Desarrollo e implementación de algoritmos de control para un enjambre de drones Crazyflie 2.0 mediante un sistema de visión de cámaras OptiTrack.

Development and implementation of control algorithms for a Crazyflie 2.0 drone swarm using an OptiTrack camera vision system.

Kenneth Andree Aldana Corado (ald18435@uvg.edu.gt)

Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica, Facultad de Ingeniería, Universidad Del Valle de Guatemala

Resumen

El estudio y desarrollo de sistemas multi-agentes ha tomado un mayor protagonismo en los últimos años. Estos sistemas pueden ser formados por distintos tipos de robots y es por ello que se han encontrado una gran cantidad de aplicaciones para esta rama de la robótica. Es por ello que el siguiente artículo presenta el desarrollo de un algoritmo para la coordinación de un enjambre de agentes, para este caso, cuadricópteros. La idea central del algoritmo es dotar a los drones de cierta autonomía para que, puedan formarse sin seguir una trayectoria previamente calculada. Para cumplir esto, se realizaron diversas pruebas con ecuaciones de consenso y sus modificaciones para poder observar el comportamiento del enjambre, al igual que el aprovechamiento que ofrece cada ecuación para la formación. Esto se realizó tomando en consideración que el algoritmo debía evitar colisión entre agentes, limitar la velocidad de movimiento y minimizar el uso de energía. Los resultados más relevantes fueron que, utilizando las ecuaciones de consenso modificadas, fue posible realizar la formación de los agentes con un uso menor de energía y permitiendo que no hubiera colisión entre ellos y permitiendo formaciones exitosas hasta un 75%.

Palabras clave: algoritmo, consenso, enjambre, cuadricóptero, energía.

Abstract

The study and development of multi-agent systems has become increasingly important in recent years. These systems can be formed by different types of robots and that is why many applications have been found for this branch of robotics. That is why the following article presents the development of an algorithm for the coordination of a swarm of agents, in this case, quadcopters. The central idea of the algorithm is to provide the drones with certain autonomy so that they can form themselves without following a previously calculated trajectory. To accomplish this, several tests were performed with consensus equations and their

modifications to observe the behavior of the swarm, as well as the use that each equation offers for the formation. This was done taking into consideration that the algorithm had to avoid collision between agents, limit the speed of movement and minimize the use of energy. The most relevant results were that, using the modified consensus equations, it was possible to perform the formation of the agents with a lower use of energy and allowing no collision between them and allowing successful formations up to 75%.

Keywords: algorithm, consensus, swarm, quadcopter, energy.

Introducción

El uso de enjambre de robots es cada vez más común para solucionar problemas específicos, es por ello que su estudio ha incrementado en los últimos años. Por lo general, el control y administración de un robot resulta un trabajo moderadamente sencillo, pero cuándo se implementan dos o más agentes, la complejidad de movimiento y elaboración de tareas complejas en conjunto presenta una mayor dificultad puesto que, un robot puede irrumpir en el área de trabajo del robot contiguo, por lo que pueden ocurrir colisiones.

Cuando se trabaja la coordinación de un enjambre en un plano, este presenta una menor dificultad que en un espacio tridimensional, es por ello por lo que coordinar un enjambre de cuadricópteros presenta otros factores y consideraciones que agregan complejidad a la problemática. Empezando por, los cuadricópteros no tienen a la capacidad de realizar desplazamiento hacia abajo, estos elementos solo permiten la disminución de velocidad en los motores para descender. Adicional, los movimientos según el ángulo al cuál se mueve, por el tamaño de los mismo, puede presentar un desequilibrio al realizar movimientos que varíen la trayectoria en un lapso corto.

Los proyectos desarrollados hasta ahora, en su mayoría, utilizan drones de baja potencia, puesto que se ha comprobado que al tener un número grande de agentes, estos al tener una distancia muy corta entre ellos, la dinámica del aire cambia por lo que ocasiona desestabilización y colisiones. Entonces, esto presenta otra dificultad, la cuál es que los drones de baja capacidad no solo no cuentan con una variedad de sensores que pueden ser útiles para la coordinación, también tiene limitaciones de energía, velocidad y calidad de actuadores.

Para ello, en el departamento de ciencias de la computación en la Universidad del Sur de California, un grupo de estudiantes desarrollaron el Crazyswarm, el cuál permite volar un enjambre de cuadricópteros de hasta 49 agentes. Este sistema permite utilizar como sistemas de localización el LPS, captura de movimiento y *Lighthouse.* Esto puede realizarse a través de 3 crazyradios, que son las antenas de Bitcraze que permiten la conexión de los cuadricópteros al ordenador.

Por lo tanto, el objetivo de este trabajo es desarrollar un algoritmo de coordinación de un enjambre de drones utilizando un sistema de captura de movimiento, para lo cuál se tiene a disposición los *Crazyflie 2.0* y el sistema OptiTrack. Con lo que se podrá cumplir con la función de realizar formaciones con 8 agentes.

Materiales y métodos

Para la formación, el controlador se orienta para dotar a los agentes la capacidad de poder reunirse en una estructura determinada. Con esto se busca que se puedan formar según la posición relativa de los demás agentes respecto a sus vecinos. Esto se logra al modificar la velocidad de cada agente respecto al error que existe entre la posición relativa esperada y la posición actual.

De los principales problemas para realizar el control de sistemas multiagentes resulta ser el de consenso, lo cual radica en que un grupo de agentes pueda reunirse en un mismo lugar. Este problema debe tener redes de intercambio de información entre los n agentes que conforman el grupo. La información relevante es la tasa de cambio de cada agente depende de la suma del estado relativo de los otros agentes, de esta afirmación se obtiene la ecuación:

$$\dot{x_i}(t) = \sum_{j \in N(i)} \left(x_i(t) - x_i(t)\right) \qquad i = 1,2 \dots, n \tag{1}$$

La ecuación 1 es la base para obtener la solución del problema canónico en sistemas multiagentes, también conocido como el problema de rendezvous. Este problema tiene como fin que un grupo de agentes coincidan en una misma posición a partir de conocer la posición relativa de sus vecinos o demás agentes. Tomando la ecuación de consenso y añadiendo pesos, se puede trabajar problemas más complejos que el de rendezvous, puesto que ya se trabaja con tensión en las aristas entre agendas. De aquí se trabaja con la función de reducir la tensión entre agentes tomando en cuenta que la tensión de una arista la define la siguiente ecuación:

$$\epsilon = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}\, \epsilon_{ij} \left(\left| \left| x_i - x_j \right| \right| \right) \tag{2}$$

al derivar se obtiene

$$\frac{\partial \epsilon_{ij}}{\partial x_i} = w_{ij} \left(\left| \left| x_i - x_j \right| \right| \right) \left(x_i - x_j\right) \tag{3}$$

De la ecuación 3 se despeja para w_{ij}, este peso varía según el problema que se resolverá. Por lo tanto, se obtienen las siguientes expresiones que definen la ecuación de consenso,

$$\epsilon_{ij} = \frac{1}{2}\left|\left|x_i - x_j\right|\right|^2 \tag{4}$$

$$w_{ij} = 1 \tag{5}$$

$$\dot{x_\iota} = -\sum_{j \in N(i)} (x_i - x_j) i = 1,2 \ldots, n \tag{6}$$

Con la ecuación 6, se puede tomar como base para distintas aplicaciones en sistemas multiagente, a continuación, se presentan algunos ejemplos de las modificaciones aplicadas a la ecuación de consenso para solucionar problemas específicos:

Velocidad normalizada (consenso bio-inspirado):

$$\epsilon_{ij} = \left|\left|x_i - x_j\right|\right| \tag{7}$$

$$w_{ij} = \frac{1}{\left|\left|x_i - x_j\right|\right|} \tag{8}$$

$$\dot{x_\iota} = -\sum_{j \in N(i)} \frac{(x_i - x_j)}{\left|\left|x_i - x_j\right|\right|} \qquad i = 1,2 \ldots, n \tag{9}$$

Control de formación:

$$\epsilon_{ij} = \frac{1}{2}\left(\left|\left|x_i - x_j\right|\right| - d_{ij}\right)^2 \tag{9}$$

$$w_{ij} = \frac{\left|\left|x_i - x_j\right|\right| - d_{ij}}{\left|\left|x_i - x_j\right|\right|} \tag{10}$$

$$\dot{x_\iota} = -\sum_{j \in N(i)} \frac{\left(\left|\left|x_i - x_j\right|\right| - d_{ij}\right)(x_i - x_j)}{\left|\left|x_i - x_j\right|\right|} i = 1,2 \ldots, n \tag{11}$$

Mantenimiento de conectividad:

$$\epsilon_{ij} = \frac{\left|\left|x_i - x_j\right|\right|^2}{\triangle - \left|\left|x_i - x_j\right|\right|} \tag{12}$$

$$w_{ij} = \frac{2\triangle - \left|\left|x_i - x_j\right|\right|}{\left(\triangle - \left|\left|x_i - x_j\right|\right|\right)^2} \tag{113}$$

$$\dot{x}_\iota = -\sum_{j\in N(i)} \frac{\left(2\triangle - \left|\left|x_i - x_j\right|\right|\right)(x_i - x_j)}{\left(\triangle - \left|\left|x_i - x_j\right|\right|\right)^2} i = 1{,}2\ldots, n \tag{14}$$

El desarrollo del algoritmo se realizó en Matlab, aprovechando la capacidad que proporciona para realizar cálculos complejos y la representación de información de forma sencilla. Para la definición del modelo matemático a utilizar, se realizaron distintas pruebas con grafos con distinto nivel de rigidez, esto con el propósito de encontrar el que mejor se adapte para la formación de agentes en un espacio tridimensional. Para esto se propuso un grafo totalmente rígido, mostrado en la figura 1, de 8 nodos.

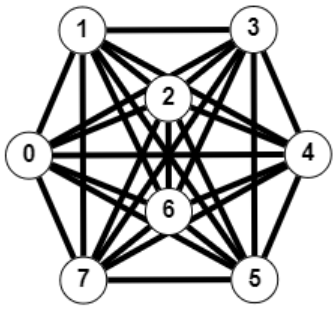

Figura 1. Grafo completamente rígido para formaciones.

Con matriz de reincidencia:

$$\begin{matrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{matrix}$$

Con esto, se hicieron dos pruebas de formaciones distintas, y para poder obtener las formaciones, se trabajó con las siguientes matrices de adyacencia:

Matriz de adyacencia 1:

$$\begin{bmatrix} 0 & 1 & 1 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{2} & \sqrt{3} \\ 1 & 0 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{3} & 1 & \sqrt{2} \\ 1 & \sqrt{2} & 0 & 1 & \sqrt{2} & 1 & \sqrt{3} & \sqrt{2} \\ \sqrt{2} & 1 & 1 & 0 & \sqrt{3} & \sqrt{2} & \sqrt{2} & 1 \\ 1 & \sqrt{2} & \sqrt{2} & \sqrt{3} & 0 & 1 & 1 & \sqrt{2} \\ \sqrt{2} & \sqrt{3} & 1 & \sqrt{2} & 1 & 0 & \sqrt{2} & 1 \\ \sqrt{2} & 1 & \sqrt{3} & \sqrt{2} & 1 & \sqrt{2} & 0 & 1 \\ \sqrt{3} & \sqrt{2} & \sqrt{2} & 1 & \sqrt{2} & 1 & 1 & 0 \end{bmatrix}$$

Que corresponde a la siguiente formación:

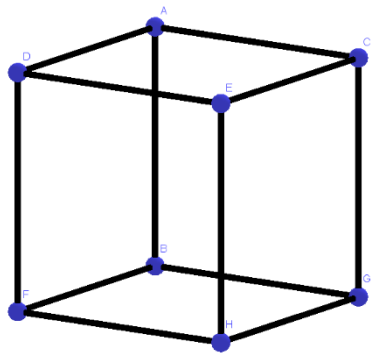

Figura 2. Primera configuración de los agentes, cubo.

Matriz de adyacencia 1:

$$\begin{bmatrix} 0 & 2 & 2 & \sqrt{3} & 2 & \sqrt{8} & \sqrt{3} & \sqrt{7} \\ 2 & 0 & 2 & 1 & \sqrt{8} & 2 & \sqrt{8} & \sqrt{5} \\ 2 & 2 & 0 & 1 & \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} \\ \sqrt{3} & 1 & 1 & 0 & \sqrt{7} & \sqrt{5} & \sqrt{5} & 2 \\ 2 & \sqrt{8} & \sqrt{8} & \sqrt{8} & 0 & 2 & 2 & \sqrt{3} \\ \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} & 2 & 0 & 2 & 1 \\ \sqrt{8} & \sqrt{8} & 2 & \sqrt{5} & 2 & 2 & 0 & 1 \\ \sqrt{7} & \sqrt{5} & \sqrt{5} & 2 & \sqrt{3} & 1 & 1 & 0 \end{bmatrix}$$

Que corresponde a la siguiente formación:

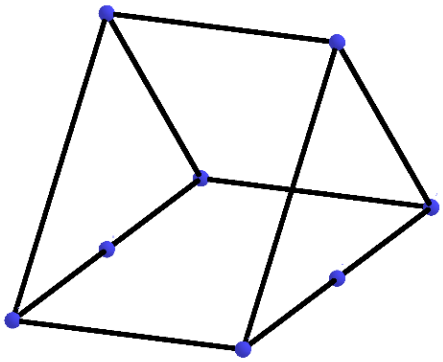

Figura 3. Segunda configuración de los agentes, prisma triangular.

En este caso, cada arista del grafo representa las distancias que deben mantener los agentes para realizar la formación, lo cual determina realmente la comunicación entre cada agente del enjambre. Es importante resaltar que, al momento de utilizar los grafos completos, se está definiendo una red de comunicación compleja debido a que cada robot debe saber la posición de todos los agentes. Es por ello que adicional al controlador, en Matlab se realiza un sub-algoritmo para que los agentes empiecen en posiciones aleatorias en un espacio de tres dimensiones.

Métricas de evaluación:

El proceso del desarrollo del algoritmo conlleva una gran cantidad de iteraciones para poder comprobar qué tan efectivo está siendo el controlador según la posición en la que empiezan los robots del enjambre. Es por ello que se define un conjunto de métricas para poder comparar y verificar cómo se comporta el algoritmo. Este conjunto consta de cuánto se aproxima la formación realizada por los agentes y la deseada y el consumo de energía de los agentes.

Energía utilizada en el sistema

Para esta métrica, se aprovecha la relación entre la energía y la velocidad de los agentes, para ello se trabajó con una aproximación para la integral de la velocidad, presentada en la siguiente ecuación:

$$\sum_{i=1}^{n}\sum_{k=0}^{T}\dot{x}_i\,k \tag{15}$$

Esta métrica es relevante puesto que los agentes a utilizar tienen un tiempo de vuelo y energía limitado, por lo tanto, es importante reducir el consumo de energía para poder aprovechar al máximo este recurso.

Éxito en la formación:

El principal factor para la métrica es si la formación cumple realmente con los pesos que se establecieron en la matriz de adyacencia. Para ello, se utiliza el error cuadrático medio:

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(x_{ij}-d_{ij}\right)^{2} \tag{16}$$

Para esto, como se trabajó con grafos totalmente rígidos, se tiene la rúbrica que la formación debe tener un error cuadrático medio menor a los límites indicados, en este caso es, si se tiene un error cuadrático menor o igual a 0.003, la formación fue exitosa, de lo contrario, se considera fallida.

Resultados y discusión

Como base del algoritmo desarrollado, es importante conocer los distintos tipos de ecuaciones existentes para poder conocer el comportamiento que tendrían. Por ello se utilizó la ecuación de consenso, la cual permite la convergencia de los robots en un punto. Posterior a su simulación, se trabajó con la ecuación de control de formaciones para la realizar la geometría indicada en la matriz de adyacencia. Debe mencionarse el problema canónico de Rendezvous, que consiste en el problema de cómo hacer converger en una posición o lugar común y específico a *N* agentes (Mesbahi *et al, 2014)*. Es por ello que se puede observar en ambas simulaciones que la formación y consenso se lleva a cabo en el centroide de la posición inicial de los robots. Para los siguientes experimentos, la longitud/posición, se mide en unidades arbitrarias (u.a.), para el tiempo, segundos.

Ecuación de consenso lineal:

En la figura 4, se puede observar cómo funciona la ecuación de consenso lineal, en dónde no presenta cambios brucos respecto a la posición y todos los agentes llegan a converger al mismo punto. En la figura 5, se tiene el registro de la velocidad de cada agente, aquí se puede observar que al converger al mismo punto, las velocidades convergen en 0, esto debido a que los agentes se encuentran en el punto deseado, por lo tanto la velocidad tiende a 0.

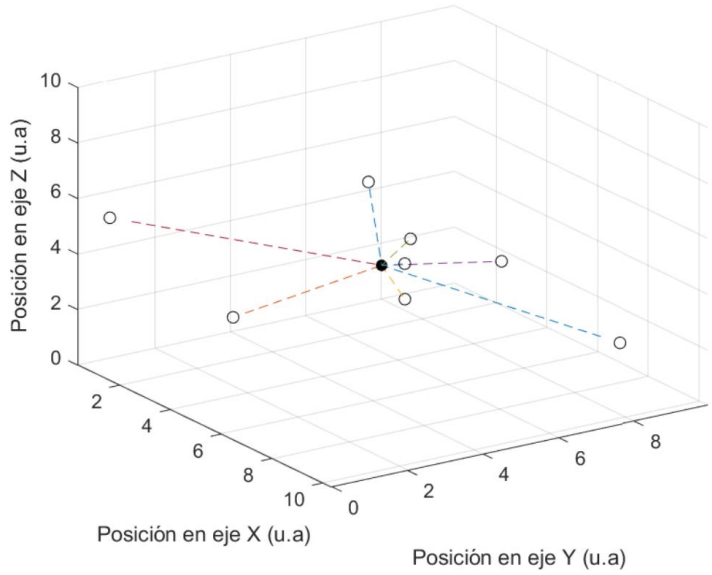

Figura 4. Trayectorias de los agentes con ecuación de consensus.

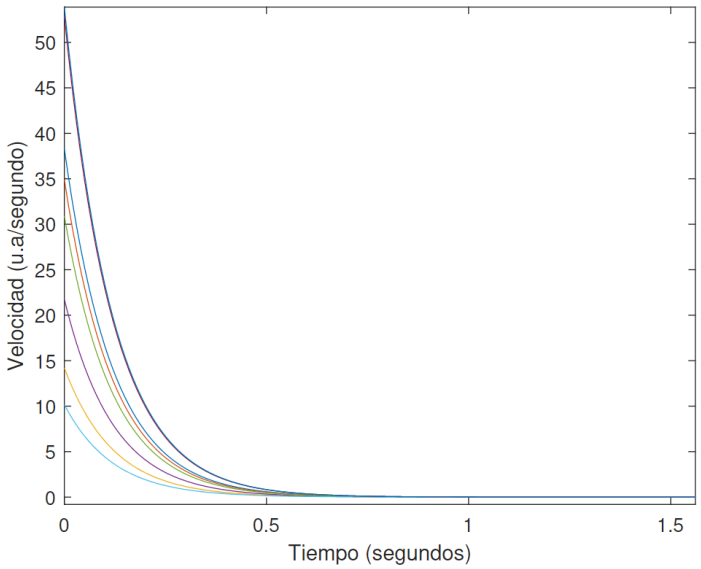

Figura 5. Magnitud de la velocidad de los agentes con ecuación de consensus.

Ecuación de formación:

En la figura 6, se ve el resultado de la ecuación de formación en dónde los agentes cumplen con realizar la formación indicada. Al igual que la ecuación de consenso, sus velocidades también convergen a 0. Para

ambos casos se mantiene que se forman en el centroide de la posición inicial de los robots. Para este experimento, se utilizó la matriz de adyacencia de la formación 1.

Al obtener los resultados de cómo se están comportando los agentes ante las modificaciones de la ecuación de consenso, se continúa realizando pruebas con la combinación de otros modelos para poder obtener más comportamientos y así poder examinar entre los experimentos, cómo optimizar el algoritmo.

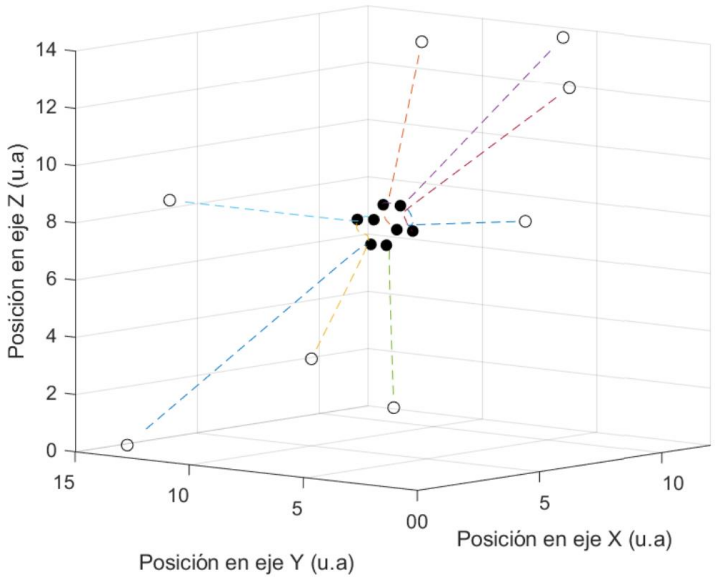

Figura 6. Trayectorias de los agentes con ecuación de formación.

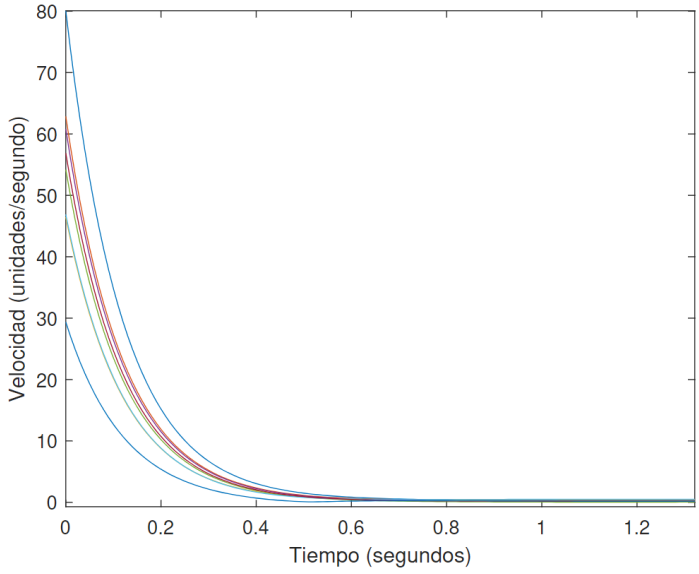

Figura 7. Magnitud de la velocidad de los agentes con ecuación de formación.

Ecuación de consenso modificadas:

Andrea generó dos ecuaciones a partir de la combinación de las ecuaciones básicas (Peña, 2019). Estas dos ecuaciones consisten en: La primera ecuación (17) tiene como objetivo principal acercar a todos los agentes hasta una distancia arbitraria, la segunda ecuación (18) consiste en una combinación entre evasión de colisión y control de formación. Tiene como principal función que los agentes realicen la formación deseada sin que interfieran en el radio físico de los otros agentes.

La ecuación de consenso con pesos tiene otras aplicaciones más relevantes, es por ello por lo que se manipuló la tensión entre agentes, esto con el propósito de aprovechar las ventajas que cada aplicación proporcionaba. Para este caso se trabajó con la ecuación de evasión de colisiones, puesto que permitía que se formaran con la restricción que, cada robot no entre en el radio físico del otro y con la ecuación de formación, que permite que los agentes se posicionen al lado de sus vecinos para que puedan formarse según la formación requerida.

$$w_{ij} = \frac{\left(\left||x_i - x_j|\right| - 2r\right)}{\left(\left||x_i - x_j|\right| - r\right)^2} \tag{17}$$

$$w_{ij} = \frac{4\left(\left||x_i - x_j|\right| - d_{ij}\right)\left(\left||x_i - x_j|\right| - r\right) - 2\left(\left||x_i - x_j|\right| - d_{ij}\right)^2}{\left(\left||x_i - x_j|\right| - r\right)^2\left(\left||x_i - x_j|\right|\right)} \tag{18}$$

Por ello, se trabajó el algoritmo con una ecuación dinámica, esto consiste en que, primero se acciona la ecuación para que los agentes se acerquen hasta un radio en dónde cada uno estuviera lo suficiente cerca del otro para poder empezar a formarse con el peso indicado del grafo, una vez en esta distancia, se cambia al controlador combinado mencionado anteriormente para que se ejecute la formación. El resultado de esta combinación se puede observar en las figuras 8 y 10, en dónde se cumple la formación definida en la matriz de adyacencia. Posteriormente, se puede visualizar en las figuras 9 y 11, cómo se comporta la velocidad de los agentes, en dónde se puede apreciar el cambio de velocidad debido al cambio de la ecuación.

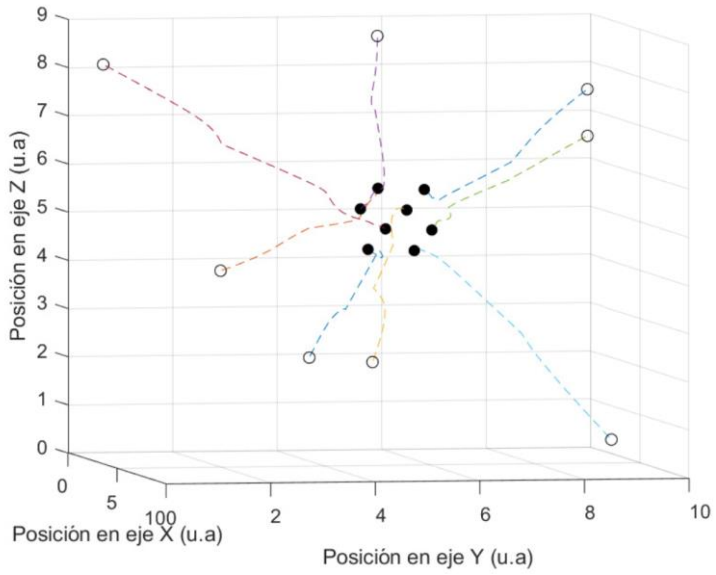

Figura 8. Trayectorias de los agentes con ecuación combinada para la formación 1.

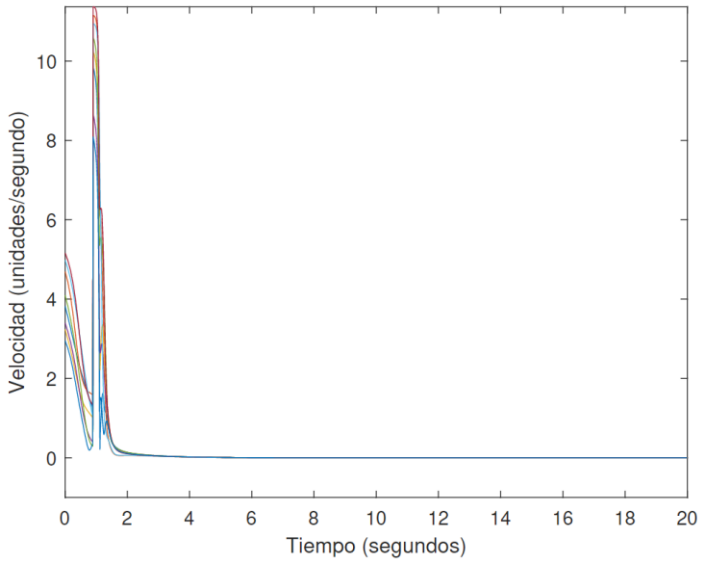

Figura 9. Magnitud de la velocidad de los agentes con ecuación combinada para la formación 1.

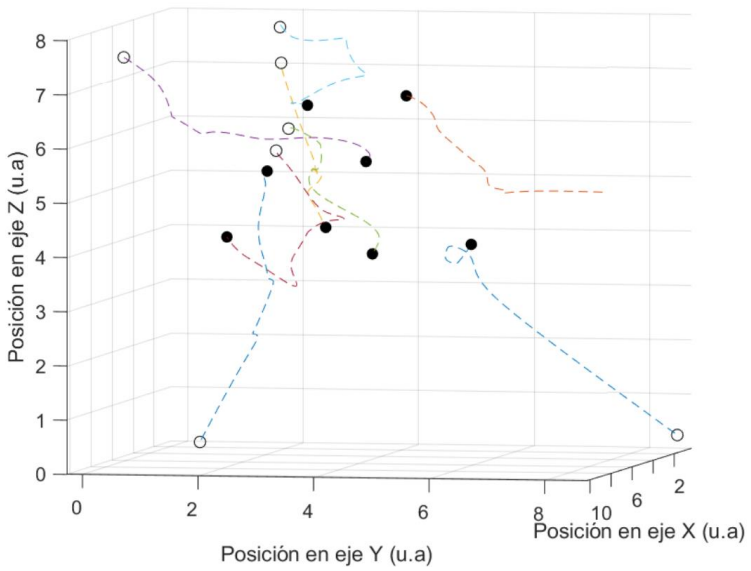

Figura 10. Trayectorias de los agentes con ecuación combinada para la formación 2.

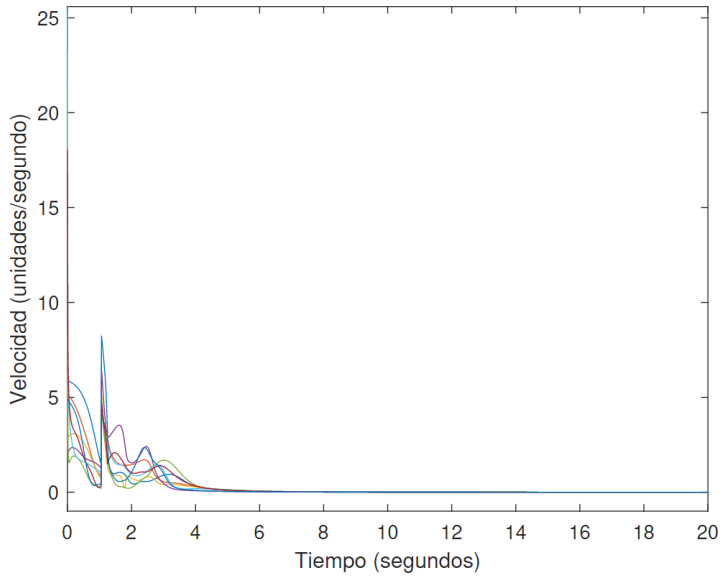

Figura 11. Magnitud de la velocidad de los agentes con ecuación combinada para la formación 2.

Estos resultados nos permiten observar que con el algoritmo planteado están funcionando satisfactoriamente, en el cuadro 1 se puede observar que se obtuvo un 68% de éxito para la primera formación y un 69% para la segunda, esto se debe a que la segunda formación es una geometría con mayor complejidad.

Cuadro 1. Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas

y fallidas de las 100 iteraciones simuladas del modelo dinámico.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad	1	100	68	32
Modificada con límite de velocidad	2	100	69	31

Cuadro 2. Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación.

Modelo	Grafo	E <50	50>E<100	100>E>200	E>200	Promedio
Modificado con límite de velocidad	1	29	56	6	9	1.23E+03
Modificado con límite de velocidad	2	6	74	9	11	1.42E+03

En las figuras 9 y 11 se puede observar que las magnitudes iniciales de la velocidad de los agentes son altas a comparación del comportamiento que presentan durante el resto de la simulación, por lo tanto, se requiere la implementación de un límite de velocidad para poder estudiar el comportamiento con agentes no ideales.

Límite de velocidad:

Se agrego un límite de velocidad de manera que, la velocidad del agente se mueva en la dirección que corresponde sin que la velocidad sea mayor al límite establecido. Esto se trabajará con el vector unitario de la velocidad obtenida por la ecuación de consenso. Por lo que, se realiza un mapeo de la velocidad con el límite establecido (Poole, 2017).

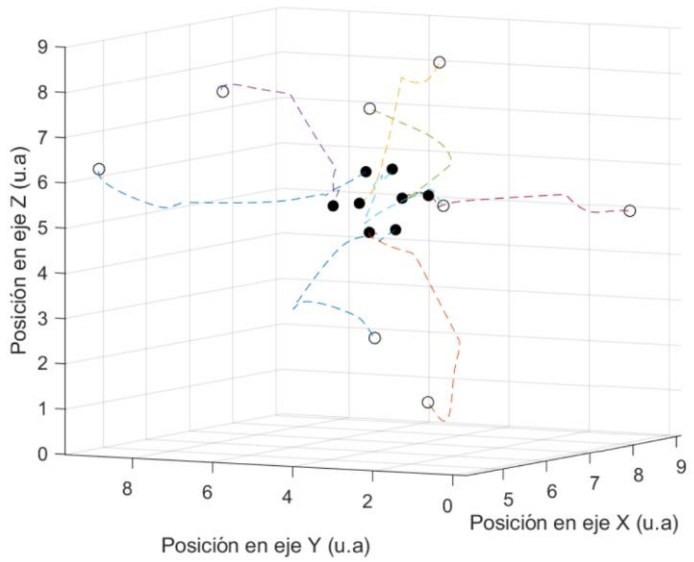

Figura 12. Trayectorias de los agentes con la ecuación dinámica y límite de velocidad para la formación 1.

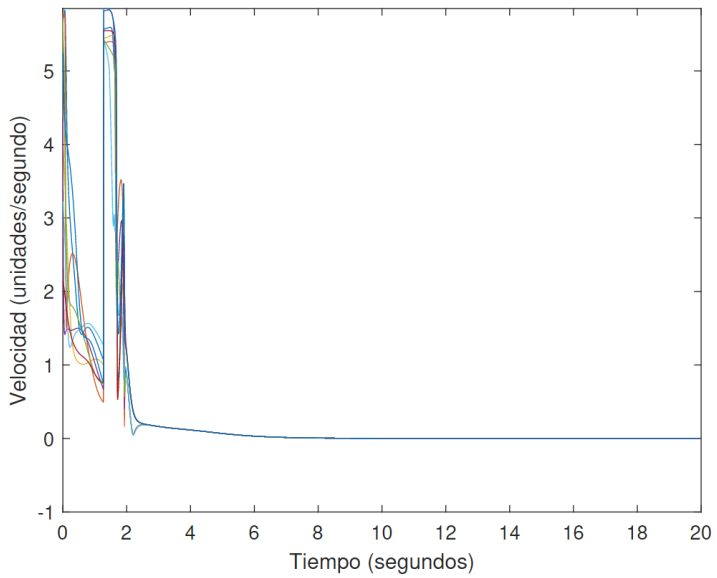

Figura 13. Magnitud de la velocidad de los agentes con la ecuación dinámica y límite de velocidad para la formación 1.

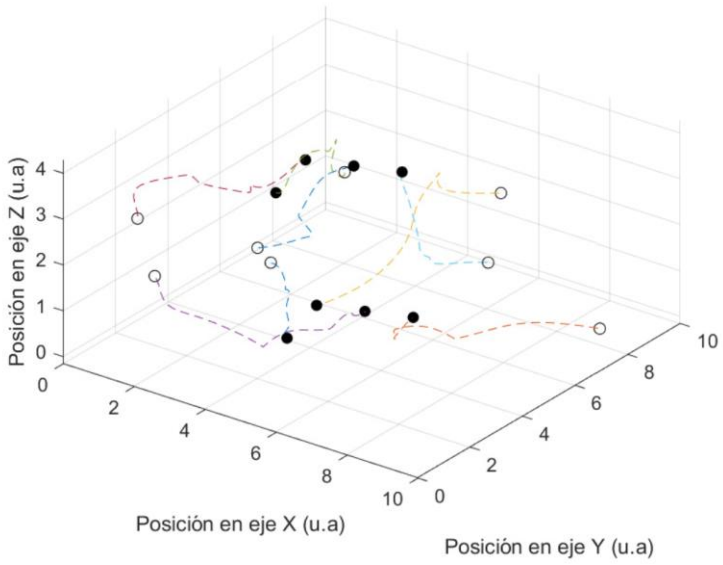

Figura 14. Trayectorias de los agentes con la ecuación dinámica y límite de velocidad para la formación 2.

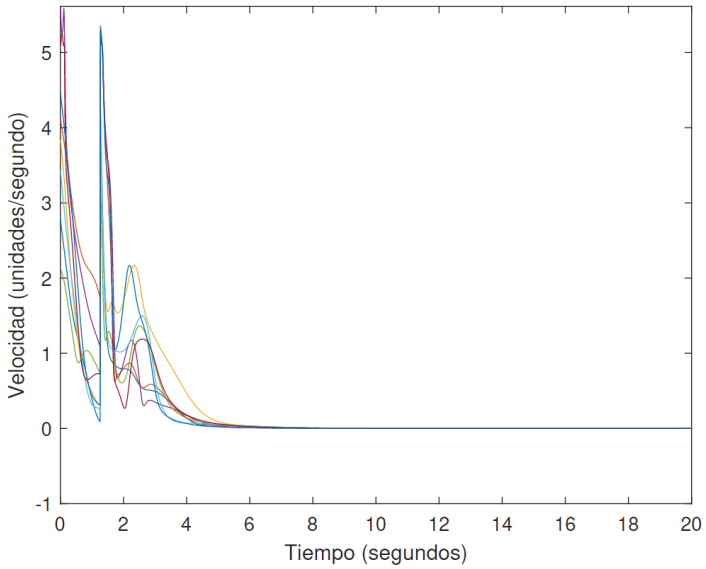

Figura 15. Magnitud de la velocidad de los agentes con la ecuación dinámica y límite de velocidad para la formación 2.

Estos resultados nos indican que la implementación de un límite de velocidad fue exitosa, pues los resultados de las 100 iteraciones realizadas por cada formación no aumentaron en la cantidad de formaciones fallidas, cómo se puede observar en el cuadro 3. En las figuras 13 y 15 se observa que la velocidad ya no presenta cambios bruscos en magnitud durante toda la trayectoria a comparación del modelo

sin límite de velocidad, por lo que se considera un algoritmo seguro para la implementación física. En las figuras 12 y 14 se pueden observar la formación exitosa de los agentes en la simulación del modelo con la implementación del límite de la velocidad en dónde respalda la capacidad de los agentes de formarse, aunque no dispongan de velocidades no limitadas.

Cuadro 3. Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad	1	100	76	24
Modificada con límite de velocidad	2	100	66	34

Cuadro 4. Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación con límite de velocidad.

Modelo	Grafo	E <50	50>E<100	100>E>200	E>200	Promedio
Modificado con límite de velocidad	1	95	5	0	0	41.73
Modificado con límite de velocidad	2	84	16	0	0	42.66

Pruebas en planos para formación en 3D

La idea de esta simulación es obtener los comportamientos lo más cercano a la realidad, es por ello que se trabajó con una posición aleatoria en el plano $Z = 0$, y así poder observar el comportamiento de los agentes al momento de la inicialización del algoritmo. El resultado de esta prueba fue que, los agentes no pudieron formarse debido a que, partiendo del problema canónico, este permite el consenso de agentes en en el centroide de la posición inicial de los agentes, acá debemos considerar que, al momento de inicializar los agentes en un plano, el centroide en un eje termina siendo la posición del plano en ese eje, por lo que les resulta imposible salir de este plano para mantener los pesos previamente definidos para la formación. El resultado de esta prueba fallida puede observarse en la figura 16 y 17, en dónde los agentes intentaron generar la formación sin salir del plano. Se realizaron 100 iteraciones para cada caso, el resultado de esto se puede observar en el cuadro 5.

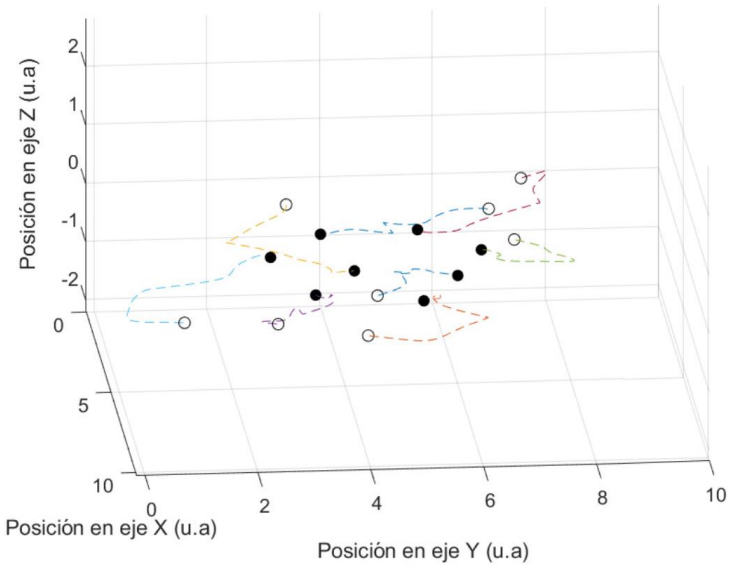

Figura 16. Trayectorias de los agentes en el plano Z = 0 para la formación 1.

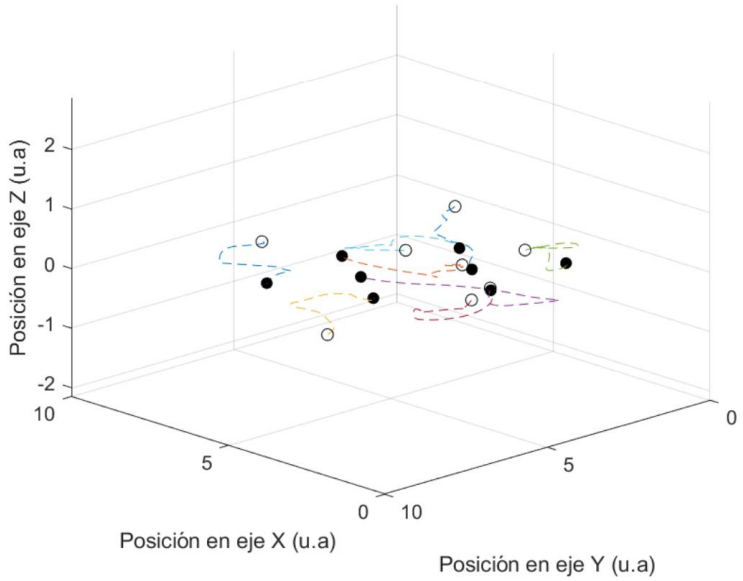

Figura 17. Trayectorias de los agentes en el plano Z = 0 para la formación 2.

Cuadro 5. Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad en el plano Z = 0.

Modelo	Grafo	Iteraciones	Exitosa	Fallida

Modificada con límite de velocidad	1	100	0	100
Modificada con límite de velocidad	2	100	0	100

Para poder solucionar el problema con la inicialización en un plano, se implementa una dinámica inicial para que los agentes al momento de iniciar se eleven durante un lapso arbitrario de tiempo y con ello, salgan del plano. Esto resulta exitoso debido a que, el algoritmo trabaja agente por agente, por lo tanto, estos no se elevan al mismo tiempo por lo que, desde que el primer agente se eleva, rompe la formación de plano y pueden iniciar con la formación. Esto se comprueba con las figuras 20 y 21, en dónde se puede observar cómo los agentes obtienen una velocidad vertical a distinto tiempo. Es importante resaltar que, se continúa con la restricción del centroide, por lo que los agentes deben elevarse lo suficiente para que la base no interfiera con los mismos. Esta dinámica resultó efectiva, debido a que se mantuvo un porcentaje aceptable de intentos exitosos. Se puede observar la trayectoria de los agentes para las dos formaciones en las Figuras 18 y 19.

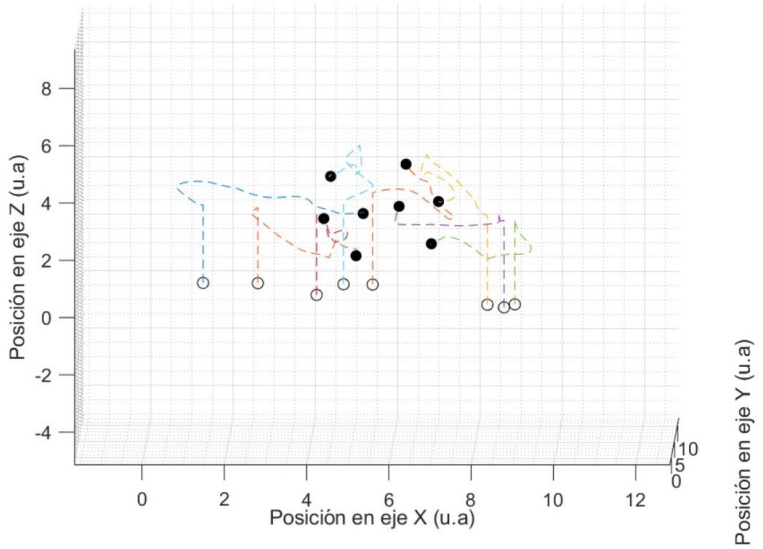

Figura 18. Trayectorias de los agentes al iniciar en el plano Z = 0 con la dinámica para romper el plano para la formación 1.

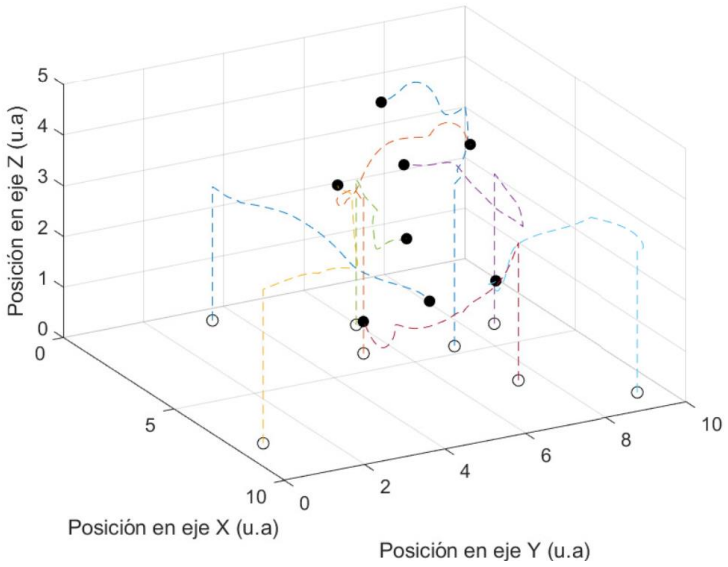

Figura 19. Trayectorias de los agentes al iniciar en el plano Z = 0 con la dinámica para romper el plano para la formación 2.

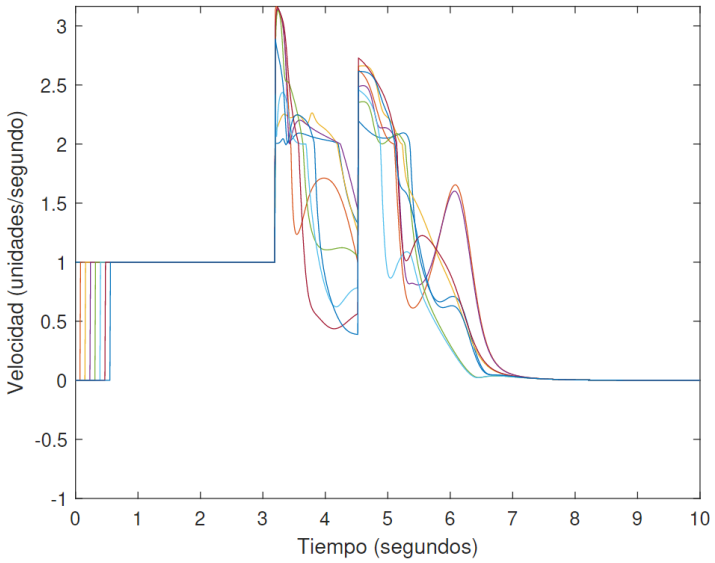

Figura 20. Velocidad de los agentes al iniciar en el plano Z = 0 con la dinámica para romper el plano para la formación 1.

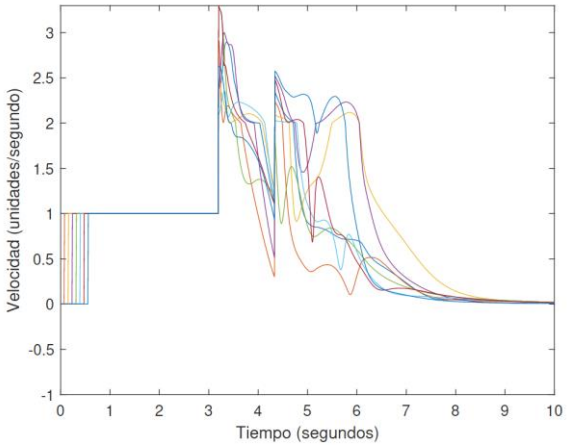

Figura 21. Velocidad de los agentes al iniciar en el plano Z = 0 con la dinámica para romper el plano para la formación 2.

Cuadro 6. Cantidad de formaciones exitosas. El cuadro muestra cuántas formaciones fueron exitosas y fallidas de las 100 iteraciones simuladas del modelo dinámico con límite de velocidad en el plano Z = 0 con rutina para romper plano.

Modelo	Grafo	Iteraciones	Exitosa	Fallida
Modificada con límite de velocidad	1	100	68	32
Modificada con límite de velocidad	2	100	65	35

Cuadro 7. Cantidad de energía utilizada. El cuadro muestra en qué rango de uso de energía se distribuyen las 100 iteraciones por formación con límite de velocidad con rutina para romper el plano Z = 0.

Modelo	Grafo	E <50	50>E<100	100>E>200	E>200	Promedio
Modificado con límite de velocidad	1	0	100	0	0	65.61
Modificado con límite de velocidad	2	0	100	0	0	64.69

En el cuadro 6 se puede observar que, para el primer modelo, la reducción de formaciones redujo en un 4%, para el segundo caso redujo únicamente un 1%. Esto nos dice que la rutina implementada no representa un cambio significativo para la ejecución de formaciones por lo que se considera una correcta implementación.

Es importante mencionar que, al observar el cuadro 7, si presenta un aumento de consumo de energía del 50\%, esto ocurre debido a que la rutina le indica a los agentes durante 3 unidades de tiempo para garantizar que los agentes no colisionen con el plano en $Z = 0$.

Conclusiones

El algoritmo desarrollado permite formarse en alguna configuración específica a un conjunto de agentes, otorgándole a cada uno la capacidad de recalcular su trayectoria según la posición de los demás agentes. El uso de grafos completamente rígidos presenta una red de comunicación compleja para la formación en 3 dimensiones, pero, resulta ser una herramienta útil para esta aplicación debido a que, por la cantidad de grados de libertad que presenta cada agente en tres dimensiones, se asegura que se tienen las restricciones suficientes para que la formación sea la esperada.

Los algoritmos en las simulaciones no toman en consideración las limitantes físicas de los actuadores de los agentes, por lo que implementar el límite de velocidad a las simulaciones nos presenta un comportamiento más cercano a las pruebas físicas.

El uso de un control dinámico permite utilizar los beneficios que presenta cada aplicación de la ecuación de consenso para aumentar el éxito de la formación.

La rutina de elevación para los drones funciona durante un tiempo especificado por el usuario, por lo que es relevante tomar en cuenta que este varía según la formación utilizada para evitar que los agentes salgan del área capturada por el sistema OptiTrack.

Los resultados de las trayectorias y estadísticas de posición nos indican que el algoritmo desarrollado permite que cada agente se comporte según lo esperado.

Agradecimiento

Agradezco al Dr. Luis Alberto Rivera por todo el apoyo durante este trabajo, principalmente con toda la ayuda brindada para utilizar de mejor manera el programa Matlab.

Agradezco al MsC. Miguel Enrique Zea Arenales por la ayuda en el análisis de grafos de adyacencia para las pruebas correspondientes.

Bibliografía

A. M. Peña Echeverría (2019) *Algoritmo de sincronización y control de sistemas de robots multi-agente para misiones de búsqueda*. Tesis de licenciatura, Universidad Del Valle de Guatemala.

D. Poole (2017) *Álgebra lineal: una introducción moderna*. Cengage Learning Latin America, 2017.

M. Mesbahi y M. Egerstedt (2014) *Graph Theoretic Methods in Multiagent Networks*. Wiley.