Implementación del algoritmo de Boids de Reynolds en robots móviles con ruedas dentro del ecosistema Robotat

Brayan José Castillo Alvarado

UVG UNIVERSIDAD DEL VALLE DE GUATEMALA

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación del algoritmo de Boids de Reynolds en robots móviles con ruedas dentro del ecosistema Robotat

Trabajo de graduación presentado por Brayan José Castillo Alvarado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación del algoritmo de Boids de Reynolds en robots móviles con ruedas dentro del ecosistema Robotat

Trabajo de graduación presentado por Brayan José Castillo Alvarado
para optar al grado académico de Licenciado en Ingeniería Mecatrónica

Guatemala,

2024

Vo.Bo.:

(f) _____
MSc. Miguel Zea

Tribunal Examinador:

(f) _____
MSc. Miguel Zea

(f) _____
Ing. Kurt Kellner

(f) _____
Ing. Christopher Chiroy

Fecha de aprobación: Guatemala, 20 de enero de 2024.

Prefacio

La elaboración del presente trabajo de graduación surge del interés personal de poder aplicar un tema tan amplio y diverso como lo es la robótica de enjambre, aplicando mis conocimientos de programación y robótica adquiridos a lo largo de estos 5 años.

Quiero agradecer principalmente a Dios y a mis padres Erick Castillo y Marleny Alvarado, al igual que mi hermano Yahir Castillo por haberme apoyado y darme la oportunidad de estudiar en la capital. A mis abuelos Rolando Castillo y Nivea López por haberme dado un segundo hogar en el cual poder llevar a cabo mis estudios superiores.

Agradezco a los catedráticos de la Universidad del Valle de Guatemala por el conocimiento compartido y principalmente me gustaría agradecer a mi asesor de tesis Msc. Miguel Zea por el apoyo y explicaciones aportadas para llevar a cabo este trabajo de graduación y también al Dr. Luis Rivera por por su apoyo en diferentes horarios y conocimientos para diferentes estrategias de programación

Por último, me gustaría agradecer a los amigos que hice en estos años los cuales me apoyaron y ayudaron a seguir adelante en la carrera, les agradezco a todos y cada uno de ustedes por todos los momentos compartidos, y por darme unos de los mejores recuerdos de mi vida.

Índice

Lista de figuras

Lista de cuadros

Resumen

El algoritmo de Boids de Reynolds se utiliza para replicar el comportamiento de parvadas y rebaños en simulaciones. En este trabajo de graduación se presenta el proceso de desarrollo del algoritmo mediante el software matemático Matlab: aplicando las 3 reglas de los Boids de Reynolds y agregando la evasión de obstáculos y depredadores. También se presenta la comprobación de la implementación física haciendo uso de los robots móviles con ruedas Pololu 3pi+ mediante un controlador de acercamiento exponencial. Asimismo se expusieron las limitaciones del algoritmo donde se posee un modelo de regresión lineal entre el número de agentes y el tiempo de ejecución de la simulación. Además, en la comparación entre simulación e implementación física se obtuvo un RMSE: no mayor a 4 cm y porcentajes de error no mayores a 3 % al utilizar tolerancias de 5 cm en el controlador. Por último, el número máximo de Pololus 3pi+ que se pueden utilizar en la implementación física fue de cinco agentes.

Abstract

The Reynolds Boids algorithm is used to replicate the behavior of flocks and herds in simulations. In this graduation work, the algorithm development process is presented using the mathematical software Matlab:, applying the 3 rules of the Reynolds Boids and adding the avoidance of obstacles and predators. The verification of the physical implementation using the Pololu 3pi+ wheeled mobile robots using an exponential approach controller is also presented. Likewise, the limitations of the algorithm where there is a linear regression model between the number of agents and the simulation execution time were exposed. Furthermore, in the comparison between simulation and physical implementation, a RMSE: no greater than 4 cm and error percentages no greater than 3 % were obtained when using tolerances of 5 cm in the controller. Finally, the maximum number of Pololus 3pi+ that can be used in the physical deployment were five agents.

CAPÍTULO 1

Introducción

El algoritmo de Boids de Reynolds ha sido utilizado a lo largo de los años tanto en películas, videojuegos como en simulaciones para replicar el comportamiento de parvadas o rebaños. Sin embargo, este algoritmo no ha sido implementado de manera física y únicamente se ha utilizado en simulaciones.

Por lo tanto, este trabajo busca replicar los comportamientos propuestos por Reynolds en un grupo de robots, dado que la principal ventaja es que no existe un líder de grupo y todos los individuos funcionan como una sola entidad, por lo que si en dado caso uno o varios individuos del grupo se pierde del rebaño este puede seguir funcionando normalmente. Asimismo, se programó la implementación de un algoritmo de evasión para que el rebaño sea capaz de evitar obstáculos o depredadores separándose y volviendo a unirse como lo haría un rebaño.

En el capítulo 7 se implementó el desarrollo del algoritmo, donde los Boids presentan movimientos fluidos y comportamientos adecuados en base a los pesos de cada una de las tres reglas. Teniendo el peso más grande la regla de separación, dado que es imprescindible evitar la colisión entre los robots a la hora de la implementación física. Asimismo en este capítulo se abordan las limitaciones que posee la simulación y como el aumento de agentes en el mismo puede generar retrasos en la ejecución del programa.

Las reglas de los Boids de Reynolds generan las posiciones y velocidades de cada individuo en cada renderización del programa para posteriormente utilizar el modelo del uniciclo junto al modelo de robots diferenciales [1] con la finalidad de hacer el envió de datos a los robots móviles con ruedas. Asimismo, también se tomó la información de uno de los robots en físico mediante un marcador para hacer una comparación entre la simulación y el comportamiento en físico dentro del ecosistema Robotat que puede encontrarse en el capítulo 10 de este trabajo.

La implementación física se inició fijando un robot con ruedas en una posición aleatoria para luego hacer el envió de dicha posición en la simulación mediante el uso de marcadores y el ecosistema Robotat.

Posteriormente se ejecutó la simulación y se realizó el envió de los datos de las velocidades de cada rueda, donde los tiempos de la simulación y los robots con ruedas era completamente distinto, por lo que se optó por diferentes estrategias como el uso de controladores que se abordaran más adelante en el trabajo.

Por último, se utilizó un controlador de acercamiento exponencial donde fue necesario realizar interpolaciones de las trayectorias obtenidas de la simulación para definir puntos más separados entre sí, esto con la finalidad de dar suficiente tiempo al controlador para realizar el envió de las velocidades a las ruedas de los Pololu 3pi+ y asegurar la recepción de datos de los marcadores del ecosistema Robotat. Esta estrategia permitió obtener RMSE: no mayores a 4 cm.

CAPÍTULO 2

Antecedentes

A lo largo de los años ha sido necesario representar parvadas y rebaños tanto en el medio de entretenimiento para mejorar la inmersión de las personas en productos audiovisuales, como en la área de investigación donde se busca representar los comportamientos de animales con la finalidad de llevar a cabo tareas de optimización. En este caso siendo los Boids de Reynolds el algoritmo que permitió satisfacer estas necesidades tanto en el ámbito digital en diferentes animaciones, como en el ambito físico con la implementación del mismo en robots como los son los ChIRP.

2.1. Bandadas, rebaños y cardúmenes: un modelo de comportamiento distribuido

En 1987 Craig W. Reynolds propuso un modelo de comportamiento para simulaciones de grandes grupos de individuos llamados Boids, siendo estos individuos que se comportarán como parvadas de pájaros, como se observa en la Figura 1. Para el caso de la simulación y el comportamiento adecuado de los Boids se propusieron tres comportamientos o restricciones principales [2].

La separación, que asegura que los individuos del rebaño no choquen entre sí, por lo que es necesario definir una distancia a la cual cada individuo estará separado de sus vecinos. La segunda restricción es la alineación del Boid, donde los individuos deben estar viendo a una misma dirección para comportarse como una parvada, no es suficiente estar equisdistantes a sus vecinos para lograr el comportamiento deseado.

El tercer comportamiento es la cohesión, la cual es mantenerse a una posición promedio a los demás individuos para que todos juntos funcionen como un rebaño, permitiendo que aunque se separen por algún objeto externo estos vuelvan a unirse. Cabe mencionar que en este trabajo únicamente se realizaron pruebas por software por lo que no hubo implementaciones físicas del algoritmo.

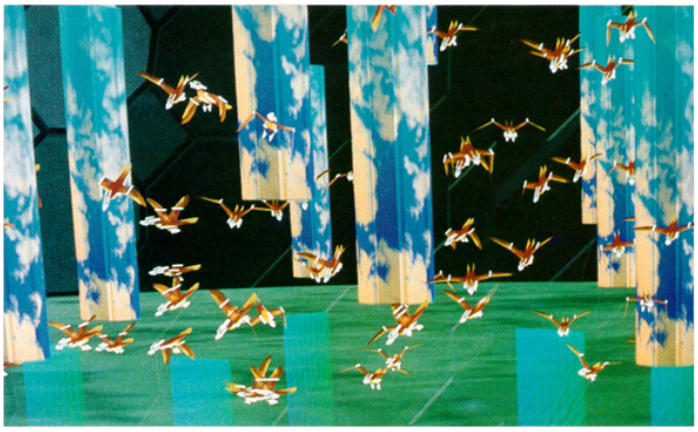

Figura 1: Implementación del algoritmo de Boids de Reynolds por software [2]

Este algoritmo permitió una aplicación cinematográfica práctica de Boids, en varias películas, como por ejemplo Batman Returns (1992) de Tim Burton[3]. Además ha sido utilizado en juegos como Unreal (Epic), Half-Life (Sierra) y Enemy Nations (Windward Studios)[4].

2.2. Implementación del comportamiento de Boids mediante robots ChIRP

En el año 2015 en la Universidad Noruega de Ciencia y Tecnología, se llevó a cabo la implementación de Boids mediante robots ChIRP los cuales contaban con seis sensores de distancia. Además de esto se utilizó un software de seguimiento de cámara, que permitió mediante una computadora externa implementar el algoritmo de Boids de Reynolds, como se observa en la Figura 2.

En este trabajo se realizaron experimentos con cuatro robots ChIRP, donde se plantearon tres escenarios diferentes. El primero, con los cuatro robots en cada punta de un cuadrado con un obstáculo pequeño en el medio y la idea era observar si se unían para formar un rebaño. El segundo caso era posicionar tres robots en la esquina superior izquierda y un único robot en la esquina inferior derecha de un cuadrado, una vez más con un obstáculo en medio con la finalidad de observar el comportamiento del rebaño [5].

El último caso a observar consistía en ubicar tanto los cuatro robots en diferentes lugares, como un obstáculo en algún lugar aleatorio y confirmar si se juntaban para volver a juntar un rebaño. En los resultados obtenidos de dichos casos se menciona que para futuros trabajos sería imprescindible el uso de más robots, ya que solo cuatro limitan el comportamiento del rebaño y al contar con más robots se podría observar de mejor manera como reaccionan ante varios obstáculos o incluso objetos de mayor tamaño.

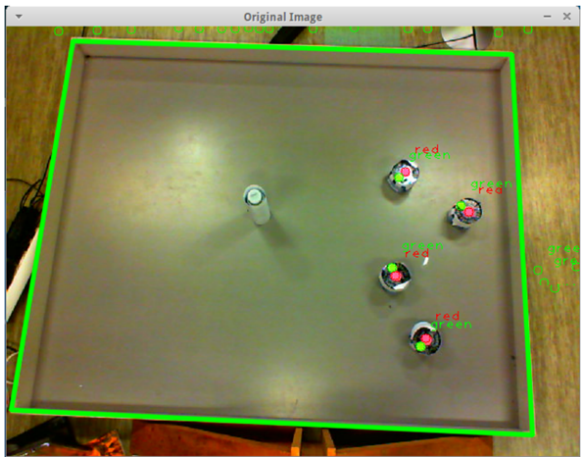

Figura 2: Robots ChIRP examinados mediante el software de captura [5]

2.3. Ecosistema Robotat

Tomando en cuenta lo mencionado anteriormente se buscó un método para controlar de forma óptima a los robots, sin depender de una cámara que muchas veces puede dar mediciones de distancias imprecisas.

Por lo que se optó por este trabajo donde se creó un ecosistema de experimentación robótico denominado Robotat, el cual mediante la implementación de una red de intercomunicación WiFi entre varios dispositivos, fue capaz de conectar el sistema de captura principal Optitrack, junto a demás agentes mediante la implementación de un microcontrolador ESP32, que utiliza el protocolo MQTT, para la comunicación de los mismos [6].

CAPÍTULO 3

Justificación

La ingeniería se ha inspirado de la naturaleza para optimizar diferentes tareas y procesos, observando que, en el caso de enjambres el hecho de imitar los comportamientos de grandes grupos de animales puede optimizar el llevar a cabo un trabajo que podría ser ineficiente o incluso imposible con un solo individuo.

Los Boids de Reynolds han sido útiles para modelos de animación a lo largo de los años, siendo utilizados en películas o incluso videojuegos, por su alta fidelidad con la representación de un rebaño o parvada real. La principal ventaja de este modelo es que no existe un líder de grupo y todos los individuos funcionan como una sola entidad, por lo que si en dado caso uno o varios individuos del grupo se pierde el rebaño puede seguir actuando normalmente.

Este es un comportamiento deseado en enjambres de robots, en donde a pesar de que en el grupo de robots alguno de estos falle o haya sido llevado a reparaciones, el resto del grupo pueda seguir funcionando y llevar a cabo la tarea, lo cual no sería posible si se cuenta con un sólo robot. Por estas razones, este trabajo busca implementar un algoritmo para enjambres de robots basado en los Boids de Reynolds con la finalidad de simular multitudes, sistemas complejos y comportamiento animal.

CAPÍTULO 4

Objetivos

4.1. Objetivo general

Desarrollar e implementar un algoritmo que replique el comportamiento de rebaños, basado en los Boids de Reynolds, en robots móviles con ruedas dentro del ecosistema Robotat.

4.2. Objetivos específicos

- Desarrollar un algoritmo que permita controlar un grupo de robots Pololu 3Pi+ contemplando las características y restricciones de los rebaños establecidas por los Boids de Reynolds.
- Implementar la capacidad que permita que el rebaño evite obstáculos y predadores, sin modificar el comportamiento regular del enjambre.
- Validar la simulación del algoritmo desarrollado en Matlab dentro del sistema Robotat con los robots Pololu 3Pi+.

CAPÍTULO 5

Alcance

El alcance de este trabajo de graduación consiste en la implementación y validación del algoritmo de Boids de Reynolds en robots móviles con ruedas dentro del ecosistema Robotat, para lo cual se desarrollo el algoritmo simulado mediante Matlab dado que las funciones con las que se utiliza el ecosistema Robotat están desarrolladas en este software. En este caso se utilizó programación orientada a objetos con la finalidad de facilitar tanto la programación de los Boids como el envió de datos.

Durante el desarrollo del proyecto se presentaron limitaciones respecto al espacio y tiempo de uso del ecosistema Robotat en los laboratorios de la universidad debido a que varios proyectos de graduación también necesitaban utilizar dicho espacio por lo que las pruebas físicas llegaban a ser limitadas en los días disponibles de trabajo. Asimismo, sólo se contaba con una cantidad de diez robots Pololu 3pi+, lo que hacia necesario compartir los robots con los demás compañeros y por lo tanto teniendo un numero limitado de agentes.

Dadas las circunstancias anteriores y la falta de sensores de distancia, este trabajo no plantean comprobar en todo momento que los robots con ruedas se comporten exactamente igual que en la simulación. Por lo que los robots serán colocados en posiciones aleatorias en el ecosistema Robotat, que serán enviadas a la computadora mediante marcadores, para posteriormente ajustar el programa y tanto la simulación como los robots en físico inicien en la misma posición, posteriormente se irán comparando los puntos obtenidos de la interpolaciones con los puntos alcanzados por cada agente mediante el controlador de acercamiento exponencial obteniendo la raíz del error cuadrático medio y porcentajes de error.

Por último, en el caso de los obstáculos y depredadores se definieron ambos como objetos estáticos con los cuales los agentes buscaran no colisionar. Se definió de esta manera para no afectar el rendimiento de la simulación dado que mientras más individuos dinámicos haya en la simulación más lenta se vuelve conforme avanza.

CAPÍTULO 6

Marco teórico

6.1. Robótica de enjambres

La robótica de enjambre enfatiza en el uso de varios robots simples en lugar de un único robot complejo, siendo su principal fortaleza la cooperación entre ellos para llevar a cabo tareas complejas de manera colectiva. El objetivo de la robótica de enjambres es el estudio del diseño de robots que logren, a partir de las interacciones entre los robots y el entorno, generar patrones de comportamiento colectivos, es decir, a partir de comportamientos individuales y sencillos obtener comportamientos complejos a nivel global[7].

La robótica de enjambres se utiliza en distintas áreas, tales como el ámbito industrial, militar, médico, entre otros. En este caso los investigadores se han inspirado en los sistema biológicos para proponer diferentes sistemas donde es apropiado utilizarlos como lo son tareas donde se debe de cubrir una región, tareas centradas en alguna entidad del entorno, tareas demasiado peligrosas, tareas que deben aumentar o disminuir en el tiempo y tareas que requieren redundancia[7].

6.1.1. Ventajas y beneficios de la robótica de enjambres

Si se compara un robot tradicional este puede necesitar componentes complejos una gran capacidad de procesamiento para llevar a cabo una tarea determinada, mientras que el uso de varios robots simples puede presentar varias ventajas como [7]:

- Son tolerantes a fallos y robustos, dado que pueden seguir en funcionamiento si falla una unidad.
- Tienen una alta escalabilidad, ya que pueden aumentar o disminuir el tamaño del enjambre según la necesidad.

- Enfatiza en el paralelismo, donde un conjunto de robots llevan a cabo una tarea más rápidamente de lo que lo haría un solo robot.
- Suelen ser más económicos, y el coste de estos sistemas permite que sea factible la reparación o sustitución de estos equipos.

6.2. Robots móviles con ruedas Pololu 3pi+

El robot Pololu 3pi+ 32U4 OLED es un robot móvil basado en la MCU ATmega32U4 el cual viene precargado con un gestor de arranque compatible con Arduino, por lo que únicamente se necesita un cable USB A a Micro-B para utilizarlo. Cuenta con dimensiones de 97L×96W×36H en mm y tiene un peso de 100g. Además de esto cuenta con otros componentes como se observa en la Figura 3.

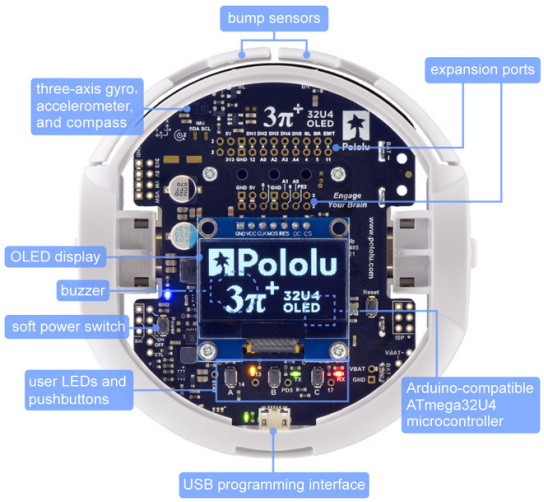

Figura 3: Robot Pololu 3pi+ Standard Edition [8]

El robot también posee codificadores de cuadratura dobles para control de posición o velocidad de circuito cerrado, sensores de línea, sensores de impacto frontal, una IMU completa (acelerómetro, magnetómetro y giroscopio de 3 ejes) y una pantalla OLED gráfica. Cabe recalcar que el modelo con el que se cuenta en la universidad es la Standard Edition la cual está ensamblada con micro motor reductores de metal 30:1 MP 6 V, los cuales proporcionan una velocidad de 1.5 m/s [8].

6.3. Cámaras de captura de movimiento

El trabajo se realizó dentro del ecosistema Robotat que está implementado mediante OptiTrack el cual está conformado por las cámaras Prime x41, que puede observarse en la Figura 4. Este sistema ha sido utilizado en diferentes proyectos como el desarrollo del videojuego *Call of Duty: Modern Warfare de Activision* o *El libro de la Selva* de Walt Disney Pictures.

Figura 4: Cámara Prime x41 [9]

Estas cámaras cuentan con una resolución de 4.1 megapíxeles, una velocidad de fotogramas nativa de 180 FPS y una velocidad máxima de fotogramas de 250 FPS. Sin embargo, las cámaras cuentan con varias configuraciones diferentes que pueden observarse en la Tabla 1.

La precisión 3D es típica para un área de seguimiento de 30'×30' (9 m × 9 m). El rango se calcula utilizando un marcador de 14 mm con cámaras con una exposición de 800, una ganancia de 6 y el número f más bajo. Las Prime x41 tienen errores de posición de menos de +/- 0.10 mm y errores de rotación de menos de 0.5 grados [9].

Cuadros por segundo	Resolución	FOV(lente estándar de 12.5mm)
180fps	2048x2048	51°×51°
240fps	2048x1440	51°×37°
500fps	2048x720	51°×18°
1000fps	2048x352	51°×9°

Cuadro 1: Tabla de especificaciones de la camara Prime x41 [9]

6.4. Algoritmo de Boids de Reynolds

6.4.1. Boids como cuerpos rígidos

En primer lugar es necesario representar a los individuos como cuerpos rígidos, muchos de las implementaciones de los algoritmos de Boids representan a los individuos como partículas sin tener que preocuparse de la orientación, pero en este caso donde se busca aplicarlo a robots móviles es más conveniente hacer la simulación con cuerpos rígidos debido a que es necesario tener orientado el frente del robot.

6.4.2. Visión local de la unidad

Teniendo un cuerpo rígido donde se puede reconocer el frente del mismo se empieza por implementar la visibilidad local de la unidad, como se sabe en los Boids el individuo debe estar pendiente de sus vecinos mas no es necesario que esté pendiente del grupo completo, por lo que para tener una referencia de la vista del individuo se utilizará un arco de visión que está dado por una distancia r del radio del arco y el ángulo theta, de la Figura 5.

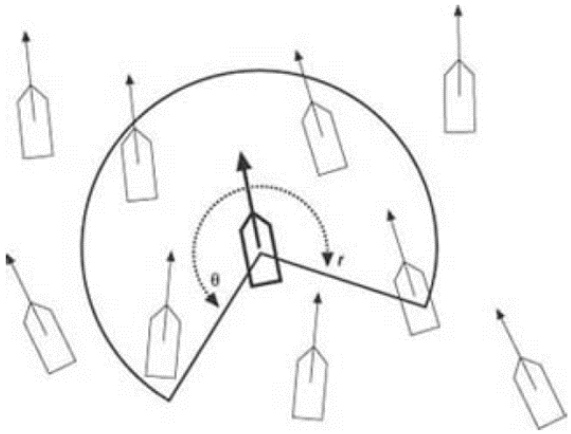

Figura 5: Ilustración de la visión local de la unidad [10]

Esta visibilidad local de la unidad es útil para el comportamiento de cohesión del rebaño, esto se debe a que un radio más grande se traduce en que los individuos estarán más conscientes de sus vecinos haciendo que el rebaño no se separe en rebaños más pequeños. Sin embargo, la restricción del ángulo servirá para tener un modelo más realista debido a que los animales en las bandadas no pueden observar a los compañeros que estén fuera de su campo de visión, es decir, el punto ciego de cada individuo que está en su espalda, por lo que esta restricción intentará ser similar a la de la Figura 5 para replicar el comportamiento. En este caso se utilizará un ángulo de 270 grados para formar el rebaño, dado que valores menores a este como lo son 45 grados tienden a formar comportamientos más similares a los de las hormigas.

Además de la visibilidad local de la unidad también será necesaria la implementación de un modelo de dirección donde se tratará cada unidad como un cuerpo rígido, aplicándole una fuerza de dirección neta en el extremo frontal de la unidad. Esta fuerza de dirección neta apuntará a estribor o babor en relación con la unidad y será la acumulación de fuerzas de dirección determinada por la aplicación de cada regla de agrupamiento. Este enfoque será útil para implementar cualquier número o combinación de reglas en el rebaño, cada regla hace una pequeña contribución a la fuerza de dirección total y el resultado neto se aplica a la unidad una vez que se consideran todas las reglas.

6.4.3. Cohesión

Luego de tomar en cuenta los aspectos anteriores se implementarán las reglas principales para los Boids de Reynolds. En primer lugar, se trabajará con la cohesión. Como se sabe esto implica que los individuos permanezcan juntos formando un rebaño, dado que no se desea que cada unidad se separe del grupo y siga su camino por separado. Para satisfacer esta regla, es necesario hacer que cada unidad se dirija hacia la posición promedio de sus vecinos.

En este caso se utilizará la posición promedio de los vecinos, siendo esta calculada mediante la suma vectorial de las respectivas posiciones divididas por el número total de vecinos, dando como resultado un vector que representa su posición promedio, en este caso, esa posición promedio se designó como un punto en la Figura 6.

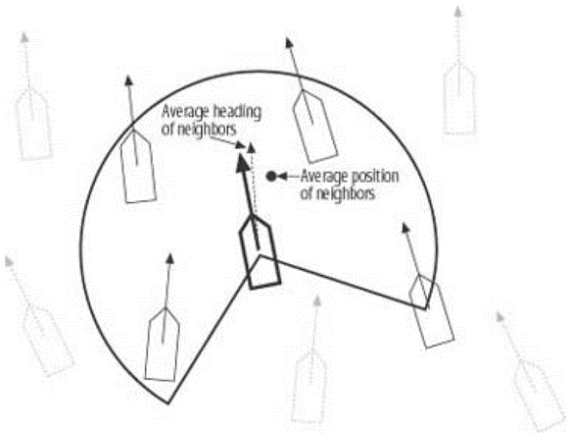

Figura 6: Ilustración de la posición promedio de los vecinos [10]

6.4.4. Alineación

La siguiente regla es la alineación la cual implica que todas las unidades del rebaño vayan generalmente en la misma dirección. Para lograr cumplir con esta restricción todas las unidades deben contar con la misma importancia o influencia al momento de calcular el rumbo promedio del rebaño, ya que no se busca que el rebaño dependa de un líder.

Observando la Figura 6, lo que se busca es que la linea del individuo se acople al promedio del rebaño que es la linea punteada, por lo que para este ejemplo, la flecha del individuo debe dirigirse hacia la derecha. Para determinar el rumbo se utilizará el vector de velocidad de cada individuo, dado que la normalización del vector de velocidad de cada individuo produce su vector de rumbo. Este caso es similar al de la cohesión donde se determinará la dirección promedio a la que apuntan los vecinos y se realizara el ajuste para que el individuo se acople al rebaño.

6.4.5. Separación

Por último, está la separación, que implica que las unidades mantengan una distancia mínima entre sí, pero sin afectar las reglas de cohesión y alineación ya que no se desea que las unidades choquen entre sí o, en dado caso, se fusionen en una posición coincidente al menos en la simulación, ya que sera necesario realizar varios ajustes preliminares antes de implementarlo de manera física con los robots móviles. Por lo tanto, será necesario hacer que los individuos se alejen de cualquier vecino que esté a la vista dentro de una distancia de separación mínima prescrita.

La implementación del algoritmo para controlar la separación es diferente al de la cohesión y la alineación, dado que para esta es necesario observar a cada vecino de forma individual y realizar las correcciones de dirección adecuadas, en lugar de tomar un valor promedio de todo el rebaño. La idea en este caso es utilizar la visibilidad local de unidad y prescribir un valor de separación determinado, como se observa en la Figura 7, donde se tiene el arco que es la visibilidad y la flecha en negro que es la distancia máxima a la que el individuo se puede acercar a un vecino.

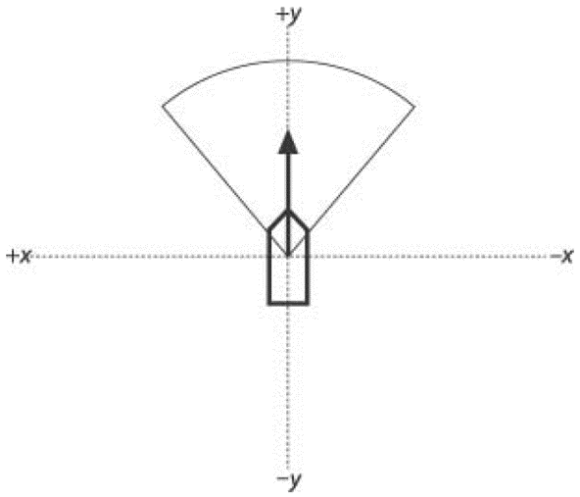

Figura 7: Ilustración de la visibilidad de unidad con una separación prescrita[10]

Es necesario mencionar que en las primeras simulaciones el algoritmo puede no funcionar adecuadamente dado que los individuos podrían separarse en varios rebaños pequeños o en dado caso chocar entre sí dando diferentes comportamientos erráticos, por lo que se llevará a cabo un proceso iterativo para obtener los mejores valores de agresividad de cada una de las tres reglas con el fin de obtener el comportamiento deseado [10].

6.5. Evasión de obstáculos

Para lograr que los rebaños esquiven obstáculos se usara el mismo sistema de visibilidad de unidad, pero en este caso el radio sera mayor para que todo el rebaño tenga tiempo de poder reconocer el obstáculo, cambiar la trayectoria y el algoritmo implementado anteriormente con las tres reglas haga que el rebaño se separe.

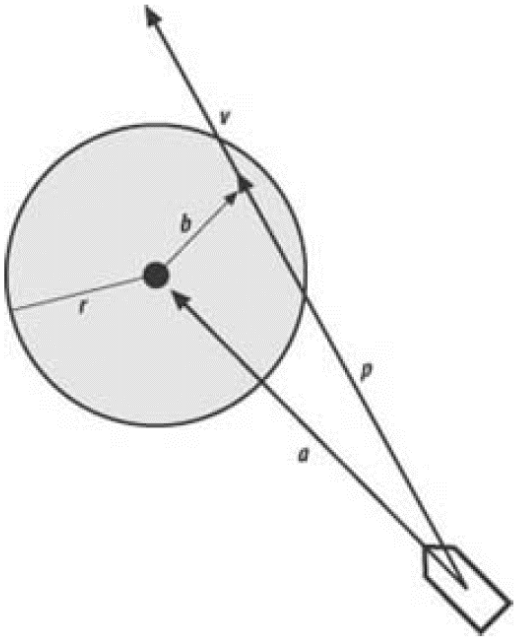

Figura 8: Ilustración de la validación del algoritmo de evasión[10]

Para poder implementar la evasión primero se calcula el vector a. Esta es simplemente la diferencia entre las posiciones de la unidad y del obstáculo. Luego, se proyecta a sobre v tomando el producto escalar. Esto produce el vector p. Esto sera útil dado que restar el vector p de a da como resultado el vector b.

Ahora, es necesario confirmar si v interseca al círculo en alguna parte, validando dos condiciones. Primero, la magnitud de p debe ser menor que la magnitud de v. Segundo, la magnitud de b debe ser menor que el radio del obstáculo, r. Si ambas pruebas son validadas, será necesario una dirección correctiva, como se observa en la Figura 8, de lo contrario el individuo puede continuar en su rumbo actual [10].

6.6. Modelo uniciclo

El modelo uniciclo consiste en el robot móvil con ruedas más simple el cual es una sola rueda giratoria vertical, este modelo aplica para cualquier sistema capaz de moverse hacia adelante y cambiar su dirección.

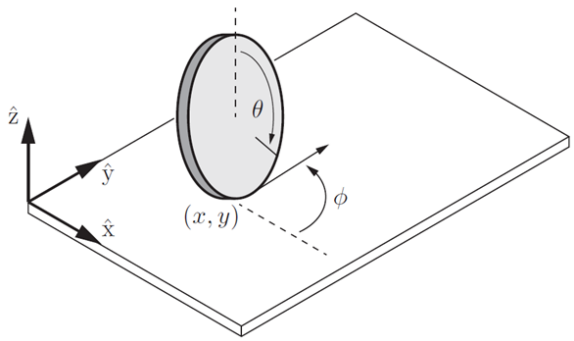

Figura 9: Modelo uniciclo [1]

En el modelo se define un punto de contacto y la dirección a la que avanza la rueda[1]. Este modelo relaciona las velocidades en un plano cartesiano donde las ecuaciones resultan en lo siguiente:

$$\dot{x} = v \cdot r \cos\theta \tag{1}$$

$$\dot{y} = v \cdot r \operatorname{sen}\theta \tag{2}$$

$$\dot{\theta} = w \tag{3}$$

Donde la velocidad lineal es v, r es el radio de las ruedas y w es la velocidad angular.

6.7. Modelo diferencial

El modelo diferencial de un robot móvil ya considera dos ruedas independientes las cuales giran sobre el mismo eje. Al tener un robot uniforme la velocidad lineal del centro de masa es el promedio de las velocidades lineales de cada llanta, de forma que se plantean las ecuaciones de velocidad de ambas llantas en función de la velocidad lineal y angular[1].

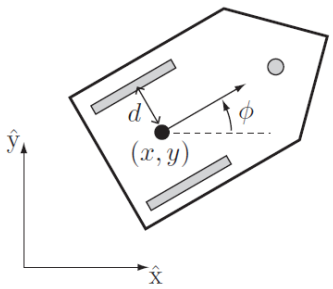

Figura 10: Modelo diferencial [1]

$$\dot{x} = \frac{r}{2}\cos\theta(uL + uR) \tag{4}$$

$$\dot{y} = \frac{r}{2}\operatorname{sen}\theta(uL + uR) \tag{5}$$

$$\dot{\theta} = \frac{r}{2d}(uR - uL) \tag{6}$$

En donde r es el radio de las ruedas, d es la distancia de las llantas al centro del robot, uL es la velocidad de la llanta izquierda y uR la velocidad de la llanta derecha.

Al juntar las ecuaciones 1, 2 y 3, con las ecuaciones 4, 5 y 6, despejamos para uL y uR, obteniendo:

$$uR = \frac{v + dw}{r} \tag{7}$$

$$uL = \frac{v - dw}{r} \tag{8}$$

6.8. Controlador PID

Un controlador PID tiene las siglas de proporcional, integral y derivativo, esto se debe a que cada uno de estos coeficiente se utiliza para realizar control de un sistema en específico. Este controlador se ha utilizado en diferentes campos, como lo puede ser un dispositivo mecánico o un dispositivo neumático.

En un controlador PID típico, estos tres elementos son controlados por una combinación del comando del sistema y la señal de retroalimentación del objeto que se está controlando (generalmente denominado "planta").

En el caso del coeficiente proporcional este es el control más fácil de implementar mediante retroalimentación dado que este generará una salida proporcional a la diferencia entre la posición actual y referencia que buscamos en la implementación, es decir, trata de alcanzar un valor deseado buscando un error de cero lo más rápido posible.

En el término integral se tiene en cuenta el error acumulado a lo largo del tiempo. Si hay un error constante en la posición, el término integral se incrementará con el tiempo y ayudará a eliminar este error acumulado. Usualmente se utiliza junto al control proporcional.

La parte derivativa busca predecir el error y actúa sobre el mismo (evitando que incremente). Proporcionando estabilidad y usualmente se emplea en menor medida que las partes proporcional e integral[11].

CAPÍTULO 7

Desarrollo del algoritmo de Boids de Reynolds en Matlab

Se decidió utilizar Matlab para el desarrollo del algoritmo dado que las funciones para la comunicación, tanto con el ecosistema Robotat como con los Pololu 3pi+, ya estaban implementadas en este lenguaje de programación. Asimismo, la decisión de utilizar los Pololu 3pi+ como los robots móviles con ruedas para este trabajo de graduación fue por ser los únicos disponibles en la Universidad del Valle de Guatemala.

7.1. Boids como partículas

En primer lugar fue necesario familiarizarse con el algoritmo y hacer pruebas del mismo, por lo que la manera más simple de probar las tres reglas de los Boids de Reynolds era simulando los agentes como simples partículas para comprobar los comportamientos y el manejo de los pesos de cada regla.

En este caso se definieron arreglos conforme al número de agentes a simular con posiciones aleatorias en el espacio de trabajo, al igual que diferentes velocidades, tanto en el eje x como el eje y.

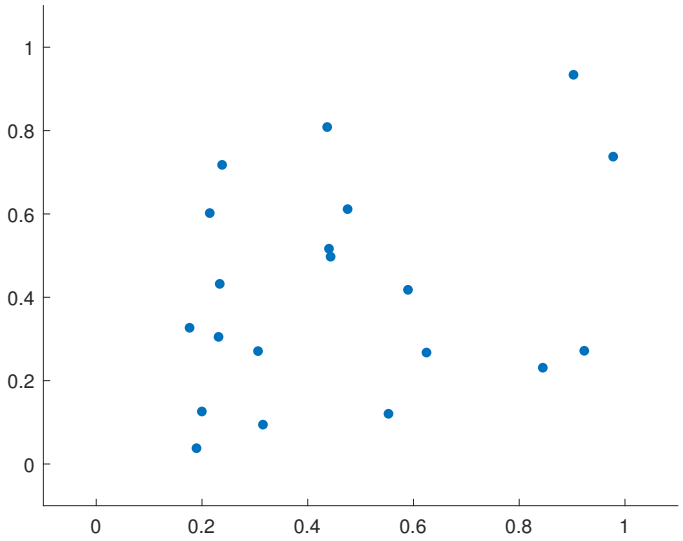

Figura 11: Posiciones iniciales de los agentes

7.1.1. Cohesión

Para la función de cohesión se definió un radio de visión que servirá para detectar a los agentes vecinos, y por lo tanto aplicar la regla y hacer un cambio de dirección. Caso contrario el agente debería seguir la misma dirección que tenia previamente. Para este proceso se utiliza el arreglo de las posiciones mencionado anteriormente y utilizando la función *pdist* de Matlab se obtuvieron las distancias entre estos.

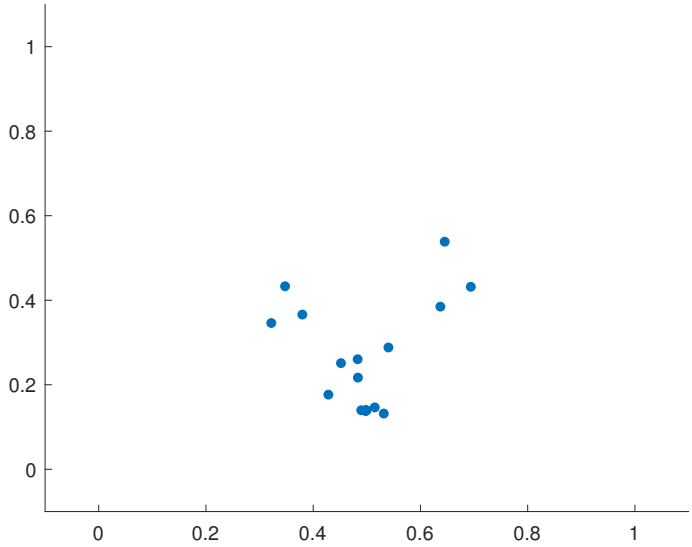

Figura 12: Regla de cohesión

Luego se hace una comprobación para conocer cuales vecinos están dentro del rango de visión y si alguno de estos se encuentra dentro, el valor de su posición se suma a una variable y aumenta el valor de un contador. Después de terminar el ciclo con todos los agentes la suma total se divide dentro del contador obteniendo la posición promedio para posteriormente restarle la posición actual del agente evaluado como se observa en la ecuación 9. El resultado de aplicar esta regla puede ser observado en la Figura 12.

$$newpos = posprom - posboide \tag{9}$$

En donde *newpos* es la nueva posición del agente evaluado, *posprom* es la posición promedio de los agentes vecinos que se encontraron dentro del rango y *posboide* es la posición actual del agente evaluado.

7.1.2. Separación

En el caso de la función de separación esta también se definió con un radio de visión dado que este suele tener un tamaño menor al de las demás reglas debido a que únicamente se debería de aplicar cuando los agentes están próximos a colisionar. Una vez más se vuelven a utilizar los arreglos de posición para los cálculos de la regla.

Volviendo a utilizar *pdist* se obtienen las distancias que estén dentro del radio del agente. Posteriormente una variable se iguala a ella misma y se le resta la diferencia entre el agente vecino y el agente evaluado, como se observa en la ecuación 10.

$$newpos = newpos - (posboidv - posboide) \tag{10}$$

Donde *newpos* es la nueva posición del agente evaluado, *posboidv* es el agente vecino dentro del rango de visión y *posboide* es el agente evaluado.

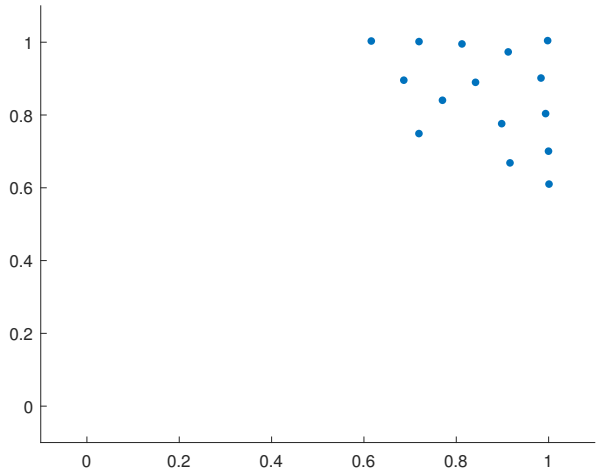

Figura 13: Regla de cohesión y separación

En este caso se muestra tanto la cohesión como la separación para entender como funciona la regla y la diferencia que se tiene con la Figura 12, donde se puede observar que los agentes van convergiendo a un solo punto sobreponiéndose entre ellos, mientras que en la Figura 13, mantienen una distancia segura sin colisionar entre ellos.

7.1.3. Alineación

Para la función de alineación también se definió un radio de visión. En este proceso a diferencia de los dos anteriores se utiliza tanto el arreglo de las velocidades como el de las posiciones. El primero para obtener la nueva velocidad resultante, y el segundo arreglo para volver a utilizar la función *pdist* y definir los vecinos dentro del rango de visión.

Teniendo los vecinos cercanos se procede a sumar el valor de sus velocidades en una variable, al mismo tiempo que aumenta el valor de un contador. Después de comprobar todos los agentes la suma total se divide dentro del contador obteniendo la velocidad promedio para posteriormente restarle la velocidad actual del agente evaluado, como se observa en la ecuación 11.

$$newvel = velprom - velboide \tag{11}$$

Donde *newvel* es la nueva velocidad del agente evaluado, *velprom* es la velocidad promedio de los agentes vecinos dentro del rango y *velboide* es la velocidad actual del agente evaluado.

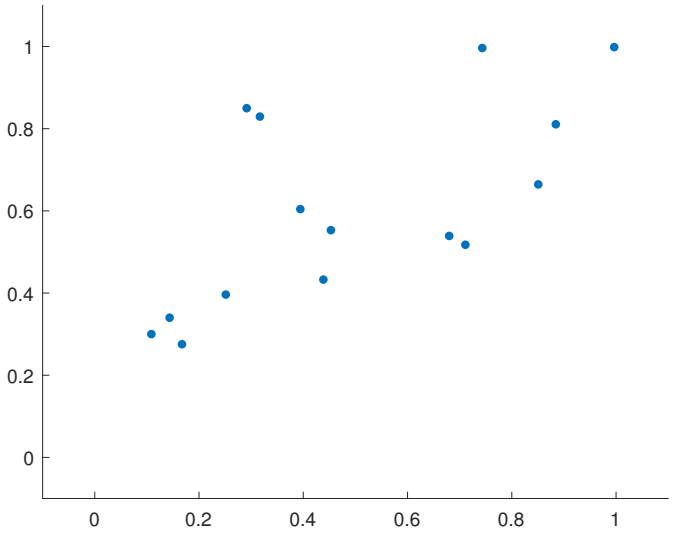

Figura 14: Regla de cohesión, separación y alineación

En este caso los agentes empiezan a formar grupos y se desplazan acoplándose a las velocidades de los vecinos. El resultado de aplicar las tres regla puede ser observado en la Figura 14.

7.1.4. Límites de simulación

Como se observa en la Figura 14 y anteriores el plano cartesiano está definido de 0 a 1. Además los agentes no están trabajando dentro de ningún límite por lo que se desplazan a donde deseen, algo que en la implementación física no sera posible.

Por lo tanto, se definieron los límites y se ajusto el tamaño de la simulación a 380 cuadros en el eje x y 480 cuadros en el eje y, dado que la plataforma del ecosistema Robotat posee estas dimensiones en centímetros.

7.2. Boids como cuerpos rígidos

Teniendo las nuevas dimensiones del plano y las reglas funcionando adecuadamente se empezó a trabajar en el cambio de los agente como partículas a los agentes como cuerpos rígidos. Esto con la finalidad de que estos puedan tener un frente y, por lo tanto, comportarse como lo haría un pez o un pájaro en la vida real al encontrarse con un obstáculo, es decir, girando progresivamente y no dando la impresión de que rebotó en una pared o en este caso los límites de la simulación.

Además de esto el código estaba empezando a ser demasiado extenso y confuso debido a la cantidad de veces que se repetían procesos. Con la finalidad de optimizar el uso de ciclos y definiciones se optó por empezar a utilizar programación orientada a objetos para que fuera más sencillo definir cada agente y sus respectivas reglas.

7.2.1. Graficar los agentes

Primero se realizó el objeto gráfico que, en este caso, será un triángulo con el frente hacia la derecha para que al momento de hacer la rotación en *theta* este sea concordante con el plano cartesiano y no sea necesario realizar ajustes.

En este caso se crearán los vértices del objeto gráfico el cual necesita como parámetros las coordenadas x y y como conjuntos de arreglos, que en este caso serán las coordenadas actuales de nuestro agente.

Para graficar el primer vértice se le restó un valor de 2.5 a la coordenada actual del agente, lo que nos dará la esquina inferior de nuestro triángulo. Luego se suma 2.5 a la coordenada actual para tener la esquina derecha (frente del triángulo), luego volvemos a restar 2.5 del valor actual de la coordenada para tener la esquina superior, y por último, se vuelve a restar 2.5 de la coordenada actual para cerrar el triángulo y, por lo tanto, completar el objeto gráfico.

Este proceso se debe de repetir con el eje y teniendo cuidado que tenga concordancia con los valores del eje x. Luego es necesario utilizar la función *patch* en Matlab que unirá los puntos que ingresamos en los arreglos y rellenará la imagen, en este caso de color negro. Esto dará un resultado como el de la Figura 15.

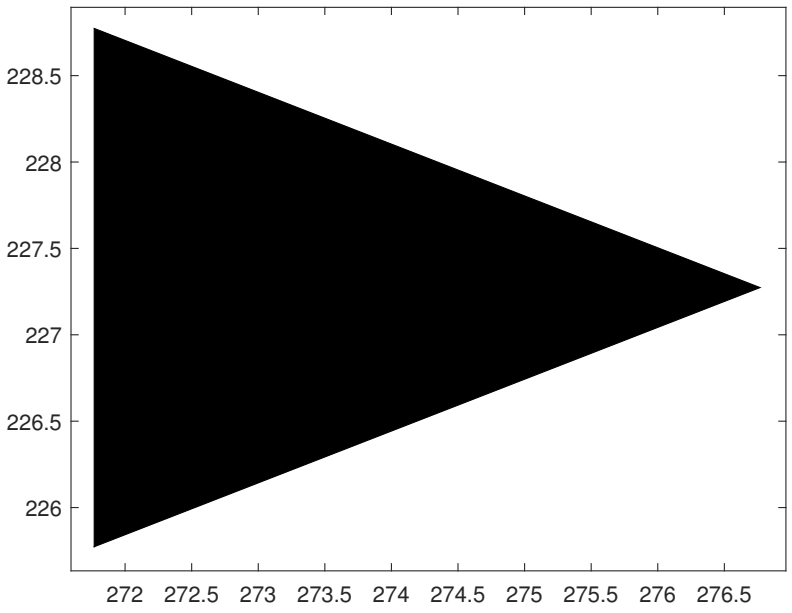

Figura 15: Agente con cuerpo rígido

Luego de esto es necesario rotar el objeto gráfico, por lo que se utiliza la función *rotate*, donde el primer parámetro es el objeto gráfico creado anteriormente, el segundo parámetro es el eje de rotación, que en este caso será el eje z, el tercer parámetro es el ángulo de rotación por lo que se tomarán las velocidades de x y y haciendo uso de la función $atan2$ para obtener *theta*. El último parámetro es el centro de rotación por lo que se ingresan las coordenadas actuales para que gire en su centro de masa.

Finalmente se hace un ciclo for para generar todos los agentes. Sin embargo, en cada renderización es necesario hacer uso de la función *delete* para borrar las posiciones anteriores de los objetos gráficos y también es recomendable usar la función *drawnow* que se utiliza para forzar una actualización inmediata de la ventana gráfica, lo que significa que todos los gráficos pendientes se representarán en ese momento.

7.2.2. Orientación de los agentes

Al ya poseer los agentes dibujados en el plano podemos proceder a proporcionarles la orientación mediante *theta*. Para esto es necesario utilizar los pesos de cada una de las tres reglas de los Boids de Reynolds mencionadas anteriormente. Luego de haberlas calculado y multiplicarlas por sus factores respectivos fue necesario aplicar dicha dirección o en este caso fuerza como se explica en [10], donde esta sera la suma de todas las reglas anteriores lo cual proporcionara una velocidad resultante para los agentes.

Teniendo la velocidad resultante se vuelve hacer el proceso mencionado en la graficación de los agentes donde se utiliza la función $atan2$ para obtener el ángulo proporcionándonos la dirección a la cual debe apuntar el frente de cada individuo haciendo que la simulación

funcione adecuadamente, dando como resultado la Figura 16.

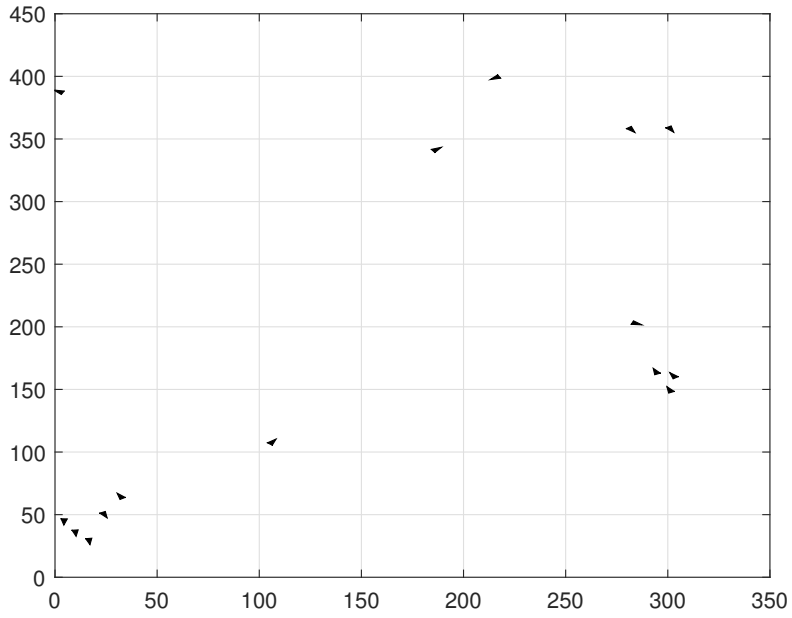

Figura 16: Agentes funcionando como cuerpos rígidos con orientación

7.2.3. Ajustes y optimización del código de las reglas de Boids de Reynlods

En el caso de la simulación se envían los cambios de posición simplemente como valores en el eje x y y, lo cual hace que los cambios de posición sean bruscos, a pesar de mitigarlo reduciendo los pesos de cada regla. Sin embargo, a la hora de la implementación física necesitamos que estos cambios sean lo más fluido posible, al igual que se necesita un envío sencillo de datos que en este caso sera la velocidad lineal y la velocidad angular, como se explicó en el Modelo diferencial.

Con la finalidad de solucionar estos problemas, en vez de simplemente enviar las posiciones se optó por obtener el ángulo $theta$ con $atan2$ al igual que cuando se graficaron los agentes, y hacer una resta entre el ángulo actual y el ángulo anterior, para posteriormente dividirlo dentro del tiempo que le tomó a la simulación ejecutarlo y obtener la velocidad angular. En el caso de la velocidad lineal esta se definió como constante para todos los agentes quedando definidos los cambios en el eje x como 12 y los cambios en el eje y como 13:

$$\dot{x} = v\cos\theta \tag{12}$$

$$\dot{y} = v\,\mathrm{sen}\,\theta \tag{13}$$

Asimismo se realizaron varias pruebas simuladas donde el comportamiento más fluido y deseado se alcanzó con diferentes pesos para cada regla. Se utilizó una proporción de 1 a 6 entre la regla de cohesión y la regla de separación, dado que es imprescindible evitar la colisión entre los robots a la hora de la implementación física. Por otro lado, la regla de alineación tiene un proporción igual a la de cohesión.

En el caso de los rangos de visión de cada regla se realizaron diferentes renderizaciones, donde los valores más óptimos fueron los siguiente; la cohesión con el radio más grande con 40 cuadros de visión, seguido de la alineación con 25 cuadros, y por último la separación con 6 cuadros de radio.

La generación del código también empezó a crecer a medida que se iban aplicando los cambios y optimizaciones mencionadas hasta el momento por lo que se optó por empezar a utilizar programación orientada a objetos para la generación de los boids, haciendo necesario buscar referencias de este tipo de estrategia por lo que una parte de la estructura del código esta basado en el repositorio de Rafal Kowalski [12].

7.3. Restricciones

Los agentes pueden llegar a colisionar entre ellos aun habiendo hecho las optimizaciones y cambios mencionados anteriormente, debido a que muchas veces la suma de todas las fuerzas de dirección es tan grande que una unidad se ve obligada a acercarse mucho a otra o justo encima de una unidad adyacente, esto también se menciona en el libro *AI for Game Developers* [10].

Asimismo se realizaron varias pruebas aumentando el números de agentes en la simulación notando un comportamiento lineal entre el número de boids y la velocidad de simulación para lograr ejecutar 150 renderizaciones obteniendo un R cuadrado de 0.993 lo que significa que el modelo de regresión se ajusta perfectamente a los datos, como se observa en la Figura 17.

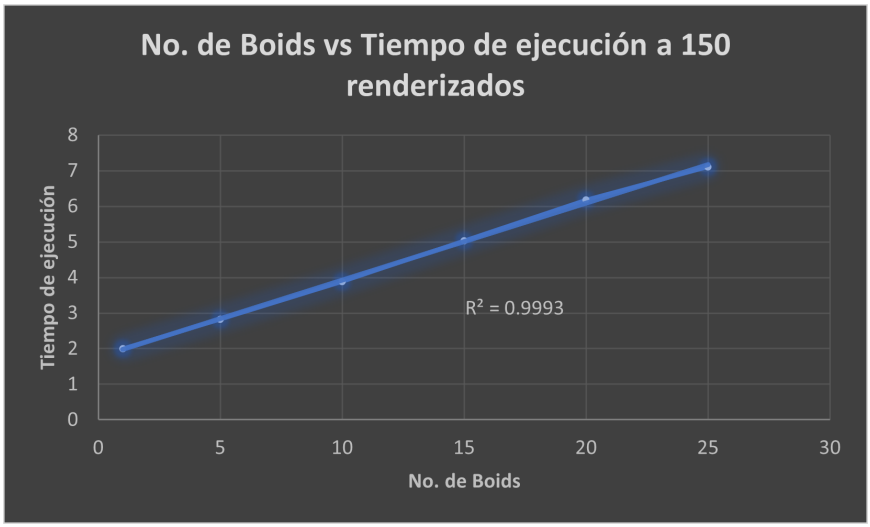

Figura 17: Gráfica No. de Boids vs Tiempo de ejecución a 150 renderizados

Con los datos y la gráfica obtenidos anteriormente podemos asegurar que mientras más agentes se agreguen a la simulación más lenta se volverá.

7.4. Generación de obstáculos y depredadores

Debido a las limitaciones de ejecución mencionadas en el capítulo anterior y con la finalidad de no afectar el rendimiento de la simulación se planteó la generación de obstáculos y depredadores como objetos estáticos. Estos también se trabajaron por medio de programación orientada a objetos y haciendo uso de la función *viscircles*, dado que esta genera círculos que se adaptan a las dimensiones del plano, es decir, si tenemos el plano con dimensiones de 380 en x y 480 en y, y decidimos graficar un obstáculo, este tendría la forma de un óvalo dadas las dimensiones, como se observa en la Figura 18, y no sería un círculo, que es un problema que se genera el graficar mediante la función *plot* de *Matlab*.

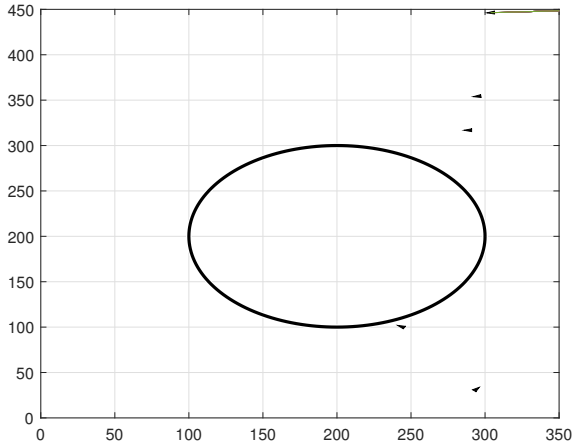

Figura 18: Gráfica No. de Boids vs Tiempo de ejecución a 150 renderizados

7.5. Implementación de la regla de evasión

Luego de haber generado el objeto se procedió a crear una nueva regla de evasión en el objeto de Boids que se utilizará para la evasión de los obstáculos y depredadores. Como se menciona en el pseudocódigo de [10], se genera un vector con la velocidad del agente para predecir a donde irá en la siguiente renderización y en caso haya un objeto colisionando con este vector se activa la regla donde se utiliza la componente en x para determinar en qué dirección se encuentra el obstáculo con respecto a la entidad y, por lo tanto, cómo debe reaccionar la entidad para evitar la colisión, como se observa en la Figura 19, donde los agentes ya esquivaron el obstáculo.

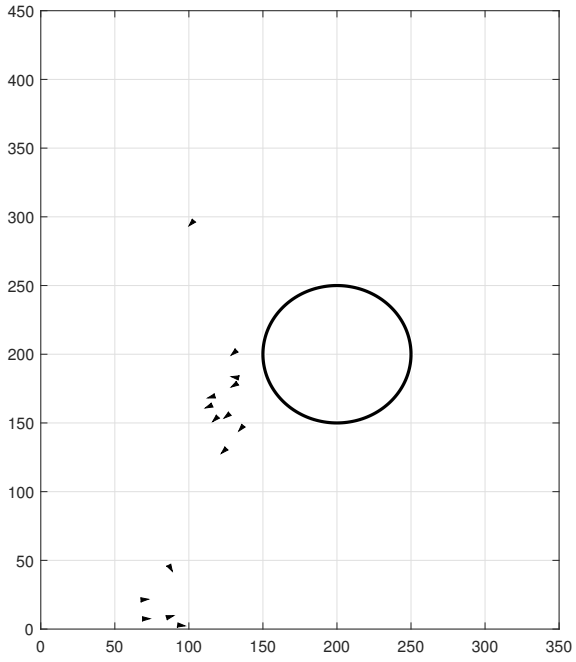

Figura 19: Regla de evasión implementada con un obstáculo de radio de 50 cuadros

Se realizaron varias ejecuciones del código donde la nueva regla de evasión no afectó el comportamiento original de las Reglas de los Boids de Reynolds, dado que cuando detectan un depredador no ignoran las demás reglas y se separan o colisionan entre sí con tal de alejarse de la zona, sino que agregan esta corrección a las reglas originales y siguen funcionando como una sola entidad para evitar el depredador.

CAPÍTULO 8

Implementación física del algoritmo

Luego de haber desarrollado y optimizado el algoritmo de Boids de Reynolds, se empezó a trabajar en la implementación física, donde es necesario configurar la orientación de los robots móviles con ruedas mediante los marcadores y también es necesario implementar el manejo de los envíos de datos a las ruedas del Pololu 3pi+.

8.1. Configurar los robots móviles con ruedas Pololu 3pi+

En primer lugar queremos que tanto la simulación como los robots en físico inicien en la misma posición, por lo que primero se coloca el Pololu 3pi+ en la plataforma acoplado con un marcador que nos permitirá obtener las posiciones en los ejes x y y. Esto no será únicamente útil al inicio, sino que también será utilizado para comparar el camino que tomó el robot en físico y compararlo con el camino que tomo el agente dentro de la simulación.

Figura 20: Marcador acoplado al robot móvil con ruedas Pololu 3pi+

Sin embargo, en el caso de la orientación es necesario obtener el ángulo de desfase que tiene el marcador con respecto del robot para tener concordancia entre los datos que registra el ecosistema Robotat y los datos de la simulación, a continuación en la Tabla 2 se muestran los ángulos utilizados para los ajustes.

No.	No. del Pololu 3pi+	Correción del ángulo de desfase
1	2	143.32
2	3	-176.44
3	4	-132.09
4	5	-94.11
5	6	-130.60
6	7	-99.55
7	8	-168.21
8	9	103.80

Cuadro 2: Corrección de los ángulos de desfase de cada Pololu 3pi+.

Luego de haber obtenido las posiciones con el marcador mediante el ecosistema Robotat es necesario hacer un ajuste de las coordenadas, dado que el origen se encuentra en el centro de la plataforma. Caso contrario de la simulación donde el origen se encuentra en la esquina inferior izquierda.

El ecosistema Robotat envía los valores en metros por lo que primero hacemos un ajuste multiplicando el dato obtenido por 100, y así tener los 380 y 480 cuadros de la simulación. Posteriormente se debe de sumar el valor de 190 a los valores obtenidos en el eje x y sumar 240 a los valores obtenidos en el eje y, esto con la finalidad de trasladar el origen y tener la concordancia entre los dos sistemas.

Asimismo, si el usuario lo desea puede hacer un ajuste similar pero menor para evitar que la simulación funcione utilizando toda la superficie del ecosistema Robotat. Esto con la finalidad de evitar que los robots móviles con ruedas puedan llegar a colisionar con las paredes.

8.2. Envío de velocidad lineal y velocidad angular a los Pololu 3pi+

Luego de ajustar los planos de coordenadas es necesario empezar a enviar las velocidades lineales y angulares a cada robot móvil con ruedas. Como se mencionó anteriormente la velocidad lineal fue definida como constantes y la velocidad angular se calculo mediante los cambios en cada renderización. En este caso se utilizaron las ecuaciones 7 y 8.

Dado que pueden presentarse velocidades demasiado altas se utilizó programación defensiva para evitar que las velocidades en las llantas superaran las 300 rpm, como se observa en la Figura 21.

```
wr = (0.5+DISTANCE_FROM_CENTER*wctrl(i))/WHEEL_RADIUS;
wl = (0.5-DISTANCE_FROM_CENTER*wctrl(i))/WHEEL_RADIUS;
% Los valores se envian en rpm


if wr > 300
    wr = wra;
end


if wl > 300
    wl = wla;
end


if wr < -300
    wr = wra;
end


if wl < -300
    wl = wla;
end
```

Figura 21: Programación defensiva contra valores altos de velocidad

8.3. Código de comparación de la simulación y la implementación física

Al momento de la implementación física se necesitaba comparar la simulación junto a lo que está sucediendo realmente. En este caso se tomó solo un robot móvil con ruedas para no saturar el envío de datos, donde se gráfica su trayectoria tanto simulado como en físico, con la finalidad de hacer comparaciones entre estos. La trayectoria simulada se puede observar en la Figura 22.

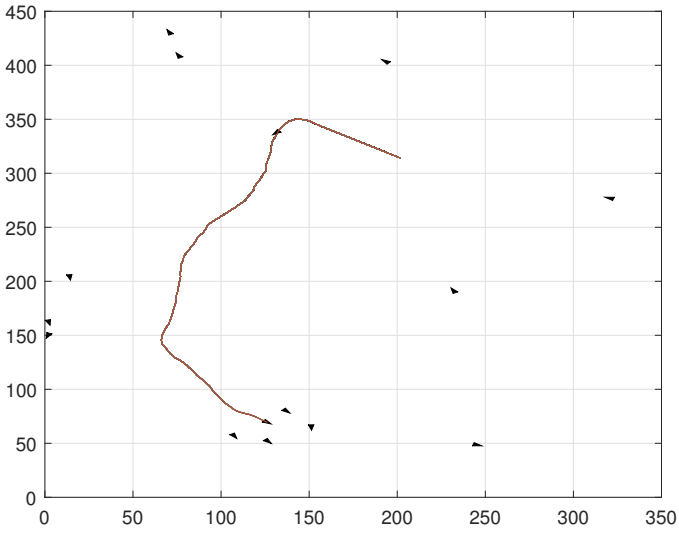

Figura 22: Gráfica de la trayectoria simulada

Para el caso de la trayectoria en físico únicamente se pide la solicitud de la posición al ecosistema Robotat mediante el marcador y se guardan los valores en un arreglo para utilizar el comando *plot* y graficarlo. Preferiblemente hacemos la gráfica luego de cada renderización, como se observa en la Figura 23.

```
for i=1:length(obj.boids)
    delete(plane.boids_figure_handles(i));
    theta = atan2(obj.boids(i).velocity(2),obj.boids(i).velocity(1));
    x = [obj.boids(i).position(1)-2.5 obj.boids(i).position(1)+2.5 obj.boids(i).position(1)-
    y = [obj.boids(i).position(2)-1.5 obj.boids(i).position(2) obj.boids(i).position(2)+1.5
    plane.boids_figure_handles(i) =  patch(x,y,'k');
    rotate(plane.boids_figure_handles(i), [0 0 1], rad2deg(theta), [obj.boids(i).position(1)
    cor = robotat_get_pose(obj.robotat, 3, 'eulxyz')

    obj.xval(obj.step_counter) = cor(1)
    obj.yval(obj.step_counter) = cor(2)
    if obj.xval(i)~=0 && obj.yval(i)~=0
        obj.xval(obj.step_counter) = cor(1)*100+380/2+10;
        obj.yval(obj.step_counter) = cor(2)*100+380/2+10;
        if obj.step_counter>30
            hold on;

            plot (obj.xval, obj.yval);

        end
    end
    grid on;
```

Figura 23: Código de la trayectoria en físico

CAPÍTULO 9

Validación y resultados

9.1. Pruebas y ajustes con un agente ejecutando simulación y envío de datos al mismo tiempo

Primero se ejecutó la simulación y se realizó el envío de los datos de las velocidades de cada rueda donde se pudo notar que los tiempos de la simulación y los robots con ruedas era completamente distinto, dado que la simulación avanzaba cinco veces más rápido que los Pololu 3pi+.

Posteriormente se optó por hacer más lenta la simulación, por lo que se estuvo ajustando la velocidad de simulación ubicando el robot móvil con ruedas en una pared y enviándolo hasta la pared de enfrente para lograr obtener un tiempo adecuado entre simulación y la implementación física.

9.2. Pruebas con dos agentes ejecutando simulación y envío de datos al mismo tiempo

Una vez obtenido una relación adecuada se procedió a utilizar dos robots para comprobar las reglas de los Boids de Reynolds. Al principio la estrategia de reducir el tiempo de simulación funcionó. Sin embargo, luego de 6 segundos los robots con ruedas empezaban a presentar errores cuadráticos medios mayores a 20 con respecto de la simulación, por lo que la implementación física superaban la velocidad de simulación.

Aun así, se realizaron pruebas para evaluar el comportamiento de los Boids de Reynolds en físico con los dos robots disponibles en el rango de tiempo efectivo, como se observa en

la Figura 24 y la Figura 25. Asimismo, para una mejor interpretación se puede encontrar un enlace a una grabación en anexos.

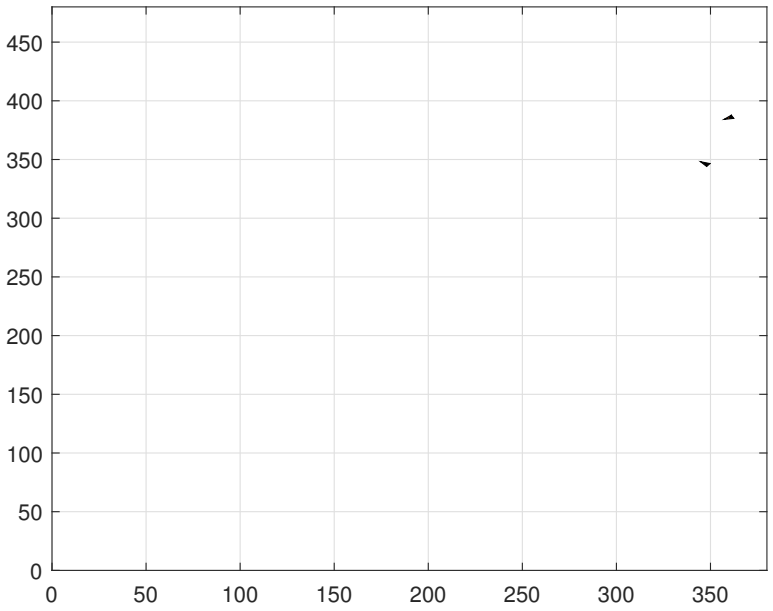

Figura 24: Simulación de dos agentes enviando datos

Figura 25: Dos agentes en físico recibiendo datos

9.3. Pruebas con varios agentes ejecutando simulación y envío de datos al mismo tiempo

Se utilizaron tres agentes pero mientras más avanzaba la simulación más lenta se volvía, mientras que la implementación física esperaba la información necesaria para seguir funcionando. Luego se utilizaron cinco agentes, como se observa en la Figura 26 y en este caso la simulación empezaba a volverse cada vez más lenta que con tres robots móviles, esto debido a que conforme más agentes se incluyan en la simulación mayores serán los procesos que debe ejecutar la computadora, tanto recepción y envío de datos.

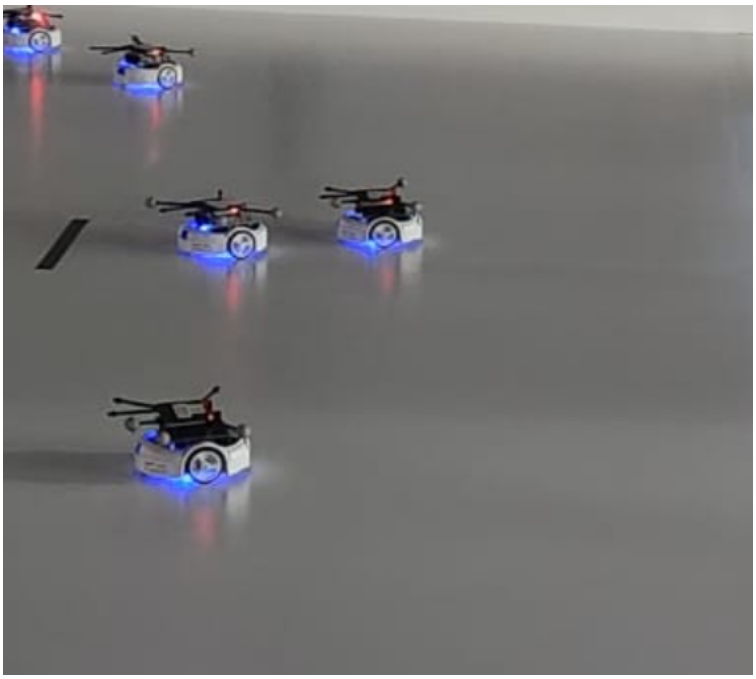

Figura 26: Implementación física con cinco agentes

Se realizaron varias pruebas con estos agentes para comprobar el comportamiento de los Boids de Reynolds. En el caso de la cohesión se pudo comprobar que los agentes se acercaban entre sí cada vez más conforme avanzaba la simulación.

El caso de la implementación de la separación también fue exitoso dado que los agentes no presentaron colisiones durante las pruebas hechas tanto con tres como con cinco agentes disponibles.

Por último, el caso de la alineación también fue exitoso dado que los agentes terminaban mirando a una sola dirección conforme se acercaban y entraban al rango de visión para la aplicación de la regla de alineación.

Sin embargo, a pesar de presentar los comportamientos deseados estos no eran similares a la simulación por lo que fue necesario cambiar de estrategia dado que las reglas anteriores solo se cumplían en un rango determinado de tiempo y conforme avanzaban el tiempo también empezaba a propagarse el error como se presenta en la siguiente sección.

9.4. Comparación de la simulación y la implementación física

9.4.1. Comparaciones de las posiciones finales de la simulación y un agente en físico

Durante las pruebas se configuraron diferentes velocidades para lograr que el agente se detuviera en el mismo punto que el de la simulación haciendo una línea recta. Sin embargo, se llego a una velocidad tan pequeña que el Pololu 3pi+ tardaba al rededor de diez segundos para desplazarse 25 cm y aun así no terminó en la misma posición que el de la simulación, como se observa en la Figura 27. teniendo un error cuadrático en el eje x de 32.0331 o en este caso un RMSE: de 5.66 cm, dado que no terminan en la misma posición. Sin embargo, en el eje y el RMSE: fue de 0.55 cm, algo esperado dado que solo nos desplazamos horizontalmente.

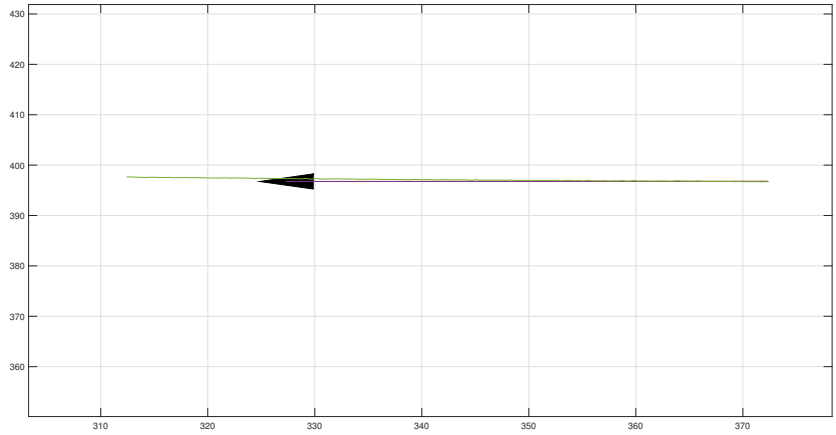

Figura 27: Comparación entre simulación e implementación física

Por lo mencionado anteriormente se decidió utilizar otra estrategia dado que aunque los comportamientos tanto físicos como en la simulación eran similares, solo lograban asegurar una semejanza durante los primeros 6 segundos para posteriormente ir incrementando el RMSE:.

El primer punto a tratar fue no ejecutar la simulación y el envió de datos a los robots móviles con ruedas al mismo tiempo, esto para reducir la carga de procesamiento en la computadora y evitar que la simulación empezara a volverse cada vez más lenta.

Por otro lado se decidió implementar controladores para lograr asegurar que el robot atraviese todas las posiciones proporcionadas por el algoritmo de la simulación. Sin embargo, vale la pena mencionar que la implementación física será considerablemente más lenta que la simulación, pero con una mayor precisión al únicamente imitar las trayectorias de la simulación.

9.4.2. Controlador de acercamiento exponencial

Haciendo uso de un controlador de acercamiento exponencial es posible asegurar que la implementación física será bastante similar a la simulación. Además de los valores de acercamiento exponencial también fue necesario ajustar tanto los valores del PID de orientación para obtener el comportamiento deseado, como se observa en la Figura 28.

```
% PID orientación
kpO = 2*5;
kiO = 0.0001;
kdO = 0;
EO = 0;
eO_1 = 0;

% Acercamiento exponencial
v0 = 0.5;
alpha = 0.7;
```

Figura 28: Ajustes de los valores del controlador de acercamiento exponencial

Como se mencionó en el capítulo 6 el coeficiente proporcional genera una salida proporcional a la diferencia entre la posición actual y la posición deseada. En el caso de la orientación fue necesario aumentarlo a 10 unidades para corregir la dirección lo más rápido posible.

Por otro lado, el término integral tiene en cuenta el error acumulado a lo largo del tiempo, por lo que si hay un error constante en la posición (por ejemplo, debido a fricciones o perturbaciones), el término integral se incrementará eliminando este error acumulado, en este caso funcionó adecuadamente un valor de 0.0001.

En la parte derivativa se predice el error y se actúa sobre este lo que proporciona estabilidad. Usualmente se emplea en menor medida que los coeficientes anteriores, incluso en este caso se anuló dicho valor.

Por último, para la velocidad inicial se definió una bastante pequeña para que la implementación física no avance demasiado rápido con la finalidad de evitar problemas con los puntos a recorrer, mientras que el *alpha* se definió con un valor de 0.7 unidades, en el caso de las tolerancias a las cuales el controlador decidirá si cambiar o no de punto se definió de 5 cm tanto para el eje x como para el eje y.

9.4.3. Interpolación de trayectorias

Dado que la simulación proporciona puntos demasiado cercanos entre sí, menores a 1 mm, esto hace que el Pololu 3pi+ ya haya pasado por varios puntos mientras el controlador aún esta en el primer ciclo de iteración. Por esta razón fue necesario hacer una separación más extensa de los puntos mediante la generación de trayectorias de múltiples ejes usando la función *bsplinepolytraj* para separar los puntos y darle tiempo al controlador de comunicarse con el ecosistema Robotat y procesar toda la información, caso contrario el Pololu 3pi+ empezara a dar vueltas al rededor de los puntos. En la Figura 29, se puede observar la nueva generación de puntos.

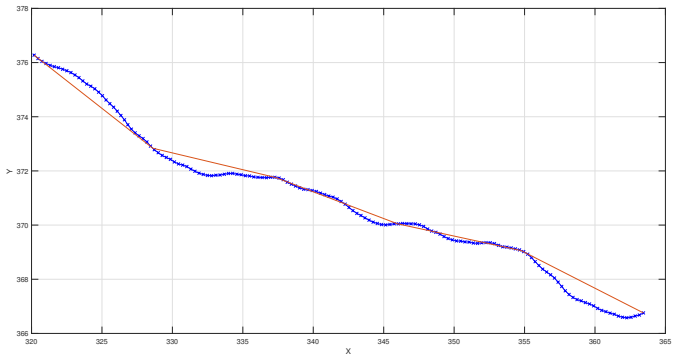

Figura 29: Interpolación de puntos sobre la trayectoria del Pololu 3pi+

Dado que se desea comparar la trayectoria en color rojo, que será la nueva trayectoria con 6 puntos, con la trayectoria original (azul) se optó por utilizar el RMSE: entre los puntos, siempre recordando que la escala esta en cm, se utilizó la función *immse* de Matlab: la cual proporciona el error cuadrático medio por lo que posteriormente se obtuvo la raíz cuadrada de este valor obteniendo una estimación de 0.6 cm. Los puntos comparados se pueden observar en la Figura 30, dado que no se puede comparar trayectorias de diferentes dimensiones se optó por tomar los puntos más alejados.

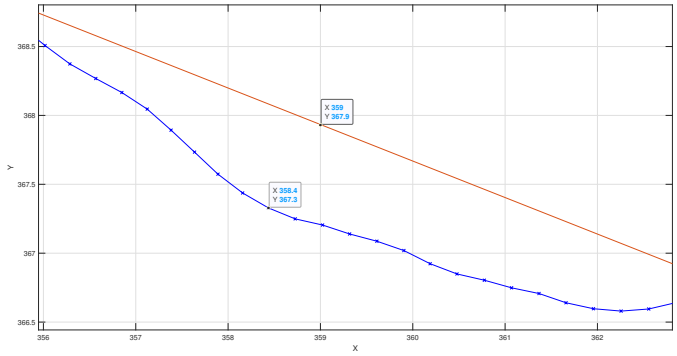

Figura 30: Comparación de los puntos más alejados entre la trayectoria original y la trayectoria generada

También es factible utilizar más puntos en la trayectoria original proporcionada por la simulación, pero es necesario reducir la velocidad a la que se mueven los Pololu 3pi+ haciendo que la implementación física sea considerablemente lenta, por lo que se definió utilizar 6 puntos hasta simulaciones de 500 renderizaciones.

9.4.4. Caso 1: Implementación sin obstáculos/depredador utilizando dos agentes

El primer caso a evaluar fue utilizando dos agentes avanzando por el eje x con la finalidad de evaluar la distancia euclidiana entre la simulación y la trayectoria recorrida en la implementación física, también se obtendrán porcentajes de error y se obtendrá la raíz del error cuadrático medio para evaluar la similitud entre estos RMSE:. Al inicio se ejecuta la simulación proporcionando las trayectorias de cada agente, como se observa en la Figura 31, donde se graficó únicamente la trayectoria del primer agente. Sin embargo, se tienen guardadas las trayectorias de ambos.

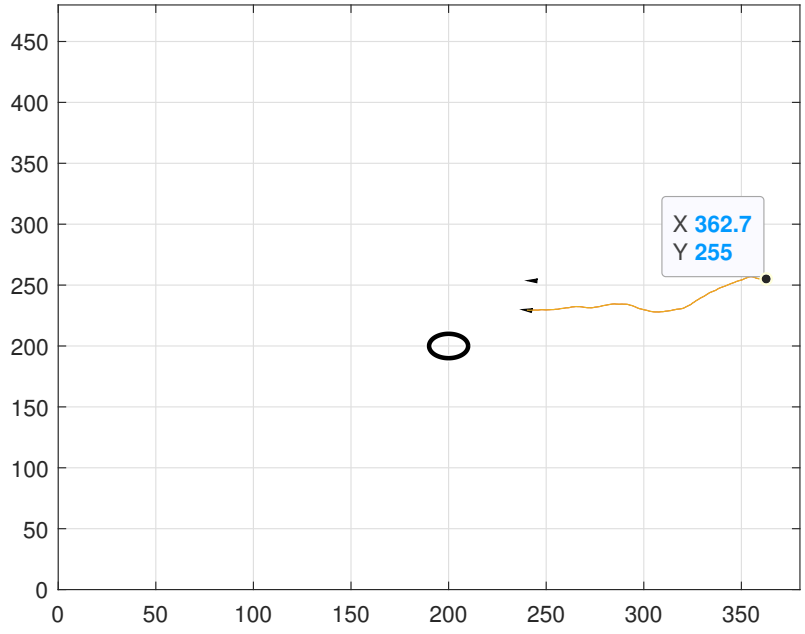

Figura 31: Caso 1. Dos agentes sin obstáculos/depredador

Posteriormente se realizó la interpolación de la trayectoria obteniendo 6 puntos a alcanzar por el controlador, en la Figura 32, se observa la trayectoria original (azul) y la nueva trayectoria (anaranjada).

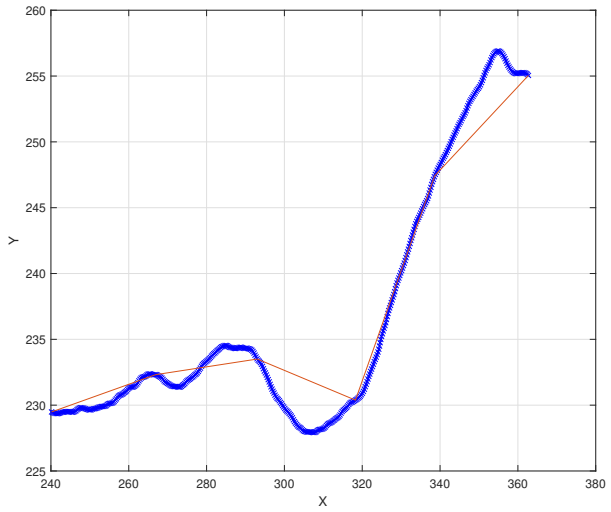

Figura 32: Interpolación de la trayectoria del caso 1

Luego de realizar la interpolación mencionada anteriormente se procedió a enviar datos al controlador exponencial para que los agentes alcanzaran las posiciones deseadas dando como resultado las trayectorias de la Figura 33, donde la trayectoria azul es la interpolada y la trayectoria de color anaranjado es la recorrida por los Pololu 3pi+.

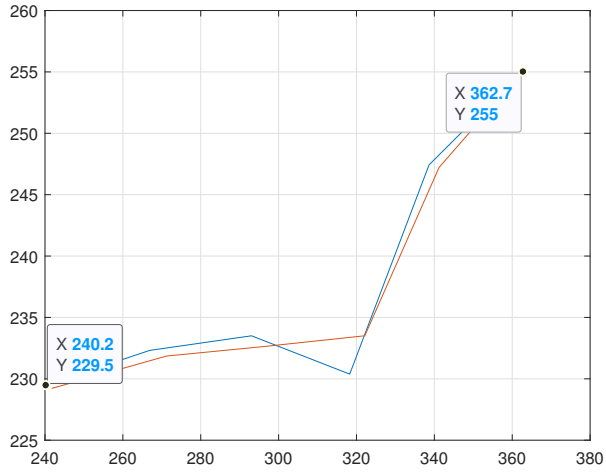

Figura 33: Caso 1. Trayectoria interpolada vs Trayectoria recorrida

Posteriormente se guardaron todos los puntos obtenidos de la Tabla 3 y se procedió a calcular la distancia euclidiana desde el origen de los valores simulados y los valores de la trayectoria recorrida, también se calculó la distancia euclidiana entre estos y el porcentaje de error dando como resultado los datos de la Tabla 4. Asimismo, se obtuvo el RMSE: al obtener la raíz cuadrada de la función *immse* de Matlab:, obteniendo un valor de 2.3078 cm.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	362.71	255.05	362.73	255.03
2	338.63	247.42	341.21	247.22
3	318.26	230.38	322.13	233.51
4	293.04	233.50	296.28	232.63
5	266.92	232.32	271.22	231.85
6	240.16	229.48	241.73	229.22

Cuadro 3: Datos del eje x y eje y obtenidos del primer agente del caso 1.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	443.41	443.41	0.03	0.01	0.01
2	419.39	421.36	2.59	0.76	0.08
3	392.89	397.86	4.97	1.22	1.36
4	374.69	376.70	3.36	1.11	0.37
5	353.86	356.81	4.32	1.61	0.20
6	332.18	333.13	1.59	0.65	0.12

Cuadro 4: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del caso 1.

En este caso fue posible notar que las distancias entre puntos no superan los 5 cm por lo que se puede asegurar que los agentes se mantienen en un rango aceptable de discrepancia dado que los porcentajes de error tampoco superan el 2 %, por lo que la implementación física es semejante a la simulación. En este prueba únicamente se tenía disponible un Pololu 3pi+, por lo que no se obtuvieron datos en físico del segundo agente. Sin embargo, las demás pruebas ya utilizan más robots con ruedas junto a sus trayectorias simuladas, interpoladas y recorridas.

9.4.5. Caso 2: Implementación con obstáculos/depredadores utilizando tres agentes

El segundo caso a evaluar fue utilizando tres agentes avanzando por el plano hasta encontrarse con el depredador u obstáculo para que posteriormente los agentes lo eviten donde una vez más se obtuvo la distancia euclidiana y porcentajes de error entre puntos. Al inicio se ejecuta la simulación proporcionando las trayectorias de cada agente, como se observa en la Figura 34.

Luego de ejecutar la simulación se guardan las trayectorias de los tres agentes en caso se quiera volver a ejecutar la simulación y posteriormente se mostrara la interpolación de la trayectoria de cada agente, dado que en este caso si se contaba con tres Pololu 3pi+ para las pruebas físicas.

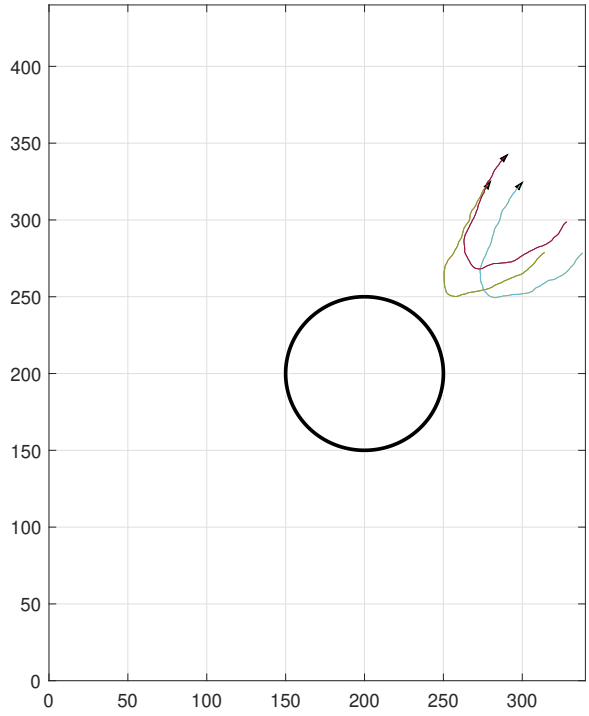

Figura 34: Caso 2. Tres agentes con un obstáculo/depredador

Posteriormente se realizaron las interpolaciones de las 500 renderizaciones donde de las tres trayectorias se obtuvieron 6 puntos para cada una, las cuales serán enviadas al controlador en cada iteración. En todas las trayectorias la original es la de color azul y la nueva trayectoria es la anaranjada, como se observa en la Figura 35, Figura 36 y Figura 37.

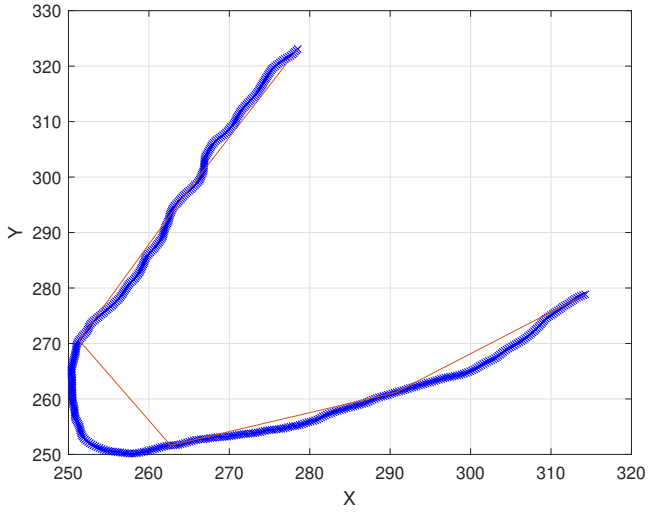

Figura 35: Interpolación de la trayectoria del primer agente para el caso 2

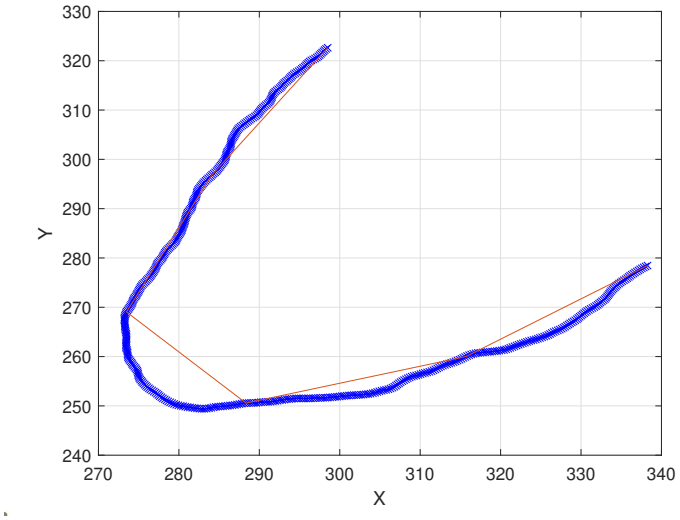

Figura 36: Interpolación de la trayectoria del segundo agente para el caso 2

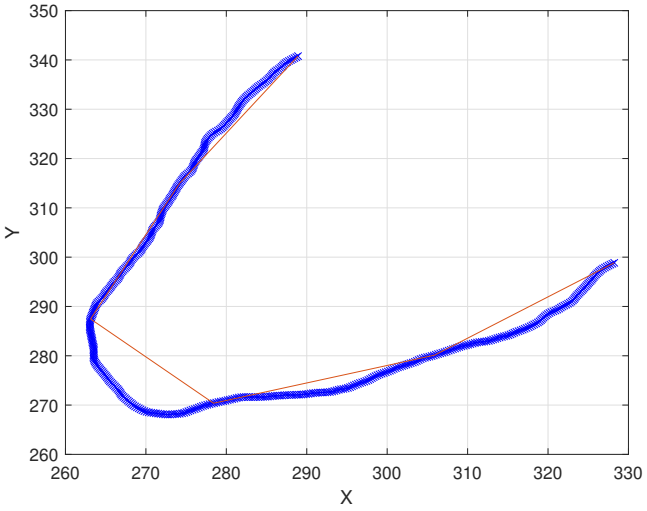

Figura 37: Interpolación de la trayectoria del tercer agente para el caso 2

Luego de realizar las interpolaciones se ejecutó la rutina del controlador exponencial, en donde cada vez que se alcance las tolerancias definidas, en este caso de 5 cm, el código le solicitará al ecosistema Robotat la posición en ese momento del Pololu 3pi+ para generar las trayectorias recorridas y comparar los resultados entre simulación e implementación física. En la Figuras 38, 39 y 40, se pueden visualizar las trayectorias de la implementación física en color anaranjado.

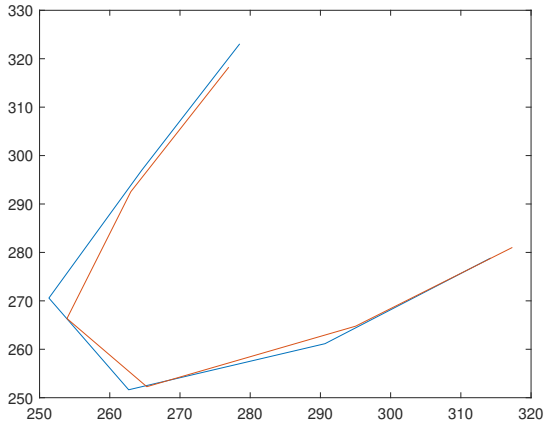

Figura 38: Trayectoria interpolada vs Trayectoria recorrida del primer agente para el caso 2

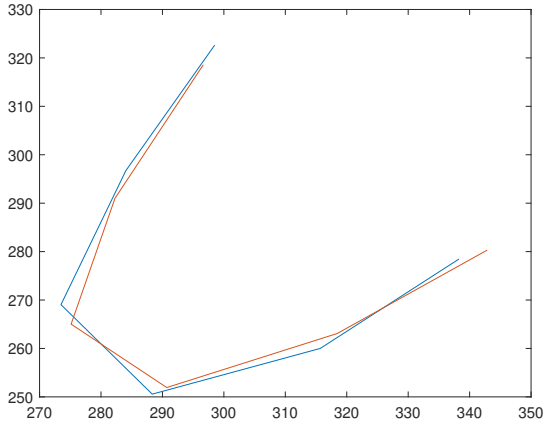

Figura 39: Trayectoria interpolada vs Trayectoria recorrida del segundo agente para el caso 2

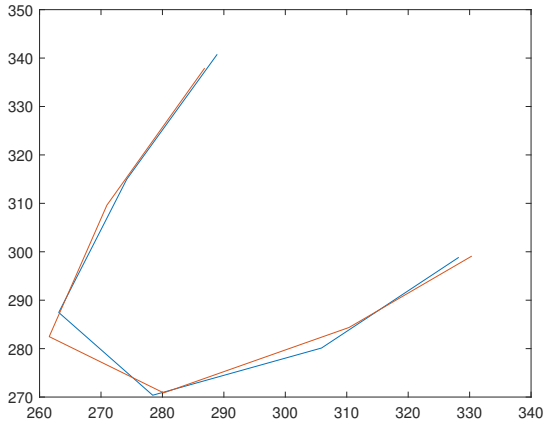

Figura 40: Trayectoria interpolada vs Trayectoria recorrida del tercer agente para el caso 2

Luego de haber obtenido los datos, al igual que el caso 1, se volvieron a generar las Tablas 5, 6 y 7 con los valores del eje x y eje y de las trayectorias simuladas y las trayectorias recorridas.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	314.26	278.84	317.33	281.03
2	290.65	261.15	295.02	264.77
3	262.69	251.64	265.27	252.27
4	251.33	270.58	253.90	266.35
5	264.54	296.95	263.00	292.55
6	278.51	323.07	276.95	318.25

Cuadro 5: Datos del eje x y eje y obtenidos del primer agente del caso 2.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	338.27	278.46	342.85	280.30
2	315.69	260.00	318.44	263.08
3	288.34	250.57	290.70	251.93
4	273.48	269.05	275.13	265.02
5	284.05	296.73	282.27	291.00
6	298.50	322.63	296.64	318.57

Cuadro 6: Datos del eje x y eje y obtenidos del segundo agente del caso 2.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	328.23	298.85	330.35	299.11
2	305.90	280.13	310.37	284.37
3	278.41	270.38	280.20	270.86
4	263.12	287.46	261.57	282.48
5	274.19	314.97	270.96	309.61
6	288.91	340.78	286.87	337.92

Cuadro 7: Datos del eje x y eje y obtenidos del tercer agente del caso 2.

Con la finalidad de comparar la similitud entre simulación e implementación física se calculó la distancia euclidiana desde el origen de los valores simulados y los valores de la trayectoria recorrida, la diferencia entre estos y el porcentaje de error, estos datos pueden observarse en las Tablas 8, 9 y 10.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	420.13	423.88	3.77	0.98	0.78
2	390.74	396.41	5.67	1.50	1.39
3	363.77	366.07	2.66	0.98	0.25
4	369.29	367.98	4.95	1.02	1.56
5	397.70	393.39	4.67	0.58	1.48
6	426.54	421.88	5.06	0.56	1.49

Cuadro 8: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del primer agente del caso 2.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	438.14	442.85	4.94	1.36	0.66
2	408.98	413.05	4.12	0.87	1.18
3	382.00	384.68	2.73	0.82	0.54
4	383.64	382.01	4.35	0.60	1.50
5	410.77	405.41	6.01	0.63	1.93
6	439.54	435.30	4.47	0.62	1.26

Cuadro 9: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del segundo agente del caso 2.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	443.90	445.64	2.14	0.65	0.09
2	414.78	420.95	6.16	1.46	1.51
3	388.10	389.71	1.85	0.64	0.18
4	389.70	384.98	5.22	0.59	1.73
5	417.60	411.43	6.26	1.18	1.70
6	446.76	443.26	3.51	0.71	0.84

Cuadro 10: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del tercer agente del caso 2.

Asimismo, se calcularon los RMSE: de cada agente, obteniendo un valor de 3.2324 cm para la primera trayectoria, un valor de 3.2124 cm para la segunda trayectoria y un valor de 3.2249 cm para la última trayectoria. Los datos anteriores indican que el RMSE: de las trayectorias no supero los 4 cm, también podemos observar las distancias euclidianas en las tablas anteriores donde no se superaron los 7 cm y los porcentajes de error tampoco superan el 2 %, por lo que podemos afirmar que la implementación física es semejante a la simulación.

9.4.6. Caso 3: Ajuste de límites utilizando tres agentes

El tercer caso que se evaluó consistió en utilizar tres agentes avanzando por el plano hasta encontrarse con uno de los límites de la simulación, se realizó este caso dado que se buscó comprobar qué tan bien responde la implementación física frente a un giro tan brusco y si los Pololu 3pi+ son capaz de ejecutarlo, además de no colisionar con las paredes de la plataforma. Al igual que en casos anteriores se evaluó la distancia euclidiana y porcentajes de error entre puntos. Luego de ejecutar la simulación se obtienen las trayectorias, como se observa en la Figura 41, donde se graficaron las trayectorias de cada agente.

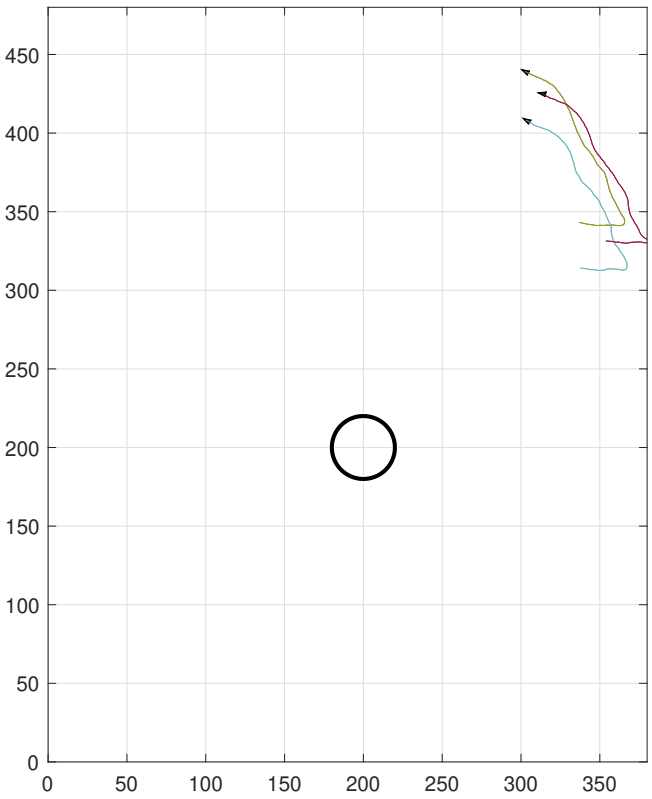

Figura 41: Caso 3. Ajuste de límites utilizando tres agentes

Este caso también cuenta con 500 renderizaciones, de las cuales se interpolaron 6 puntos para cada agente, las cuales fueron enviadas al controlador en cada iteración. En todas las trayectorias la original es la de color azul y la nueva trayectoria es la anaranjada, como se observa en la Figura 42, Figura 43 y Figura 44.

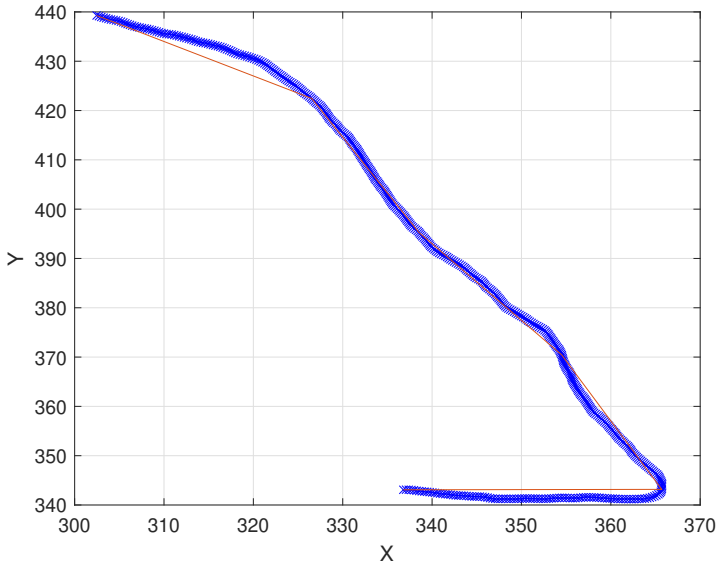

Figura 42: Interpolación de la trayectoria del primer agente para el caso 3

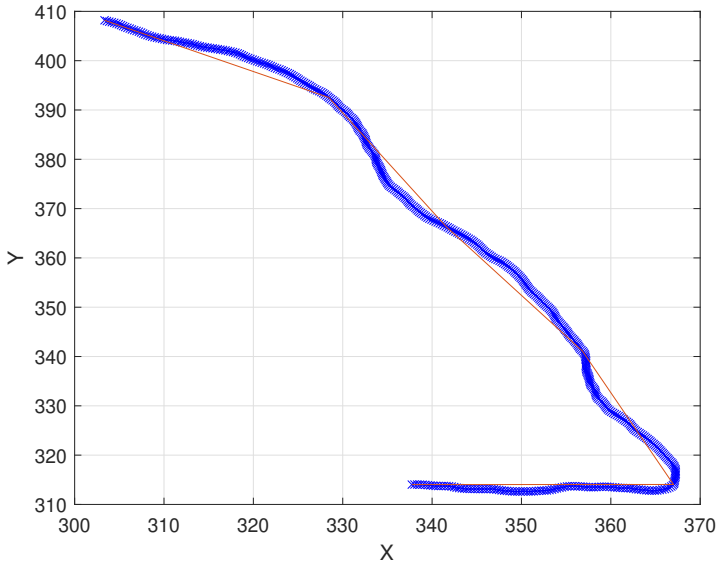

Figura 43: Interpolación de la trayectoria del segundo agente para el caso 3

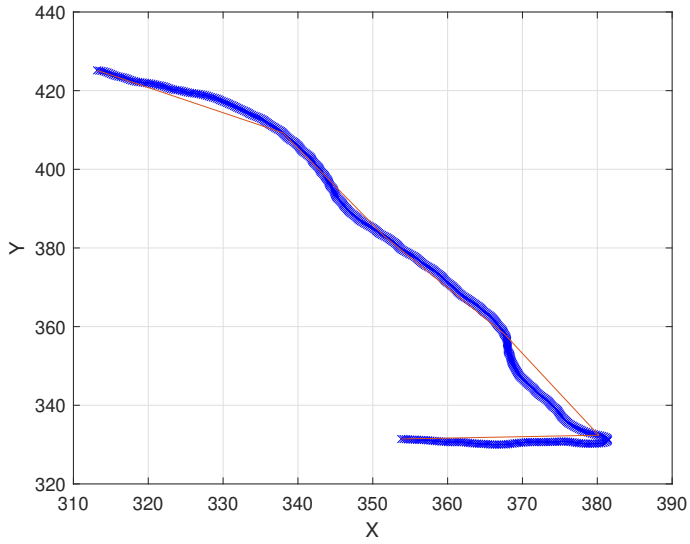

Figura 44: Interpolación de la trayectoria del tercer agente para el caso 3

Se puede notar que las interpolaciones son bastantes similares a las trayectorias por lo que para este caso se decidió aumentar la tolerancia a la cual el controlador cambia de punto, originalmente este estaba definido a 5 cm, pero en este caso fue necesario ajustarlo a 20 cm, que es el tamaño aproximado del Pololu 3pi+ junto al marcador, esto dado que se buscó reducir el giro brusco de la simulación para lograr que la implementación física tenga un giro más natural y realista a la hora de detectar un muro. Al ejecutar la implementación física se obtuvieron las Figuras 45, 46 y 47, donde se pueden visualizar las trayectorias de la implementación física en color anaranjado y las interpoladas en color azul.

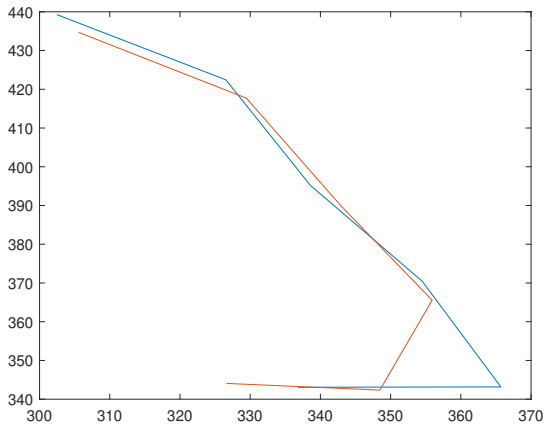

Figura 45: Trayectoria interpolada vs Trayectoria recorrida del primer agente para el caso 3

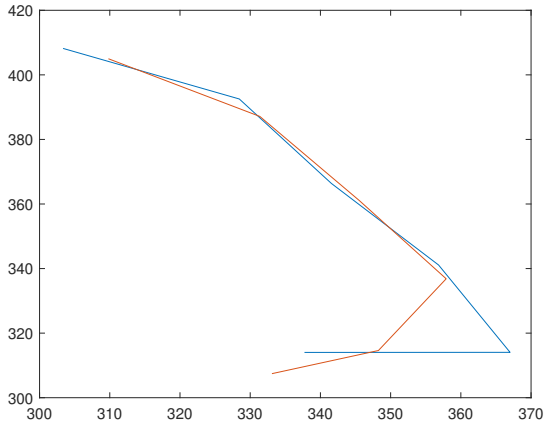

Figura 46: Trayectoria interpolada vs Trayectoria recorrida del segundo agente para el caso 3

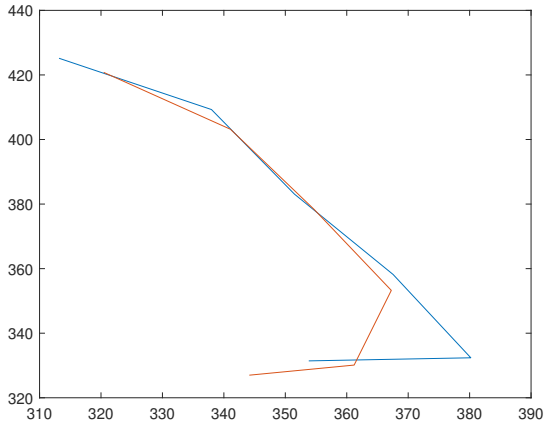

Figura 47: Trayectoria interpolada vs Trayectoria recorrida del tercer agente para el caso 3

Al igual que en los casos anteriores, se volvieron a generar las Tablas 11, 12 y 13 con los valores del eje x y eje y de las trayectorias simuladas y las trayectorias recorridas todas en centímetros.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	336.78	343.08	326.65	344.07
2	365.70	343.17	348.47	342.36
3	354.50	370.45	355.92	365.54
4	338.54	395.25	342.96	389.85
5	326.53	422.45	329.46	417.69
6	302.46	439.27	305.55	434.70

Cuadro 11: Datos del eje x y eje y obtenidos del primer agente del caso 3.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	337.73	314.03	333.09	307.45
2	367.01	314.05	348.26	314.62
3	356.82	341.11	357.91	336.84
4	341.57	366.36	345.50	361.14
5	328.45	392.53	331.37	387.12
6	303.35	408.19	309.81	404.99

Cuadro 12: Datos del eje x y eje y obtenidos del segundo agente del caso 3.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	353.82	331.43	344.14	327.00
2	380.19	332.39	361.20	330.13
3	367.55	358.19	367.25	353.26
4	351.50	383.07	354.69	378.35
5	337.98	409.28	341.08	403.17
6	313.17	425.16	320.45	420.79

Cuadro 13: Datos del eje x y eje y obtenidos del tercer agente del caso 3.

Además, con la finalidad de comparar la similitud entre simulación e implementación física se calculó la distancia euclidiana desde el origen (DE) de los valores simulados y los valores de la trayectoria recorrida, la diferencia entre estos (DEP) y el porcentaje de error tanto en el eje x como el eje y, estos datos pueden observarse en las Tablas 14, 15 y 16.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	480.76	474.43	10.18	3.01	0.29
2	501.50	488.50	17.25	4.71	0.24
3	512.74	510.20	5.11	0.40	1.32
4	520.41	519.23	6.98	1.30	1.37
5	533.93	531.99	5.59	0.90	1.13
6	533.33	531.34	5.52	1.02	1.04

Cuadro 14: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del primer agente del caso 3.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	461.17	453.29	8.05	1.37	2.10
2	483.04	469.33	18.76	5.11	0.18
3	493.64	491.49	4.41	0.30	1.25
4	500.89	499.79	6.53	1.15	1.42
5	511.82	509.58	6.15	0.89	1.38
6	508.57	509.90	7.21	2.13	0.78

Cuadro 15: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del segundo agente del caso 3.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	484.80	474.72	10.64	2.73	1.34
2	505.00	489.34	19.13	5.00	0.68
3	513.22	509.58	4.94	0.08	1.38
4	519.89	518.60	5.70	0.91	1.23
5	530.79	528.09	6.85	0.92	1.49
6	528.05	528.91	8.49	2.32	1.03

Cuadro 16: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del tercer agente del caso 3.

Asimismo, se calcularon los RMSE: de cada agente, obteniendo un valor de 6.6939 cm para la primera trayectoria, un valor de 6.8822 cm para la segunda trayectoria y un valor de 7.3877 cm para la última trayectoria. Los datos anteriores indican que el RMSE: de las trayectorias no superó los 7.5 cm, también podemos observar las distancias euclidianas en las tablas anteriores donde no se superaron los 20 cm y los porcentajes de error tampoco superaron el 6 %, por lo que podemos afirmar que la implementación física es semejante a la simulación.

Aun así, se tomó la decisión de reducir el espacio donde funciona la plataforma de 380 cm en x a 340 cm y de 480 cm en y a 440 cm para no tener que variar las tolerancias de 5 cm a 20 cm, esto con la finalidad de tener una trayectoria más semejante a la interpolada y también evitar que los Pololu 3pi+ colisionen con los muros, así evitando tener una distancia euclidiana entre los puntos de 20 cm que es 4 veces mayor a la de los dos casos anteriores.

9.4.7. Caso 4: Implementación con obstáculos/depredadores utilizando cinco agentes

El cuarto caso consistió en utilizar cinco agentes avanzando por el plano hasta encontrarse con uno de los depredadores de la simulación, se realizó este caso dado que se buscó comprobar que tan bien funciona la implementación física con un mayor número de agentes y si es necesario realizar ajustes en el controlador. Al igual que en casos anteriores se evaluó la distancia euclidiana y porcentajes de error entre puntos para asegurar la semejanza entre simulación e implementación física. Luego de ejecutar la simulación se obtienen las trayectorias, como se observa en la Figura 48, donde se graficaron las trayectorias de los 5 agentes.

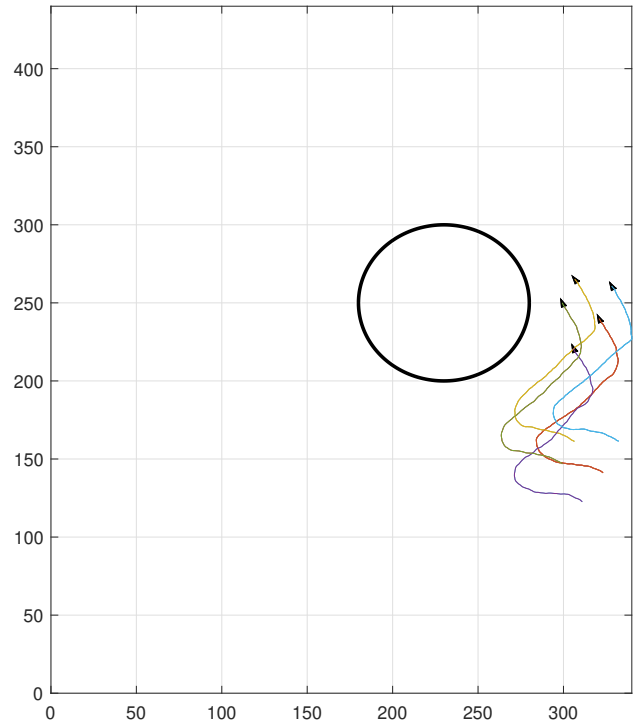

Figura 48: Caso 4. Cinco agentes con un obstáculo/depredador

Este caso también cuenta con 500 renderizaciones, de las cuales se interpolaron 6 puntos para cada agente siendo estas enviadas al controlador en cada iteración. En todas las trayectorias la original es la de color azul y la nueva trayectoria es la anaranjada, como se observa en la Figura 49, Figura 50, Figura 51, Figura 52 y Figura 53.

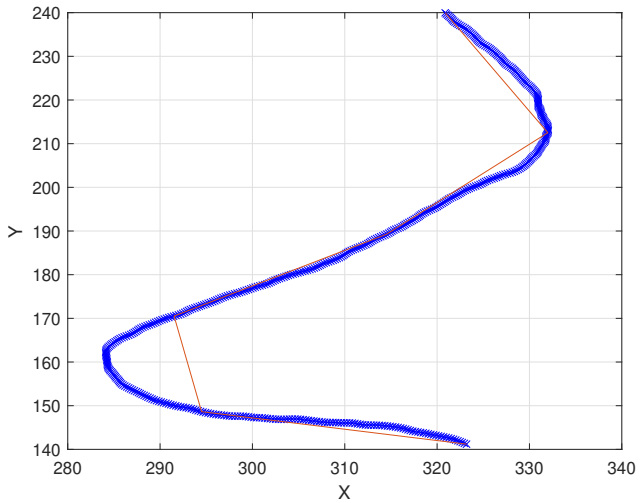

Figura 49: Interpolación de la trayectoria del primer agente para el caso 4

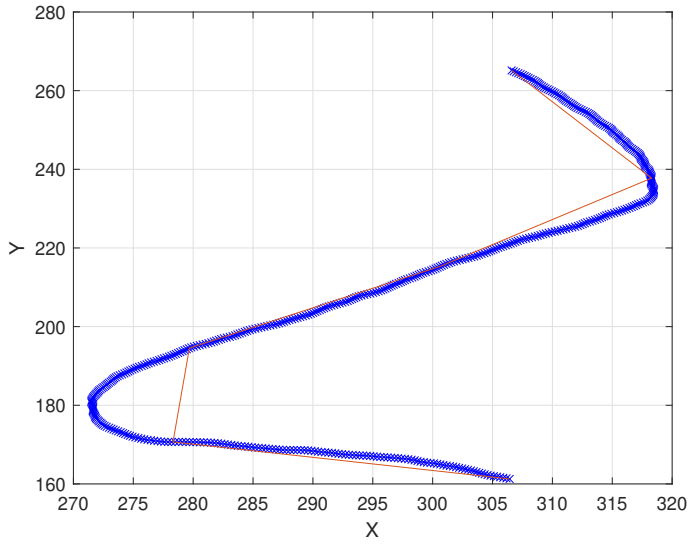

Figura 50: Interpolación de la trayectoria del segundo agente para el caso 4

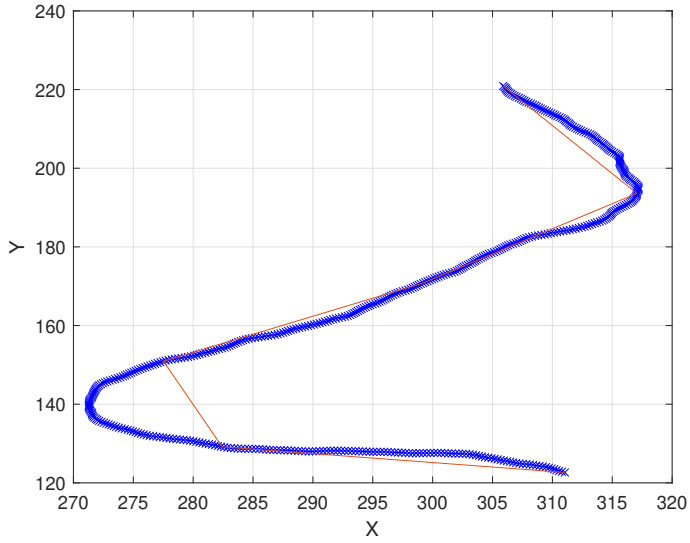

Figura 51: Interpolación de la trayectoria del tercer agente para el caso 4

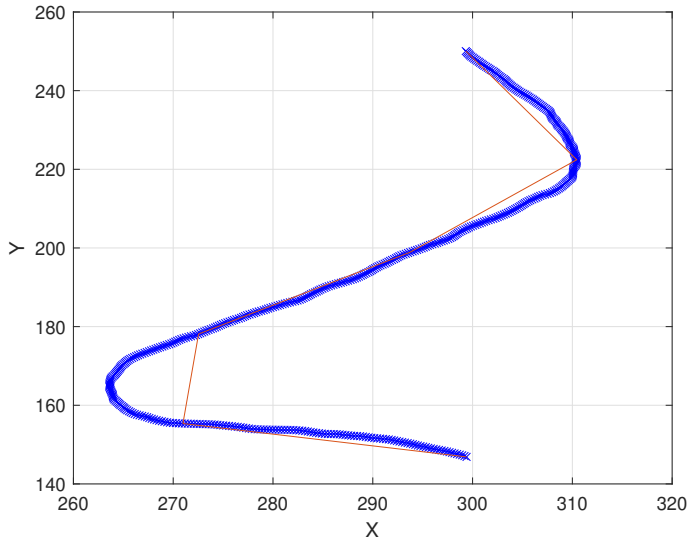

Figura 52: Interpolación de la trayectoria del cuarto agente para el caso 4

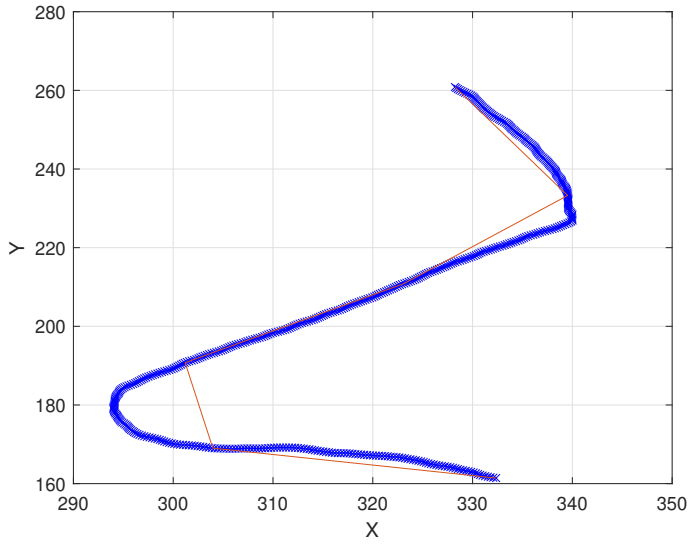

Figura 53: Interpolación de la trayectoria del quinto agente para el caso 4

Podemos notar que las interpolaciones son bastantes similares a las trayectorias, aun así, en el primer giro de cada agente la trayectoria interpolada varia visualmente de la simulada, la más alta tienen una diferencia de 9 cm con un porcentaje de error de 3.06 % en el eje x y un porcentaje de error de 0.03 % en el eje y por lo que está en un rango aceptable. Además, esta interpolación evitaría un cambio demasiado brusco en el giro a la hora de ejecutar la

implementación física.

En este caso se decidió mantener la tolerancia de 5 cm a la cual el controlador cambia de punto. Al ejecutar la implementación física se obtuvieron las Figuras 54, 55, 56, 57 y 58, donde se pueden visualizar las trayectorias de la implementación física en color anaranjado y las interpoladas en color azul.

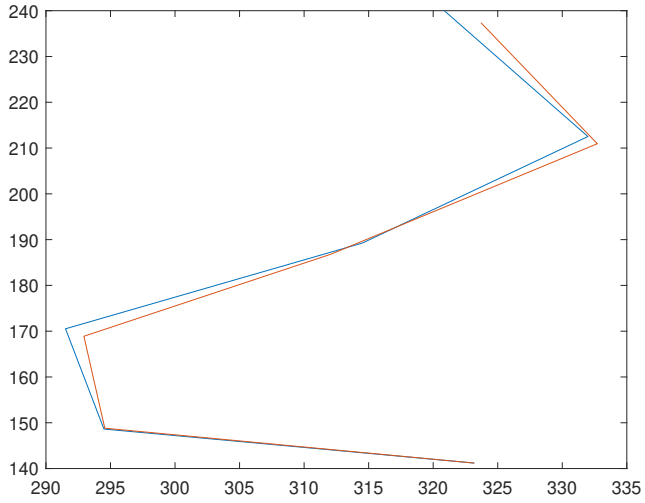

Figura 54: Trayectoria interpolada vs Trayectoria recorrida del primer agente para el caso 4

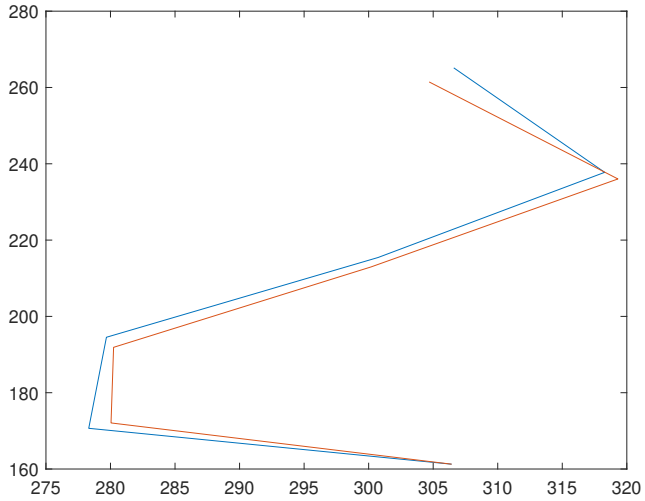

Figura 55: Trayectoria interpolada vs Trayectoria recorrida del segundo agente para el caso 4

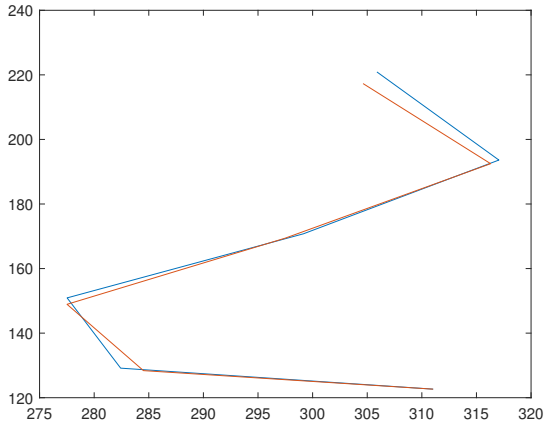

Figura 56: Trayectoria interpolada vs Trayectoria recorrida del tercer agente para el caso 4

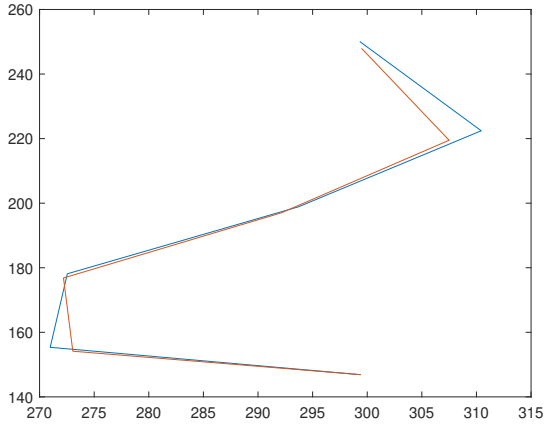

Figura 57: Trayectoria interpolada vs Trayectoria recorrida del cuarto agente para el caso 4

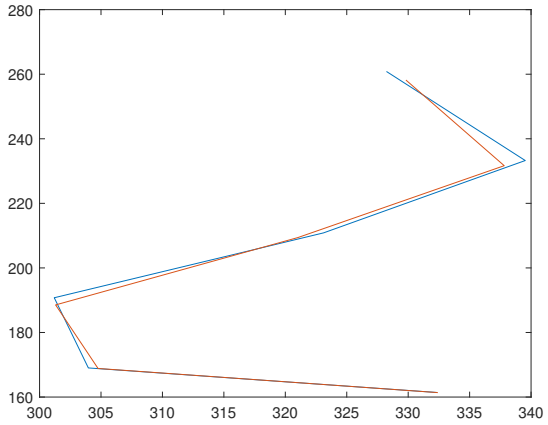

Figura 58: Trayectoria interpolada vs Trayectoria recorrida del quinto agente para el caso 4

Nuevamente se volvieron a generar las Tablas 17, 18, 19, 20 y 21 con los valores obtenidos del eje x y eje y tanto de las trayectorias simuladas como de las trayectorias recorridas en centímetros.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	323.19	141.19	323.18	141.19
2	294.49	148.63	294.56	148.82
3	291.52	170.52	292.94	168.89
4	314.51	189.25	312.06	186.80
5	331.99	212.51	332.72	210.94
6	320.88	239.96	323.70	237.36

Cuadro 17: Datos del eje x y eje y obtenidos del primer agente del caso 4.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	306.43	161.27	306.43	161.27
2	278.32	170.67	280.04	172.08
3	279.69	194.53	280.24	191.89
4	300.73	215.44	300.18	212.96
5	318.28	237.76	319.31	236.02
6	306.59	265.15	304.69	261.45

Cuadro 18: Datos del eje x y eje y obtenidos del segundo agente del caso 4.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	311.04	122.65	311.03	122.65
2	282.43	129.15	284.51	128.39
3	277.52	150.92	277.52	148.93
4	299.16	170.77	297.48	169.41
5	317.05	193.61	316.30	192.49
6	305.89	220.90	304.62	217.30

Cuadro 19: Datos del eje x y eje y obtenidos del tercer agente del caso 4.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	299.40	146.87	299.40	146.87
2	270.97	155.35	273.05	154.15
3	272.54	178.12	272.19	176.82
4	293.71	198.88	292.16	197.04
5	310.44	222.43	307.50	219.50
6	299.32	250.06	299.48	247.94

Cuadro 20: Datos del eje x y eje y obtenidos del cuarto agente del caso 4.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	332.36	161.41	332.41	161.38
2	303.97	169.02	304.76	168.85
3	301.20	190.76	301.31	188.54
4	323.11	210.86	321.02	209.40
5	339.53	233.26	337.82	231.66
6	328.23	260.85	329.82	258.20

Cuadro 21: Datos del eje x y eje y obtenidos del quinto agente del caso 4.

Una vez más, con la finalidad de comparar la similitud entre simulación e implementación física se calculó la distancia euclidiana desde el origen (DE) de los valores simulados y los valores de la trayectoria recorrida, la diferencia entre estos (DEP) y el porcentaje de error tanto en el eje x como el eje y, estos datos pueden observarse en las Tablas 22, 23, 24, 25 y 26.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	352.68	352.68	0.00	0.00	0.00
2	329.87	330.02	0.20	0.02	0.13
3	337.73	338.14	2.17	0.49	0.96
4	367.06	363.70	3.46	0.78	1.29
5	394.18	393.95	1.73	0.22	0.74
6	400.68	401.40	3.84	0.88	1.08

Cuadro 22: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del primer agente del caso 4.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	346.27	346.27	0.00	0.00	0.00
2	326.48	328.68	2.23	0.62	0.82
3	340.69	339.64	2.70	0.20	1.36
4	369.93	368.05	2.53	0.18	1.15
5	397.28	397.07	2.02	0.32	0.73
6	405.34	401.49	4.16	0.62	1.39

Cuadro 23: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del segundo agente del caso 4.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	334.35	334.34	0.01	0.00	0.00
2	310.56	312.14	2.22	0.74	0.59
3	315.90	314.95	1.99	0.00	1.32
4	344.47	342.34	2.16	0.56	0.80
5	371.50	370.26	1.36	0.24	0.58
6	377.31	374.18	3.82	0.41	1.63

Cuadro 24: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del tercer agente del caso 4.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	333.48	333.48	0.00	0.00	0.00
2	312.35	313.55	2.40	0.76	0.78
3	325.58	324.58	1.34	0.13	0.73
4	354.71	352.40	2.41	0.53	0.93
5	381.90	377.80	4.15	0.95	1.32
6	390.03	388.80	2.12	0.06	0.85

Cuadro 25: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del cuarto agente del caso 4.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	369.48	369.51	0.07	0.02	0.02
2	347.80	348.41	0.81	0.26	0.10
3	356.52	355.43	2.22	0.04	1.16
4	385.83	383.28	2.54	0.65	0.69
5	411.93	409.62	2.34	0.50	0.69
6	419.26	418.86	3.09	0.48	1.02

Cuadro 26: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del quinto agente del caso 4.

Asimismo, se calcularon los RMSE: de cada agente, obteniendo un valor de 1.6948 cm para la primera trayectoria, un valor de 1.8253 cm para la segunda trayectoria, un valor de 1.5810 cm para la tercera trayectoria, 1.7099 cm para la cuarta trayectoria y 1.5023 cm para la última trayectoria. Los datos anteriores indican que el RMSE: de las trayectorias no supero los 2 cm, también se observa que las distancias euclidianas en las tablas anteriores no superaron los 5 cm y los porcentajes de error tampoco sobrepasaron el 2 %, por lo que podemos afirmar que la implementación física es comparable a la simulación.

En este caso particular se realizaron cambios al controlador de acercamiento exponencial cambiando la velocidad inicial a un valor de 0.1 y el *alpha* a un valor de 0.5. Asimismo, esta simulación funcionó en el primer intento y es por lo que se observa que los primeros puntos tanto en el eje x como en el eje y presentan errores de casi 0 % en algunos casos.

9.4.8. Caso 5: Implementación con obstáculos/depredadores utilizando ocho agentes

Luego de haber comprobado que el algoritmo funcionaba con cinco agentes, se planteó aumentar el número de Pololu 3pi+ a utilizar. En este caso el laboratorio de la Universidad del Valle de Guatemala contaba con ocho robots móviles al momento de las pruebas, por lo que se ejecutó la simulación de la Figura 59, para obtener las trayectorias de cada agente.

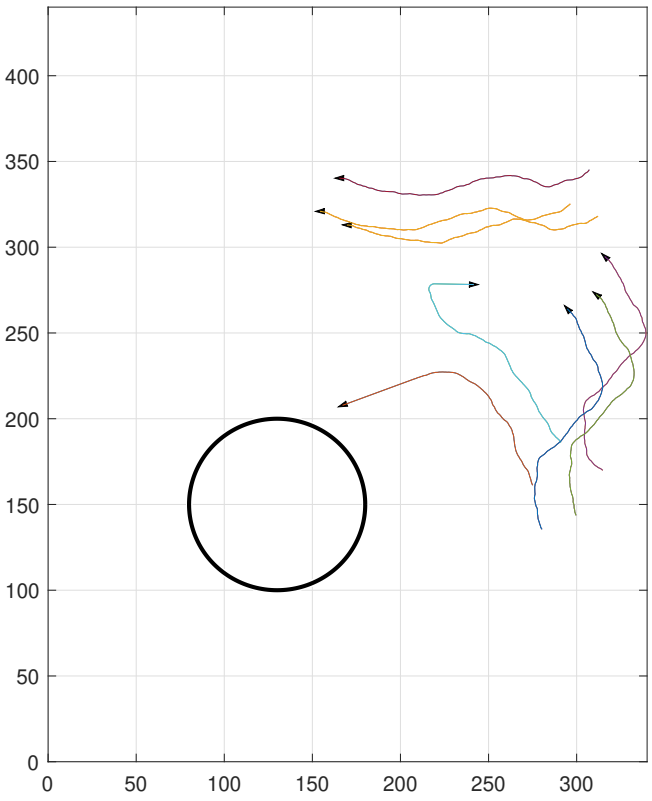

Figura 59: Caso 5. Implementación con ocho agentes

Al igual que en casos anteriores se obtuvieron las trayectorias de la simulación, se calcularon las nuevas trayectorias interpoladas, pero al momento de la implementación física los Pololu 3pi+ no fueron capaces de seguirlas, esto debido a las limitaciones del envío y recepción de datos.

Se realizaron varios intentos, pero en repetidas veces el servidor se desconectó de la computadora, como se observa en la Figura 60, y en las ocasiones donde no se desconectó el servidor, el envío y recepción de datos tardó tanto que la implementación física empezó a presentar complicaciones y propagaciones de error donde no es capaz de alcanzar los puntos interpolados, en anexos puede encontrarse un vídeo de esta prueba, donde al inicio siguen la trayectoria pero terminan colisionando entre sí.

```
ERROR: Could not receive data from server.
Output argument "mocap_data" (and maybe others) not assigned during call to "robotat_get_pose".

Error in boids_model (line 335)
                roba = robotat_get_pose(robot, 1+i, 'eulxyz'); % i + 1, Obtener la posicion actual
```

Figura 60: Desconexión del servidor

Incluso se realizaron cambios en la velocidad inicial del controlador exponencial con valores menores a 0.1, donde algunos de los agentes lograron alcanzar el tercer punto de la trayectoria interpolada, pero luego de ello volvieron a colisionar. Habiendo observado los resultados de esta prueba se decidió únicamente utilizar un máximo de cinco robots móviles con ruedas dado que un número mayor puede presentar las complicaciones mencionadas anteriormente.

9.4.9. Caso 6: Implementación con obstáculos/depredadores utilizando cinco agentes separados en dos grupos

Luego de haber comprobado las limitaciones de la implementación física se volvió a plantear un caso utilizando cinco agentes donde estos se encontrarían con el depredador y posteriormente se separarían en dos grupos como se observa en la Figura 61, donde se obtuvieron las trayectorias de cada agente.

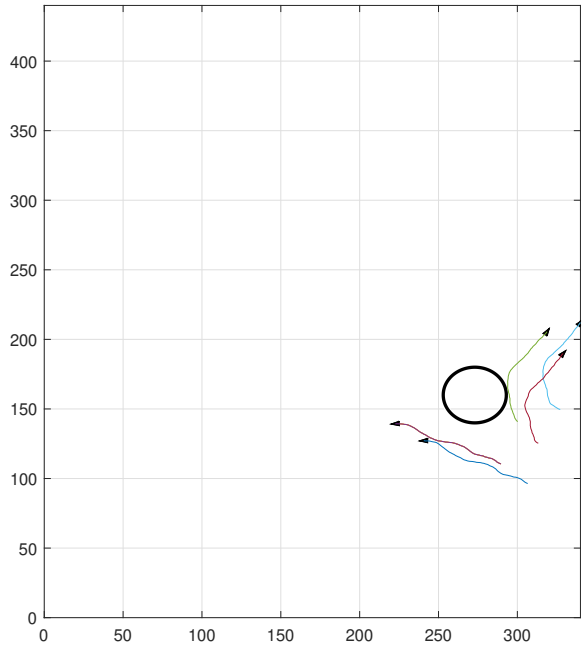

Figura 61: Caso 6. Implementación con cinco agentes separándose en dos grupos

Este caso contó con 250 renderizaciones, a diferencia de los casos pasados con 500 renderizaciones, de las cuales se interpolaron 6 puntos para cada agente igual que los casos anteriores, por lo que a diferencia del caso 4 los puntos que deben alcanzar los Pololu 3pi+ estarán más cercanos entre ellos. Esto se realizó con la finalidad de evaluar si el mismo número de puntos interpolados es adecuado para menores recorridos de 250 renderizaciones y, si es posible, mantener las configuraciones originales del controlador o es necesario realizar ajustes.

Al igual que en casos anteriores se trazaron las interpolaciones donde la original es la de color azul y la nueva trayectoria es la anaranjada, como se observa en la Figura 62, Figura 63, Figura 64, Figura 65 y Figura 66.

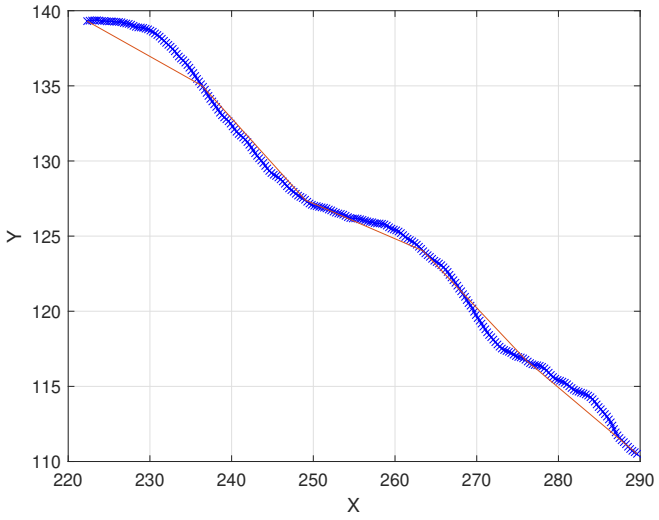

Figura 62: Interpolación de la trayectoria del primer agente para el caso 6

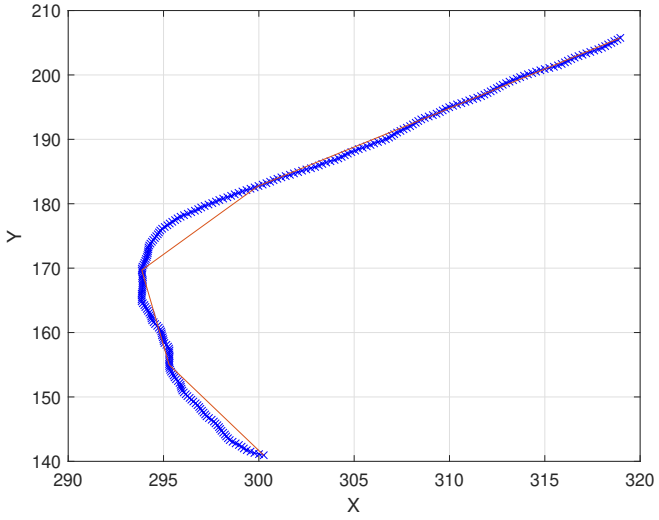

Figura 63: Interpolación de la trayectoria del segundo agente para el caso 6

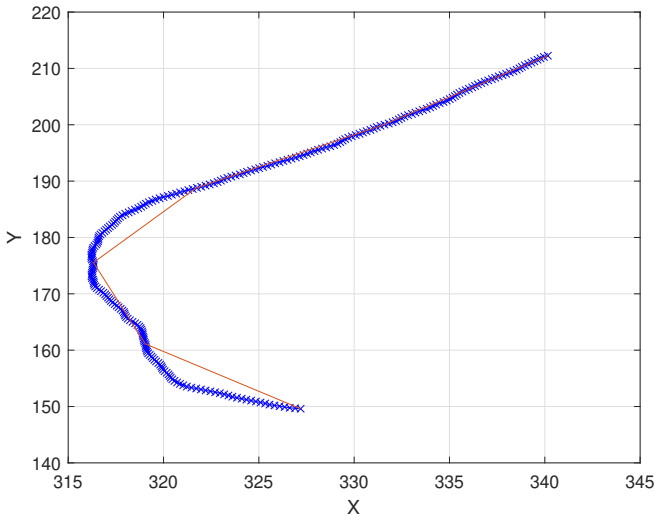

Figura 64: Interpolación de la trayectoria del tercer agente para el caso 6

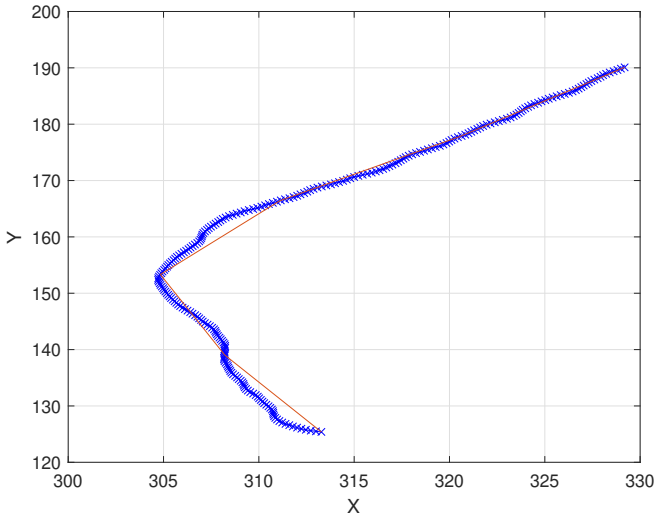

Figura 65: Interpolación de la trayectoria del cuarto agente para el caso 6

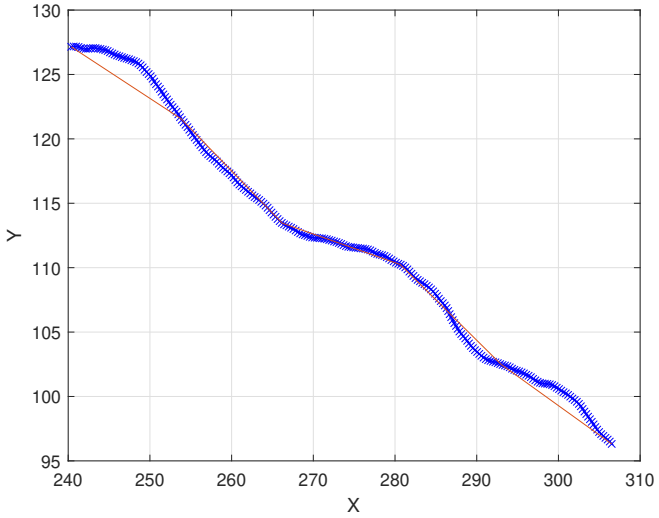

Figura 66: Interpolación de la trayectoria del quinto agente para el caso 6

Una vez más las interpolaciones son bastantes similares a las trayectorias originales e incluso más precisas que los casos anteriores dado que al solo tener 250 renderizaciones y mantener 6 puntos de interpolación la función *bsplinepolytraj* se adapta mejor a la trayectoria. En este caso se decidió mantener la tolerancia de 5 cm a la cual el controlador cambia de punto lo que al ejecutar la implementación física proporcionó las Figuras 67, 68, 69, 70 y 71, donde se pueden visualizar las trayectorias de la implementación física en color anaranjado y las interpoladas en color azul.

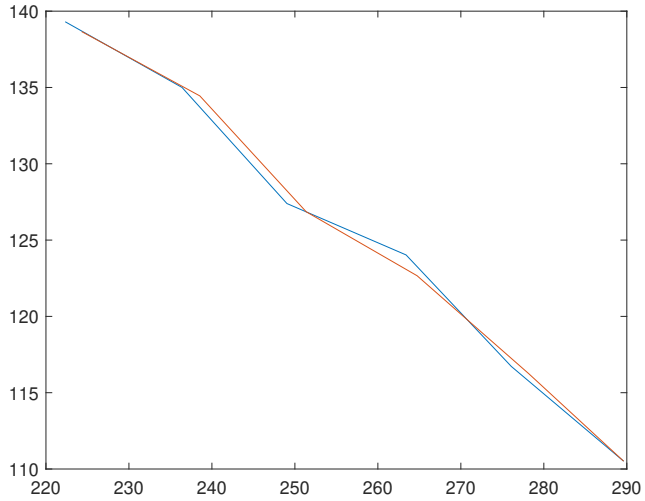

Figura 67: Trayectoria interpolada vs Trayectoria recorrida del primer agente para el caso 6

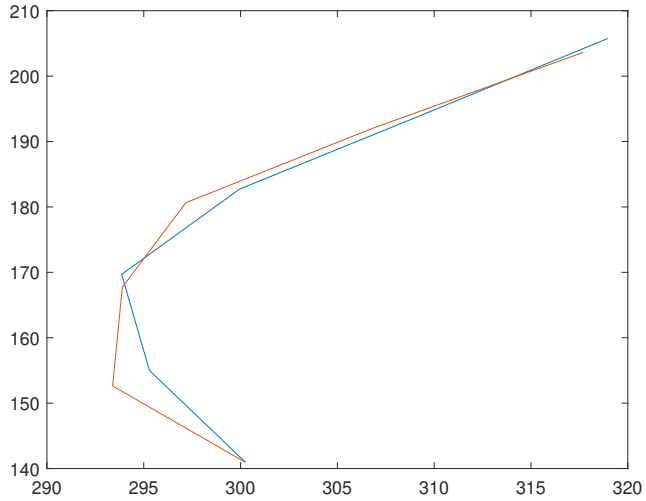

Figura 68: Trayectoria interpolada vs Trayectoria recorrida del segundo agente para el caso 6

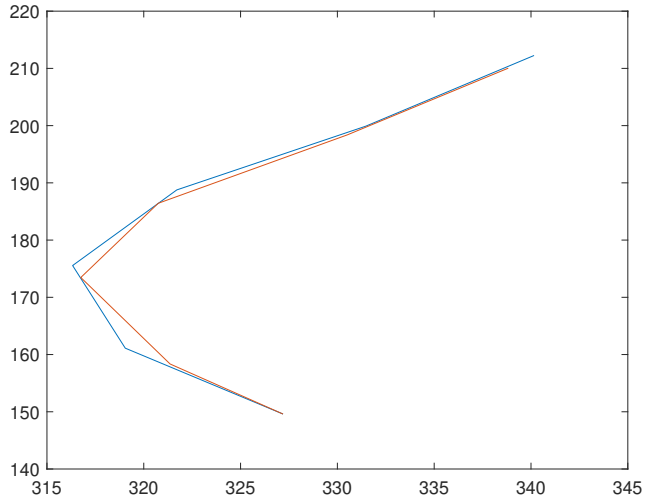

Figura 69: Trayectoria interpolada vs Trayectoria recorrida del tercer agente para el caso 6

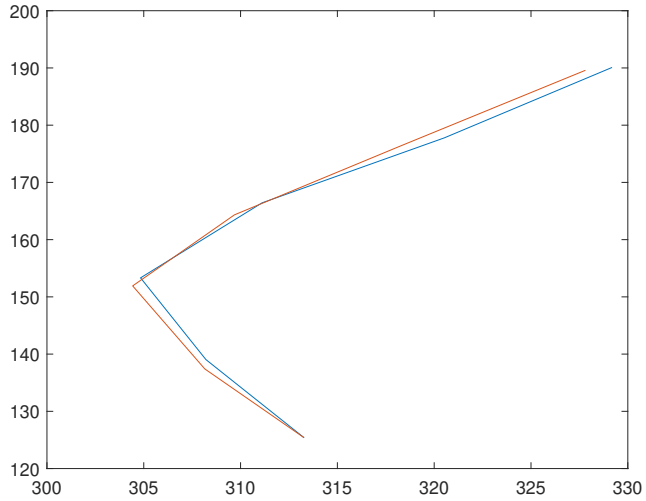

Figura 70: Trayectoria interpolada vs Trayectoria recorrida del cuarto agente para el caso 6

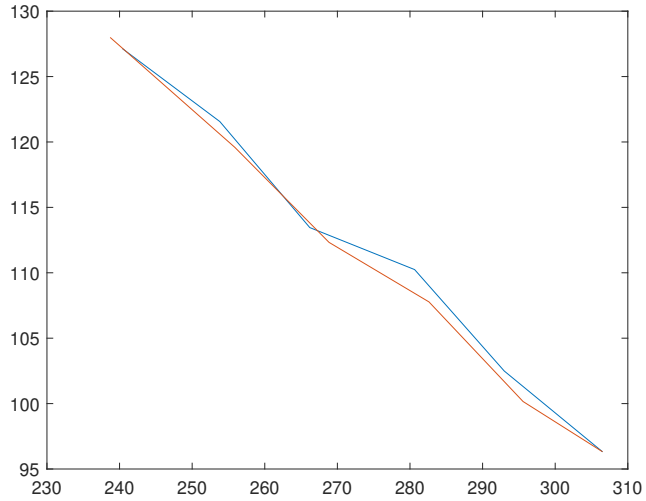

Figura 71: Trayectoria interpolada vs Trayectoria recorrida del quinto agente para el caso 6

Reiteradamente se volvieron a generar las Tablas 27, 28, 29, 30 y 31 con los valores obtenidos del eje x y eje y tanto de las trayectorias simuladas como de las trayectorias recorridas en centímetros.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	289.62	110.51	289.62	110.51
2	276.08	116.72	278.18	116.25
3	263.40	124.03	264.73	122.66
4	249.04	127.39	251.36	126.86
5	236.40	135.00	238.54	134.45
6	222.32	139.31	224.31	138.66

Cuadro 27: Datos del eje x y eje y obtenidos del primer agente del caso 6.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	300.26	140.95	300.26	140.95
2	295.29	155.00	293.39	152.63
3	293.86	169.70	293.89	167.73
4	299.94	182.72	297.18	180.65
5	309.39	194.09	306.81	192.01
6	318.96	205.75	317.69	203.63

Cuadro 28: Datos del eje x y eje y obtenidos del segundo agente del caso 6.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	327.19	149.60	327.19	149.60
2	319.05	161.11	321.36	158.35
3	316.33	175.55	316.75	173.46
4	321.71	188.77	320.75	186.43
5	331.46	199.90	330.51	198.37
6	340.16	212.25	338.83	210.04

Cuadro 29: Datos del eje x y eje y obtenidos del tercer agente del caso 6.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	313.28	125.39	313.28	125.39
2	308.21	139.04	308.15	137.42
3	304.83	153.32	304.43	151.92
4	311.11	166.43	309.68	164.33
5	320.52	177.78	318.27	176.38
6	329.19	190.09	327.81	189.59

Cuadro 30: Datos del eje x y eje y obtenidos del cuarto agente del caso 6.

No.	Sim. eje x (cm)	Sim. eje y (cm)	Recorrido eje x (cm)	Recorrido eje y (cm)
1	306.52	96.32	306.52	96.32
2	292.99	102.49	295.57	100.16
3	280.64	110.25	282.62	107.77
4	266.20	113.46	268.85	112.33
5	253.81	121.56	255.87	119.59
6	240.37	127.15	238.71	127.99

Cuadro 31: Datos del eje x y eje y obtenidos del quinto agente del caso 6.

Nuevamente, con la finalidad de comparar la similitud entre simulación e implementación física se calculó la distancia euclidiana desde el origen (DE) de los valores simulados y los valores de la trayectoria recorrida, la diferencia entre estos (DEP) y el porcentaje de error tanto en el eje x como el eje y, estos datos pueden observarse en las Tablas 32, 33, 34, 35 y 36.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	309.99	309.99	0.00	0.00	0.00
2	299.74	301.50	2.16	0.76	0.41
3	291.14	291.76	1.90	0.50	1.10
4	279.73	281.56	2.38	0.93	0.42
5	272.23	273.82	2.21	0.91	0.41
6	262.36	263.71	2.09	0.89	0.46

Cuadro 32: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del primer agente del caso 6.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	331.69	331.69	0.00	0.00	0.00
2	333.50	330.72	3.04	0.64	1.53
3	339.34	338.39	1.97	0.01	1.16
4	351.21	347.77	3.45	0.92	1.13
5	365.23	361.93	3.32	0.83	1.07
6	379.57	377.35	2.47	0.40	1.03

Cuadro 33: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del segundo agente del caso 6.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	359.77	359.77	0.00	0.00	0.00
2	357.42	358.26	3.60	0.73	1.71
3	361.77	361.13	2.13	0.13	1.19
4	373.00	370.99	2.53	0.30	1.24
5	387.08	385.47	1.80	0.29	0.76
6	400.95	398.65	2.58	0.39	1.04

Cuadro 34: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del tercer agente del caso 6.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	337.44	337.44	0.00	0.00	0.00
2	338.12	337.40	1.63	0.02	1.17
3	341.22	340.23	1.46	0.13	0.92
4	352.83	350.59	2.53	0.46	1.26
5	366.52	363.87	2.65	0.70	0.79
6	380.13	378.69	1.46	0.42	0.26

Cuadro 35: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del cuarto agente del caso 6.

No.	DE Sim. (cm)	DE Rec. (cm)	DEP (cm)	% de Error eje x	% de Error eje y
1	321.29	321.30	0.00	0.00	0.00
2	310.40	312.08	3.48	0.88	2.27
3	301.51	302.47	3.17	0.71	2.24
4	289.37	291.37	2.88	1.00	1.00
5	281.42	282.44	2.85	0.81	1.62
6	271.92	270.86	1.86	0.69	0.66

Cuadro 36: Distancia euclidiana desde el origen (DE), distancia euclidiana entre puntos (DEP) y porcentajes de error del quinto agente del caso 6.

Además de las tablas se calcularon los RMSE: de cada agente, obteniendo un valor de 1.3904 cm para la primera trayectoria, un valor de 1.8746 cm para la segunda trayectoria, un valor de 1.6795 cm para la tercera trayectoria, 1.3031 cm para la cuarta trayectoria y 1.8711 cm para la última trayectoria. Los datos anteriores indican que el RMSE: de las trayectorias no superó los 2 cm, también se observa que las distancias euclidianas en las tablas anteriores no superaron los 4 cm y los porcentajes de error tampoco sobrepasaron el 3 %, por lo que podemos afirmar que la implementación física es comparable a la simulación. En este caso se realizaron cambios al controlador de acercamiento exponencial cambiando la velocidad inicial a un valor de 0.15 y el *alpha* se mantuvo con un valor de 0.5.

CAPÍTULO 10

Conclusiones

- La regla de cohesión, separación y alineación se comprobó tanto en el desarrollo del algoritmo, como en la implementación física cuando se utilizaron más de dos robots móviles con ruedas dado que estos se acercaron, evitaron colisiones y también se alinearon conforme avanzaba el tiempo, todo esto con un RMSE no mayor a 4 cm y porcentajes de error no mayores al 3 % al utilizar tolerancias en el controlador de acercamiento exponencial de 5 cm.
- Los pesos que presentan las mejores características de un rebaño tienen una proporción de 1 a 6 para la cohesión y la separación. Mientras que la regla de alineación tiene la misma proporción que la de cohesión.
- Los agentes pueden llegar a colisionar entre ellos aún habiendo hecho diferentes optimizaciones debido a que muchas veces la suma de todas las fuerzas de dirección es tan grande que un agente se ve obligada a acercarse mucho a uno de sus vecinos.
- El envío y recepción de información del servidor y los robots móviles con ruedas es demasiado limitado para ejecutar al mismo tiempo la simulación y la implementación física obteniendo un RMSE de 5.66 cm en el eje x, en los primeros 6 segundos de simulación la cual aumenta conforme avanza el tiempo haciéndola ineficiente.
- Mientras más agentes se incluyan en la simulación más aumentará el tiempo de ejecución haciendo que la simulación se vuelva más lenta. Teniendo un modelo de regresión lineal con un R cuadrado de 0.993.
- Debido a las limitaciones del servidor el número máximo de agentes que se pueden controlar con un controlador de acercamiento exponencial es de cinco Pololus 3pi+. Un número mayor puede hacer que el servidor se desconecte de la computadora o simplemente el envío y recepción de datos tarde tanto que la implementación física empiece a presentar complicaciones y propagaciones de error donde no es capaz de alcanzar cada uno de los 6 puntos de la interpolación.

- La implementación de detección de obstáculos y depredadores no afectó el comportamiento original de las 3 reglas de los Boids de Reynolds, dado que evaden a los depredadores como grupo y no individualmente.
- Reducir el espacio de la plataforma a de 380 cm en x a 340 cm y de 480 cm en y a 440 cm evita tener que aumentar la tolerancia del controlador de acercamiento exponencial, esto con el fin de no perder precisión pero también evitando que los Pololu 3pi+ colisionen con las paredes de la plataforma.

CAPÍTULO 11

Recomendaciones

- Para futuras investigaciones seria adecuado agregar un sistema de sensores para ser redundante con la simulación y no depender únicamente del envió de datos a las llantas de los robots móviles con ruedas.
- Implementar paralelismo para mejorar el rendimiento del envío y recepción de datos puede permitir aumentar el número de agentes que se pueden utilizar actualmente en la implementación física.
- Experimentar el uso de otros controladores diferentes al de acercamiento exponencial con la finalidad de evaluar cual tiene un mejor rendimiento con el algoritmo de Boids de Reynolds.
- En futuros trabajos pueden agregarse más comportamientos como hacer que los agentes siempre tiendan a una coordenada especifica o incluso que se alejen de esta haciendo que los agentes formen círculos en el plano.

CAPÍTULO 12

Bibliografía

[1] K. M. L. y F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," en *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, 1987, págs. 25-34.

[3] I. Failes, *Befores & Afters: Issue #4 editor Ian Failes, illustration Aidan Roberts. Before and After*. 2022.

[4] A. Davison, *Killer game programming in Java*. .º'Reilly Media, Inc.", 2005.

[5] H.-D. Lam, "Implementing Boids behaviors on the ChIRP robots," Tesis de mtría., Norwegian University of Science y Technology, 2015.

[6] C. Perafán Montoya, "Robotat: un ecosistema robótico de captura de movimiento y comunicación inalámbrica.," es, Tesis de licenciatura, Universidad del Valle de Guatemala, Guatemala, 2022. dirección: `https://repositorio.uvg.edu.gt/handle/123456789/4250`.

[7] M. L. S. Tortosa, ""Agentes y enjambres artificiales: modelado y comportamientos para sistemas de enjambre robóticos,"," Tesis doct., Universidad de Alicante, 2013.

[8] P. R.
bibinitperiod Electronics, *3pi+ 32U4 OLED Robot - Standard Edition (30:1 MP Motors) Assembled*. dirección: `https://www.pololu.com/product/4975/blog`.

[9] I. D. O. NaturalPoint, *Primex 41*. dirección: `https://optitrack.com/cameras/primex-41/`.

[10] D. Bourg y G. Seemann, *AI for Game Developers* (O'Reilly Series). O'Reilly, 2004, ISBN: 9780596005559. dirección: `https://books.google.com.gt/books?id=Sz-Sqvm-hSYC`.

[11] T. Wescott, "PID without a PhD," *Embedded Systems Programming*, vol. 13, n.º 11, págs. 1-7, 2000.

[12] R. Kowalski. "boids-model." Última visita: 3 de agosto del 2023. (2017), dirección: `https://github.com/b3rnoulli/boids-model`.

CAPÍTULO 13

Anexos

13.1. Código de las reglas de los Boids de Reynolds

13.1.1. Repositorio en Github

A continuación se adjunta un enlace al repositorio de Github donde se encuentra la programación orientada a objetos desarrollada de la simulación final de los Boids de Reynolds junto al código de la implementación física.

Enlace al repositorio: Repositorio en Github

13.2. Vídeos de las diferentes pruebas

13.2.1. Primer prototipo

Se adjunta un enlace del primer prototipo desarrollado para simular los Boids de Reynolds, en este caso funcionaba únicamente la cohesión y la separación dado que la alineación aún no estaba desarrollada correctamente.

Enlace al vídeo de Youtube: Vídeo Primer Prototipo

13.2.2. Implementación física

A continuación se adjunta un enlace de la implementación en físico de los Boids de Reynolds, en este caso se utilizaron únicamente dos robots móviles con ruedas para representar las diferente reglas.

Enlace al vídeo de Youtube: Vídeo Implementación física

13.2.3. Implementación del Caso 1

En el siguiente enlace se mostrará la implementación en físico del caso 1 presentado en este trabajo, donde se utilizaron únicamente dos robots móviles con ruedas para representar las diferente reglas en físico.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 1

13.2.4. Implementación del Caso 2

A continuación se adjunta un enlace de la implementación en físico del caso 2 presentado en este trabajo. En este caso se utilizaron tres robots móviles con ruedas para representar las diferentes reglas y la evasión de depredadores.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 2

13.2.5. Implementación del Caso 3

El siguiente enlace mostrará la implementación en físico del caso 3 presentado en este trabajo, donde se utilizaron tres robots móviles con ruedas para ajustar los límites de la implementación física.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 3

13.2.6. Implementación del Caso 4

El siguiente enlace mostrará la implementación en físico del caso 4 presentado anteriormente en este trabajo, donde se utilizaron cinco robots móviles con ruedas evitando un obstáculo/depredador.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 4

13.2.7. Implementación del Caso 5 (Fallido)

A continuación se adjunta un enlace de la implementación en físico del caso 5 presentado en este trabajo, en este caso se utilizaron ocho robots móviles con ruedas donde no fue posible alcanzar todos los puntos de las interpolaciones.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 5

13.2.8. Implementación del Caso 6

En el siguiente enlace se muestra la implementación en físico del caso 6 donde se utilizaron cinco robots móviles con ruedas donde estos se separaron en dos grupos esquivando el

obstáculo/depredador.

Enlace al vídeo de Youtube: Vídeo de la Implementación física del caso 6

CAPÍTULO 14

Glosario

Matlab: Es una plataforma de programación y cálculo numérico utilizada por millones de ingenieros y científicos para analizar datos, desarrollar algoritmos y crear modelos. xi, xii, 38, 40

RMSE: Es la raíz del error cuadrático medio y se utiliza para comparar puntos en el plano dado que el valor obtenido se expresa en las mismas unidades que los valores originales y proporciona una medida de la dispersión de los errores. xi, xii, 2, 36, 38–40, 46, 52, 60, 70