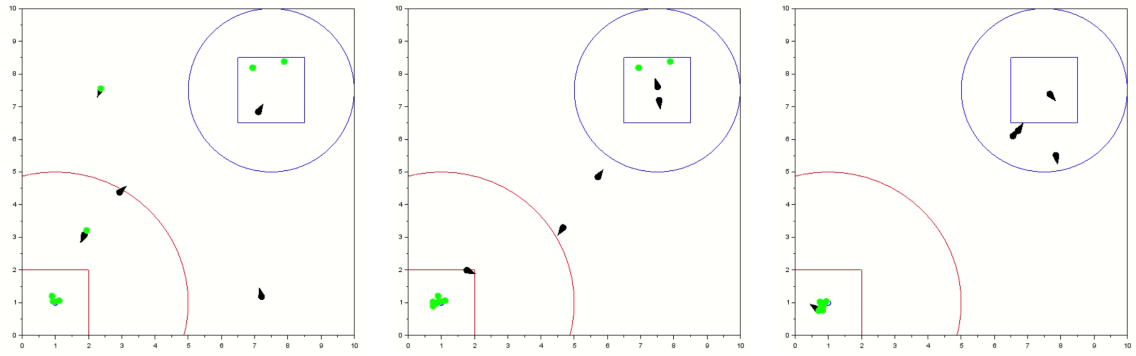
(a) $\kappa = 0$ (b) $\kappa = 845$ (c) $\kappa = 1690$

(d) $\kappa = 2535$ (e) $\kappa = 3380$ (f) $\kappa = 4226$

Figura 4.23: Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$.

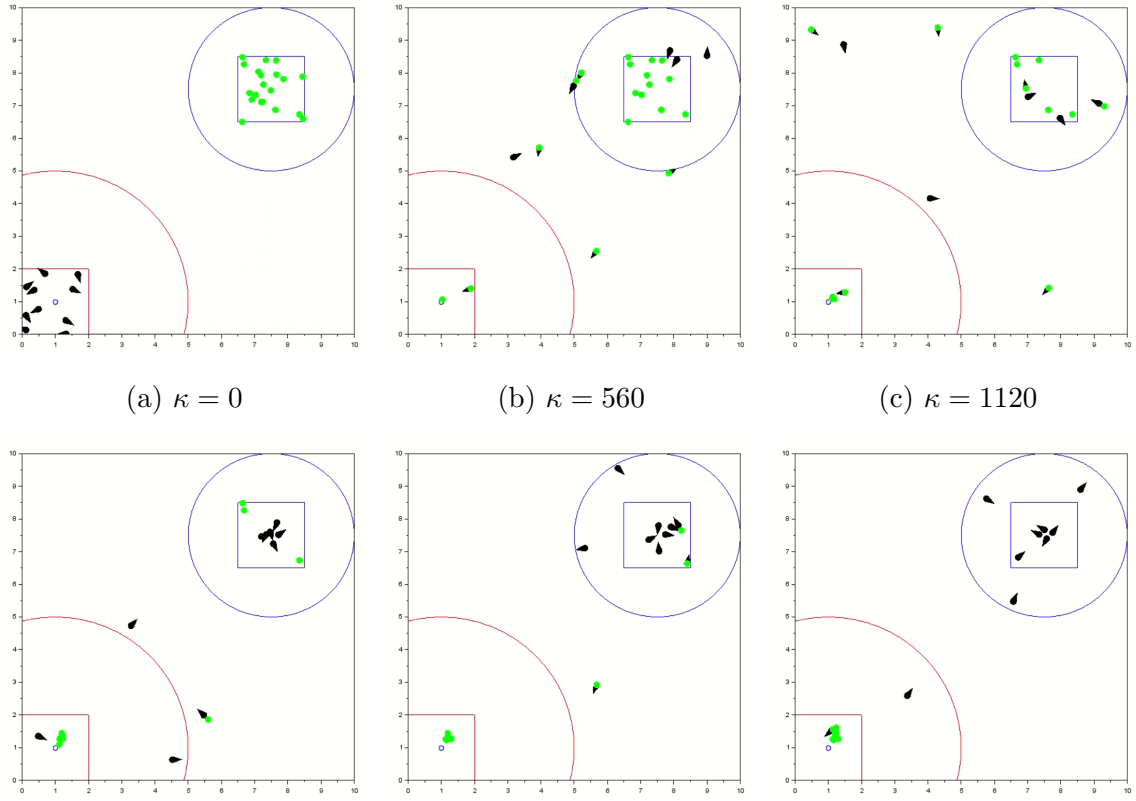


(a) $\kappa = 0$ (b) $\kappa = 875$ (c) $\kappa = 1750$



(d) $\kappa = 2625$ (e) $\kappa = 3500$ (f) $\kappa = 4387$

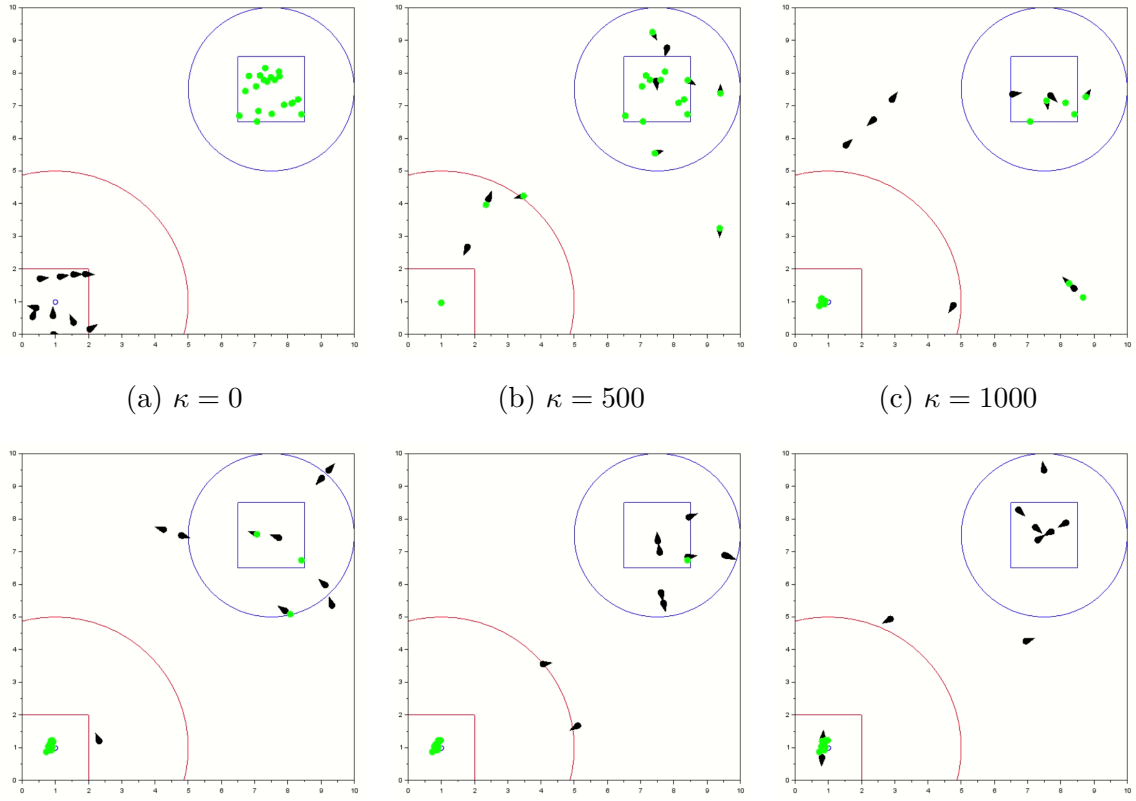
Figura 4.24: Simulación de enjambre de 5 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$.



(a) $\kappa = 0$ (b) $\kappa = 560$ (c) $\kappa = 1120$

(d) $\kappa = 1680$ (e) $\kappa = 2240$ (f) $\kappa = 2794$

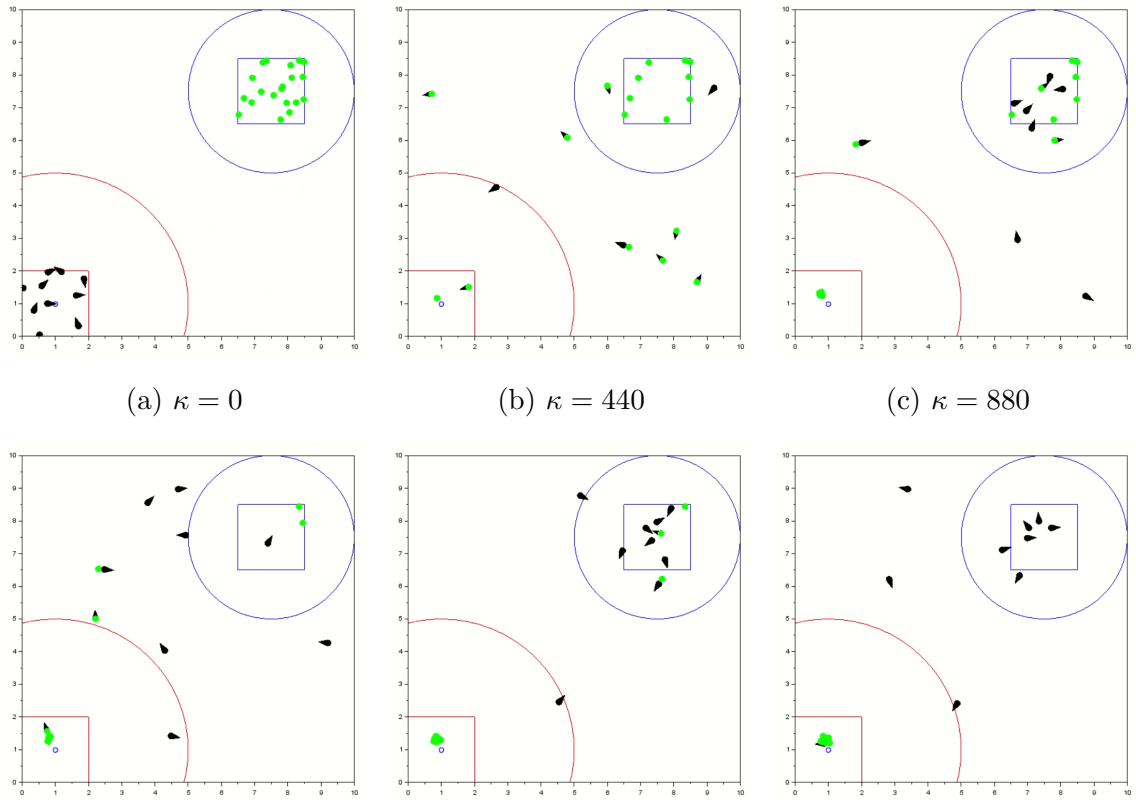
Figura 4.25: Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$.



(a) $\kappa = 0$ (b) $\kappa = 500$ (c) $\kappa = 1000$

(d) $\kappa = 1500$ (e) $\kappa = 2000$ (f) $\kappa = 2569$

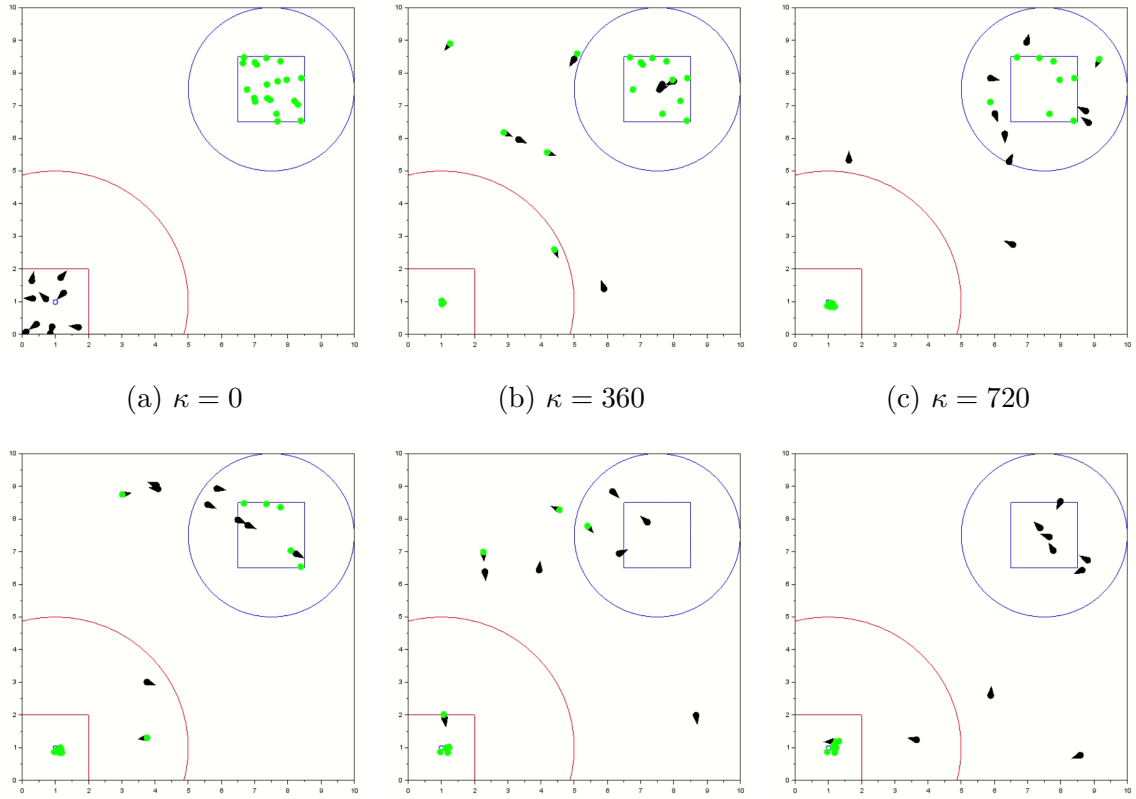
Figura 4.26: Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$.



(a) $\kappa = 0$ (b) $\kappa = 440$ (c) $\kappa = 880$

(d) $\kappa = 1320$ (e) $\kappa = 1760$ (f) $\kappa = 2171$

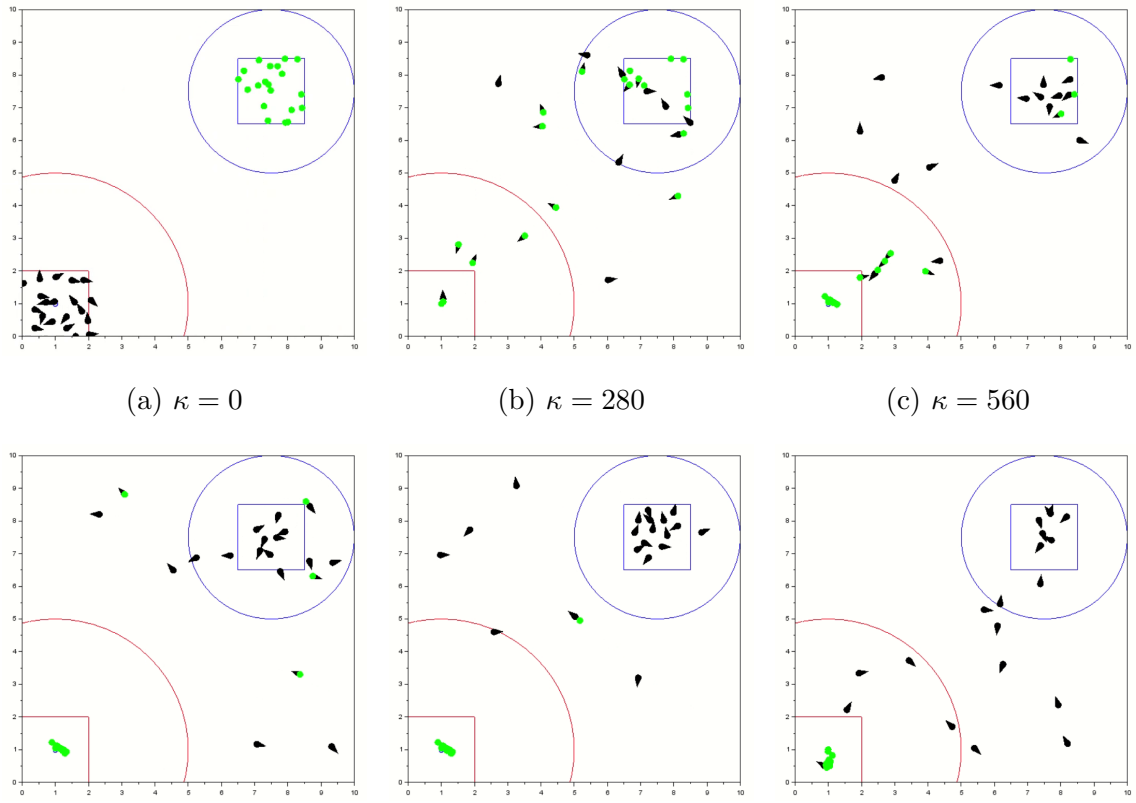
Figura 4.27: Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$.



(a) $\kappa = 0$ (b) $\kappa = 360$ (c) $\kappa = 720$

(d) $\kappa = 1080$ (e) $\kappa = 1440$ (f) $\kappa = 1804$

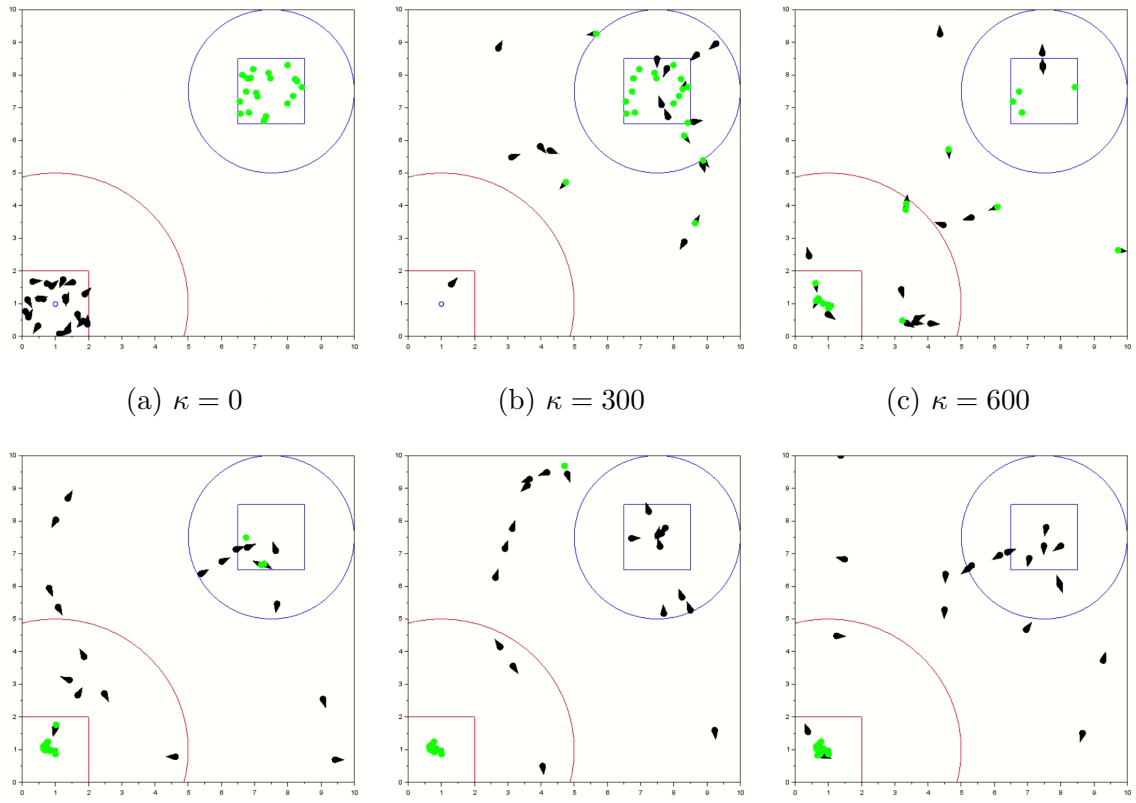
Figura 4.28: Simulación de enjambre de 10 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$.



(a) $\kappa = 0$ (b) $\kappa = 280$ (c) $\kappa = 560$

(d) $\kappa = 840$ (e) $\kappa = 1120$ (f) $\kappa = 1435$

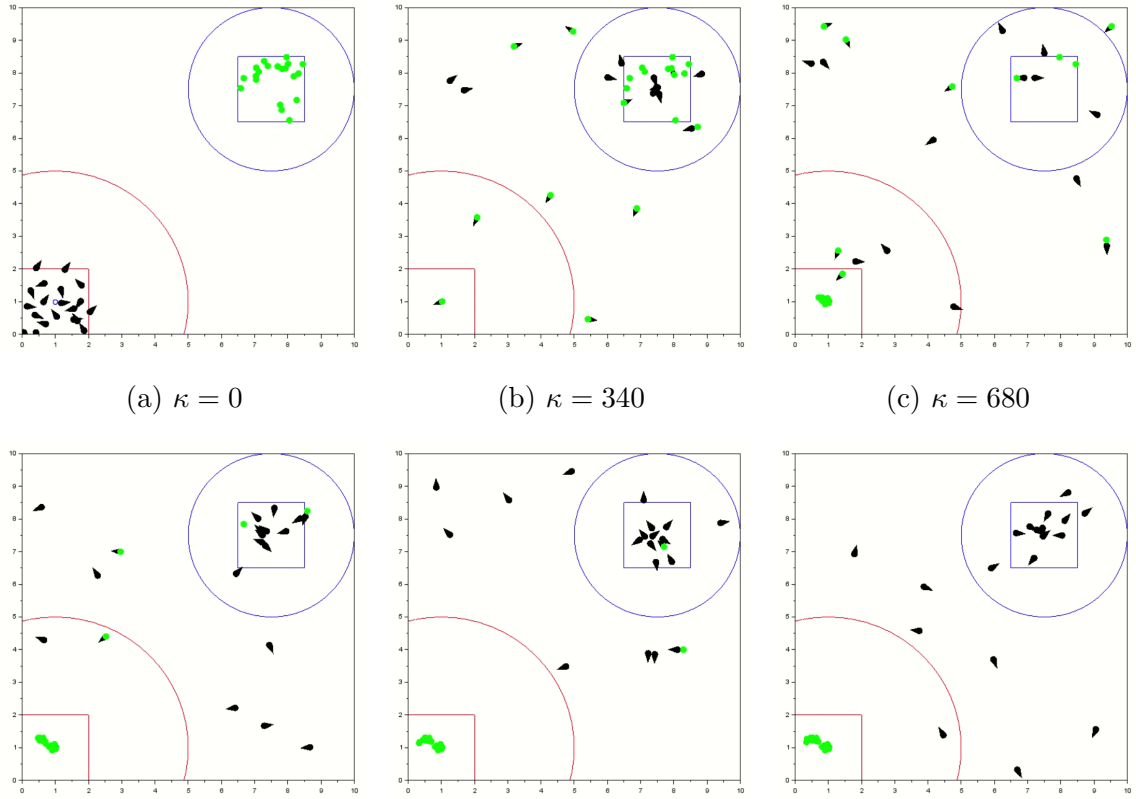
Figura 4.29: Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 0.2$.



(a) $\kappa = 0$ (b) $\kappa = 300$ (c) $\kappa = 600$

(d) $\kappa = 900$ (e) $\kappa = 1200$ (f) $\kappa = 1524$

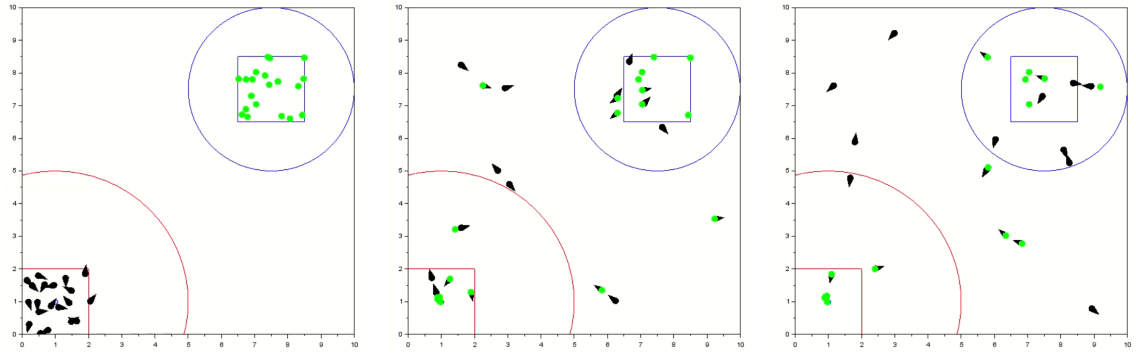
Figura 4.30: Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.01$ y $r_a = 1$.



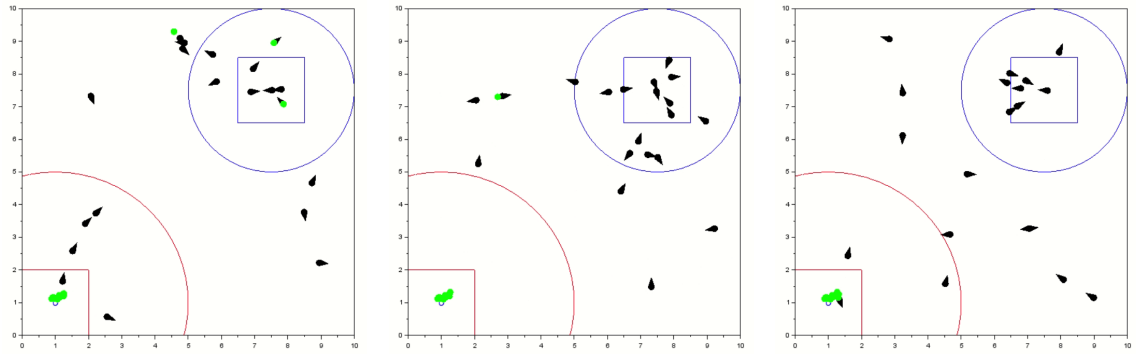
(a) $\kappa = 0$ (b) $\kappa = 340$ (c) $\kappa = 680$

(d) $\kappa = 1020$ (e) $\kappa = 1360$ (f) $\kappa = 1688$

Figura 4.31: Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$.



(a) $\kappa = 0$ (b) $\kappa = 340$ (c) $\kappa = 380$



(d) $\kappa = 1020$ (e) $\kappa = 1360$ (f) $\kappa = 1710$

Figura 4.32: Simulación de enjambre de 20 robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$.

Tabla 4.6: Resultados de tarea de transporte de objetos con 5 *BugBot*, basados en simulación.

Test	r_r (m)	$r_a(m)$	No. de prueba	t_d (s)	t_s (s)	λ (m)	t_e (s)
1	0.01	0.2	1	1223.2	5988.8	1397.775	7212
			2	859	5017	1067.0172	5876
			3	804.8	3392.2	819.16538	4197
			Media	**962.33**	**4799.33**	**1094.65**	**5761.67**
2	0.05	0.2	1	740	5538	1213.2369	6278
			2	1251	3374	904.28499	4625
			3	1331.6	2705.4	790.4595	4037
			Media	**1107.53333**	**3872.46667**	**969.32713**	**4980**
3	0.1	0.2	1	1408.4	2817.6	827.6661	4226
			2	837.6	3628.4	872.90912	4466
			3	1055.6	2519.4	705.56913	3575
			Media	**1100.53333**	**2988.46667**	**802.048117**	**4089**
4	0.01	0.6	1	1083.8	3950.2	945.19066	5034
			2	1124	4673	1127.3257	5797
			3	859.8	6289.2	1331.3649	7149
			Media	**1022.53333**	**4970.8**	**1134.62709**	**5993.33**
5	0.05	0.6	1	943.6	3628.4	909.62507	4572
			2	1417.4	2947.6	873.72054	4365
			3	1007.6	3446.4	877.00453	4454
			Media	**1122.86667**	**3340.8**	**886.78338**	**4463.67**
6	0.1	0.6	1	848.2	2393.8	644.89407	3242
			2	1121.2	2989.8	809.0406	4111
			3	823.8	3355.2	815.53713	4179
			Media	**931.066667**	**2912.93333**	**756.4906**	**3844**
7	0.01	1	1	1098.8	2768.2	719.34674	3867
			2	950.4	4899.6	1132.9665	5850
			3	1013.4	4620.6	948.40398	5634
			Media	**1020.86667**	**4096.13333**	**933.572407**	**5117**
8	0.05	1	1	1447	3784	1042.3873	5231
			2	1054.4	2636.6	745.84321	3691
			3	1240	3131	873.87974	4371
			Media	**1247.13333**	**3183.86667**	**887.370083**	**4431**
9	0.1	1	1	936.4	2190.6	627.46429	3127
			2	1029.8	3664.2	938.45867	4694
			3	1424	2963	882.01124	4387
			Media	**1130.06667**	**2939.26667**	**815.978067**	**4069.33**

Tabla 4.7: Resultados de tarea de transporte de objetos con 10 *BugBot*, basados en simulación.

Test	r_r (m)	$r_a(m)$	No. de prueba	t_d (s)	t_s (s)	λ (m)	t_e (s)
1	0.01	0.2	1	611.3	2182.7	581.99541	2794
			2	535.4	1621.6	441.31862	2157
			3	724.8	2360.2	649.51133	3085
			Media	**623.833333**	**2054.83333**	**557.608453**	**2678.67**
2	0.05	0.2	1	554.9	1431.1	408.52134	1986
			2	503.7	2228.3	561.86088	2732
			3	613	1562	442.00975	2175
			Media	**557.2**	**1740.46667**	**470.797323**	**2297.67**
3	0.1	0.2	1	547	1624	433.94157	2171
			2	631.5	1122.5	350.05068	1754
			3	619.2	1649.8	452.10877	2269
			Media	**599.233333**	**1465.43333**	**412.033673**	**2064.67**
4	0.01	0.6	1	506.5	1922.5	485.35309	2429
			2	687	2091	549.90817	2778
			3	723.9	1670.1	468.89406	2394
			Media	**639.133333**	**1894.53333**	**501.385107**	**2533.67**
5	0.05	0.6	1	608.8	1666.2	452.07028	2275
			2	426.5	1478.5	381.78889	1905
			3	611.9	1329.1	382.99894	1941
			Media	**549.066667**	**1491.26667**	**405.61937**	**2040.33**
6	0.1	0.6	1	484.3	1314.7	372.85726	1799
			2	571.4	1683.6	459.42923	2255
			3	478.4	1239.6	358.48943	1718
			Media	**511.366667**	**1412.63333**	**396.925307**	**1924**
7	0.01	1	1	668.8	2037.2	531.30784	2706
			2	606.8	1962.2	501.03696	2569
			3	487.3	1684.7	422.02339	2172
			Media	**587.633333**	**1894.7**	**484.789397**	**2482.33**
8	0.05	1	1	401.5	1101.5	296.24567	1503
			2	424	1632	400.61104	2056
			3	564.1	1664.9	431.31441	2229
			Media	**463.2**	**1466.13333**	**376.05704**	**1929.33**
9	0.1	1	1	487.5	1316.5	349.68935	1804
			2	690.2	1415.8	414.9434	2106
			3	390.3	1151.7	306.12019	1542
			Media	**522.666667**	**1294.66667**	**356.917647**	**1817.33**

Tabla 4.8: Resultados de tarea de transporte de objetos con 20 *BugBot*, basados en simulación.

Test	r_r (m)	$r_a(m)$	No. de prueba	t_d (s)	t_s (s)	λ (m)	t_e (s)
1	0.01	0.2	1	328.6	1106.4	295.72616	1435
			2	343.2	1632.8	399.28536	1976
			3	274.7	1085.3	281.09291	1360
			Media	**315.5**	**1274.83333**	**325.368143**	**1590.33**
2	0.05	0.2	1	260.25	1226.75	296.93608	1487
			2	184.3	1335.7	295.22959	1520
			3	210.55	1185.45	273.59756	1396
			Media	**218.366667**	**1249.3**	**288.587743**	**1467.67**
3	0.1	0.2	1	285.55	1675.45	424.2902	1961
			2	295.65	1392.35	369.31882	1688
			3	291.4	1321.6	341.67229	1613
			Media	**290.866667**	**1463.13333**	**378.427103**	**1754**
4	0.01	0.6	1	258.7	1476.3	356.91841	1735
			2	412.25	1391.75	362.18216	1804
			3	200.45	1010.55	244.97289	1211
			Media	**290.466667**	**1292.86667**	**321.35782**	**1583.33**
5	0.05	0.6	1	339.05	1047.95	298.54704	1387
			2	366.2	1100.8	317.52619	1467
			3	247.85	1095.15	296.43198	1343
			Media	**317.7**	**1081.3**	**304.168403**	**1399**
6	0.1	0.6	1	225.35	1163.65	270.79002	1389
			2	298	1689	384.18459	1987
			3	240.9	1349.1	307.92706	1590
			Media	**254.75**	**1400.58333**	**320.967223**	**1655.33**
7	0.01	1	1	289.15	1303.85	327.74213	1593
			2	311.2	941.8	256.30709	1253
			3	314.7	1209.3	311.42216	1524
			Media	**305.016667**	**1151.65**	**298.49046**	**1456.67**
8	0.05	1	1	217.55	795.45	198.91763	1013
			2	258.7	758.3	200.34202	1017
			3	192.45	959.55	224.12042	1152
			Media	**222.9**	**837.766667**	**207.793357**	**1060.67**
9	0.1	1	1	366.35	1343.65	364.40739	1710
			2	317	1457	383.85453	1774
			3	271.55	1255.45	328.18434	1527
			Media	**318.3**	**1352.03333**	**358.81542**	**1670.33**

Las Figuras 4.33 - 4.36, ilustran algunas instantáneas de los experimentos físicos y muestran la evolución de la tarea de transporte de objetos en tiempos específicos. Considerando todos los experimentos, cambios de parámetros y réplicas por experimento, se generó la Tabla 4.9, que incluye el tiempo de ejecución (t_e) cuando la tarea es completada.

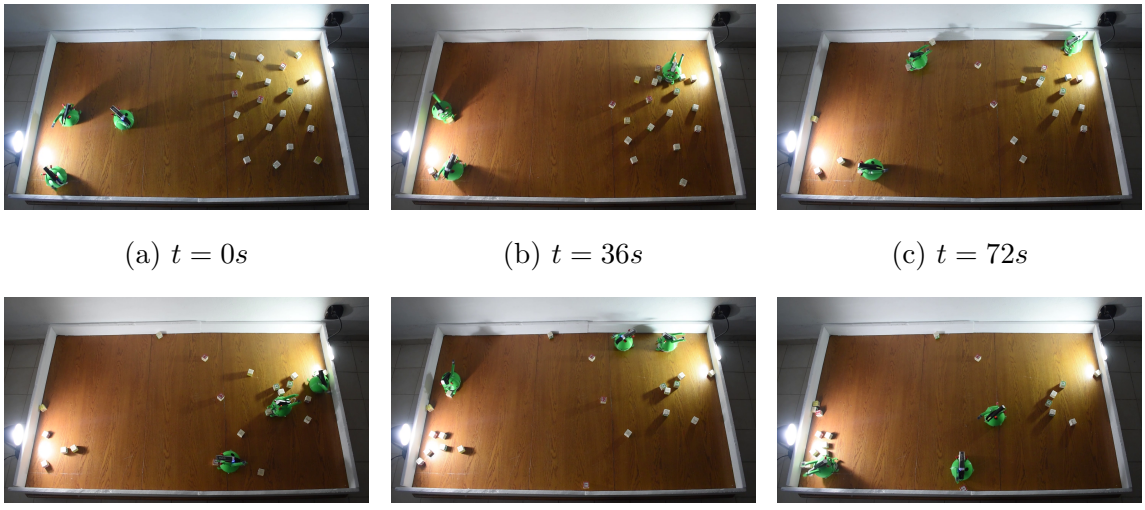
4.2.3 Análisis de datos y modelado de la emergencia de colaboración en el enjambre, a partir de los resultados obtenidos en la tarea de transporte de objetos

Las Figuras 4.37 y 4.38, ilustran el efecto de repulsión y atracción en el enjambre, con respecto al tiempo de ejecución (t_e). Los datos fueron extraídos de las Tablas 4.6, 4.7, 4.8 y 4.9, con los valores promedio normalizados en un rango de 0 a 100.

Para conocer una aproximación al modelo de comportamiento, y conocer el impacto que tienen los parámetros r_r y r_a, se realiza una regresión lineal múltiple (Ec. 4.1). Los resultados de la Tabla 4.10 y 4.12, nos muestran información sobre la ecuación de regresión, su correlación y coeficientes.

Tabla 4.10: Estadísticas de la regresión para tarea de transporte de objetos.

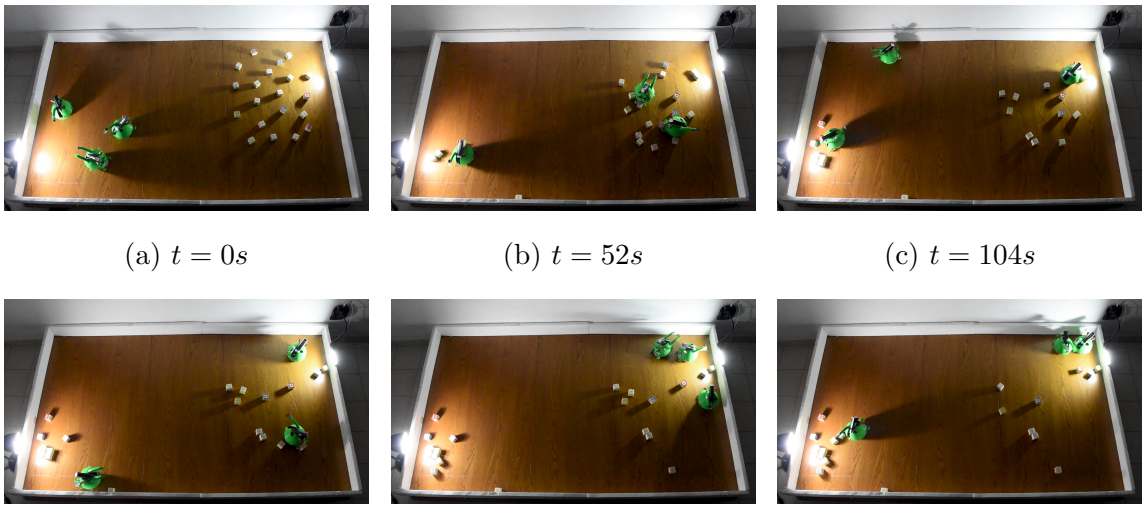
Modelo	R	R^2	R^2 ajustado	Error típico
Figura 4.37a	0.939458	0.882581	0.843441	14.052822
Figura 4.37b	0.940012	0.883622	0.844829	14.181679
Figura 4.37c	0.570217	0.325147	0.100196	28.0721651
Figura 4.38	0.985491	0.971193	0.913582	13.536585



(a) $t = 0s$ (b) $t = 36s$ (c) $t = 72s$

(d) $t = 108s$ (e) $t = 144s$ (f) $t = 182s$

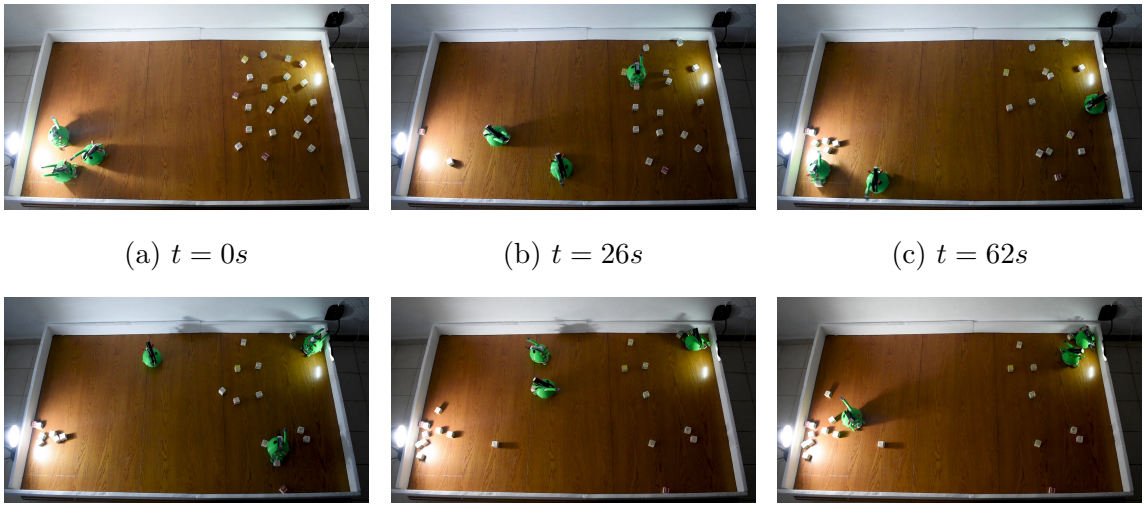
Figura 4.33: Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.05$ y $r_a = 0.2$.



(a) $t = 0s$ (b) $t = 52s$ (c) $t = 104s$

(d) $t = 156s$ (e) $t = 208s$ (f) $t = 259s$

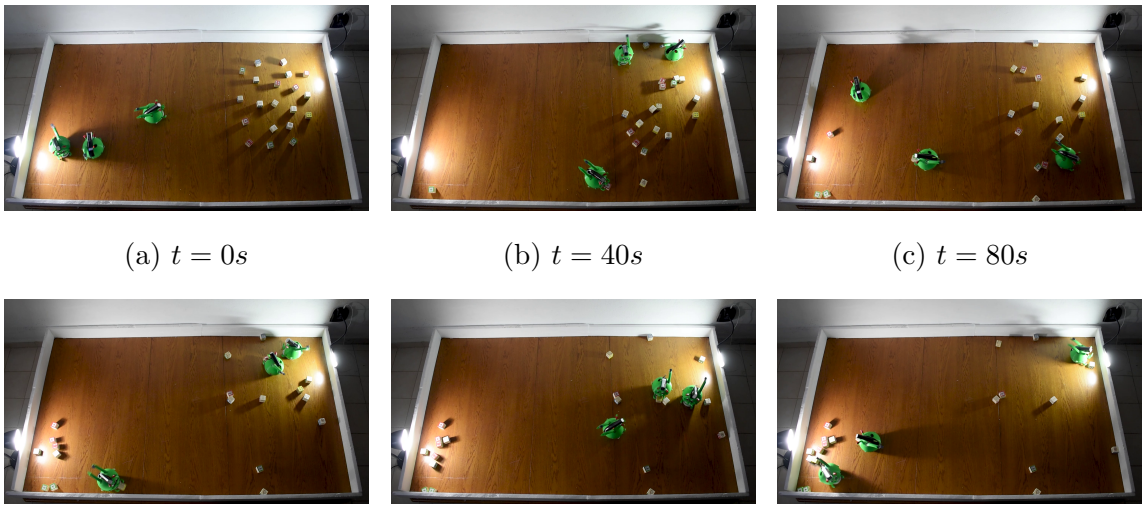
Figura 4.34: Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 0.2$.



(a) $t = 0s$ (b) $t = 26s$ (c) $t = 62s$

(d) $t = 88s$ (e) $t = 114s$ (f) $t = 136s$

Figura 4.35: Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.05$ y $r_a = 1$.



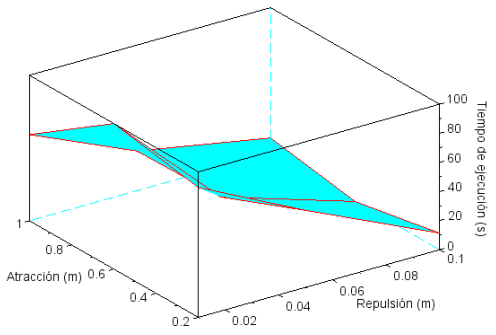
(a) $t = 0s$ (b) $t = 40s$ (c) $t = 80s$

(d) $t = 120s$ (e) $t = 160s$ (f) $t = 206s$

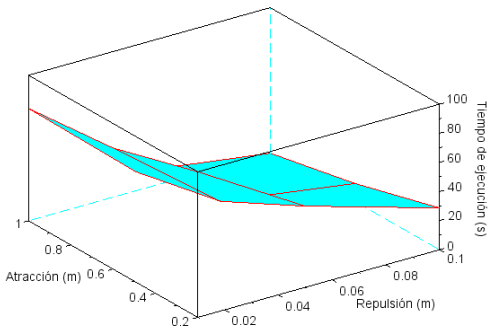
Figura 4.36: Manada de robots ejecutando tarea de transporte de objetos con $r_r = 0.1$ y $r_a = 1$.

Tabla 4.9: Resultados de tarea de transporte de objetos con 3 *BugBot*, basados en implementación física.

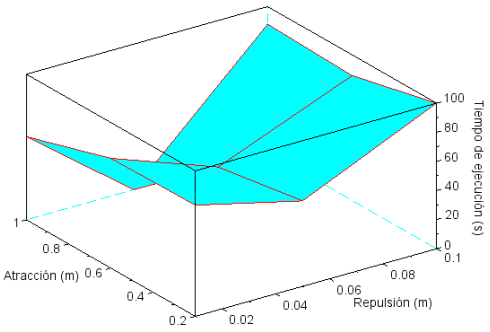
Test	r_r (m)	r_a (m)	No. de prueba	t_e (s)
1	0.05	0.2	1	181.2
			2	190.2
			3	183
			4	139.2
			Media	**173.4**
2	0.1	0.2	1	332.4
			2	269.4
			3	259.8
			4	264
			Media	**281.4**
3	0.05	1	1	184.8
			2	130.8
			3	181.8
			4	136.2
			Media	**158.4**
4	0.1	1	1	268.8
			2	258
			3	205.2
			4	200.4
			Media	**233.1**



(a) 5 robots



(b) 10 robots



(c) 20 robots

Figura 4.37: Efecto de repulsión y atracción en el tiempo de ejecución para tarea de transporte de objetos, basado en simulaciones.

Tabla 4.11: Estadísticas de regresión para la superficie de comportamiento de la Figura 4.37c.

Rango	R	R^2	R^2 ajustado	Error típico
$0 \leq r_r \leq 0.6$	0.907788	0.824079	0.706798	15.169274
$0.6 < r_r \leq 1$	0.935131	0.874469	0.790782	16.739363

El modelo para la Figura 4.37c presenta un valor bajo en el coeficiente de determinación debido a la no linealidad de la superficie. Por lo que se propone dividir

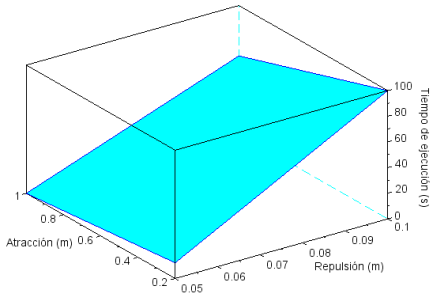


Figura 4.38: Efecto de repulsión y atracción en el tiempo de ejecución para tarea de transporte de objetos, basado en implementación física.

el modelo de comportamiento en dos partes con respecto al valor del parámetro de repulsión. Primero, se obtiene un modelo en el rango de valores de repulsión de 0 a 0.6, y el segundo modelo en un rango de valores de repulsión de 0.6 a 1, como se muestra en la Tabla 4.11.

De modo que el modelo aproximado de las superficies en la Figuras 4.37a, 4.37b, 4.37c y 4.38, corresponden a las Ecuaciones (4.6), (4.7), (4.8) y (4.9), con una certeza del 84.4 %, 84.4 %, 70.6 % para $0 \leq r_r \leq 0.6$ ó 79 % para $0.6 < r_r \leq 1$, y 91.3 %, respectivamente. Por último, en la Tabla 4.13, se muestra el porcentaje de correlación de las variables r_r y r_a con la variable dependiente.

Tabla 4.13: Correlación de parámetros de tarea de transporte de objetos.

Modelo	**Repulsión**		**Atracción**	
	Correlación	**Porcentaje**	**Correlación**	**Porcentaje**
Ec. 4.6	- 0.911014	82.99 %	- 0.229419	5.26 %
Ec. 4.7	- 0.860681	74.07 %	-0.377955	14.28 %
Ec. 4.8 ($0 \leq r_r \leq 0.6$)	- 0.660801	43.6 %	- 0.62243	38.7 %
Ec. 4.8 ($0.6 < r_r \leq 1$)	0.829154	68.74 %	- 0.432403	18.69 %
Ec. 4.9	0.931185	86.71 %	- 0.322627	10.41 %

Tabla 4.12: Relación de coeficientes para modelos de tarea de transporte de objetos.

Modelo		**Coeficiente**	**Error típico**	**P-valor**
Figura 4.37a	b_0	100.449414	11.918105	0.000152
	b_1	-828.545139	127.228277	0.000624
	b_2	-23.521505	14.342602	0.152121
Figura 4.37b	b_0	109.900988	12.027388	0.000096
	b_1	-793.470537	128.394897	0.000825
	b_2	-39.280185	14.47411	0.034928
Fig. 4.37c $(0 \leq r_r \leq 0.6)$	b_0	107.323718	15.939772	0.006689
	b_1	-844.951925	309.641522	0.072012
	b_2	-48.737981	18.961593	0.082467
Fig. 4.37c $(0.6 < r_r \leq 1)$	b_0	6.971154	24.992529	0.021603
	b_1	1108.01282	273.35266	0.027051
	b_2	-44.230769	20.924204	0.124908
Figura 4.38	b_0	-11.737804	18.224186	0.635726
	b_1	742.682925	135.365855	0.114774
	b_2	-32.164634	16.920732	0.308304

$$t_{e1} = 100.449414 - 828.545139 r_r - 23.521505 r_a \tag{4.6}$$

$$t_{e2} = 109.900988 - 793.470537 r_r - 39.2801852 r_a \tag{4.7}$$

$$t_{e3} = \begin{cases} 107.323718 - 844.951925 r_r - 48.737981 r_a, & \text{si} \quad (0 \leq r_r \leq 0.6) \\ 6.971154 + 1108.01282 r_r - 44.230769 r_a, & \text{si} \quad (0.6 < r_r \leq 1) \end{cases} \tag{4.8}$$

$$t_{e4} = -11.737804 + 742.682925 r_r - 32.164634 r_a \tag{4.9}$$

Las Ecuaciones (4.6) - (4.9), son sólo una aproximación para los modelos de comportamiento de las superficies generadas en las Figuras 4.37 y 4.38, para condiciones especificas y están acotadas al rango de los valores en las configuraciones paramétricas utilizadas.

4.2.4 Discusión

En esta tarea se establece un comportamiento general en el enjambre, que incluye un conjunto de reglas que permiten a los robots deambular en una arena desconocida, buscar objetos, tomarlos y navegar hacia donde deben colocarse. Uno de los principales desafíos, es transportar y descargar el objeto correctamente de modo emergente. Dependiendo de la configuración paramétrica, el comportamiento del enjambre cambia, provocando que los robots tomen autonomía unos con otros o se muevan juntos en grupos. En la evolución de la tarea, la interacción con los vecinos y el entorno provoca la emergencia de colaboración, ya que el movimiento en grupo facilita la navegación y localización entre las zonas (búsqueda y entrega). Esta tarea tiene similitud con la búsqueda de alimento en insectos (*foraging*), aunque es simple, puede considerarse como una abstracción con muchos puntos en común con aplicaciones más complejas, como la búsqueda y remoción de minas o rescate [108].

Con base en los resultados obtenidos, se observa similitud en las pruebas basadas en simulaciones cuando el número de robots es igual a 5 y 10; sin embargo, al aumentar a 20 robots, el comportamiento del enjambre es diferente al modificar el valor de r_r. Esto se refleja en la Figura 4.37c, ya que a diferencia de la Figuras 4.37a y 4.37b, la eficacia mejora con valores de r_r intermedios; mientras que, en los otros casos, mejora al aumentar su valor. Por otro lado, en los experimentos por implementación física (Figura 4.38), el tiempo de ejecución aumenta con valores altos de repulsión. La Tabla 4.13 afirma estos resultados, ya que existe una correlación negativa (el tiempo disminuye) para los resultados por simulación con 5

y 10 robots; en cambio, existe una correlación positiva (el tiempo aumenta) con los resultados por simulación con 20 robots en un rango de $0.6 < r_r \leq 1$ y por implementación física. En relación al parámetro de atracción (r_a), para todos los casos (simulación e implementación física), el tiempo de ejecución disminuye al aumentar su valor; esto debido a que se generan cadenas entre los robots, que facilitan el traslado entre las zonas de búsqueda y entrega (ver Figura 4.30).

En configuraciones con valores bajos de repulsión y altos de atracción (Figuras 4.26 y 4.30), la entrega de objetos es consistente, es decir, los robots se encuentran en todo momento buscando y entregando objetos. Esto debido a que se forman cadenas de robots de diferentes longitudes manteniendo al enjambre más unido, lo cual ayuda a transportarse de forma coordinada entre la zonas de búsqueda y entrega. Sin embargo, en algunos casos, las cadenas que se forman con esta configuración, están encabezadas por robots que están en el proceso de entrega y desvían la atención de aquellos que están en un proceso de búsqueda, y viceversa. Por el contrario, con valores altos de repulsión y bajos de atracción (Figuras 4.27 y 4.31), los robots se separan al tratar de evitar a los vecinos, pero debido a la influencia se reúnen en las zonas de búsqueda y entrega. Esto provoca que se entregue el mayor porcentaje de objetos en los primeros instantes de tiempo; sin embargo, al quedar pocos objetos, la mayor concentración del enjambre se encuentra en la zona de búsqueda, provocando que los robots se eviten y tarden más en tomar los últimos objetos.

Por medio de la plataforma de simulación, es posible explorar distintos elementos en un enjambre de robots, permitiendo experimentar con distintas cantidades de individuos; que, debido al costo de fabricación de un enjambre real, sería difícil probar. Además, se posibilita la homogeneidad en el enjambre, es decir, individuos con dinámica y capacidades de percepción idénticas entre sí. De manera que, por simulaciones, se probó el concepto de una tarea de transporte de objetos, empleando un algoritmo basado en el comportamiento de animales sociales, antes de ser implementado en un enjambre de robots real. Para esto, se propuso la arquitectura de robot tipo *BugBot*, que aún es un prototipo con mejoras

potenciales en el diseño y capacidades sensoriales. No obstante, se logró ejecutar la tarea implementando las reglas de comportamiento en tres *BugBot*; que, a pesar de tratar de mantener la uniformidad, existen errores de lectura en los sensores, perturbaciones en el voltaje aplicado en los actuadores y ruido de luz ambiental. Sin embargo, mediante la interacción entre sí y con estímulos por influencia, cumplen el objetivo y se demuestra que es posible trasladar las reglas de comportamiento en pequeños robots con limitaciones sensoriales.

Capítulo 5

Conclusiones y trabajo futuro

5.1 Conclusiones

La *RE* es una aproximación para la coordinación de múltiples robots que ha recibido mucha atención en los últimos años, su objetivo es desarrollar sistemas que sean robustos, escalables y flexibles; a pesar de su potencial, se conocen pocos casos donde se ha utilizado este enfoque para abordar una aplicación del mundo real. El propósito principal de la *RE*, es obtener comportamientos colectivos deseados y comprender sus propiedades ante las limitaciones de hardware de los robots disponibles.

La principal contribución de esta tesis, consiste en la implementación de reglas de comportamiento basadas en parámetros *RAOI* a un enjambre de robots real, sin intervención humana o métodos centralizados. Es decir, una vez definidos los parámetros a utilizar en una tarea específica, el enjambre tiene la libertad de encontrar soluciones sin que el humano participe en el proceso. Para esto, se propone un algoritmo que proporciona un medio simple y efectivo para ejecutar tareas colaborativas, donde la relación de los parámetros provoca la emergencia de comportamientos colectivos; que son fundamentales para desplegar la *RE* en el

futuro, especialmente cuando no se conocen instrucciones a priori o están codificadas en el sistema.

En base a los resultados obtenidos por simulación, los robots exploran el espacio de prueba bajo las reglas locales de comportamiento implementadas, manteniendo su dirección hasta llegar a un límite o reunirse con otros miembros del enjambre y formar grupos; de modo que, los parámetros de repulsión, orientación y atracción mantienen la forma del enjambre y cambian su distribución de navegación. Por otro lado, al agregar formas de percepción, es posible incluir diferentes mecanismos de influencia y asociar cada estímulo con tareas específicas. De esta forma, al integrar los paramétros $RAOI$ en un algoritmo, se abre la posibilidad de gobernar en el comportamiento de un enjambre de robots, explorar completamente las ventajas que ofrecen las arquitecturas de enjambre distribuidas y manejar entornos diversos y dinámicos. Estas reglas de comportamiento local, se diseñaron inspiradas en el comportamiento de las formas de vida social, por lo que se genera un comportamiento similar en nuestra mandada de robots; los cuales se construyeron bajo un modelo de arquitectura simple.

Al implementar las reglas de comportamiento en robots que operan en el mundo físico, nos enfrentamos a desafíos como: limitaciones sensoriales, perturbaciones en sensores y actuadores, cambios dinámicos contínuos y eventos con condiciones externas que son difíciles de modelar o predecir; por lo que es difícil obtener resultados idénticos a los observados en las simulaciones, sin embargo, se logra observar ciertas similitudes en los comportamientos durante instantes de tiempo. Esto se puede afirmar con los resultados de las Tablas 4.5 y 4.13, donde el efecto de los parámetros r_r y r_a, es similar en el área de cobertura y tiempo de ejecución.

De este modo, este trabajo de tesis proporciona un algoritmo de comportamiento capaz de establecer emergencia de colaboración en el enjambre, en ausencia de líderes y control centralizado. Además, el impacto de los parámetros r_r

y r_a en el comportamiento de un enjambre de robots físicos, el cual se cuantifica con indicadores de eficiencia, proporciona una aproximación del comportamiento ante casos específicos de tareas como: depredador-presa y transporte de objetos. También, abre nuevas posibilidades de exploración en la *RE* para trabajo futuro que se explican la siguiente sección.

5.2 Trabajo futuro

A pesar de que los resultados presentados son prometedores, existen una serie de oportunidades que deben de abordarse en trabajo futuro. Con el creciente uso de la *RE* y la necesidad de su aplicación al mundo real, es necesario comprender en su totalidad el efecto de todos los parámetros en el comportamiento del enjambre; por lo cual, se deben incluir metodologías para la inclusión del parámetro de orientación (r_o), el cual permite un movimiento coordinado entre los miembros del enjambre; además, considerar el uso de sensores más complejos para evitar zonas muertas o puntos ciegos en *ZOR*, *ZOO* y *ZOA*. Por otro lado, explorar otros mecanismos de percepción y, de este modo, probar nuevas tareas. De esta forma, ante procesos complejos que puedan dividirse en subtareas, es posible conmutar los valores paramétricos r_r, r_o y r_a, en función del estímulo por influencia activo; y así, mejorar la eficiencia de cada subtarea. Otra necesidad es realizar una verificación y validación del comportamiento del enjambre con métricas bien definidas, ya que a menudo suelen estar demasiado relacionadas con soluciones específicas y, por lo tanto, no se pueden utilizar para otros sistemas o comparaciones. Además, calcular un factor que determine el grado de variabilidad respecto al modelo simulado y físico. Así mismo, es posible realizar un estudio que considere las condiciones dinámicas de los robots, para evitar colisiones y condiciones del terreno.

Por otra parte, debido a que la mayor parte de trabajos se realizan en entornos estructurados de laboratorio, se requiere un escenario más realista para

aplicar los comportamientos generados en las tareas ya exploradas (depredador-presa y transporte de objetos) en aplicaciones reales como captura o rescate. Finalmente, los parámetros de repulsión, orientación, atracción e influencia, pueden establecerse adecuadamente utilizando enfoques de optimización, para mejorar el desempeño del enjambre en tareas colectivas específicas.

5.3 Divulgación científica

5.3.1 Artículos de revista

- Erick Ordaz-Rivas, Angel Rodriguez-Liñan, Mario Aguilera-Ruiz, Luis Torres-Treviño. Collective Tasks for a Flock of Robots Using Influence Factor. October 2018. Journal of Intelligent and Robotic Systems.

5.3.2 Artículos de congreso

- Erick Ordaz-Riva, Angel Rodriguez-Liñan, Luis Torres-Treviño. Collective Behaviors in Swarms of Builder Robots. December 2019. Research in Computing Science 148(11):103-114.
- Erick Ordaz-Rivas, Angel Rodriguez-Liñan, Luis Torres-Treviño. Collaboration of Robot Swarms with a Relation of Individuals with Prey-Predator Type. December 2017. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.

Apéndice A

Reporte técnico de BugBots

A.1 Diseño

La estructura del robot está formada por diferentes niveles, en cada nivel se incorporan los sensores, actuadores y el microcontrolador. El diámetro total del robot es de 12 cm y el de cada rueda de 1.2 cm. Una vista de la estructura mecánica puede verse en la Figura A.1.

A.2 Hardware

A.2.1 Microcontrolador

El procesamiento del robot es realizado por un módulo *ESP32*, que incorpora un chip *ESP32-D0WDQ6*; el cual cuenta con dos núcleos de CPU que se pueden controlar individualmente, y la frecuencia del reloj es ajustable. Además, integra *Bluetooth*® y $Wi-Fi$, ideal para desarrollar productos de IoT. Las especificaciones del módulo se encuentran en la Tabla A.1.

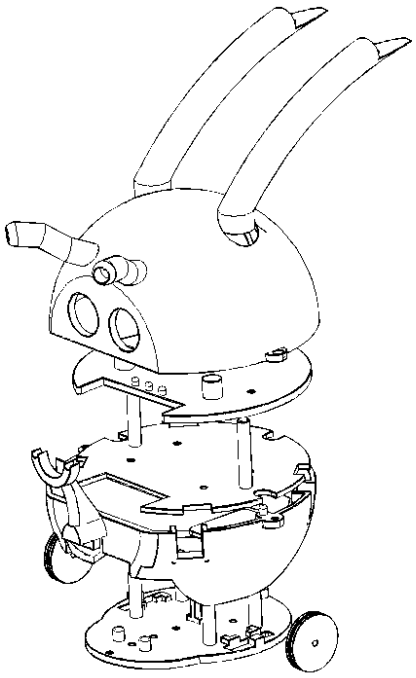


Figura A.1: Vista explosionada de la estructura del robot.

A.2.2 Sensores y actuadores

Cada BugBot cuenta con sensores de proximidad y luz, para detectar y medir distancias con objetos cercanos e interactuar con el entorno. Al frente del robot se ubica un sensor ultrasónico HC-SR04 (Figura A.2), mientras que en los costados, sensores infrarrojos TCRT5000 (FiguraA.3) y resistencias dependientes de luz (LDR) (Figura A.4), cuyas características se muestran en la Tablas A.2 - A.4.

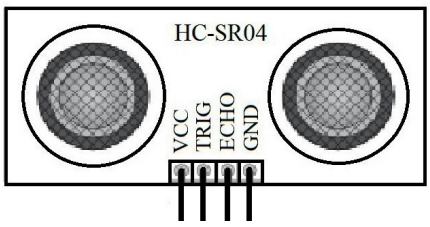


Figura A.2: Sensor ultrasónico HC-SR04.

Por otro lado, cada robot utiliza dos motores de CC con reducción de velocidad

Tabla A.1: Características de módulo ESP32.

Voltaje de alimentación	3.3V DC (2.7 - 3.6V)
Corriente de operación	*80mA (fuente superior a 500mA)
CPU	Dual Core Tensilica LX6 (32 bit)
Frecuencia de reloj	240MHz
SRAM	520KB
Memoria flash externa	4MB
Pines Digitales GPIO	34 (Incluyendo todos los pereféricos)
Canales PWM	16 canales independientes (16-bits)
ADC	2 (12-bit)
DAC	2 (8-bit)
Wi-Fi	802.11n up to 150 Mbps
Bluetooth	V4.2 BR/EDR

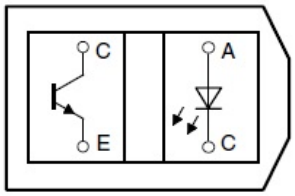


Figura A.3: Sensor infrarrojo TCRT5000.

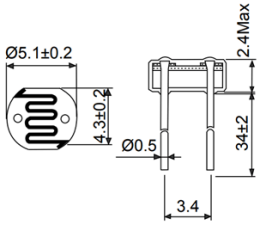


Figura A.4: Resistencia dependiente de luz (LDR).

y un circuito integrado SN754410 para controlarlos; las características de los motores se muestran en las Tabla A.5.

Tabla A.2: Características de sensor ultrasónico HC-SR04.

Voltaje de alimentación	5V DC
Corriente de operación	15 mA
Rango de salida	2 - 400 cm
Resolución	3 mm
Ángulo efectivo	<15°
Ángulo de medición	30°

Tabla A.3: Características de sensor infrarrojo TCRT5000.

Voltaje de alimentación	3.5 - 5V DC
Corriente operación	60 mA
Distancia focal	2.55mm

A.2.3 Alimentación

Cada robot cuenta con una banco de energía de 5 V y con capacidad de 2600 mAh, para alimentar el módulo de *ESP32* y componentes del robot (Figura A.5a).

A.3 Software

La programación del microcontrolador se realiza en el entorno de desarrollo integrado de *Arduino*^{TM} (*Arduino IDE 1.8.12*); el código implementado en cada robot se muestra en el Apéndice B. Para la comunicación *Bluetooth*®, se utiliza la aplicación *Bluetooth Serial Terminal* para *Windows 10*, mientras que para la comunicación con un celular la aplicación *Bluetooth Terminal* para *Android*.

Tabla A.4: Características de LDR.

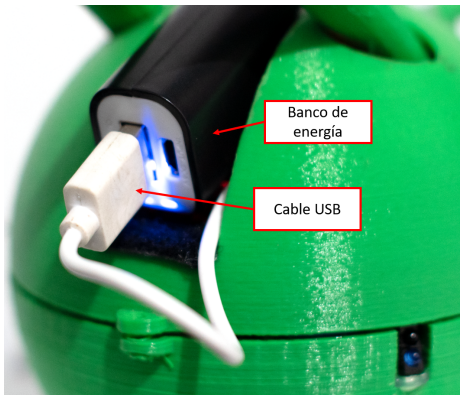
Pico de voltaje AC/DC	320 V max
Corriente operación	60 mA max
Disipación de potencia	100 mW max
Resistencia de celda	10 - 1000 LUX

Tabla A.5: Características de motor CC.

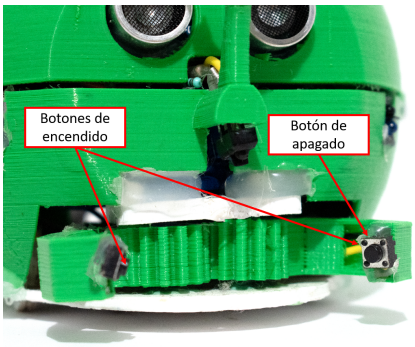
Voltaje de operación	6 V
Par	2 kgf.cm
Relación de transmición	100:1

A.4 Funcionamiento

Para encender el robot es necesario conectar el cable USB, incorporado en el robot, al banco de energía y presionar los botones de inicialización, como se muestra en la Figura A.5; una vez encendido, el robot inicia con el proceso de búsqueda y recolección de objetos. Para desactivarlo, sólo hay que presionar el botón izquierdo.



(a) Conexión a banco de energía.



(b) Botones de encendido y apagado.

Figura A.5: Alimentación y encendido del robot.

Apéndice B

Códigos en Arduino - Reglas de comportamiento

B.1 Tarea de relación tipo depredador-presa

```
1  //*********************************************************************
2  //Estudio de comportamintos emergentes en enjambres de robots basados
3  //factores de influencia
4  //DIE Mecatronica
5  //UANL - FIME
6  //M.Sc. Erick de Jesus Ordaz Rivas
7  //*********************************************************************
8
9  // Libraries
10 #include <NewPing.h>
11 #include <SoftwareSerial.h>
12
13 #define Trig  5          // Ultrasonic sensor digital output.
14 #define Echo 6           // Ultrasonic sensor digital input.
15 int button = 4;          // Push button digital input.
16 int motorRD = 12;        // Right DC motor digital output.
17 int motorLD = 10;        // Left DC motor digital output.
```

```
18 int motorRA = 11;          // Right DC motor digital output.
19 int motorLA = 9;           // Left DC motor digital output.
20 int ldrD = A2;             // Left LDR analog input.
21 int ldrI = A4;             // Right LDR analog input.
22 int irD = A3;              // Right TCRT analog input.
23 int irI = A5;              // Left TCRT analog input.
24
25 //Variables and constants initialization.
26 int statebutton = 0;
27 int previousstate = 0;
28 int swarmon = 0;
29
30 int distance;       // Calculated distance by ultrasonic sensor
31 int velR,velO,velA;// Motors speed
32 int vel, velI;      // Motors speed
33 int val;            // Value read by serial port.
34 int bI;
35 int blue = 1;      // Bluetooth flag
36 int ldrI_val, ldrD_val; // LDRs.
37 int irI_val, irD_val;   // TCRTs.
38 int dR=5;           // Repulsion distance.
39 int dO=15;          // Orientation distance.
40 int dA=100;         // Attraction distance.
41 int Qf[]={0, 1};  // Direction vector by ultrasonic sensor.
42 int Ql[]={-0.707,0.707}; // Direction vector by left TCRT.
43 int Qr[]={0.707,0.707};  // Direction vector by right TCRT.
44 int qfA,qlA,qrA;  // Elements detected by sensor in attraction zone.
45 int qfR,qlR,qrR;  // Elements detected by sensor in repulsion zone.
46 int detR;          // Elements detected in repulsion zone.
47 int detO;          // Elements detected in orientation zone.
48 int detA;          // Elements detected in attraction zone.
49 int DQf;           // Measured distance by ultrasonic sensor.
50 int DQl;           // Measured distance by left TCRT.
51 int DQr;           // Measured distance by right TCRT.
52 int DLl,ILl;       // Light information by left LDR.
53 int DLr,ILr;       // Light information by right LDR.
```

```
54 int ddx,ddy;        // Components of the desired direction.
55 int dd,dd2;         // Desired direction.
56
57 void setup() {
58   // Initialize serial communication
59   Serial.begin(9600);
60   Serial1.begin(9600);
61   // Pin configuration of motors
62   pinMode(motorLD, OUTPUT);
63   pinMode(motorRD, OUTPUT);
64   pinMode(motorLA, OUTPUT);
65   pinMode(motorRA, OUTPUT);
66   digitalRead(button);
67 }
68
69 void loop() {
70   if (Serial.available()){
71     blue = '0'; // Check for commands.
72   }
73   while (blue == '0') {
74     if (Serial.available()) {
75       val = Serial.read();
76     }
77     // Change parameter values.
78     if (val == '1') {
79       Stop();
80       dR=5;
81       dO=15;
82       dA=20;
83       blue = '1';
84     }
85     if (val == '2') {
86       Stop();
87       dR=10;
88       dO=15;
89       dA=20;
```

```
90          blue = '1';
91        }
92        if (val == '3') {
93          Stop();
94          dR=15;
95          dO=15;
96          dA=20;
97          blue = '1';
98        }
99        if (val == '4') {
100         Stop();
101         dR=2;
102         dO=15;
103         dA=60;
104         blue = '1';
105       }
106       if (val == '5') {
107         Stop();
108         dR=10;
109         dO=15;
110         dA=60;
111         blue = '1';
112       }
113       if (val == '6') {
114         Stop();
115         dR=15;
116         dO=15;
117         dA=60;
118         blue = '1';
119       }
120       if (val == '7') {
121         Stop();
122         dR=5;
123         dO=15;
124         dA=100;
125         blue = '1';
```

```
126       }
127       if (val == '8') {
128         Stop();
129         dR=10;
130         dO=15;
131         dA=100;
132         blue = '1';
133       }
134       if (val == '9') {
135         Stop();
136         dR=15;
137         dO=15;
138         dA=100;
139         blue = '1';
140       }
141     }
142
143     statebutton = digitalRead(button);
144     if ((statebutton == HIGH) && (previousstate == LOW)){
145       swarmon = 1 - swarmon;
146     }
147     previousstate = statebutton;
148
149   // Sensor reading.
150     ldrI_val = analogRead(ldrI);
151     ldrD_val = analogRead(ldrD);
152     DLl = map(ldrI_val, 0, 1023, 1000, 0);
153     DLr = map(ldrD_val, 0, 1023, 1000, 0);
154
155     irI_val = analogRead(irI);
156     irD_val = analogRead(irD);
157
158     DQl = 126.43*pow(irI_val,-0.692);
159     DQr = 126.43*pow(irD_val,-0.692);
160
161     DQf = Ultrasonic();
```

```
162
163   //Influence parameters.
164   int DLm = (DLl +DLr)/2;
165   int DLsup = DLm + 25;
166   int DLinf = DLm - 25;
167   int Rango = 120;
168
169   // Initialize  variables to zero
170   detR=0; detO=0; detA=0;
171   qfR=0; qlR=0; qrR=0; qfA=0; qlA=0; qrA=0;
172   velR = 50; velO = velI; velA=(220+velI)/2; vel=velO;
173
174   if (swarmon == 1){
175
176   // Repulsion.
177   if (DQf<=dR || DQl<=dR || DQr<=dR){
178     if (DQf<=dR){
179       qfR=1;
180     }
181     if (DQl<=dR){
182       qlR=1;
183     }
184     if (DQr<=dR){
185       qrR=1;
186     }
187     ddx= -(Qf[0]*qfR + Ql[0]*qlR + Qr[0]*qrR);
188     ddy= -(Qf[1]*qfR + Ql[1]*qlR + Qr[1]*qrR);
189     vel = velR; detR=1;
190   }
191
192   //Influence.
193   if (detR == 0) {
194     if (DLinf < DLl && DLl < DLsup && DLinf < DLr && DLr < DLsup) {
195       //Attraction.
196       if (dO < DQf && DQf <= dA) {
197         detA=1; vel=velA;
```

```
198           ddx = 0; ddy = 1;
199         }
200       }
201       else {
202         dd2 = 30;
203         if (DLl > DLr) {
204           TurnRight(dd2);
205           Stop();
206         }
207         else {
208           TurnLeft(dd2);
209           Stop();
210         }
211       }
212     }
213
214     // Orientation
215     if (detR==0 && detA==0){
216       if (DQf<=dO || DQl<=dO || DQr<=dO){
217         detO = 1; vel = velO;
218         ddx = 0; ddy = 1;
219       }
220       // Out of range
221       if (detO == 0){
222         vel = velO;
223         ddx=0; ddy=1;
224       }
225     }
226
227     // Calculation of the desired direction in radians.
228     float ddrad=atan2(ddy,ddx);
229     dd = ddrad*180/3.14159;
230
231     // Conversion to the robot reference system.
232     if (dd>-90){
233       dd2=dd-90;
```

```
234     }
235     if (dd<=-90){
236       dd2=dd+270;
237     }
238
239     if (dd2>=90 && dd2<180){
240       TurnLeft(90);
241       Forward();
242     }
243     if(dd2<90 && dd2>0){
244       TurnLeft(dd2);
245       Forward();
246     }
247
248     if (dd2<=-90 && dd2>-180){
249       TurnRight(90);
250       Forward();
251     }
252
253     if (dd2>-90 && dd2<0){
254       dd2=abs(dd2);
255       TurnRight(dd2);
256       Forward();
257     }
258
259     if (dd2 == 0) {
260       if (DQf > dR) {
261         Forward();
262       }
263       else {
264         int randomnumber = random(0,1);
265         if (randomnumber == 1) {
266           TurnRight(20);
267         }
268         else {
269           TurnLeft(20);
```

```
270         }
271       }
272     }
273
274     if (dd2>=178 && dd2<=182){
275       if (irI_val>irD_val){
276           TurnRight(90);
277           Forward();
278       }
279       if (irI_val<=irD_val){
280           TurnLeft(90);
281           Forward();
282       }
283       }
284     }
285
286     else {
287       Stop();
288     }
289
290 }
291
292 //Motion functions
293 void Forward(){
294       digitalWrite(motorLD, 0);
295       digitalWrite(motorRD, 0);
296       analogWrite(motorLA, (vel));
297       analogWrite(motorRA, vel);
298 }
299
300 void Backward(){
301       digitalWrite(motorLD, 1);
302       digitalWrite(motorRD, 1);
303       digitalWrite(motorLA, 0);
304       digitalWrite(motorRA, 0);
305 }
```

```
306
307 void TurnRight(int grados){
308     digitalWrite(motorLD, 0);
309     digitalWrite(motorRD, 1);
310     digitalWrite(motorLA, 1);
311     digitalWrite(motorRA, 0);
312     int time= 3.39*grados;
313     delay(time);
314 }
315
316 void TurnLeft(int grados){
317     digitalWrite(motorLD, 1);
318     digitalWrite(motorRD, 0);
319     digitalWrite(motorLA, 0);
320     digitalWrite(motorRA, 1);
321     int time= 3.39*grados;
322     delay(time);
323 }
324
325 void Stop(){
326     digitalWrite(motorLD, 1);
327     digitalWrite(motorRD, 1);
328     digitalWrite(motorLA, 1);
329     digitalWrite(motorRA, 1);
330 }
331
332 int Ultrasonic(){
333   digitalWrite(Trig, LOW);
334   delayMicroseconds(5);
335   digitalWrite(Trig, HIGH);
336   delayMicroseconds(10);
337   digitalWrite(Trig, LOW);
338   int t = pulseIn(Echo, HIGH);
339   int d = (t/58);
340   return d;
341 }
```

B.2 Tarea de transporte de objetos

```
1  //*********************************************************************
2  //Estudio de comportamintos emergentes en enjambres de robots basados
3  //factores de influencia
4  //DIE Mecatronica
5  //UANL - FIME
6  //M.Sc. Erick de Jesus Ordaz Rivas
7  //*********************************************************************
8
9  // Libraries
10 #include <ESP32Servo.h>
11 #include <analogWrite.h>
12 #include <Wire.h>
13 #include <MechaQMC5883.h>
14
15 // Definition of I/O pins
16 const int Trig = 19;  // Ultrasonic sensor digital output.
17 const int Echo = 18;  // Ultrasonic sensor digital input.
18 #define MotorRB 27    // Left DC motor digital output.
19 #define MotorRF 26    // Left DC motor digital output.
20 #define MotorLB 25    // Right DC motor digital output.
21 #define MotorLF 33    // Right DC motor digital output.
22 #define LDRl 34       // Left LDR analog input.
23 #define LDRr 35       // Right LDR analog input.
24 #define TCRTbr 13     // TCRT analog input.
25 #define TCRTbl 39     // TCRT analog input.
26 #define TCRTfr 15     // TCRT analog input.
27 #define TCRTfl 36     // TCRT analog input.
28 #define TCRTg 4       // TCRT analog input (grip).
29 #define PUSHr 16      // Push button digital input.
30 #define PUSHl 17      // Push button digital input.
31 Servo SERVO;
32
33 //Magnetometer
```

```
34  MechaQMC5883 qmc;
35  int x, y, z;
36  float acimut, geografico;
37  float declinacion=4.58;
38
39  static int gn;
40  int gd, gsup, ginf;
41  int bg = 0;
42
43  //Influence by magnetometer
44  int MSsup, MSinf, MDsup, MDinf;
45  int MSi, MSf, MDi, MDf;
46  int Mumbral, MSl, MSr, MDl, MDr;
47  static int MS, MD;
48
49  //Declaration of sensor variables
50  int dUS;              //Ultrasonic sensor
51  int dLDRl,dLDRr;      //Light information by LDRs.
52  int dTCRTg;           //Gripper
53  int dTCRTbl, dTCRTbr, dTCRTfr, dTCRTfl; //TCRTs
54  int uTCRT;
55  int tempfr; int tempfl;
56  int tempfrs = 0;
57  int tempfls = 0;
58
59  int DLm, DLsup, DLinf, Rango, uDLm, uDLmg;
60  int sLDRl, sLDRr, uLDR, dsLDR;
61  int bS = 0;
62  int bc= 2;
63  int fb = 0;
64  int rn = 0;
65
66  long d, t;
67  int mdUS;
68  int ddUS;
69  int tempdUS;
```

```
70  static int sTCRTg;
71  int grip = 0;
72
73  int velR,velO,velA; //Motors speed
74  int vel, velI;
75
76  int dPUSHr, dPUSHl; //Push button states.
77  int swarmON = 0;
78
79  // Repulsion, orientation and attraction parameters.
80  int dR=10;
81  int dO=10;
82  int dA=100;
83
84  //Direction vectors by sensors.
85  float Qf[] ={1, 0};            // 0 Direction vector by ultrasonic sensor.
86  float Qbr[]={-0.707, -0.707};// -135 Direction vector by TCRT.
87  float Qbl[]={-0.707, 0.707}; // 135 Direction vector by TCRT.
88  float Qfr[]={0.707, -0.707}; // -45 Direction vector by TCRT.
89  float Qfl[]={0.707, 0.707};  // 45 Direction vector by TCRT.
90
91  int qfR,qfrR,qflR,qbrR,qblR; // Elements detected by sensor in repulsion zone.
92  int qfA,qfrA,qflA,qbrA,qblA; // Elements detected by sensor in attraction zone.
93  int detR,detO,detA,detI;     // Elements detected in zones.
94  float ddx,ddy,dd;            // Components of the desired direction.
95  int TR,TL,F;
96  static int randomnumber;
97
98  void setup() {
99    Serial.begin (115200);
100
101   //Magnetometer
102   Wire.begin(21,22,40000);
103   qmc.init();
104
105   // I/O pin configuration.
```

```
106     pinMode(PUSHr,INPUT_PULLUP);
107     pinMode(PUSHl,INPUT_PULLUP);
108     pinMode(TCRTg, INPUT);
109     pinMode(TCRTbr, INPUT);
110     pinMode(TCRTbl, INPUT);
111     pinMode(TCRTfr, INPUT);
112     pinMode(TCRTfl, INPUT);
113     pinMode(LDRl, INPUT);
114     pinMode(LDRr, INPUT);
115     pinMode(MotorLB, OUTPUT);
116     pinMode(MotorLF, OUTPUT);
117     pinMode(MotorRB, OUTPUT);
118     pinMode(MotorRF, OUTPUT);
119     pinMode(Trig, OUTPUT);
120     pinMode(Echo, INPUT);
121     SERVO.attach(32);
122
123     //Calibration for magnetometer.
124     Magnetometer();
125     MS=geografico; MD=geografico+180; Mumbral=180;
126     if (MD>360){MD=MD-360;}
127     MSsup=MS+Mumbral; MSinf=MS-Mumbral;
128     MDsup=MD+Mumbral; MDinf=MD-Mumbral;
129     if (MSinf<0){MSinf=MSinf+360;}
130     if (MSsup>360){MSsup=MSsup-360;}
131     if (MSinf<MSsup){MSi=0;}
132     else if (MSinf>MSsup){MSi=1;}
133     if (MDinf<0){MDinf=MDinf+360;}
134     if (MDsup>360){MDsup=MDsup-360;}
135     if (MDinf<MDsup){MDi=0;}
136     else if (MDinf>MDsup){MDi=1;}
137
138     //Calibration for gripper.
139     sTCRTg = analogRead(TCRTg);
140     sTCRTg = map(sTCRTg, 4095, 0, 0, 4095);
141     sTCRTg = sTCRTg + 500;
```

```
142   }
143
144   void loop() {
145     Magnetometer();
146
147     // Sensors reading.
148     dLDRl = analogRead(LDRl); // LDRs.
149     dLDRr = analogRead(LDRr);
150     dLDRl = map(dLDRl, 0, 4096, 4096, 0);
151     dLDRr = map(dLDRr, 0, 4096, 4096, 0);
152
153     dTCRTg  = analogRead(TCRTg); // TCRTs.
154     dTCRTbr = analogRead(TCRTbr);
155     dTCRTbl = analogRead(TCRTbl);
156     dTCRTfr = analogRead(TCRTfr);
157     dTCRTfl = analogRead(TCRTfl);
158
159     dTCRTbr = 126.43*pow(dTCRTbr,-0.692);
160     dTCRTbl = 126.43*pow(dTCRTbl,-0.692);
161     dTCRTfr = 126.43*pow(dTCRTfr,-0.692);
162     dTCRTfl = 126.43*pow(dTCRTfl,-0.692);
163
164     dUS = Ultrasonic();
165
166     //Push buttons.
167     dPUSHr = digitalRead(PUSHr); //
168     dPUSHl = digitalRead(PUSHl);
169     if (dPUSHr == LOW && dPUSHl == LOW && grip==0){swarmON=1;}
170     if (dPUSHr == HIGH && dPUSHl == LOW && grip==0){swarmON=0;}
171
172     //Influence parameters.
173     DLm = (dLDRl + dLDRr) / 2;
174     dsLDR = sqrt((pow(abs(dLDRl - DLm), 2) + pow(abs(dLDRr - DLm), 2)) / 2);
175     uDLm = 2800;
176     uDLmg = 3500;
177
```

```
178     if (grip==0 && DLm>uDLm && (dUS<=dR || dTCRTfr<=dR || dTCRTfl<=dR)){
179       bS = bS + 1;
180     }
181     if (dTCRTfr<=dR && dTCRTfl<=dR){
182       bS = bS + 1;
183     }
184
185     // Initialize  variables to zero
186     TL=0; TR=0; F=0;
187     detR=0; detO=0; detA=0; detI=0;
188     qfR=0; qfrR=0, qflR=0, qbrR=0, qblR=0;
189     qfA=0; qfrA=0, qflA=0, qbrA=0, qblA=0;
190     velR = 100; velO = velI; velA=(220+velI)/2; vel=velO;
191
192     //Gripper.
193     if(dTCRTg>sTCRTg)
194     {
195       grip=1; //Close gripper
196       SERVO.write(45);
197     }
198     else
199     {
200       grip=0; //Open gripper
201       SERVO.write(90);
202     }
203
204     //Angle by magnetometer
205     if (MSi==0)
206     {
207       if (MSinf<geografico && geografico<MSsup){MSf=1;}
208       else {MSf=0;}
209     }
210     else if (MSi==1)
211     {
212       if (MSinf<geografico || geografico<MSsup){MSf=1;}
213       else {MSf=0;}
```

```
214     }
215     if (MDi==0)
216     {
217       if (MDinf<geografico && geografico<MDsup){MDf=1;}
218       else {MDf=0;}
219     }
220     else if (MDi==1)
221     {
222       if (MDinf<geografico || geografico<MDsup){MDf=1;}
223       else {MDf=0;}
224     }
225
226     if (rn == 1)
227     {
228       if (fb >= 2) {rn = 0; bS = 0;}
229       BehaviorRules();
230       if (dUS<=dR){fb= fb + 1;}
231     }
232
233     if (grip == 0 & rn == 0)
234     {
235       if (MSf==1 && bS<=bc)
236       {
237         BehaviorRules();
238       }
239
240       else if (bS > bc)
241       {
242         fb = 0; bS = 0; rn = 1;
243       }
244
245       else if (MSf == 0 & bS <= bc)
246       {
247         MSr = abs(MSsup-geografico);
248         MSl = abs(MSinf-geografico);
249         if (MSr<MSl)
```

```
250           {
251             ddx=1; ddy=1;
252           }
253           else
254           {
255             ddx=1; ddy=-1;
256           }
257         }
258       }
259
260       else if (grip==1 && rn == 0)
261       {
262         if (DLm>uDLmg && MDf ==1)
263         {
264           SERVO.write(90);
265           grip=0;
266         }
267
268         if (MDf == 1 && bS <= bc)
269         {
270           BehaviorRules();
271         }
272
273         else if (bS > bc)
274         {
275           bS = 0; fb = 0;rn = 1;
276         }
277
278         else if (MDf == 0 & bS <= bc)
279         {
280           MDr = abs(MDsup-geografico);
281           MDl = abs(MDinf-geografico);
282           if (MDr<MDl)
283           {
284             ddx=1; ddy=1;
285           }
```

```
286           else
287           {
288             ddx=1; ddy=-1;
289           }
290         }
291       }
292
293       // Calculation of the desired direction in radians.
294       float ddrad=atan2(ddy,ddx);
295       dd=ddrad*180/3.14159;
296
297       //Conversion to the robot reference system.
298       if (swarmON == 1)
299       {
300         if (0<dd && dd<180)
301         {
302           TL=1; TR=0; F=0;
303           TurnLeft(dd);
304         }
305
306         if (0>dd && dd>-180)
307         {
308           TR=1; TL=0; F=0;
309           TurnRight(dd);
310         }
311
312         if (dd== 0)
313         {
314           TR=0; TL=0; F=1;
315           Forward();
316           //Stop();
317         }
318       }
319
320       else
321       {
```

```
322         Stop();
323       }
324     }
325
326     void Magnetometer()
327     {
328       qmc.read(&x, &y, &z, &acimut);
329       geografico = acimut + declinacion;
330       if (geografico < 0)
331       {
332         geografico= geografico+360;
333       }
334       if (geografico >360)
335       {
336         geografico= geografico-360;
337       }
338     }
339
340     int Ultrasonic()
341     {
342       digitalWrite(Trig, LOW);
343       delayMicroseconds(5);
344       digitalWrite(Trig, HIGH);
345       delayMicroseconds(10);
346       digitalWrite(Trig, LOW);
347       t = pulseIn(Echo, HIGH);
348       d = (t/58);
349       return d;
350     }
351
352     void Backward()
353     {
354       analogWrite(MotorLB,255);
355       analogWrite(MotorLF,0);
356       analogWrite(MotorRB,220);
357       analogWrite(MotorRF,0);
```

```
358   }
359
360   void Forward()
361   {
362     analogWrite(MotorLB,0);
363     analogWrite(MotorLF,vel);
364     analogWrite(MotorRB,0);
365     analogWrite(MotorRF,vel);
366   }
367
368   void Stop()
369   {
370     analogWrite(MotorLB,0);
371     analogWrite(MotorLF,0);
372     analogWrite(MotorRB,0);
373     analogWrite(MotorRF,0);
374   }
375
376   void TurnRight(int grados)
377   {
378     grados = (-1 * grados);
379     analogWrite(MotorLB, 0);
380     analogWrite(MotorLF, 255);
381     analogWrite(MotorRB, 255);
382     analogWrite(MotorRF, 0);
383     int time = 4.4 * grados;
384     delay(time);
385   }
386
387   void TurnLeft(int grados)
388   {
389     analogWrite(MotorLB, 255);
390     analogWrite(MotorLF, 0);
391     analogWrite(MotorRB, 0);
392     analogWrite(MotorRF, 255);
393     int time = 3.6 * grados;
```

```
394     delay(time);
395   }
396
397   void BehaviorRules{
398     //Repulsion
399     if (dUS<=dR || dTCRTbr<=dR || dTCRTbl<=dR || dTCRTfr<=dR || dTCRTfl<=dR)
400     {
401       ddx= -(Qf[0]*qfR + Qfr[0]*qfrR + Qfl[0]*qflR + Qbr[0]*qbrR + Qbl[0]*qblR);
402       ddy= -(Qf[1]*qfR + Qfr[1]*qfrR + Qfl[1]*qflR + Qbr[1]*qbrR + Qbl[1]*qblR);
403       if (ddx<=-1 || ddy==0)
404       {
405         randomnumber = random(0,2);
406         if (randomnumber == 0) {ddx=-0.707; ddy=0.707;}
407         if (randomnumber == 1) {ddx=-0.707; ddy=-0.707;}
408       }
409       vel= velR; detR= 1;
410       fb= fb + 1;
411     }
412
413     //Influence
414     if (detR == 0)
415     {
416       if (dsLDR<1000)
417       {
418         detI=0;
419         //Attraction
420         if (dO < dUS && dUS > dA)
421         {
422           vel=velA; detA=1;
423           ddx= 1; ddy= 0;
424         }
425       }
426       else
427       {
428         detI=1;
429         if (dLDRl < dLDRr)
```

```
430           {
431             ddx=1; ddy=-0.035;
432           }
433           else
434           {
435             ddx=1; ddy=0.035;
436           }
437         }
438       }
439
440       //Orientation
441       if (detR==0 && detA==0 && detI==0)
442       {
443         if (dUS<=dO || dTCRTbr<=dO || dTCRTbl<=dO || dTCRTfr<=dO || dTCRTfl<=dO)
444         {
445           detO= 1; vel= velO; ddx= 1; ddy= 0;
446         }
447         else
448         {
449           vel= velO; ddx= 1; ddy= 0;
450         }
451       }
452     }
```

Apéndice C

Código en Scilab - Reglas de comportamiento

C.1 Tarea de relación tipo depredador-presa

```
1  //*********************************************************************
2  //Estudio de comportamintos emergentes en enjambres de robots basados
3  //factores de influencia
4  //DIE Mecatronica
5  //UANL - FIME
6  //M.Sc. Erick de Jesus Ordaz Rivas
7  //*********************************************************************
8
9  function [norm_val] = normalize(val, l_min, l_max)
10     norm_val = (val - l_min)/((l_max - l_min) + %eps);
11     if norm_val > 1 then
12         norm_val = 1;
13     elseif norm_val < 0 then
14         norm_val = 0;
15     end
16 endfunction
17
```

```
18 function [val]=denormalize(norm_val, l_min, l_max)
19     val = norm_val * (l_max - l_min) + l_min;
20     if val > l_max then
21         val = l_max;
22     elseif val < l_min then
23         val = l_min;
24     end
25 endfunction
26
27 function [dxdt]=dynamicModel(t,c)
28     // c(1,:) x position
29     //c(2,:) y position
30     //c(3,:) movement
31     //c(4,:) orientation
32     //c(5,:) speed
33     //c(6,:) angular speed
34
35     //Parameters
36     m = .38;     //mass
37     Ip = 0.005; //inertia moment
38     d = 0.02;    //distance centroide to axis wheel
39     r = 0.03;    //radio wheel
40     R = 0.05;    //distance wheel-center
41
42     M = [m 0 ; 0 (Ip+m*d^2)];           //Inertial matrix
43     H = [-m*d*c(6)^2 ; m*d*c(5)*c(6)]; //Coriolis and centrifugal forces
44     B = [1/r 1/r ; R/r -R/r];           //Conversion torque-wheel-movil force
45     A = [r/2 r/2 ; r/(2*R) -r/(2*R)];  //Speed ratio
46     F = [0; 0];                         //Friction
47     Ts = [.434 0 ; 0 .434];
48     Ks = [2.745 0 ; 0 2.745];
49     Kl = [1460.2705 0 ; 0 1460.2705];
50
51     dxdt = [[cos(c(4)) -d*sin(c(4)); sin(c(4)) d*cos(c(4))]*[c(5); c(6)]; ...
52     ... c(5);c(6); inv(M + B*inv(Kl)*Ts*inv(A)) * (B*inv(Kl)*Ks*u - ...
53     ... (H + F + B*inv(Kl)*inv(A)*[c(5); c(6)]))];
```

```
54 endfunction
55
56 function [c,t]= mov(u,ci)
57     ti = 0;                 //Initial time
58     tf = 1;                 //Final time
59     tspan = [ti:0.1:tf]; //Time vector
60     c = ode(ci,ti,tspan,dynamicModel);
61     t = tspan;
62 endfunction
63
64 function [ReportPredators]= ppBehavior(Predators, Preys, dRd, dOd, dAd)
65     res = zeros(3,1);           //Center of mass and dispersion report
66     Tv = Predators + Preys;     //Total of vehicles
67     C = zeros(Tv, 6);
68     for v=1:Tv
69         C(v,4) = denormalize(rand(1,'uniform'),0,2*%pi); ;
70     end                         //Vehicle states
71     iPreys = zeros(Preys,3)     //Influence by prey
72     uO=0; uR =0; uA = 0; uI =0; //Initial speeds (voltages)
73     ud = zeros(Tv,2)+uO;        //Desired speed (voltage)
74     lims= 10;                   //Test area limits
75     dInf = 2;                   //Maximum distance of prey influence 2
76     cp = 0;                     //Collisions by prey
77     m=0;                        //Time
78     collisions = 3;
79     dirExp = C(:,4);
80     temp = zeros(Tv,1)
81     count = zeros(Tv,1)
82
83     for i=1: Preys
84         iPreys(i,1) = C(i,1);
85         iPreys(i,2) = C(i,2);
86         iPreys(i,3) = 0;
87     end
88
89     dR = 0.075+dRd; dRp = 0.25; //Respulsion ratio
```

```
90      dO = 0.075+dOd; dOp = 0.25; //Orientation ratio
91      dA = 0.075+dAd; dAp = 0.25; //Atraction ratio
92
93      for v = 1:Tv  //Initialize predators
94          if v == 1 then
95              C(v,1) = lims*denormalize(rand(1,'uniform'),0.4,0.6);
96              C(v,2) = lims*denormalize(rand(1,'uniform'),0.5,0.6);
97          else
98              while %T
99                  if v <= Preys then
100                     safe = 0;
101                     C(v,1) = lims*denormalize(rand(1,'uniform'),0.4,0.6);
102                         C(v,2) = lims*denormalize(rand(1,'uniform'),0.5,0.6);
103                     for j = 1:v-1
104                         dother = sqrt((C(v,1)-C(j,1))^2 + (C(v,2)-C(j,2))^2);
105                         if dother > 0.3 then
106                                 safe = safe + 1;
107                         end
108                     end
109                     if safe == j then
110                         break;
111                     end
112                 else
113                     safe = 0;
114                     C(v,1) = lims*denormalize(rand(1,'uniform'),0.33,0.66);
115                     C(v,2) = lims*denormalize(rand(1,'uniform'),0,0.33);
116                     for j = 1:v-1
117                         dother = sqrt((C(v,1)-C(j,1))^2 + (C(v,2)-C(j,2))^2);
118                         if dother > 0.3 then
119                             safe = safe + 1;
120                         end
121                     end
122                     if safe == j then
123                         break;
124                     end
125                 end
```

```
126             end
127         end
128         C(v,3) = 0;
129         C(v,4) = denormalize(rand(1,'uniform'),0,2*%pi);
130         C(v,5) = 0;
131         C(v,6) = 0;
132     end
133
134     scf();
135     f = get("current_figure");
136     f.BackgroundColor = [1 1 1];
137     f.figure_position = [10,10];
138     f.figure_size=[700,800];
139
140     while cp < collisions
141         m = m + 1;
142         for v=1:Tv
143             if v <= Preys then
144                 dR = dRp; dO = dOp; dA = dAp;
145             else
146                 dR = dRd; dO = dOd; dA = dAd;
147             end
148
149             //Elements detected
150             DetR = 0;       //Repulsion
151             DetO = 0;       //Orientation
152             DetA = 0;       //Attraction
153             DetI = 0;       //Influence (Objects)
154             ElemRx = [];    //Elements in repulsion
155             ElemOx = [];    //Elements in orientation
156             ElemAx = [];    //Elements in attraction
157             ElemIx = [];    //Elements in influence
158             ElemRy = [];    //Elements in repulsion
159             ElemOy = [];    //Elements in orientation
160             ElemAy = [];    //Elements in attraction
161             ElemIy = [];    //Elements in influence
```

```
162
163             //Detection range for repulsion, orientation, attraction, influence,
164             //nest and objects
165             rangR = 2.0944;
166             rangO = 2.0944;
167             rangA = 0.5235988;
168             rangI = 2.44346;
169
170             //Verify each sensor for repulsion of walls
171             for i = 1:5
172                 if i == 1 then
173                     dirObs = C(v,4) -3.8397244;
174                 else
175                     dirObs = dirObs+1.9198622;
176                 end
177                 if dirObs < 0 then
178                     dirObs = dirObs+(2*%pi);
179                 elseif dirObs > (2*%pi) then
180                     dirObs = dirObs-(2*%pi);
181                 end
182                 Dir = [cos(dirObs),sin(dirObs)];
183                 limitX = C(v,1) + (Dir(1)*dR);
184                 limitY = C(v,2) + (Dir(2)*dR);
185
186                 //Resulting direction due exploration
187                 if limitX > lims | limitX < 0 | limitY > lims | limitY < 0 then
188                     explore(v) = 0;
189                     dirExp(v) = dirObs+(3*%pi/4)+(rand()*%pi/2);
190                     if dirExp(v) > (2*%pi) then
191                         dirExp(v) = dirExp(v)-(2*%pi);
192                     end
193                 end
194             end
195
196             for z=1:Tv
197                 if v <> z then //It must not be the same
```

```
198                     //Angle of the individual with respect to other
199                     //members of the swarm
200                     ang = atan(C(z,2)-C(v,2),C(z,1)-C(v,1))
201                     if ang<0 then
202                         ang=ang+(2*%pi);
203                     elseif ang > (2*%pi) then
204                         ang=ang-(2*%pi);
205                     end
206
207                     //Calculation of angles of repulsion and attraction with
208                     //respect to other individuals
209                     Beta = ang - C(v,4);
210                     if  Beta < 0 then
211                         Beta = Beta + (2*%pi);
212                     end
213
214                     Gama = C(v,4) - ang;
215                     if  Gama < 0 then
216                         Gama = Gama + (2*%pi);
217                     end
218
219                     if Gama<Beta then
220                         Delta = Gama;
221                     else
222                         Delta = Beta;
223                     end
224
225                     //Calculation of the repulsion distance with respect
226                     //to other individuals
227                     if Delta < rangR/2 then
228                         dr = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
229                     else
230                         dr = %inf;
231                     end
232
233                     //Calculation of the orientation distance with respect to objec
```

```
234                     if Delta < rangO/2 then
235                         dor = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
236                     else
237                         dor = %inf;
238                     end
239
240                      //Calculation of the attraction distance with respect
241                      //to other individuals
242                     if Delta < rangA/2 then
243                         da = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
244                     else
245                         da = %inf;
246                     end
247
248                     //Number of individuals detected in the radius of repulsion,
249                     //orientation and attraction
250                     if  dr <= dR then
251                         ElemRx = [ElemRx; cos(ang)];
252                         ElemRy = [ElemRy; sin(ang)];
253                         DetR = DetR+1; //A vehicle is detected (repulsion)
254                     end
255
256                     if  dor > dR & dor <= dO & DetR == 0 & DetA == 0 then
257                         ElemOx = [ElemOx; cos(C(z,4))];
258                         ElemOy = [ElemOy; sin(C(z,4))];
259                         DetO = DetO+1; //A vehicle is detected (orientation)
260                     end
261
262                     if  da > dO & da <= dA & DetR == 0 then
263                         ElemAx = [ElemAx; cos(ang)];
264                         ElemAy = [ElemAy; sin(ang)];
265                         DetA = DetA+1; //A vehicle is detected (attraction)
266                     end
267                 end
268             end //of Z
269
```

```
270             //Influence prey angle
271             if v > Preys then
272                 for p=1 : Preys
273                     if iPreys(p,3) < collisions then
274                         dirP(p) = atan(iPreys(p,2)-C(v,2),iPreys(p,1)-C(v,1));
275                         if dirP(p)<0 then
276                             dirP(p)=dirP(p)+(2*%pi);
277                         elseif dirP(p) > (2*%pi) then
278                             dirP(p)=dirP(p)-(2*%pi);
279                         end
280
281                         //Calculation of influence angle
282                         BetaI = dirP(p) - C(v,4);
283                             if  BetaI < 0 then
284                                 BetaI = BetaI + (2*%pi);
285                             end
286
287                         GamaI = C(v,4) - dirP(p);
288                         if  GamaI < 0 then
289                             GamaI = GamaI + (2*%pi);
290                         end
291
292                         if GamaI<BetaI then
293                             DeltaI = GamaI;
294                         else
295                             DeltaI = BetaI;
296                         end
297
298                         //Calculation of the influence distance
299                         if DeltaI<rangI then
300                             di(p)=sqrt((iPreys(p,1)-C(v,1))^2+(iPreys(p,2)- ...
301                             ... C(v,2))^2);
302                         else
303                             di(p) = %inf;
304                         end
305                     else
```

```
306                         di(p) = %inf;
307                     end
308                 end
309
310                 minPrey = find(di==min(di))
311                 if  di(minPrey) <= dInf then
312                     ElemIx = [ElemIx; cos(dirP(minPrey))];
313                     ElemIy = [ElemIy; sin(dirP(minPrey))];
314                     DetI = DetI+1; //A vehicle is detected (repulsion)
315                 end
316
317                 Ilim = 0.05;
318                 if (di(minPrey)>=Ilim) then
319                     di(minPrey) = di(minPrey);
320                 else
321                     di(minPrey) = Ilim;
322                 end
323
324                 [norm_val]=normalize(di(minPrey),Ilim,dInf)
325                 [val]=denormalize(norm_val,1.2,2)
326                 uI = val;
327             end
328
329             uO=uI; uR=1; uA=(uI+2)/2;
330             uDO=1.2; uDR=1.2; uDA=(uDO+1.2)/2;
331
332             //Average of detected elements
333             dirR = atan(-sum(ElemRy),-sum(ElemRx));
334             if dirR < 0 then
335                 dirR = dirR+(2*%pi);
336             end
337
338             dirO = atan(sum(ElemOy),sum(ElemOx));
339             if dirO < 0 then
340                 dirO = dirO+(2*%pi);
341             end
```

```
342
343             dirA = atan(sum(ElemAy),sum(ElemAx));
344             if dirA < 0 then
345                 dirA = dirA+(2*%pi);
346             end
347
348             dirI = atan(sum(ElemIy),sum(ElemIx));
349             if dirI < 0 then
350                 dirI = dirI+(2*%pi);
351             end
352
353             //Behavior Policies
354             //Repulsion rules
355             if DetR > 0 then
356                 if (v > Preys) then
357                     ud(v,:) = zeros(1,2)+uR;
358                     xT = cos(dirR);
359                     yT = sin(dirR);
360                     C(v,4) = atan(yT,xT);
361                     dirExp(v) = dirR;
362                 else
363                     if iPreys(v,3) < collisions
364                         if count(v) < 6 & m > 10 then
365                             temp(v) =  temp(v) + 1;
366                         else
367                             temp(v) = 0;
368                             count(v) = 0;
369                         end
370                         ud(v,:) = zeros(1,2)+uR;
371                         xT = cos(dirR);
372                         yT = sin(dirR);
373                         C(v,4) = atan(yT,xT);
374                         dirExp(v) = dirR;
375                     end
376                 end
377             end
```

```
378             //Collisions by prey
379             if temp(v) >= 1 then
380                 count(v) = count(v) + 1;
381             end
382             if temp(v) == collisions then
383                 iPreys(v,3) = collisions;
384             end
385             cp =  mean(iPreys(:,3));
386
387             //Influence rules
388             if v > Preys then
389                 if (DetI>0 & DetR==0) then
390                     if DetA == 0 then
391                         ud(v,:) = zeros(1,2)+uO;
392                         xT = cos(dirI);
393                         yT = sin(dirI);
394                         C(v,4) = atan(yT,xT);
395                         dirExp(v) = dirI;
396                     else
397                         ud(v,:) = zeros(1,2)+uO;
398                         xT = cos(dirI)*0.5 + cos(dirA);
399                         yT = sin(dirI)*0.5 + sin(dirA);
400                         C(v,4) = atan(yT,xT);
401                         dirExp(v) = dirI;
402                     end
403                 end
404             end
405
406             //Attraction rules
407             if (DetA>0 & DetR==0 & DetO==0 & DetI==0 ) then
408                 if (v > Preys) then
409                     ud(v,:) = zeros(1,2)+uA;
410                     xT = cos(dirA);
411                     yT = sin(dirA);
412                     C(v,4) = atan(yT,xT);
413                     dirExp(v) = dirA;
```

```
414                 else
415                     if iPreys(v,3) < collisions then
416                         ud(v,:) = zeros(1,2)+uDA;
417                         xT = cos(dirA);
418                         yT = sin(dirA);
419                         C(v,4) = atan(yT,xT);
420                         dirExp(v) = dirA;
421                     end
422                 end
423             end
424
425             //Orientation rules
426             if (DetO>0 & DetR==0 & DetA==0 & DetI==0) then
427                 if (v > Preys) then
428                     ud(v,:) = zeros(1,2)+uO;
429                     xT = cos(dirO);
430                     yT = sin(dirO);
431                     C(v,4) = atan(yT,xT);
432                     dirExp(v) = dirO;
433                 else
434                     if iPreys(v,3) < collisions then
435                         ud(v,:) = zeros(1,2)+uDO;
436                         xT = cos(dirO);
437                         yT = sin(dirO);
438                         C(v,4) = atan(yT,xT);
439                     dirExp(v) = dirO;
440                     end
441                 end
442             end
443
444             //Orientation-Attraction rules
445             if (DetO>0 & DetA>0 & DetR==0 & DetI==0) then
446                 if (v > Preys) then
447                     ud(v,:) = zeros(1,2)+uO;
448                     xT = 0.5*cos(dirO)+0.5*cos(dirA);
449                     yT = 0.5*sin(dirO)+0.5*sin(dirA);
```

```
450                     C(v,4) = atan(yT,xT);
451                     dirExp(v) = dirO;
452                 else
453                     if iPreys(v,3) < collisions then
454                         ud(v,:) = zeros(1,2)+uDO;
455                         xT = 0.5*cos(dirO)+0.5*cos(dirA);
456                         yT = 0.5*sin(dirO)+0.5*sin(dirA);
457                         C(v,4) = atan(yT,xT);
458                     dirExp(v) = dirO;
459                     end
460                 end
461             end
462
463             //Out of range
464             if (DetO==0 & DetR==0 & DetA==0 & DetI==0) then
465                 if (v > Preys) then
466                     ud(v,:) = zeros(1,2)+uO;
467                     xT = cos(dirExp(v));
468                     yT = sin(dirExp(v));
469                     C(v,4) = atan(yT,xT);
470                 else
471                     if iPreys(v,3) < collisions then
472                         ud(v,:) = zeros(1,2)+uDO;
473                         xT = cos(dirExp(v));
474                         yT = sin(dirExp(v));
475                         C(v,4) = atan(yT,xT);
476                     end
477                 end
478             end
479
480             //ReportsPredators
481             if v > Preys then
482                 ReportPredators(m,v-Preys,1) = C(v,1);
483                 ReportPredators(m,v-Preys,2) = C(v,2);
484                 ReportPredators(m,v-Preys,3) = C(v,4);
485                 ReportPredators(m,v-Preys,4) = C(v,5);
```

```
486             else
487                 ReportPreys(m,v,1) = C(v,1);
488                 ReportPreys(m,v,2) = C(v,2);
489                 ReportPreys(m,v,3) = C(v,4);
490                 ReportPreys(m,v,4) = C(v,5);
491             end
492
493             Dm(m,v) = C(v,3);
494             Dr(v) = max(Dm(:,v))
495
496             Cpast = C(v,:);
497             [cs,t] = mov(ud(v,:)',C(v,:)'); //Vehicle movement
498             [r, c] = size(cs);
499             C(v,:) = cs(:,c)'; //Next initial condition, actual condition
500
501             if (v <= Preys) then
502                 if iPreys(v,3) >= collisions then
503                     C(v,:) = Cpast;
504                 end
505             end
506
507             for i=1:Preys
508                 iPreys(i,1) = C(i,1)
509                 iPreys(i,2) = C(i,2)
510             end
511
512             if C(v,1) < 0 | C(v,1) > lims | C(v,2) < 0 | C(v,2) > lims then
513                 C(v,:) = Cpast;
514             end
515
516             //This avoids an infinite increment of radians
517             if C(v,4) > (2*%pi) then
518                 C(v,4) = C(v,4)-(2*%pi);
519             elseif C(v,4) < 0 then
520                 C(v,4) = C(v,4)+(2*%pi);
521             end
```

```
522     end
523
524     drawlater();
525     clf();
526     scatter([0,0],[0,0]);
527     d = 0.18;          //Size of arrow
528     RadRobot = 0.06; //Radius of robot
529
530     for v=1:(Tv)
531         if v <= Preys then
532             t1 = C(v,4)-acos(RadRobot/d);
533             t2 = C(v,4)+acos(RadRobot/d);
534             xf = [C(v,1),C(v,1)+RadRobot*cos(t1),C(v,1)+d*cos(C(v,4)),C(v,1)+ ...
535             ... RadRobot*cos(t2)];
536             yf = [C(v,2),C(v,2)+RadRobot*sin(t1),C(v,2)+d*sin(C(v,4)),C(v,2)+ ...
537             ... RadRobot*sin(t2)];
538             xfpoly(xf,yf,[-1]);
539             if iPreys(v,3) < collisions then
540                 gce().background = 7;
541             else
542                 gce().background = 3;
543             end
544             xfarc(C(v,1)-RadRobot,C(v,2)+RadRobot,RadRobot*2,RadRobot*2,0,360*64);
545             if iPreys(v,3) < collisions then
546                 gce().background = 7;
547             else
548                 gce().background = 3;
549             end
550         else
551             t1 = C(v,4)-acos(RadRobot/d);
552             t2 = C(v,4)+acos(RadRobot/d);
553             xf = [C(v,1),C(v,1)+RadRobot*cos(t1),C(v,1)+d*cos(C(v,4)),C(v,1)+ ...
554             ... RadRobot*cos(t2)];
555             yf = [C(v,2),C(v,2)+RadRobot*sin(t1),C(v,2)+d*sin(C(v,4)),C(v,2)+ ...
556             ... RadRobot*sin(t2)];
557             xfpoly(xf,yf,[-1]);
```

```
558                 xfarc(C(v,1)-RadRobot,C(v,2)+RadRobot,RadRobot*2,RadRobot*2,0,360*64);
559             end
560         end
561
562         a = get("current_axes");
563         a.data_bounds = [0,0; lims,lims];
564         a.tight_limits = ["on","on"];
565         drawnow();
566     end //of m
567
568     cx = sum(ReportPredators(m,:,1))/Predators;
569     cy = sum(ReportPredators(m,:,2))/Predators;
570     dy=sqrt(sum((ReportPredators(m,:,2)-cy).^2)/Predators);
571     dx=sqrt(sum((ReportPredators(m,:,1)-cx).^2)/Predators);
572     area = (%pi*dy*dx)
573     set(gca(),"auto_clear","off");
574     plot(cx,cy,'rx');
575     xset("color",1);
576     xarc(cx-(dx*1.3),cy+(dy*1.3),(dx*1.3)*2,(dy*1.3)*2,0,360*64);
577     //title('k=250, con influencia');
578     set(gca(),"auto_clear","on");
579     xlabel('x-axis','Fontsize',3);
580     ylabel('y-axis','Fontsize',3);
581     res(1,1)=cx;
582     res(2,1)=cy;
583     res(3,1)=dy;
584     res(4,1)=dx;
585     res(5,1)=area;
586     endfunction
```

C.2 Tarea de transporte de objetos

```
1
2   //*********************************************************************
```

```
3  //Estudio de comportamintos emergentes en enjambres de robots basados
4  //factores de influencia
5  //DIE Mecatronica
6  //UANL - FIME
7  //M.Sc. Erick de Jesus Ordaz Rivas  M.Sc. Luis Angel Marquez Vega
8  //*********************************************************************
9
10 function [norm_val]=normalize(val,l_min,l_max)
11     norm_val = (val-l_min)/((l_max-l_min)+%eps);
12     if norm_val > 1 then
13         norm_val = 1;
14     elseif norm_val < 0 then
15         norm_val = 0;
16     end
17 endfunction
18
19 function [val]=denormalize(norm_val,l_min,l_max)
20     val = norm_val*(l_max-l_min)+l_min;
21     if val > l_max then
22         val = l_max;
23     elseif val < l_min then
24         val = l_min;
25     end
26 endfunction
27
28 function [dxdt]=DynamicModel(t,c)
29     //c(1,:) x position
30     //c(2,:) y position
31     //c(3,:) movement
32     //c(4,:) orientation
33     //c(5,:) speed
34     //c(6,:) angular spee
35
36     //Parameters
37     m = .38;     //mass
38     Ip = 0.005; //inertia moment
```

```
39      d = 0.02;    //distance centroid to axis wheel
40      r = 0.03;    //radio wheel
41      R = 0.05;    //distance wheel-center
42
43      M = [m 0 ; 0 (Ip+m*d^2)];          //Inertial matrix
44      H = [-m*d*c(6)^2 ; m*d*c(5)*c(6)]; //Coriolis and centrifugal forces
45      B = [1/r 1/r ; R/r -R/r];          //Conversion torque-wheel-movil force
46      A = [r/2 r/2 ; r/(2*R) -r/(2*R)];  //Speed ratio
47      F = [0; 0];                        //Friction
48
49      Ts = [.434 0 ; 0 .434];
50      Ks = [2.745 0 ; 0 2.745];
51      Kl = [1460.2705 0 ; 0 1460.2705];
52
53      dxdt = [[cos(c(4)) -d*sin(c(4)); sin(c(4)) d*cos(c(4))]*[c(5); c(6)]; ...
54      ... c(5); c(6); inv(M + B*inv(Kl)*Ts*inv(A)) * (B*inv(Kl)*Ks*u - ...
55      ... (H + F + B*inv(Kl)*inv(A)*[c(5); c(6)]))];
56  endfunction
57
58  function [c,t]= Mov(u,ci)
59      ti = 0;                //Initial time
60      tf = 1;                //Final time
61      tspan = [ti:0.1:tf]; //Time vector
62      c = ode(ci,ti,tspan,DynamicModel);
63      t = tspan;
64  endfunction
65
66  function [Report,Oend,Oini,st,Tr,Dr]=SimMov2(Ob,Tv,dRU,dOU,dAU)
67      st = 0;                          //Iterations
68      Report = zeros(st,Tv,4);         //Report
69      Tr = zeros(Tv,3);                //Delivery time,Search time,collected objects
70      C = zeros(Tv,6);                 //vehicle states
71      rm = rand(1,1,'normal')*0.01;    //White noise in DC motors (1%)
72      uO = 0; uR = 0; uA = 0; uI = 0;//Initial speeds (voltages)
73      ud = zeros(Tv,2)+uO;             //Desired speed (voltage)
74      lims = 10;                       //Test area limits
```

```
75      dNest = 4;                          //Maximum distance of influence (nest)
76      dObBox = 3;                         //Maximum distance of influence (Objects box)
77
78      dR = 0.075+dRU;
79      dO = 0.075+dOU;
80      dA = 0.075+dAU;
81
82      Gripc = zeros(Tv,1); //Open grip
83      NestFull = Ob;       //Nest full (End task)
84
85      //Objects box
86      centerBox = 0.75;
87      limsBox = 0.2;
88      ObBox = [centerBox*lims,centerBox*lims];
89      ObBoxOn = zeros(Tv,1);
90
91      Objects = zeros(Ob,2);     //Objects location
92      if Ob == 0 then
93          Object = zeros(1,1)    //Objects vector
94          gov = zeros(1,1)       //Objects gripped by vehicle
95      else
96          Object = zeros(Ob,1); //Objects vector
97          gov = zeros(Ob,1)      //Objects gripped by vehicle
98      end
99
100     //Random objects position
101     for o = 1:Ob
102         Obrand1 = denormalize(rand(1,'uniform'),centerBox-(limsBox/2),
103         centerBox+(limsBox/2));
104         Obrand2 = denormalize(rand(1,'uniform'),centerBox-(limsBox/2),
105         centerBox+(limsBox/2));
106         Objects(o,1) = [lims*Obrand1()];
107         Objects(o,2) = [lims*Obrand2()];
108         Oini(o,1) = Objects(o,1);       //Initial position of object on X axis
109         Oini(o,2) = Objects(o,2);       //Initial position of object on Y axis
110         Object(o,1) = 1;
```

```
111     end
112
113     //Nest
114     limsNest = 0.2;
115     dot = zeros(1,2)+lims*(limsNest/2); //Nest dot
116     Nest = [dot(1,1),dot(1,2)];         //Nest location (Dotted line)
117     NestOn = zeros(Tv,1);               //Influence of nest activated for vehicle
118
119     for v = 1:Tv                        //Initialize vehicles
120         if v == 1 then
121             C(v,1) = lims*denormalize(rand(1,'uniform'),0,limsNest);
122             C(v,2) = lims*denormalize(rand(1,'uniform'),0,limsNest);
123         else
124             while %T
125                 safe = 0;
126                 C(v,1) = lims*denormalize(rand(1,'uniform'),0,0.2);
127                 C(v,2) = lims*denormalize(rand(1,'uniform'),0,0.2);
128                 for j = 1:v-1
129                     dother = sqrt((C(v,1)-C(j,1))^2 + (C(v,2)-C(j,2))^2);
130                     if dother > 0.3 then
131                         safe = safe + 1;
132                     end
133                 end
134                 if safe == j then
135                     break;
136                 end
137             end
138         end
139         C(v,3) = 0;
140         C(v,4) = denormalize(rand(1,'uniform'),0,2*%pi);
141         C(v,5) = 0;
142         C(v,6) = 0;
143     end
144
145     dirExp = C(:,4);
146     explore = zeros(1,Tv);
```

```
147
148     /*scf();
149     f = get("current_figure");
150     f.BackgroundColor = [1 1 1];
151     f.figure_position = [10,10];
152     f.figure_size=[700,800];*/
153
154     //Finish task when nest is full
155     while NestFull <> 0
156
157         st = st +1; //Iteration counter
158
159         for v = 1:Tv
160
161             //Elements detected
162             DetR = 0;       //Repulsion
163             DetO = 0;       //Orientation
164             DetA = 0;       //Attraction
165             DetN = 0;       //Nest and influence (Objects)
166             DetObBox = 0; //Object Box
167             ElemRx = [];    //Elements in repulsion
168             ElemOx = [];    //Elements in orientation
169             ElemAx = [];    //Elements in attraction
170             ElemRy = [];    //Elements in repulsion
171             ElemOy = [];    //Elements in orientation
172             ElemAy = [];    //Elements in attraction
173
174             //Detection range for repulsion, orientation, attraction, influence,
175             //nest and objects
176             rangR = 5.75959;
177             rangO = 2.44346
178             rangA = 0.523599;
179             rangI = 0.523599;
180             rangN = 2.44346;
181             rangOB = 2.44346;
182
```

```
183             //Verify each sensor for repulsion of walls
184             for i = 1:5
185                 if i == 1 then
186                     dirObs = C(v,4)-3.8397244;
187                 else
188                     dirObs = dirObs+1.9198622;
189                 end
190                 if dirObs < 0 then
191                     dirObs = dirObs+(2*%pi);
192                 elseif dirObs > (2*%pi) then
193                     dirObs = dirObs-(2*%pi);
194                 end
195                 Dir = [cos(dirObs),sin(dirObs)];
196                 limitX = C(v,1) + (Dir(1)*dR);
197                 limitY = C(v,2) + (Dir(2)*dR);
198
199                 //Resulting direction due exploration
200                 if limitX > lims | limitX < 0 | limitY > lims | limitY < 0 then
201                     explore(v) = 0;
202                     dirExp(v) = dirObs+(3*%pi/4)+(rand()*%pi/2);
203                     if dirExp(v) > (2*%pi) then
204                         dirExp(v) = dirExp(v)-(2*%pi);
205                     end
206                 end
207             end
208
209             //Resulting direction due object box
210             dirOb = atan(ObBox(1,2)-C(v,2),ObBox(1,1)-C(v,1));
211             if dirOb < 0 then
212                 dirOb = dirOb+(2*%pi);
213             elseif dirOb > (2*%pi) then
214                 dirOb = dirOb-(2*%pi);
215             end
216
217             //Calculation of influence angles by object box
218             BetaOB = dirOb-C(v,4);
```

```
219             if  BetaOB < 0 then
220                 BetaOB = BetaOB+(2*%pi);
221             end
222             GamaOB = C(v,4)-dirOb;
223             if  GamaOB < 0 then
224                 GamaOB = GamaOB+(2*%pi);
225             end
226             if GamaOB < BetaOB then
227                 DeltaOB = GamaOB;
228             else
229                 DeltaOB = BetaOB;
230             end
231
232             //Calculated distance between the robots and the object zone
233             if DeltaOB < rangOB/2 then
234                 dOB = sqrt(((C(v,1)-ObBox(1,1))^2)+((C(v,2)-ObBox(1,2))^2));
235             else
236                 dOB = %inf;
237             end
238
239             [norm_valOB] = normalize(dOB,0,dObBox)  //normalize distance of
240             //influence by nest
241             [valOB] = denormalize(norm_valOB,1,2.8) //desnormalize distance of
242             //influence by nest
243             uI = valOB;
244
245             for z = 1:Tv
246                 if v <> z then //it must not be the same
247
248                     //Angle of the individual with respect to other
249                     // members of the swarm
250                     ang = atan(C(z,2)-C(v,2),C(z,1)-C(v,1))
251                     if ang < 0 then
252                         ang = ang+(2*%pi);
253                     elseif ang > (2*%pi) then
254                         ang = ang-(2*%pi);
```

```
255                     end
256
257                     //Calculation of angles of repulsion and attraction with
258                     //respect to other individuals
259                     Beta = ang-C(v,4);
260                     if Beta < 0 then
261                         Beta = Beta+(2*%pi);
262                     end
263                     Gama = C(v,4)-ang;
264                     if Gama < 0 then
265                         Gama = Gama+(2*%pi);
266                     end
267                     if Gama < Beta then
268                         Delta = Gama;
269                     else
270                         Delta = Beta;
271                     end
272
273                     //Calculation of the repulsion distance with respect
274                     //to other individuals
275                     if Delta < rangR/2 then
276                         dr = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
277                     else
278                         dr = %inf;
279                     end
280
281                     //Calculation of the orientation distance with respect
282                     //to objects
283                     if Delta < rangO/2 then
284                         dor = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
285                     else
286                         dor = %inf;
287                     end
288
289                     //Calculation of the attraction distance with respect
290                     //to other individuals
```

```
291                     if Delta < rangA/2 then
292                         da = sqrt((C(v,1)-C(z,1))^2+(C(v,2)-C(z,2))^2);
293                     else
294                         da = %inf;
295                     end
296
297                     //Number of individuals detected in the radius of repulsion,
298                     //orientation and attraction
299                     if  dr <= dR then
300                         ElemRx = [ElemRx; cos(ang)];
301                         ElemRy = [ElemRy; sin(ang)];
302                         DetR = DetR+1; //A vehicle is detected (repulsion)
303                     end
304                     if  dor > dR & dor <= dO & DetR == 0 & DetA == 0 then
305                         ElemOx = [ElemOx; cos(C(z,4))];
306                         ElemOy = [ElemOy; sin(C(z,4))];
307                         DetO = DetO+1; //A vehicle is detected (orientation)
308                     end
309                     if  da > dO & da <= dA & DetR == 0 then
310                         ElemAx = [ElemAx; cos(ang)];
311                         ElemAy = [ElemAy; sin(ang)];
312                         DetA = DetA+1; //A vehicle is detected (attraction)
313                     end
314
315                 end
316             end //of z
317
318             for o = 1:Ob
319                 //Search object
320                 if Object(o,1) == 1 & Gripc(v,1) == 0 then
321
322                     //Angle of influence (respect to objects)
323                     angI = atan(Objects(o,2)-C(v,2),Objects(o,1)-C(v,1));
324                     if angI < 0 then
325                         angI = angI+(2*%pi);
326                     elseif angI > (2*%pi) then
```

```
327                     angI = angI-(2*%pi);
328                 end
329
330                 //Calculation of influence angles (respect to objects)
331                 BetaI = angI-C(v,4);
332                 if  BetaI < 0 then
333                     BetaI = BetaI+(2*%pi);
334                 end
335                 GamaI = C(v,4)-angI;
336                 if  GamaI < 0 then
337                     GamaI = GamaI+(2*%pi);
338                 end
339                 if GamaI < BetaI then
340                     DeltaI = GamaI;
341                 else
342                     DeltaI = BetaI;
343                 end
344
345                 //Noise for distance value in influence
346                 //(standard distribution) of 5%
347                 ds1 = 0;
348                 ds2 = 0;
349                 for i = 1:12
350                     ds1 = ds1+rand(1,1,'normal');
351                 end
352                 for j = 1:6
353                     ds2 = ds2+rand(1,1,'normal');
354                 end
355                 rds= (ds1-ds2)*0.05;
356
357                 //Calculated influence distance (respect to objects)
358                 if DeltaI < rangI/2 then
359                     di = sqrt((C(v,1)-Objects(o,1))^2+((C(v,2)- ...
360                     ... Objects(o,2))^2))+rds;
361                 else
362                     di = %inf;
```

```
363                     end
364
365                     //Distance between objects and robot
366                     Ilim = 0.02;
367                     if di <= Ilim then
368                         NestOn(v) = 1;
369                         gov(o) = v;
370                         Tr(v,3) = Tr(v,3)+1;
371                         Gripc(v,1) = 1;  //Close grip
372                         Object(o,1) = 0; //Object taken by robot
373                     end
374
375                 end
376
377                 //Nest
378                 if Object(o,1) == 0 & NestOn(v) == 1 then
379
380                     //Angle respect to nest
381                     dirNest = atan(Nest(1,2)-C(v,2),Nest(1,1)-C(v,1));
382                     if dirNest < 0 then
383                         dirNest = dirNest+(2*%pi);
384                     elseif dirNest > (2*%pi) then
385                         dirNest = dirNest-(2*%pi);
386                     end
387
388                     //Calculation of influence angles
389                     BetaN = dirNest-C(v,4);
390                     if BetaN < 0 then
391                         BetaN = BetaN+(2*%pi);
392                     end
393                     GamaN = C(v,4)-dirNest;
394                     if GamaN < 0 then
395                         GamaN = GamaN+(2*%pi);
396                     end
397                     if GamaN < BetaN then
398                         DeltaN = GamaN;
```

```
399                     else
400                         DeltaN = BetaN;
401                     end
402
403                     //Calculated influence distance (respect to nest)
404                     if DeltaN < rangN/2 then
405                         dN = sqrt((Nest(1,1)-C(v,1))^2+(Nest(1,2)-C(v,2))^2);
406                     else
407                         dN = %inf;
408                     end
409
410                     //Distance between nest and robot
411                     Nlim = 0.05;
412                     if dN < Nlim then
413                         Objects(o,1) = Nest(1,1);
414                         Objects(o,2) = Nest(1,2);
415                         dot(1,1) = dot(1,1) + denormalize(rand(1,'uniform'),-0.1,0.
416                         dot(1,2) = dot(1,2) + denormalize(rand(1,'uniform'),-0.1,0.
417                         Nest(1,1) = dot(1,1);
418                         Nest(1,2) = dot(1,2);
419                         NestFull = NestFull - 1;
420                         Gripc(v,1) = 0;
421                         NestOn(v) = 0;
422                     end
423
424                     [norm_val] = normalize(dN,Nlim,dNest) //normalize distance of
425                     //influence by nest
426                     [valN] = denormalize(norm_val,1.2,2.4) //desnormalize
427                     //distance of influence by nest
428                     uI = valN;
429
430                     if Gripc(gov(o,1),1) == 1 then
431                         Objects(o,1) = C(gov(o,1),1);
432                         Objects(o,2) = C(gov(o,1),2);
433                     end
434                 end
```

```
435
436             end //of o
437
438             uR = 1.2;
439             uO = uI;
440             uA = (uI+2.4);
441
442             //Average of detected elements
443             dirR = atan(-sum(ElemRy),-sum(ElemRx));
444             if dirR < 0 then
445                 dirR = dirR+(2*%pi);
446             end
447
448             dirO = atan(sum(ElemOy),sum(ElemOx));
449             if dirO < 0 then
450                 dirO = dirO+(2*%pi);
451             end
452
453             dirA = atan(sum(ElemAy),sum(ElemAx));
454             if dirA < 0 then
455                 dirA = dirA+(2*%pi);
456             end
457
458             //Behavior Policies
459             //Repulsion behavior
460             if DetR > 0 then
461                 if NestOn(v) == 0 then
462                     if dOB < dObBox then
463                         ud(v,:) = zeros(1,2)+uR+rm;
464                         xT = 0.8*cos(dirR)+0.2*cos(dirOb);
465                         yT = 0.8*sin(dirR)+0.2*sin(dirOb);
466                         C(v,4) = atan(yT,xT);
467                         //dirExp(v) = dirR;
468                         explore(v) = 1;
469                     else
470                         ud(v,:) = zeros(1,2)+uR+rm;
```

```
471                         xT = cos(dirR);
472                         yT = sin(dirR);
473                         C(v,4) = atan(yT,xT);
474                         dirExp(v) = dirR;
475                     end
476                 else
477                     if dN < dNest then
478                         ud(v,:) = zeros(1,2)+uR+rm;
479                         xT = 0.8*cos(dirR)+0.2*cos(dirNest);
480                         yT = 0.8*sin(dirR)+0.2*sin(dirNest);
481                         C(v,4) = atan(yT,xT);
482                     else
483                         ud(v,:) = zeros(1,2)+uR+rm;
484                         xT = cos(dirR);
485                         yT = sin(dirR);
486                         C(v,4) = atan(yT,xT);
487                         dirExp(v) = dirR;
488                     end
489                 end
490             end
491
492             //Orientation of vehicles
493             if DetR == 0 & DetO > 0 & DetA == 0 then
494                 if NestOn(v) == 0 then
495                     if dOB < dObBox then
496                         ud(v,:) = zeros(1,2)+uO+rm;
497                         xT = cos(dirOb);
498                         yT = sin(dirOb);
499                         C(v,4) = atan(yT,xT);
500                         explore(v) = 1;
501                     else
502                         ud(v,:) = zeros(1,2)+uO+rm;
503                         xT = cos(dirO);
504                         yT = sin(dirO);
505                         C(v,4) = atan(yT,xT);
506                         dirExp(v) = dirO;
```

```
507                     end
508                 else
509                     if dN < dNest then
510                         ud(v,:) = zeros(1,2)+uO+rm;
511                         xT = cos(dirNest);
512                         yT = sin(dirNest);
513                         C(v,4) = atan(yT,xT);
514                     else
515                         ud(v,:) = zeros(1,2)+uO+rm;
516                         xT = cos(dirO);
517                         yT = sin(dirO);
518                         C(v,4) = atan(yT,xT);
519                         dirExp(v) = dirO;
520                     end
521                 end
522             end
523
524             //Attraction of vehicles
525             if DetR == 0 & DetO == 0 & DetA > 0 then
526                 if NestOn(v) == 0 then
527                     if dOB < dObBox then
528                         ud(v,:) = zeros(1,2)+uO+rm;
529                         xT = cos(dirOb);
530                         yT = sin(dirOb);
531                         C(v,4) = atan(yT,xT);
532                         explore(v) = 1;
533                     else
534                         ud(v,:) = zeros(1,2)+uA+rm;
535                         xT = cos(dirA);
536                         yT = sin(dirA);
537                         C(v,4) = atan(yT,xT);
538                         dirExp(v) = dirA;
539                     end
540                 else
541                     if dN < dNest then
542                         ud(v,:) = zeros(1,2)+uO+rm;
```

```
543                         xT = cos(dirNest);
544                         yT = sin(dirNest);
545                         C(v,4) = atan(yT,xT);
546                     else
547                         ud(v,:) = zeros(1,2)+uA+rm;
548                         xT = cos(dirA);
549                         yT = sin(dirA);
550                         C(v,4) = atan(yT,xT);
551                         dirExp(v) = dirA;
552                     end
553                 end
554             end
555
556             //Orientation and Attraction of vehicles
557             if DetR == 0 & DetO > 0 & DetA > 0 then
558                 if NestOn(v) == 0 then
559                     if dOB < dObBox then
560                         ud(v,:) = zeros(1,2)+uO+rm;
561                         xT = cos(dirOb);
562                         yT = sin(dirOb);
563                         C(v,4) = atan(yT,xT);
564                         explore(v) = 1;
565                     else
566                         ud(v,:) = zeros(1,2)+uO+rm;
567                         xT = 0.5*cos(dirO)+0.5*cos(dirA);
568                         yT = 0.5*cos(dirO)+0.5*sin(dirA);
569                         C(v,4) = atan(yT,xT);
570                         dirExp(v) = atan(sin(dirO)+sin(dirA),cos(dirO)+cos(dirA));
571                     end
572                 else
573                     if dN < dNest then
574                         ud(v,:) = zeros(1,2)+uO+rm;
575                         xT = cos(dirNest);
576                         yT = sin(dirNest);
577                         C(v,4) = atan(yT,xT);
578                     else
```

```
579                         ud(v,:) = zeros(1,2)+uO+rm;
580                         xT = 0.5*cos(dirO)+0.5*cos(dirA);
581                         yT = 0.5*cos(dirO)+0.5*sin(dirA);
582                         C(v,4) = atan(yT,xT);
583                         dirExp(v) = atan(sin(dirO)+sin(dirA),cos(dirO)+cos(dirA));
584                     end
585                 end
586             end
587
588             //Out of range
589             if DetR == 0 & DetO == 0 & DetA == 0 then
590                 if NestOn(v) == 0 then
591                     if dOB < dObBox then
592                         ud(v,:) = zeros(1,2)+uO+rm;
593                         xT = cos(dirOb);
594                         yT = sin(dirOb);
595                         C(v,4) = atan(yT,xT);
596                         explore(v) = 1;
597                     else
598                         ud(v,:) = zeros(1,2)+uO+rm;
599                         xT = cos(dirExp(v));
600                         yT = sin(dirExp(v));
601                         C(v,4) = atan(yT,xT);
602                     end
603                 else
604                     if dN < dNest then
605                         ud(v,:) = zeros(1,2)+uO+rm;
606                         xT = cos(dirNest);
607                         yT = sin(dirNest);
608                         C(v,4) = atan(yT,xT);
609                     else
610                         ud(v,:) = zeros(1,2)+uO+rm;
611                         xT = cos(dirExp(v));
612                         yT = sin(dirExp(v));
613                         C(v,4) = atan(yT,xT);
614                     end
```

```
615                 end
616             end
617
618             if explore(v) == 1 & dOB > dObBox & rand(1,'uniform') < 0.1 then
619                 explore(v) = 0;
620                 dirExp(v) = dirExp(v)+(3*%pi/4)+(rand()*%pi/2);
621                 if dirExp(v) > (2*%pi) then
622                     dirExp(v) = dirExp(v)-(2*%pi);
623                 end
624             end
625
626             //Reports
627             Report(st,v,1) = C(v,1);
628             Report(st,v,2) = C(v,2);
629             Report(st,v,3) = C(v,3);
630             Report(st,v,4) = C(v,4);
631
632             Dm(st,v) = C(v,3);
633             Dr(v) = max(Dm(:,v))
634
635             Cpast = C(v,:);
636             [cs,t] = Mov(ud(v,:)',C(v,:)'); //Vehicle movement
637             [r, c] = size(cs);
638             C(v,:) = cs(:,c)'; //Next initial condition, actual condition
639
640             if C(v,1) < 0 | C(v,1) > lims | C(v,2) < 0 | C(v,2) > lims then
641                 C(v,:) = Cpast;
642             end
643
644             //This avoids an infinite increment of radians
645             if C(v,4) > (2*%pi) then
646                 C(v,4) = C(v,4)-(2*%pi);
647             elseif C(v,4) < 0 then
648                 C(v,4) = C(v,4)+(2*%pi);
649             end
650
```

```
651                 //Delivery time
652                 if Gripc(v,1) == 1 then //Close grip
653                     Tr(v,1) = Tr(v,1) + 1;
654                 end
655
656             end //of v
657
658             //animation(C,Tv, Objects ,Ob, lims , limsNest , centerBox , limsBox , dNest , dObBox);
659
660         end //of while
661
662         Tr(:,2) = st - Tr(:,1);
663         Oend(o,1) = Objects(o,1);
664         Oend(o,2) = Objects(o,2);
665
666 endfunction
667
668 function animation(C,Tv,Objects,Ob,lims,limsNest,centerBox,limsBox,dNest,dObBox)
669
670         drawlater();
671         clf();
672         scatter([1,1],[1,1]);
673         xrect(0,limsNest*lims,limsNest*lims,limsNest*lims);
674         gce().foreground = color("red");
675         xarc(lims*(limsNest/2)-dNest,lims*(limsNest/2)+dNest,2*dNest,2*dNest,0,360*64);
676         gce().foreground = color("red");
677         xrect(lims*(centerBox-(limsBox/2)),lims*(centerBox+(limsBox/2)), ...
678         ... lims*limsBox,lims*limsBox);
679         gce().foreground = color("blue");
680         xarc(lims*centerBox-2.5,lims*centerBox+2.5,2*2.5,2*2.5,0,360*64);
681         gce().foreground = color("blue");
682
683         d = 0.3;          //Size of arrow
684         RadRobot = 0.1; //Radius of robot
685         for v = 1:Tv
686             t1 = C(v,4)-acos(RadRobot/d);
```

```
687             t2 = C(v,4)+acos(RadRobot/d);
688             xf = [C(v,1),C(v,1)+RadRobot*cos(t1),C(v,1)+d*cos(C(v,4)),C(v,1)+ ...
689             ... RadRobot*cos(t2)];
690             yf = [C(v,2),C(v,2)+RadRobot*sin(t1),C(v,2)+d*sin(C(v,4)),C(v,2)+ ...
691             ... RadRobot*sin(t2)];
692             xfpoly(xf,yf,[-1]);
693             xfarc(C(v,1)-RadRobot,C(v,2)+RadRobot,RadRobot*2,RadRobot*2,0,360*64);
694         end
695
696         RadOb = 0.1; //Radius of object
697         for o = 1:Ob
698             xfarc(Objects(o,1)-RadOb,Objects(o,2)+RadOb,RadOb*2,RadOb*2,0,360*64);
699             gce().background = color("green");
700         end
701
702         a = get("current_axes");
703         a.data_bounds = [0,0; lims,lims];
704         a.tight_limits = ["on","on"];
705         drawnow();
706
707 endfunction
708
709 function [rst,rDr,rTr1,rTr2,Report]=replicas(Ob,Tv,dRU,dOU,dAU)
710         for i=1:3
711             [Report,Oend,Oini,st,Tr,Dr]=SimMov2(Ob,Tv,dRU,dOU,dAU)
712             Dr = mean(Dr);
713             Tr1 = mean(Tr(:,1));
714             Tr2 = mean(Tr(:,2));
715             for j=1:1
716                 rst(i,j) = st;
717                 rDr(i,j) = Dr;
718                 rTr1(i,j) = Tr1;
719                 rTr2(i,j) = Tr2;
720             end
721         end
722         prst = mean(rst);
```

```
723         prDr = mean(Dr);
724         prTr1 =mean(rTr1);
725         prTr2 =mean(rTr2);
726         Report = [prTr1 prTr2 prst prDr]
727 endfunction
```

Bibliografía

[1] Ian D. Couzin, Jens Krause, Richard James, Graeme D. Ruxton, and Nigel R. Franks. Collective Memory and Spatial Sorting in Animal Groups. *Journal of Theoretical Biology*, 2002(218):1–11, 2002.

[2] Belkacem Khaldi and Cherif Foudil. An overview of swarm robotics: Swarm intelligence applied to multi-robotics. *International Journal of Computer Applications*, 126(2):31–37, 09 2015.

[3] Kshitij Tiwari and Nak Young Chong. 12 - multi-robot systems: The cost of scalability. In Kshitij Tiwari and Nak Young Chong, editors, *Multi-robot Exploration for Environmental Monitoring*, pages 159 – 169. Academic Press, 2020.

[4] Zhiguo Shi, Jun Tu, Qiao Zhang, Lei Liu, and Junming Wei. A survey of swarm robotics system. In Ying Tan, Yuhui Shi, and Zhen Ji, editors, *Advances in Swarm Intelligence*, pages 564–572, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[5] Bing Zhu, Lihua Xie, Duo Han, Xiangyu Meng, and Rodney Teo. A survey on recent progress in control of swarm systems. *Science China Information Sciences*, 60(7):070201, Jun 2017.

[6] Eugene Kagan, Nir Shvalb, Shlomi Hacohen, and Alexander Novoselsky. *Multi-Robot Systems and Swarming*, chapter 9, pages 199–241. John Wiley & Sons, Ltd, 2019.

[7] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246, Sep 2014.

[8] Eliseo Ferrante, Ali Emre Turgut, Edgar Duéñez-Guzmán, Marco Dorigo, and Tom Wenseleers. Evolution of self-organized task specialization in robot swarms. *PLOS Computational Biology*, 11(8):1–21, 08 2015.

[9] Patricia Suárez, Andrés Iglesias, and Akemi Gálvez. Make robots be bats: specializing robotic swarms to the bat algorithm. *Swarm and Evolutionary Computation*, 44:113–129, 2019.

[10] Fabrício R. Inácio, Douglas G. Macharet, and Luiz Chaimowicz. Pso-based strategy for the segregation of heterogeneous robotic swarms. *Journal of Computational Science*, 31:86–94, 2019.

[11] Gerardo Beni. From swarm intelligence to swarm robotics. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, pages 1–9, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *From Natural to Artificial Swarm Intelligence*. Oxford University Press, Inc., USA, 1999.

[13] A. Eiben and Jim Smith. *Introduction To Evolutionary Computing*, volume 45. Springer, Berlin, Heidelberg, 01 2003.

[14] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 04 1992.

[15] Stephane Doncieux, Nicolas Bredeche, Jean-Baptiste Mouret, and Agoston E. (Gusz) Eiben. Evolutionary robotics: What, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015.

[16] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[17] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 26(1):29–41, 1996.

[18] Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39:459–471, 11 2007.

[19] Szymon Łukasik and Sławomir Żak. Firefly algorithm for continuous constrained optimization tasks. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, and Shyi-Ming Chen, editors, *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, pages 97–106, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[20] X. Yang and Suash Deb. Cuckoo search via lévy flights. In *2009 World Congress on Nature Biologically Inspired Computing (NaBIC)*, pages 210–214, 2009.

[21] Pinar Civicioglu. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219(15):8121 – 8144, 2013.

[22] Ying Tan and Zhong yang Zheng. Research advance in swarm robotics. *Defence Technology*, 9(1):18 – 39, 2013.

[23] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7, 04 2020.

[24] Scott Camazine, Nigel R. Franks, James Sneyd, Eric Bonabeau, Jean-Louis

Deneubourg, and Guy Theraula. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, USA, 2001.

[25] Vito Trianni. *Multi-Robot Systems, Swarm Robotics and Self-Organisation*, pages 23–46. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[26] V. V. Isaeva. Self-organization in biological systems. *Biology Bulletin*, 39(2):110–118, Apr 2012.

[27] Simon Garnier, Jacques Gautrais, and Guy Theraulaz. The biological principles of swarm intelligence. *Swarm Intelligence*, 1(1):3–31, Jun 2007.

[28] Chris Taylor, Colin Luzzi, and Cameron Nowzari. On the effects of collision avoidance on emergent swarm behavior. In *2020 American Control Conference (ACC)*, pages 931–936, 07 2020.

[29] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, pages 10–20, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[30] O. Linda and M. Manic. Fuzzy manual control of multi-robot system with built-in swarm behavior. In *2009 2nd Conference on Human System Interactions*, pages 4–9, 2009.

[31] Benjamin T. Fine and Dylan A. Shell. Unifying microscopic flocking motion models for virtual, robotic, and biological flock members. *Autonomous Robots*, 35(2):195–219, 2013.

[32] Christoph Moeslinger, Thomas Schmickl, and Karl Crailsheim. A minimalist flocking algorithm for swarm robots. In George Kampis, István Karsai, and Eörs Szathmáry, editors, *Advances in Artificial Life. Darwin Meets von Neumann*, pages 375–382, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[33] Levent Bayindir. A review of swarm robotics tasks. *Neurocomputing*, 172:292–321, 2016.

[34] Nadia Nedjah and Luneque Silva Junior. Review of methodologies and tasks in swarm robotics towards standardization. *Swarm and Evolutionary Computation*, 50:100565, 08 2019.

[35] Micael S. Couceiro, Patricia A. Vargas, Rui P. Rocha, and Nuno M.F. Ferreira. Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems*, 62(2):200 – 213, 2014.

[36] Y. U. Cao, A. S. Fukunaga, A. B. Kahng, and F. Meng. Cooperative mobile robotics: antecedents and directions. In *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, volume 1, pages 226–234 vol.1, 1995.

[37] A. S. Anoop and P. Kanakasabapathy. Review on swarm robotics platforms. In *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, pages 1–6, Dec 2017.

[38] Jan Carlo Barca and Y. Ahmet Sekercioglu. Swarm robotics reviewed. *Robotica*, 31(3):345–359, 2013.

[39] Frederick Ducatelle, Gianni A. Di Caro, and Luca M. Gambardella. Cooperative self-organization in a heterogeneous swarm robotic system. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, page 87–94, New York, NY, USA, 2010. Association for Computing Machinery.

[40] Heiko Hamann. *Scenarios of Swarm Robotics*, pages 65–93. Springer International Publishing, Cham, 2018.

[41] Simon Fong, Suash Deb, and Ankit Chaudhary. A review of metaheuristics in robotics. *Comput. Electr. Eng.*, 43(C):278–291, apr 2015.

[42] Luneque Silva Junior and Nadia Nedjah. Wave algorithm applied to collective navigation of robotic swarms. *Applied Soft Computing*, 57:698–707, 2017.

[43] Veysel Gazi and Kevin Passino. *Swarm Stability and Optimization*. Springer, Berlin, Heidelberg, 01 2011.

[44] Reza Olfati-Saber. Flocking for multi-agent dynamic systems: Algorithms and theory. *Automatic Control, IEEE Transactions on*, 51:401 – 420, 04 2006.

[45] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *AMC SIGGRAPH'87 Comput. Graph.*, 21(4):25–34, August 1987.

[46] Olaf Witkowski and Takashi Ikegami. Emergence of swarming behavior: Foraging agents evolve collective motion based on signaling. *PloS one*, 11:e0152756, 04 2016.

[47] Tamás Vicsek, András Czirók, Eshel Ben-Jacob, Inon Cohen, and Ofer Shochet. Novel type of phase transition in a system of self-driven particles. *Phys. Rev. Lett.*, 75:1226–1229, Aug 1995.

[48] Christopher Hartman and Bedrich Benes. Autonomous boids. *Computer Animation and Virtual Worlds*, 17(3-4):199–206, 2006.

[49] Felipe Cucker and Cristian Huepe. Flocking with informed agents. *MathematicS In Action [electronic only]*, 1, 01 2008.

[50] Housheng Su and Xiaofan Wang. Flocking of multi-agents with a virtual leader. *Automatic Control, IEEE Transactions on*, 54:293 – 307, 03 2009.

[51] Wenwu Yu, Guanrong Chen, and Ming Cao. Distributed leader-follower flocking control for multi-agent dynamical systems with time-varying velocities. *Systems & Control Letters*, 59:543–552, 01 2010.

[52] Ellips Masehian and Mitra Royan. Characteristics of and approaches to flocking in swarm robotics. *Applied Mechanics and Materials*, 841:240–249, 06 2016.

[53] Ali Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Sahin. Self-organized flocking with a mobile robot swarm. In *Swarm Intelligence*, volume 1, pages 39–46, 01 2008.

[54] M. R. Pac, A. M. Erkmen, and Ismet Erkmen. Control of robotic swarm behaviors based on smoothed particle hydrodynamics. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4194–4200, 2007.

[55] L. C. A. Pimenta, G. A. S. Pereira, N. Michael, R. C. Mesquita, M. M. Bosque, L. Chaimowicz, and V. Kumar. Swarm coordination based on smoothed particle hydrodynamics technique. *IEEE Transactions on Robotics*, 29(2):383–399, 2013.

[56] Z. Kira and M. A. Potter. Exerting human control over decentralized robot swarms. In *2009 4th International Conference on Autonomous Robots and Agents*, pages 566–571, 2009.

[57] M. A. Goodrich, B. Pendleton, P. B. Sujit, and J. Pinto. Toward human interaction with bio-inspired robot teams. In *2011 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2859–2864, 2011.

[58] S. Jung and M. A. Goodrich. Multi-robot perimeter-shaping through mediator-based swarm control. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6, 2013.

[59] Colin Torney, Andrew Berdahl, and Iain Couzin. Signalling and the evolution of cooperative foraging in dynamic environments. *PLoS computational biology*, 7:e1002194, 09 2011.

[60] S. Bashyal and G. K. Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8, 2008.

[61] F. Martinez, E. Jacinto, and D. Acero. Brownian motion as exploration strategy for autonomous swarm robots. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2375–2380, 2012.

[62] David Ball, Patrick Ross, Andrew English, Tim Patten, Ben Upcroft, Robert Fitch, Salah Sukkarieh, Gordon Wyeth, and Peter Corke. Robotics for sustainable broad-acre agriculture. In *Field and service robotics*, pages 439–453. Springer, 2015.

[63] T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, and C. Schlegel. Managing a mobile agricultural robot swarm for a seeding task. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6879–6886, 2016.

[64] Dario Albani, Tiziano Manoni, Arikhan Arik, Daniele Nardi, and Vito Trianni. Field coverage for weed mapping: Toward experiments with a uav swarm. In Adriana Compagnoni, William Casey, Yang Cai, and Bud Mishra, editors, *Bio-inspired Information and Communication Technologies*, pages 132–146, Cham, 2019. Springer International Publishing.

[65] Joan saez pons, Lyuba Alboul, Jacques Penders, and Leo Nomdedeu. Multi-robot team formation control in the guardians project. *Industrial Robot: An International Journal*, 37:372–383, 01 2010.

[66] Sabine Hauert, Jean-Christophe Zufferey, and Dario Floreano. Evolved swarming without positioning information: An application in aerial communication relay. *Autonomous Robots*, 26, 10 2009.

[67] E. Feo Flushing, L. M. Gambardella, and G. A. Di Caro. A mathematical programming approach to collaborative missions with heterogeneous teams. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 396–403, 2014.

[68] Maja Varga, Meysam Basiri, Gregoire Heitz, and Dario Floreano. Distributed formation control of fixed wing micro aerial vehicles for area coverage. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 669–674. IEEE, 2015.

[69] Alessandro Farinelli, Masoume M Raeissi, Nathan Brooks, and Paul Scerri. Interacting with team oriented plans in multi-robot systems. *Autonomous Agents and Multi-Agent Systems*, 31(2):332–361, 2017.

[70] Daniel A Real-Arce, Tania Morales, C Barrera, J Hernández, and O Llinás. Smart and networking underwater robots in cooperation meshes: the swarms ecsel: H2020 project. In *Instrumentation viewpoint*, pages 19–19. SARTI, 2016.

[71] T. Maccready. Multiscale vorticity from a swarm of drifters. In *2015 IEEE/OES Eleveth Current, Waves and Turbulence Measurement (CWTM)*, pages 1–6, 2015.

[72] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, Mar 2013.

[73] Lorenzo Garattoni and Mauro Birattari. *Swarm Robotics*, pages 1–19. American Cancer Society, 2016.

[74] Heiko Hamann and Thomas Schmickl. Modelling the swarm: Analysing biological and engineered swarm systems. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):1–12, 2012.

[75] Nikolaus Correll and Heiko Hamann. Probabilistic modeling of swarming systems. In *Springer handbook of computational intelligence*, pages 1423–1432. Springer, 2015.

[76] Alexandru Bara and Sanda Dale. Dynamic modeling and stabilization of wheeled mobile robot. In *Proceedings of the 5th WSEAS International Conference on Dynamical Systems and Control*, CONTROL'09, pages 87–92,

Stevens Point, Wisconsin, USA, 2009. World Scientific and Engineering Academy and Society (WSEAS).

[77] Rafael Kelly, Victor Santibáñez Davila, and Julio Antonio Loría Perez. *Control of robot manipulators in joint space*. Springer Science & Business Media, 2006.

[78] H.K. Khalil. *Nonlinear Systems*. Prentice Hall, 1996.

[79] Erick Ordaz-Rivas, Angel Rodriguez-Liñan, Mario Aguilera-Ruíz, and Luis Torres-Treviño. Collective tasks for a flock of robots using influence factor. *Journal of Intelligent & Robotic Systems*, Oct 2018.

[80] P. Liu, H. Yu, and S. Cang. Modelling and dynamic analysis of underactuated capsule systems with friction-induced hysteresis. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 549–554, 2016.

[81] Pengcheng Liu, Hongnian Yu, and Shuang Cang. Modelling and analysis of dynamic frictional interactions of vibro-driven capsule systems with viscoelastic property. *European Journal of Mechanics - A/Solids*, 74:16 – 25, 2019.

[82] Vladimir Stojanovic, Novak Nedic, Dragan Prsic, and Ljubisa Dubonjic. Optimal experiment design for identification of arx models with constrained output in non-gaussian noise. *Applied Mathematical Modelling*, 40(13):6676 – 6689, 2016.

[83] Vojislav Filipović, Novak Nedic, and Vladimir Stojanovic. Robust identification of pneumatic servo actuators in the real situations. *Forschung im Ingenieurwesen*, 75, 12 2011.

[84] Vladimir Stojanovic and Vojislav Filipović. Adaptive input design for identification of output error model with constrained output. *Circuits, Systems, and Signal Processing*, 33, 01 2014.

[85] José Antonio Martín H., Javier de Lope, and Darío Maravall. Analysis and solution of a predator-protector-prey multi-robot system by a high-level reinforcement learning architecture and the adaptive systems theory. *Robot. Auton. Syst.*, 58(12):1266–1272, December 2010.

[86] Stefano Nolfi. Co-evolving predator and prey robots. *Adaptive Behavior*, 20(1):10–15, 2012.

[87] S. Mukhopadhyay and H. Leung. Cluster synchronization of predator prey robots. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2753–2758, Oct 2013.

[88] T. Yasuda, K. Ohkura, T. Nomura, and Y. Matsumura. Evolutionary swarm robotics approach to a pursuit problem. In *2014 IEEE Symposium on Robotic Intelligence in Informationally Structured Space (RiiSS)*, pages 1–6, Dec 2014.

[89] Jorge C. Gomes, Pedro Mariano, and Anders Lyhne Christensen. Systematic derivation of behaviour characterisations in evolutionary robotics. *CoRR*, abs/1407.0577, 2014.

[90] Tiago Rodrigues, Miguel Duarte, Margarida Figueiró, Vasco Costa, Sancho Moura Oliveira, and Anders Lyhne Christensen. Overcoming limited onboard sensing in swarm robotics through local communication. In Ngoc Thanh Nguyen, Ryszard Kowalczyk, Béatrice Duval, Jaap van den Herik, Stephane Loiseau, and Joaquim Filipe, editors, *Transactions on Computational Collective Intelligence XX*, pages 201–223, Cham, 2015. Springer International Publishing.

[91] Randal S. Olson, David B. Knoester, and Christoph Adami. Evolution of swarming behavior is shaped by how predators attack. *Artificial Life*, 22(3):299–318, 2016.

[92] Arthur Bernard, Jean-Baptiste André, and Nicolas Bredeche. To cooperate or

not to cooperate: Why behavioural mechanisms matter. *PLOS Computational Biology*, 12(5):1–14, 05 2016.

[93] Jorge Gomes, Miguel Duarte, Pedro Mariano, and Anders Lyhne Christensen. Cooperative coevolution of control for a real multirobot system. In Julia Handl, Emma Hart, Peter R. Lewis, Manuel López-Ibáñez, Gabriela Ochoa, and Ben Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, pages 591–601, Cham, 2016. Springer International Publishing.

[94] Nigel Franks and A. Sendova-Franks. Brood sorting by ants: distributing the workload over the work-surface. *Behavioral Ecology and Sociobiology*, 30:109–123, 03 1992.

[95] P.P. Grassé. *La Reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes Sp. ; la théorie de la stigmergie: essai d'interprétation, comportement des termites constructeurs.* Masson, 1959.

[96] Jens Wawerla, Gawav Sukhatme, and Maja Mataric. Collective construction with multiple robots. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 3, pages 2696 – 2701 vol.3. IEEE, 02 2002.

[97] Justin Werfel and Radhika Nagpal. Extended stigmergy in collective construction. *IEEE Intelligent Systems*, 21:20–28, 03 2006.

[98] Justin Werfel and Radhika Nagpal. Three-dimensional construction with mobile robots and modular blocks. *I. J. Robotic Res.*, 27:463–479, 03 2008.

[99] Andrew Vardy. Orbital construction: Swarms of simple robots building enclosures. In *2018 IEEE 3rd International Workshops on Foundations and Applications of Self Systems (FAS W)*, pages 147–153. IEEE, 09 2018.

[100] Kirstin H Petersen, Nils Napp, Robert Stuart-Smith, Daniela Rus, and Mirko Kovac. A review of collective robotic construction. *Science Robotics*, 4(28), 2019.

[101] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Distributed multi-robot algorithms for the termes 3d collective construction system. In *Proceedings of Robotics: Science and Systems*. Institute of Electrical and Electronics Engineers, 05 2020.

[102] Bernard Chazelle. An algorithmic approach to collective behavior. *Journal of Statistical Physics*, 158(3):514–548, Feb 2015.

[103] Anil Özdemir, Melvin Gauci, and Roderich Gross. Shepherding with robots that do not compute. *The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE)*, pages 332–339, 2017.

[104] Mitchell A. Potter, Lisa Meeden, and Alan C. Schultz. Heterogeneity in the coevolved behaviors of mobile robots: The emergence of specialists. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'01, pages 1337–1343, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[105] Luke Ludwig and Maria Gini. Robotic swarm dispersion using wireless intensity signals. In Maria Gini and Richard Voyles, editors, *Distributed Autonomous Robotic Systems 7*, pages 135–144, Tokyo, 2006. Springer Japan.

[106] James McLurkin and Jennifer Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In Rachid Alami, Raja Chatila, and Hajime Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 399–408, Tokyo, 2007. Springer Japan.

[107] S. Nouyan, R. Gross, M. Bonani, F. Mondada, and M. Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, Aug 2009.

[108] Mikhail Efremov and Ivan Kholod. Swarm robotics foraging approaches. In *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pages 299–304, 01 2020.