ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ESTUDIO, CONTROL E IMPLEMENTACIÓN DE SISTEMAS ROBÓTICOS AVANZADOS

APLICACIÓN DE UN ALGORITMO BASADO EN CONSENSO PARA UN SISTEMA MULTI-AGENTE ROBÓTICO SIMULADO EN COPPELIASIM Y COMANDADO DESDE MATLAB.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

ALVARO MISSAEL TIPANGUANO ASTUDILLO

alvaro.tipanguano@epn.edu.ec

DIRECTOR: Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

patricio.cruz@epn.edu.ec

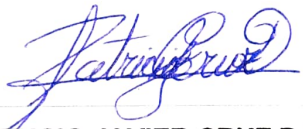
DMQ, octubre 2022

CERTIFICACIONES

Yo, ALVARO MISSAEL TIPANGUANO ASTUDILLO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ALVARO MISSAEL TIPANGUANO ASTUDILLO

Certifico que el presente trabajo de integración curricular fue desarrollado por ALVARO MISSAEL TIPANGUANO ASTUDILLO, bajo mi supervisión.



Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

ALVARO MISSAEL TIPANGUANO ASTUDILLO

Ing. PATRICIO JAVIER CRUZ DÁVALOS, PhD.

Ing. EDISON PATRICIO CRUZ CANDO

DEDICATORIA

Dedico con mucho cariño a mis padres, hermanos y mi querida sobrina. Quienes han estado conmigo siempre apoyándome, para alcanzar mis metas.

AGRADECIMIENTO

A Dios, por brindarme la salud y sobre todo la fortaleza en los momentos más complicados de la vida.

A mis queridos padres, mi mayor inspiración; quienes han sabido crear un maravilloso hogar, siendo un gran ejemplo para mí y mis hermanos. Nunca decidieron rendirse con los desafíos de la vida, enseñándonos el valor del esfuerzo y trabajo que al final siempre traen sus recompensas.

A mis queridos hermanos Jenny, Alex, Tatiana y Andres que son mis guías y mi motivación para seguir esforzándome cada día; también a mi querida sobrina Victoria por brindarme su compañía y la oportunidad de verla crecer. Agradezco también a nuestra unidad familiar, que siempre nos ha permitido apoyarnos mutuamente.

A mi tutor, profesor y director Dr. Patricio Cruz, por su paciencia, comprensión y sabios consejos, que siempre me ayudaron en cada etapa universitaria. Su gran profesionalismo y excelencia, en cada ámbito en el cual se desempeña es admirable.

A mi familia y personas conocidas, que siempre me mostraron su apoyo y supieron animarme positívame a seguir.

A mis amigos/as dentro y fuera de la universidad, con quienes he compartido gratos momentos, llenos de alegrías y tristezas, que me han ayudado a sobrellevar mi día a día.

A la UME y los capítulos técnicos RAS & CSS de la IEEE EPN, donde tuve buenas experiencias y conocí nuevos amigos.

A la Escuela Politécnica Nacional por brindarme la oportunidad de vivir una grandiosa experiencia universitaria.

ÍNDICE DE CONTENIDO

RESUMEN

Dentro de la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional se han realizado diversos proyectos de investigación con respecto al consenso de sistemas multi-agentes, dentro de los cuales se han realizado varios trabajos de titulación que han inspirado al desarrollo de este Trabajo de Integración Curricular. La presente aplicación se enfoca en el área de robótica móvil para lo cual se utiliza la plataforma Pioneer 3-DX, y el algoritmo de consenso para formaciones tipo líder-seguidor para el seguimiento de trayectoria de contención, utilizando un entorno virtual dentro de CoppeliaSim trabajando en conjunto a MATLAB. El desarrollo de esta aplicación inicia con el control de un robot aplicado al seguimiento de trayectoritas, para posteriormente continuar con la aplicación del algoritmo de consenso a un conjunto de robots, los cuales cumplen la función de agentes seguidores. Para aquellos agentes denominados como líderes se utiliza el concepto de líderes virtuales los cuales cumplen con las características necesarias para participar en el consenso y permiten encerrar un área de contención dentro de la cual los agentes robóticos interactúan en base al consenso. De los resultados obtenidos, se puede comprobar el correcto desempeño del seguimiento de trayectoria del sistema multi-agente, cuyos resultados se aprecian mediante una interfaz gráfica diseñada para el comando del entorno virtual desarrollado.

PALABRAS CLAVE: Simulink, CoppeliaSim, Sistema Multi-agente, Consenso, Pioneer 3-DX.

ABSTRACT

Within the School of Electrical and Electronic Engineering at Escuela Politécnica Nacional, several research projects regarding the consensus of multi-agent systems have been carried out. Therefore, several final degree works developed based on this topic have inspired this work. This application focuses on the area of mobile robots since the Pioneer 3-DX platform is used together with a consensus algorithm for leader-follower type formations for containment trajectory tracking. To achieve this goal, a virtual environment based on CoppeliaSim working together with MATLAB is implemented. The development of this application begins with the trajectory tracking control of one robot and then it continues with the application of the consensus algorithm to a set of robots, which are considered as follower agents. For the leader agents, the concept of virtual leaders is used and they meet the necessary characteristics to participate in the consensus and allow to enclose a containment area within which the robotic agents interact based on consensus. From the test results, it is possible to verify the correct performance of the trajectory tracking of the multi-agent system, whose results are shown through a graphical interface designed for the command of the developed virtual environment.

KEYWORDS: Simulink, CoppeliaSim, Multi-agente system, Consensus, Pioneer 3-DX.

1. INTRODUCCIÓN

Los sistemas multi-agente hoy en día se han desarrollado en muchas áreas debido a su versatilidad, por lo que, se los está aplicando a diversas tareas en diferentes campos. Algunas de sus aplicaciones más conocidas se encuentran en: Telecomunicaciones, Robótica, Sistemas de Generación & Distribución Eléctrica, entre otras; que ayuden a la operación desde aplicaciones básicas, hasta procesos industriales complejos.

En particular, la Robótica, que con el pasar de los años, ha ido alcanzando un alto nivel de madurez debido a su desarrollo tecnológico especialmente en cuanto al tipo de plataformas aéreas, terrestres o una combinación de ellos. Para la robótica terrestre se encuentran plataformas móviles muy útiles en operaciones de mapeo, navegación, entre otras. Este tipo de robots se utilizan desde operaciones (tareas) militares, como el reconocimiento de espacios peligrosos, hasta en aplicaciones civiles como en almacenes automatizados o en grandes supermercados.

De la unión de los sistemas multi-agente y la robótica, resulta los sistemas MARS o Multi-Agent Robotic Systems (Sistemas Multi-Agente robóticos); los cuales consisten en grupos o formaciones de robots, operando de manera cooperativa bajo el control de algún algoritmo cooperativo. Sus implementaciones físicas y costos dependerán del tipo de sistema, y aplicación a construir. Por esto una opción práctica, es el uso de plataformas de simulación por sus buenas prestaciones, que permiten obtener una dinámica similar a la real.

Las plataformas de simulación robóticas son herramientas muy útiles para desarrollos investigativos; debido a que las implementaciones virtuales reducen costos económicos. Por esto son la primera opción para pruebas preliminares de controladores y algoritmos. Entre las más comunes se encuentra CoppeliaSim, que mediante su versión educativa proporciona muchas herramientas útiles para diseños de todo tipo de escenarios de simulación; además, provee de una amplia biblioteca de funciones API que permiten la operación en conjunto de otros softwares como MATLAB. Sus modelos robóticos disponibles, como el Pioneer 3-DX, permiten realizar proyectos de investigación de manera rápida y sencilla sin la necesidad de la creación de un prototipo nuevo.

Debido a la poca información de sistemas multi-agente robóticos sobre una plataforma simulación, en este trabajo se propone una aplicación basada en consenso de un grupo de agentes robóticos móviles que permita ampliar el conocimiento dentro de los sistemas multi-agente robóticos y conocer un nuevo ambiente de simulación, como lo es CoppeliaSim. Además, que será un aporte al Proyecto de investigación PIS-19-01 “Estudio,

caracterización y control de sistemas industriales interconectados analizados desde el punto de vista de la teoría de redes complejas", en el cual se ha desarrollado la teoría relacionada a los sistemas multi-agente.

1.1 OBJETIVO GENERAL

Aplicar un algoritmo de consenso para un sistema multi-agente robótico simulado en CoppeliaSim y comandado desde MATLAB.

1.2 OBJETIVOS ESPECÍFICOS

- Revisar e investigar sobre algoritmos de consenso aplicados a sistemas multi-agente robóticos conformados por plataformas móviles en formación líder-seguidor, así como del uso de CoppeliaSim y su integración con MATLAB
- Diseñar un sistema multi-agente robótico bajo una formación líder-seguidor usando el robot móvil Pioneer 3-DX virtual disponible dentro del entorno de CoppeliaSim
- Diseñar e implementar un algoritmo de consenso para el sistema multi-agente robótico en formación líder-seguidor, usando las herramientas de MATLAB que interactúe con la plataforma de simulación CoppeliaSim
- Diseñar un interfaz gráfico para comandar la integración MATLAB-CoppeliaSim y mostrar resultados de la simulación
- Verificar el funcionamiento del entorno de simulación implementado para el sistema multi-agente robótico, construido en el entorno de CoppeliaSim y comandado desde MATLAB, considerando la tarea del seguimiento de trayectorias

1.3 ALCANCE

- Se realiza la revisión bibliográfica referente a la robótica móvil, en particular enfocada en la plataforma comercial Pioneer 3-DX, considerando algunos temas de interés como su modelamiento matemático, el control y aplicaciones.
- Se realiza la revisión bibliográfica acerca de algoritmos de consenso para sistemas multi-agente aplicados a robots móviles, bajo una formación líder-seguidor.
- Se hará una investigación de la plataforma de simulación CoppeliaSim, junto a sus prestaciones que puede proporcionar este software para comunicarse y operar con MATLAB.

- Se diseña el escenario virtual de simulación dentro del entorno de CoppeliaSim, para un sistema multi-agente robótico, bajo una formación líder-seguidor conformado por al menos 3 líderes virtuales y 2 seguidores robóticos móviles.
- Se diseña e implementa el controlador de posición y movimiento para el robot móvil Pioneer 3-DX en base a su modelo cinemático inverso, usando las herramientas de MATLAB.
- Se diseña e implementa un algoritmo de consenso usando las herramientas del software MATLAB para el sistema multi-agente construido en la plataforma de simulación.
- Se integra la comunicación entre MATLAB y la plataforma de simulación CoppeliaSim usando su Remote API, que permitirá el intercambio de datos.
- Se diseña un interfaz gráfico para la visualización de resultados importantes, como: señales de control, referencia de seguimiento, trayectorias recorridas y los respectivos errores de posición.
- Se verifica la tarea de seguimiento de trayectoria del sistema multi-agente robótico implementado en CoppeliaSim, operando con MATLAB para 4 trayectorias: Circular, Lemniscata, Senoidal y Cuadrada.

1.4 MARCO TEÓRICO

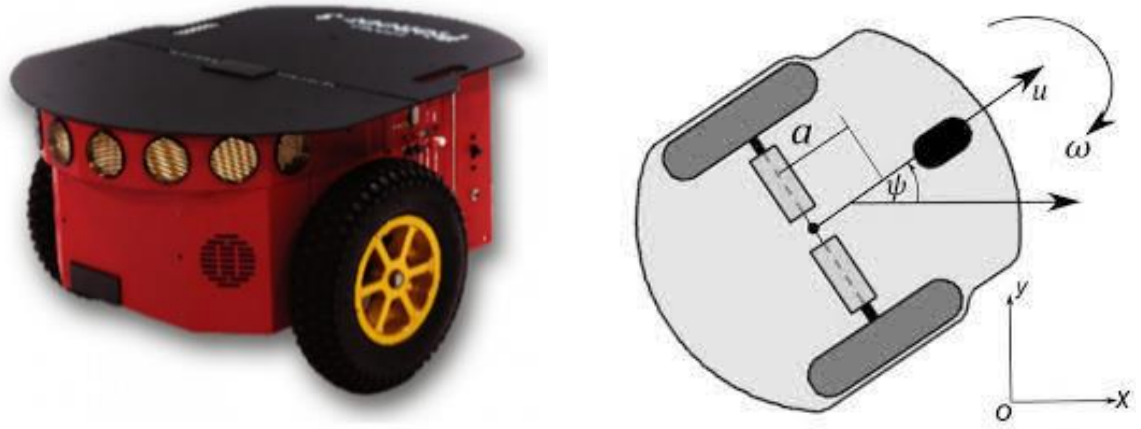
En este apartado se describe la teoría relacionada al robot móvil a usar en la aplicación propuesta. También se realiza un resumen explicativo de los sistemas multi-agente aplicados en el área de la robótica; además, de una descripción del algoritmo de consenso. Finalmente, se presenta una revisión respecto a los softwares de simulación y sus herramientas necesarias para la implementación del sistema multi-agente.

1.4.1. ROBOT MÓVIL PIONEER 3-DX

El Pioneer 3-DX es un robot móvil que pertenece a la familia *Pioneer 3* fabricado por *MobileRobots Inc*, se encuentra entre las más populares en el área de la robótica por su versatilidad, confiabilidad, resistencia, por esto es muy común encontrarlo en desarrollos de investigación, exhibiciones y otros proyectos.

Este robot cuenta con dos ruedas cuya diferencia de velocidades permite darle dirección durante su trayecto, lo cual se denomina como “tracción diferencial”, y una rueda adicional para mantener la estabilidad conocida como “rueda loca”. Estas características

estructurales que se pueden ver en Figura 1.1, y responde con una cinemática conocida como tipo uniciclo, con una restricción denominada no holonómica que limita la dirección de movimiento instantánea. Además, incorpora una serie de sensores de proximidad para la percepción del medio, como se especifican en [1].



a) b)

Figura 1.1 Robot Móvil Pioneer 3-DX; a) estructura real física, b) modelo estructural [2]

Modelamiento Cinemático

El modelamiento de este robot se basa en el modelo, como el que se muestra en la Figura 1.1 (b). Este modelo es denominado como mejorado debido al valor de a que permite mover el centro de masa del robot, esto se describe con mayor detalle en [3], [4]. La ecuación (1.1), representa la cinemática del robot tipo uniciclo como el Pioneer 3-DX, y es la base fundamental para múltiples aplicaciones y ensayos experimentales.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}cos(\psi) & -asen(\psi)\\ sen(\psi) & acos(\psi)\\ 0 & 1\end{bmatrix}\begin{bmatrix}u\\ \omega\end{bmatrix} \tag{1.1}$$

donde, x, y son las coordenadas de la posición del robot; y ψ es su orientación; a es la distancia entre la recta que une los centros de ambas ruedas y el centro de masa; u, ω son las señales de control de la velocidades lineal y angular del robot, respectivamente.

Una representación más real y precisa, se la puede obtener mediante su modelo dinámico descrito detalladamente en [5], que se resume a continuación.

Modelamiento Dinámico

El modelo dinámico es muy utilizado en diversas aplicaciones donde la precisión tiene una mayor prioridad [3]. Su estructura se basa en un complemento sobre el modelamiento

cinemático, y al que se le adicionan dos nuevos términos como se ven en la ecuación (1.2), y constantes dinámicas que se las encuentra en base a pruebas sobre la plataforma física.

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\psi}\\ \dot{u}\\ \dot{\omega}\end{bmatrix}=\begin{bmatrix}ucos(\psi)-a\omega sen(\psi)\\ usen(\psi)+a\omega cos(\psi)\\ \omega\\ \frac{\theta_3}{\theta_1}\omega^2-\frac{\theta_4}{\theta_1}u\\ -\frac{\theta_5}{\theta_2}u\omega-\frac{\theta_6}{\theta_2}\omega\end{bmatrix}+\begin{bmatrix}0 & 0\\ 0 & 0\\ 0 & 0\\ \frac{1}{\theta_1} & 0\\ 0 & \frac{1}{\theta_2}\end{bmatrix}\begin{bmatrix}u_c\\ \omega_c\end{bmatrix}+\begin{bmatrix}\delta_x\\ \delta_y\\ 0\\ \delta_u\\ \delta_\omega\end{bmatrix} \quad (1.2)$$

donde, u_c, ω_c, son las señales de control para la velocidad lineal y angular, provenientes del controlador cinemático; u, ω son las velocidades lineal y angular del robot; ψ, es la orientación del robot; $\delta_x, \delta_y, \delta_u, \delta_\omega$, son incertidumbres en las posiciones y velocidades, debido a varios factores como las perturbaciones; $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ y θ_6 representan de manera resumida términos de fuerzas y torques.

Del procedimiento de identificación de parámetros y constantes de este modelo de robot móvil Pioneer 3-DX, se ha realizado varios desarrollos de investigación como en [5], estas constantes se resumen en la Tabla 1.1.

Tabla 1.1. Constantes modelo dinámico

θ_1	0.24089
θ_2	0.2424
θ_3	-0.00093603
θ_4	0.99629
θ_5	-0.0037256
θ_6	1.0915

1.4.2. SISTEMA MULTI-AGENTE

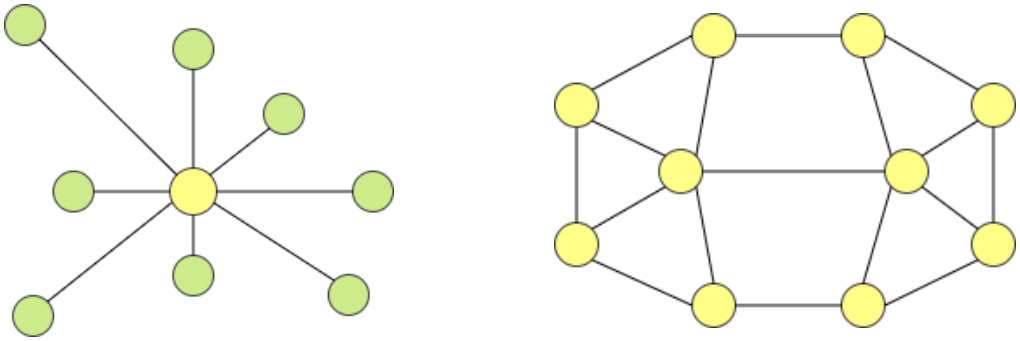
Se denominan sistemas multi-agentes, a un conjunto o grupos de agentes dispuestos estratégicamente. Estos agentes son unidades funcionales; que acorde a [4], [6], son entidades físicas o abstractas inteligentes que resuelven problemas de manera cooperativa. Estas entidades deben cumplir con las siguientes características: sensado local, autonomía, y descentralización. Estos sistemas son de gran utilidad para muchas aplicaciones dentro de áreas como: telecomunicaciones, sistemas eléctricos, robótica, entre otros.

En robótica estos sistemas multi-agentes están conformador por plataformas del mismo tipo o de diferentes, por lo que son conocidos como sistemas homogéneos y heterogéneos, respectivamente. Estas plataformas inteligentes con capaces de percibir el ambiente a

través de sensores, y capaces de evaluar el estado mediante un procesador lógico simple o complejo [6]. Así también pueden comunicarse con otros robots para obtener datos y evaluarlos en base a un algoritmo de control. Algunas ventajas de estos sistemas son: escalabilidad, mayor alcance geográfico y operación distribuida.

1.4.2.1. Control distribuido

Se denomina así, a la distribución de operaciones en estaciones autónomas capaces de trabajar por un objetivo en común, como por ejemplo los sistemas multi-agentes. Este tipo de control es distante de la arquitectura clásica centralizada donde las operaciones se realizan en una estación principal, donde estas últimas suelen ser muy vulnerables a constantes fallos, debido a que dependen un procesador central. Estas dos arquitecturas están representadas en la Figura 1.2, donde se representa las estaciones con color amarillo y verde según la jerarquía dentro del sistema, que esquematizan las arquitecturas centralizada y descentralizada respectivamente.



a) b)

Figura 1.2 Arquitecturas de control; a) Arquitectura centralizada b) Arquitectura distribuida. [Autoría propia]

Para una arquitectura distribuida, como se representa en la Figura 1.2 (b); al no depender de un procesador central reduce el riesgo de perder una operación total, sino que existe una autonomía en el desarrollo de operaciones divididas. Para esto el flujo de información se intercambia entre agentes interactuantes. En esta arquitectura no se requiere un intercambio de información entre cada uno de las estaciones sino de las que realmente lo requiera, mejorando el procesamiento de datos [3].

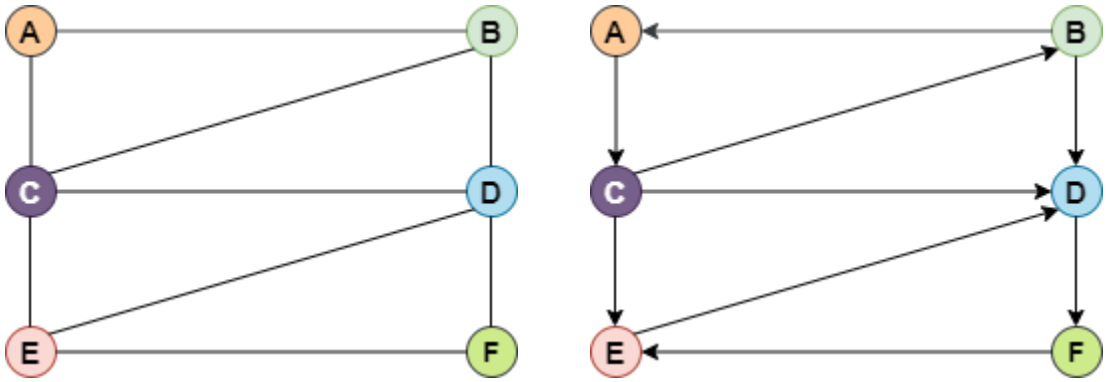
Una de las técnicas para el modelado y diseño sistemas multi-agente se basa en la Teoría de Grafos, por lo que, a continuación, se provee un resumen breve de los conceptos más importantes asociados a la misma.

1.4.2.2. Teoría de Grafos

Esta es una rama consolidada de las matemáticas discretas aplicadas a diversas ciencias como: genética, lingüística, geografía, sociología, telecomunicaciones, robótica, entre otras. Además, constituye una base fundamental para la comprensión del control distribuido, su procedimiento de diseño [7].

Un grafo es un par ordenado asociado, definido como $G = (V, E)$, constituido por un conjunto finito de vértices $V = \{v_1, v_2, \dots, v_n\}$ y un subconjunto de aristas $E \subseteq VxV$. Para este trabajo los vértices v_i, para $i = 1, \cdots, n$, son considerados como agentes, ubicados en el plano XY y las aristas representan las relaciones entre agentes (v_i, v_j), tal que $v_i, v_j \in V$.

Los grafos pueden ser de tipo no dirigido o dirigido dependiendo del tipo de enlace entre vértice, como se observa en la Figura 1.3 donde se aprecia ambos casos.



a) b)

Figura 1.3 Tipos de grafos: a) no dirigido b) dirigido [Autoría propia]

Dependiendo de la aplicación, un grafo no dirigido, como el que se muestra en la Figura 1.3 (a), se utiliza para casos en los cuales los vértices tengan interacción sin importancia en el sentido que tengan, como por ejemplo los amigos en común en las redes sociales. Por otro lado, un grafo dirigido, como el que se muestra en la Figura 1.3 (b), se utilizan para aplicaciones en las cuales la dirección del enlace toma importancia, como por ejemplo él envió de mensajes en las redes sociales, donde el flujo de mensajería puede ser unidireccional y/o bidireccional. Las aristas asociadas pueden tener un valor que presentan una relación entre los vértices. Basado en [3], para este trabajo este valor representa el grado de influencia entre agentes de la formación, como el que se observa en la Figura 1.4. Para este tipo de grafos su representación algebraica tiene mucha importancia, las cuales consisten en matrices asociadas al grafo, en particular la matriz Adyacencia y la matriz Laplaciana.

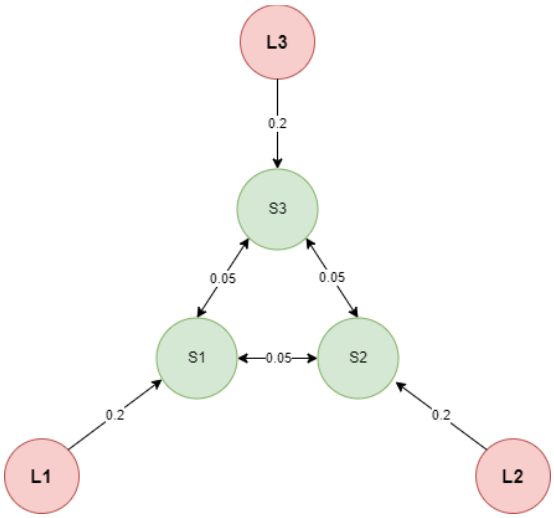


Figura 1.4. Ejemplo de grafo dirigido con ponderación [Autoría propia]

***Matriz de Adyacencia (A)*:** es una matriz cuadrada A_n, en donde n es el número de nodos del grafo G, también conocida como matriz de incidencia. Representa relaciones binarias o interacciones entre los vértices. Los elementos de la matriz A se definen mediante la ecuación (1.3):

$$a_{ij} = \begin{cases} > 0 & si(v_i, v_j) \in E \\ 0 & otro\ caso \end{cases} \tag{1.3}$$

donde el elemento a_{ij} es el valor de ponderación del nodo j sobre el nodo i, $\forall i,j \in \{1,2,\dots,n\}$, además el valor de i y j ubican en la fila y columna sobre la matriz A. En el caso de un grafo sin ponderación, se considera una ponderación valor igual a 1.

***Matriz Laplaciana (L)*:** conocida también como matriz de Kirchhoff [3]; esta matriz depende del grado de entrada y salida del vértice, expresadas en las ecuaciones (1.4), respectivamente; cuyo valor cuantifica el valor de interacción de cada vértice v_i.

$$\begin{aligned} \delta^+(v_i) &= \sum_{j=1}^{n} a_{ij} \\ \delta^-(v_i) &= \sum_{j=1}^{n} a_{ij} \end{aligned} \tag{1.4}$$

Para el caso de un grafo no dirigido la dirección de las aristas son indiferentes por lo cual los grados de entrada y salida son iguales ($\delta^+(v_i) = \delta^-(v_i)$). En el caso de un grafo dirigido como los de este trabajo, solo se considera las interacciones de entrada; por lo tanto, podemos definirlo simplemente como $\delta(v_i)$. Entonces, los elementos de la matriz L se definen de la siguiente manera:

$$\ell_{ij} = \begin{cases} \delta(v_i) & si\ i = j \\ -a_{ij} & si\ i \neq j \\ 0 & otro\ caso \end{cases} \tag{1.5}$$

donde, ℓ_{ij} es el elemento de i,j según sea el caso especificado en la ecuación (1.5), $\forall i,j \in \{1,2,\dots,n\}$, ubicado en la fila y columna de la matriz L.

Una relación directa entre ambas matrices se expresa en la ecuación (1.6).

$$L = D - A \tag{1.6}$$

donde D es una matriz diagonal, formada por los grados de los nodos v_i del grafo G.

Resumiendo todo lo mencionado, se presenta en la Tabla 1.2, un ejemplo práctico utilizando el grafo dirigido presentado en la Figura 1.4.

Tabla 1.2. Ejemplo de matriz de Adyacencia y Laplaciana de un grafo dirigido ponderado

Matriz de Adyacencia (A)	**Matriz Laplaciana (L)**
$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0.05 & 0.05 \\ 0 & 0.2 & 0 & 0.05 & 0 & 0.05 \\ 0 & 0 & 0.2 & 0.05 & 0.05 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2 & 0 & 0 & 0.21 & -0.05 & -0.05 \\ 0 & -0.2 & 0 & -0.05 & 0.21 & -0.05 \\ 0 & 0 & -0.2 & -0.05 & -0.05 & 0.21 \end{bmatrix}$

ÁRBOL DE EXPANSIÓN DIRIGIDA

Esta característica del grafo asegura la existencia de un camino directo desde cada uno de los nodos hacia otro especifico [4]. Para una mejor comprensión se resaltada esta característica para el grafo presentado anteriormente, véase la Figura 1.5.

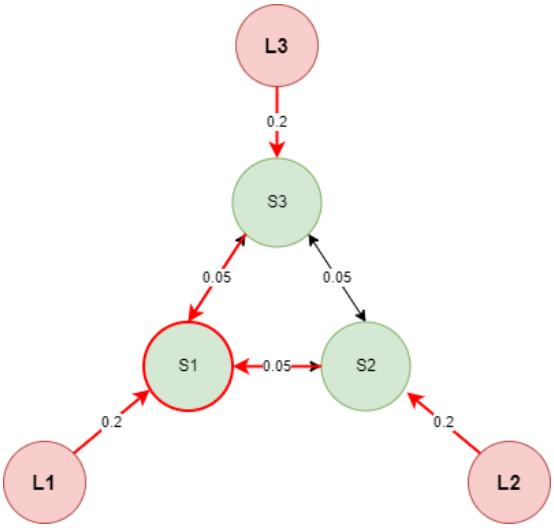


Figura 1.5. Característica de árbol de expansión dirigida [Autoría propia]

Este ejemplo muestra como mediante el uso de los caminos unidireccionales y bidireccionales todos los nodos tienen al menos un camino directo que llegan hacia el nodo denominado como S1.

1.4.3. ALGORITMO DE CONSENSO

En un sistema de múltiples agentes, cuando varios de estos llegan a un acuerdo en el valor de una determinada variable, se dice que ha llegado a un consenso. Para llegar a esto, es necesario que esta variable denominada estado de información sea compartida por los agentes involucrados, mediante un método algorítmico apropiado, denominado como algoritmo de consenso [4].

La estructura de estos algoritmos de control se basa en arquitecturas distribuidas, para esto se asume una interacción local de los agentes involucrados, de manera que el estado de información de interés, esté determinado por los estados de información de los agentes vecinos; así el objetivo es obtener una ley de control cuyo estado converja a un valor común [4]. Para esto se busca imponer una dinámica similar en los estados de información de cada agente, en el que cada estado se modele como una ecuación diferencial o ecuación en diferencias. En este trabajo la variable de interés estará enfocada en la posición en x y y, aplicado al seguimiento de trayectoria por contención similar a [3], [4].

Los algoritmos de consenso pueden diferenciarse en base al tipo de dinámica de los agentes como en [8], [9], donde se proponen una varios algoritmos para agentes que poseen una dinámica de simple o de doble integrador; también dependiendo de la jerarquía se define el consenso de líderes y seguidores, todos estos casos se presentan a continuación.

1.4.3.1. Consenso de agentes

Este algoritmo contiene un enfoque más general en la que sus agentes no precisan distinción alguna es decir que todos tienen la misma jerarquía. Para este caso se presentan algoritmos diferenciados por su dinámica ya sea de simple integrador, es decir que su modelamiento se basa en su cinemática $V = \frac{dx}{dt}$; y doble integrador o sistemas masa-fuerza $F = m\frac{d^2x}{dt^2}$, expresadas en las ecuaciones (1.7),(1.8) respectivamente.

Dinámica de simple integrador

$$\dot{x}_i(t) = u_i(t)$$

$$\dot{x}_i(t) = -\sum_{j=1}^{n} a_{ij}\left[x_i(t) - x_j(t)\right] \quad \forall\, i = \{1,2,3,\dots,n\} \tag{1.7}$$

Donde $u_i(t)$ es la señal de control en forma de velocidad del agente robótico i para un tiempo t; $x_i(t)$, $x_j(t)$ son las variables de interés de los agentes i,j respectivamente; a_{ij} corresponde al elemento i,j de la matriz de Adyacencia, asociada al grafo G de la formación multi-agente.

Dinámica de doble integrador

$$\ddot{x}_i(t) = u_i(t)$$

$$\ddot{x}_i(t) = \sum_{j=1}^{n} a_{ij}[(x_i(t) - x_j(t) + \gamma\left(\dot{x}_i(t) - \dot{x}_j(t)\right)\big] \quad \forall\, i = \{1,2,3,\dots,n\} \tag{1.8}$$

Donde $u_i(t)$ es la señal de control en forma de aceleración del agente robótico i para un tiempo t; $x_i(t)$, $x_j(t)$ son las variables de interés y sus derivadas $\dot{x}_i(t)$, $\dot{x}_j(t)$ de los agentes i,j respectivamente para un tiempo t; a_{ij} corresponde al elemento i,j de la matriz de Adyacencia, asociada al grafo G de la formación multi-agente. Además, se incluye la constante γ, que representa la fuerza de acoplamiento[3].

1.4.3.2. Consenso de agentes líderes y seguidores

Estos algoritmos precisan de agentes con jerarquía distinta como líderes y seguidores, esta jerarquía puede verse desde distintas formas como tamaño, tecnología, locomoción y otras características propias del agente robótico. También, se pueden clasificar según las características de software y hardware [3]. Un ejemplo sencillo puede relacionarse con las características de cada robot, en el cual los agentes líderes disponen de mayor equipamiento tecnológico como sensores, localizador GPS y detectores de objetos/colores, mientras que los agentes seguidores no disponen de detección sino esperan que el estado de información sea compartido por los agentes de mayor jerarquía.

Este tipo de consenso es útil en aplicaciones enfocadas al control por contención [4], donde los agentes denominados líderes encierran de un área de interacción, en el cual deben permanecer los agentes seguidores; estas superficies de restricción denominadas también como convexas, son polígonos geométricos, formados por ángulos inferiores a 180°.

Líderes estáticos

En este caso los líderes forman un área de contención sin movimiento, dentro de la cual ocurre la interacción de los agentes seguidores. Para esto, los algoritmos (1.9),(1.10) para simple y doble integrador respectivamente, permiten obtener el consenso entre los agentes

involucrados.

$$\dot{x}_i(t) = u_i(t) = 0 \quad i \in \mathcal{R}$$

$$\dot{x}_i(t) = -\sum_{j \epsilon \mathcal{Q} \cup \mathcal{R}} a_{ij}[x_i(t) - x_j(t)] \quad i \in \mathcal{Q} \tag{1.9}$$

$$\ddot{x}_i(t) = -\beta \dot{x}_\iota(t) - \sum_{j \epsilon \mathcal{Q} \cup \mathcal{R}} a_{ij}\{\beta[x_i(t) - x_j(t)] + [\dot{x}_i(t) - \dot{x}_j(t)]\} \quad i \in \mathcal{Q} \tag{1.10}$$

donde $\mathcal{R}$ es el conjunto de agentes líderes; $\mathcal{Q}$ es el conjunto de agentes seguidores, β, es una constante positiva; y el resto de los términos de ambos algoritmos son similares al caso del consenso de agentes sin distinción entre líderes y seguidores.

Líderes dinámicos

Una aplicación más completa y apegada a los procesos reales son formaciones con movimiento o dinámica; una aplicación muy común es el seguimiento de trayectoria por contención. Para estos sistemas, los agentes líderes forman un área de restricción en donde los agentes de menor jerarquía o seguidores pueden interactuar con el resto de las agentes mientras la formación de líderes se moviliza hacia el objetivo meta.

Aplicado a una situación real existe la posibilidad de encontrarnos con líderes cuyas velocidades sean la mismas o incluso donde sean distintitas, para ambos casos se presentan algoritmos en las ecuaciones (1.11) y (1.13) .

$$\ddot{x}_i(t) = u_i(t), \quad i \in \mathcal{R}$$

Líderes con la misma velocidad

$$\ddot{x}_i(t) = -\gamma\, sgn\left(\sum_{j \epsilon \mathcal{Q} \cup \mathcal{R}} a_{ij}\{\beta(x_i(t) - x_j(t)) + (\dot{x}_i(t) - \dot{x}_j(t))\}\right) - \beta \dot{x}_i(t) \tag{1.11}$$

donde γ es una constante positiva y $sgn(x)$ es la función signo definida por la ecuación (1.12); y el resto de los términos de ambos algoritmos, son similares al caso de consenso de líderes estacionarios.

$$sgn = \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 1 \end{cases} \tag{1.12}$$

Líderes con velocidad distinta

Considerando un ambiente más real existe la posibilidad en la cual los líderes no tienen la misma velocidad por lo cual el siguiente algoritmo considera este caso.

$$\ddot{x}_i(t) = u_i(t)$$

$$\ddot{x}_i(t) = -\sum_{j\epsilon\mathcal{Q}\cup\mathcal{R}} a_{ij}\left[\left(x_i(t) - x_j(t)\right) + \alpha\left(\dot{x}_i(t) - \dot{x}_j(t)\right)\right] - \beta sgn\left\{\sum_{j\epsilon\mathcal{Q}\cup\mathcal{R}}^{n} a_{ij}\left[\gamma\left(x_i(t) - x_j(t)\right) + \left(\dot{x}_i(t) - \dot{x}_j(t)\right)\right]\right\} \quad (1.13)$$

$$i \in \mathcal{Q}$$

donde aparece α, que es una constante positiva; y el resto de los términos son similares al caso anterior.

De la ecuación (1.13) se puede tomar en cuenta el comportamiento de la función signo de donde se puede notar que existe un cambio de -1 a 1 para $x = 0$. Esta señal en prototipos reales puede generar inconvenientes en la señal de control resultante conocido como *chattering*, por lo tanto, los actuadores pueden ser comprometidos en su vida útil. Para esto se recomienda utilizar la función sigmoide [4] [7], que proporciona una señal más atenuada en el cambio de -1 a 1 dependiendo del factor e.

Función sigmoide

$$sigm(x) = \frac{x}{|x| + e} \quad (1.14)$$

Con esta función aplicado al algoritmo de la ecuación (1.13), la expresión matemática del algoritmo queda expresada por la ecuación (1.15).

$$\ddot{x}_i(t) = u_i(t)$$

$$\ddot{x}_i(t) = -\sum_{j\epsilon\mathcal{Q}\cup\mathcal{R}} a_{ij}\left\{\left[x_i(t) - x_j(t)\right] + \alpha[\dot{x}_i(t) - \dot{x}_j(t)]\right\} - \beta\, sigm\left(\sum_{j\epsilon\mathcal{Q}\cup\mathcal{R}} a_{ij}\left\{\gamma\left[x_i(t) - x_j(t)\right] + \left[\dot{x}_i(t) - \dot{x}_j(t)\right]\right\}\right) \quad (1.15)$$

Para este trabajo los líderes están ubicados en los vértices del área de restricción, que encierran la superficie de interacción de los agentes seguidores. Los agentes denominados como seguidores son robots Pioneers, que se desplazan por la trayectoria, cuya referencia la obtienen a partir de consenso.

Las implementaciones de los algoritmos para este trabajo son realizadas empleando MATLAB; y el escenario de simulación dentro del ambiente de CoppeliaSim. Las características más relevantes de ambas plataformas para la realización del presente proyecto se describen a continuación.

1.4.4. COPPELIASIM Y MATLAB

CoppeliaSim es una plataforma de simulación robótica que sustituye a su versión clásica V-REP con mejoras significativas. Su entorno y biblioteca de prototipos robóticos, como el que se muestra en la Figura 1.6, junto con su amplia librería de funciones API para el control de robots, permiten una fácil y ágil implementación de escenarios de simulación. Por lo mencionado este software le hace frente a otras plataformas de simulación como Gazebo y otras; que han sido comparados en [10], [11]. El uso de este software alcanza investigaciones, aplicaciones industriales y desarrollos académicos; para esto dispone de 2 versiones gratuitas *Player* & *Edu* y una versión profesional no gratuita *Pro*. Actualmente este software se encuentra en constante desarrollo de la mano de Coppelia Robotics.

Para este trabajo se utilizará versión V4.0.0, revisión 4, disponible en la página oficial de CoppeliaSim.

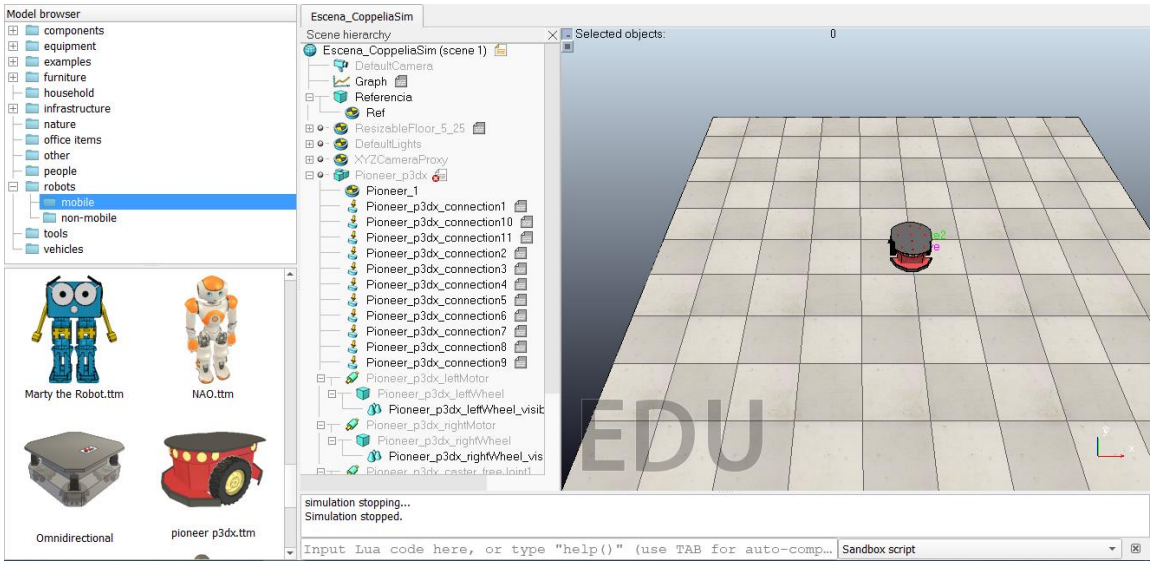


Figura 1.6 CoppeliaSim [Autoría propia]

Algunas de sus características más importantes son [12]:

Multiplataforma: está disponible para los sistemas operativos Windows, MacOS y Linux.

Motores de simulación gráficos: ODE, Bullet, Vortex y Newton.

Navegador: drag and drop (arrastrar y soltar).

Enfoques de programación: permite su integración mediante scripts incrustados, plugins, add-ons, nodos ROS, o APIs de clientes remotos.

Lenguajes de programación: se acopla a C/C++, Python, Java, Lua, Matlab u Octave.

Registro de datos y visualización: representación de gráficas en el tiempo, gráfico X / Y o curvas 3D.

La aplicación propuesta, se estará operando en conjunto con MATLAB, aprovechando sus características y ventajas que se describen rápidamente a continuación.

1.4.4.1. MATLAB

Es un software de ingeniería desarrollado por *MathWorks* que permite el desarrollo científico en muchas áreas. Por todas sus herramientas y complementos (*toolboxes*) se ha convertido en una plataforma muy utilizada para trabajos de investigación de ingenieros y científicos. Las prestaciones más importantes de MATLAB [13], son:

- Análisis de datos
- Gráficas
- Desarrollo de algoritmos
- Creación de aplicaciones
- Uso con otros lenguajes
- Cálculo paralelo
- Despliegue en escritorio y web
- Cálculo en la nube

En este trabajo se utilizan las herramientas de Simulink para las implementaciones de algoritmos y controladores; así también App Designer, para el diseño de una interfaz gráfica.

Simulink

Esta herramienta de MATLAB permite la programación visual en base a diagramas de bloques; es utilizado ampliamente en muchas áreas de ingeniería como: telecomunicaciones, eléctrica & electrónica, robótica, entre otras; y especialmente en ingeniería de control. Gracias a su amplia biblioteca de bloques y funciones para la programación, resulta sencilla la implementación de modelos y sistemas. Por lo tanto, es muy común encontrarse en muchos trabajos de investigación y desarrollos científicos [14].

App Designer

Esta herramienta de MATLAB permite la creación de apps personalizados; su fácil metodología de arrastrar y colocar los elementos visuales permite crear interfaces profesionales. Su diseño consiste principalmente en 2 partes: la primera donde los elementos visuales están disponibles en la librería de componentes; y la segunda consiste en el código fuente de cada uno de los componentes [15].

En este trabajo se integrará la aplicación dentro de los entornos de CoppeliaSim y MATLAB; la cual será comanda desde una interfaz gráfica creada en App Designer. Para este propósito, se requiere comunicar ambas plataformas, lo cual se describe a continuación.

1.4.4.2. Comunicación MATLAB y CoppeliaSim

CoppeliaSim ha creado herramientas para operar en conjunto al software de MATLAB y sus herramientas mediante el uso de un conjunto de funciones conocidas como *Remote API funtions*[16], las cuales han sido diseñadas específicamente para el intercambio de datos entre estas dos plataformas. Las funciones API desarrolladas por CoppeliaSim para el uso con MATLAB permiten el intercambio de múltiples datos, cada una de las funciones se encuentran adecuadamente descritas en [17]. Además, se encuentran disponibles una serie de instrucciones que permiten manejar directamente los parámetros de run y stop de la simulación.

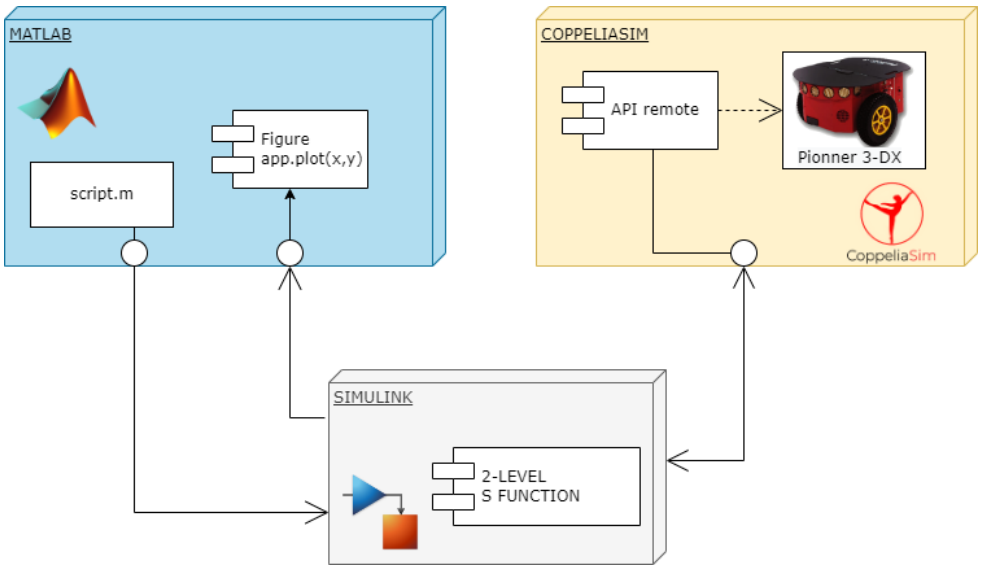


Figura 1.7. Diagrama de comunicación MATLAB – CoppeliaSim [Autoría propia].

La conexión entre ambos programas se realiza mediante un puerto de comunicación entre programas del mismo PC; esta comunicación puede ser de tipo sincrónica o asincrónica, dependiendo de los requerimientos de la simulación. Para este trabajo la metodología

utilizada se resume en la Figura 1.7, en base a una comunicación sincrónica. Para más detalles con respecto a la metodología utilizada para el desarrollo de esta aplicación, se puede revisar el ANEXO III, donde, se pueden encontrar las instrucciones y recomendaciones necesarias para comunicar estas dos herramientas.

2. METODOLOGÍA

Este capítulo describe el desarrollo de la aplicación en base a una metodología aplicada, para lo cual se ha recopilado la teoría asociada, descrita en el capítulo anterior. En base a esto, la presente aplicación consiste en la construcción de un sistema multi-agente robótico, de uno, dos y hasta tres robots Pioneers 3-DX, bajo una formación líder-seguidor; enfocada a la tarea del seguimiento de trayectoria. Para esto se requiere una serie de etapas como: generación de trayectoria, aplicación del algoritmo de consenso, controlador de movimiento; las cuales serán implementadas dentro MATLAB-Simulink. Todo esto operando en conjunto con CoppeliaSim donde se encuentra la implementación de la formación. Además, para el control de la aplicación se cuenta con una interfaz gráfica que permita comando de la simulación. Todas estas etapas mencionadas han sido resumidas de manera gráfica en la Figura 2.1.

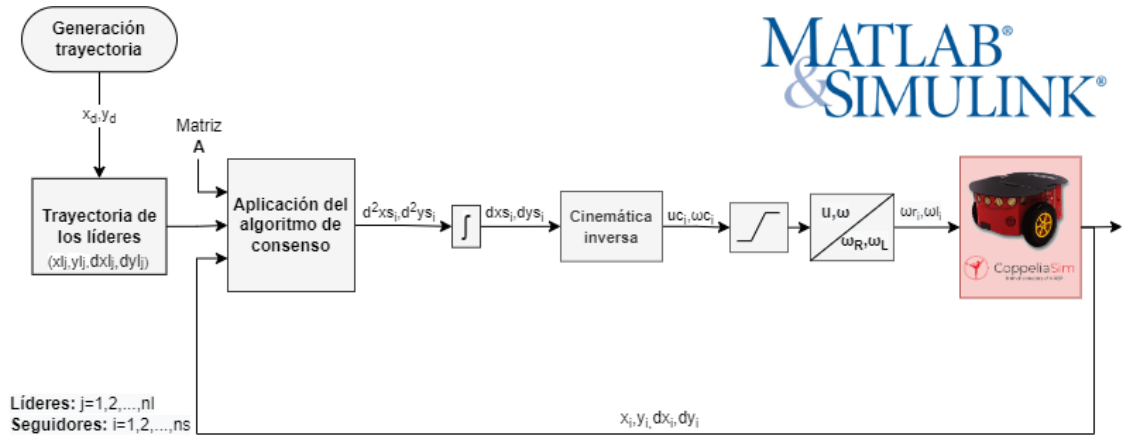


Figura 2.1. Esquema de control de la aplicación [Autoría propia]

Todas las etapas de color gris son implementación dentro de MATLAB-Simulink; por otro lado, la única etapa marcada con el color rojo estará dentro de CoppeliaSim. Todo este desarrollo se describe detalladamente a continuación.

2.1. CONTROL DE MOVIMIENTO PIONEER 3-DX

Se parte definiendo el controlador de movimiento adecuado para el seguimiento de trayectoria, en base al modelo matemático del robot. Para esto se necesita en primer lugar

una etapa dedicada a la generación de trayectorias según Figura 2.1, la cual se describe a continuación.

2.1.1. GENERACIÓN DE TRAYECTORIAS

Las trayectorias de seguimiento están establecidas por las ecuaciones descritas en la Tabla 2.1, cuya parametrización permite obtener las coordenadas en el plano XY de cada una en función del tiempo t.

Tabla 2.1. Trayectorias parametrizadas

Trayectoria	$x[m]$	$y[m]$
Circular	$x(t) = Rcos(\omega_f t)$	$y(t) = Rsin(\omega_f t)$
Lemniscata	$x(t) = Rcos(\omega_f t)$	$y(t) = Rsin(2\omega_f t)$
Sinusoidal	$x(t) = mt + b$	$y(t) = Asin(1.5\omega_f t)$
Cuadrada	$x(t) = \begin{cases} l/2, & 0 < t \leq t_l \\ -mt + l, & t_l < t \leq 3t_l \\ -l/2\,, & 3t_l < t \leq 5t_l \\ mt - 3l, & 5t_l < t \leq 7t_l \\ l/2, & 7t_l < t \leq 8t_l \end{cases}$	$y(t) = \begin{cases} mt, & 0 < t \leq t_l \\ l/2, & t_l < t \leq 3t_l \\ -mt + 2l, & 3t_l < t \leq 5t_l \\ -l/2, & 5t_l < t \leq 7t_l \\ mt - 4l, & 7t_l < t \leq 8t_l \end{cases}$

En la Tabla 2.1, las constantes R determinan la amplitud de sus respectivas trayectorias y $\omega_f = 2\pi/T$, es su frecuencia de oscilación, donde T es el tiempo de simulación. Para la trayectoria sinusoidal, m, b, son las constantes de la ecuación lineal de la posición x. Para la trayectoria cuadrada $l, m = (4L/T)$ son parámetros que dependen de la longitud del lado y t_l es $1/8T$.

Una vez dada la parametrización de las trayectorias, estas son implementadas dentro de MATLAB-Simulink; como siguiente paso, se definen los controladores de movimiento los cuales se describen a continuación.

2.1.2. CINEMÁTICA INVERSA

Actualmente se pueden encontrar múltiples controladores basados en PID, Fussy Logic, Slinding mode, etc; los cuales siguen en constate desarrollo. En algunos trabajos como [18], se analizan algunos controladores, describiendo sus características y eficiencia.

En este trabajo se utiliza el controlador basado en el modelo matemático de la plataforma robótica, empleando la cinemática inversa.

2.1.2.1. Controlador cinemático

Partiendo del modelamiento dado en la ecuación (1.1), tomando las expresiones correspondientes a las variables de estado de x, y, se la puede expresar de manera simbólica matricial como en la ecuación(2.1), a la cual se la denomina cinemática directa.

$$\dot{H} = JV \tag{2.1}$$

donde, V, es un vector columna con las velocidades lineal y angular; $\dot{H}$, es un vector columna con las velocidades en los ejes x y y; y J, es el Jacobiano.

Aplicando el concepto de cinemática inversa, se despeja el vector columna V de la ecuación (2.2).

$$V = J^{-1}\dot{H} \tag{2.2}$$

Esta sencilla relación permite relacionar un espacio bidimensional de velocidades cartesianas $(\dot{x}, \dot{y})$ a un espacio de dos velocidades: lineal (u) y angular (ω). Esta implementación se resume gráficamente, en la Figura 2.2, donde se nota adicionalmente los saturadores en las señales de control, obtenidos de las especificaciones del fabricante [19] y son de ±1.2 m/s y ±5.24 rad/s para las velocidades lineal y angular, respectivamente.

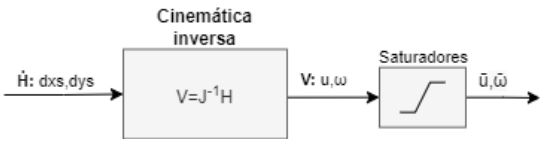


Figura 2.2. Controlador de movimiento propuesto [Autoría propia]

Una vez obtenido el seguimiento de trayectoria para un solo robot Pioneer 3-DX, el siguiente paso es realizar el control de todo un conjunto en base al consenso. Para esto se ha define utilizar el robot móvil como agente seguidor dentro una formación multi-agente, los cuales se describen en la siguiente sección.

2.2. CONFORMACIÓN DEL SISTEMA MULTI-AGETE: LÍDERES-SEGUIDORES

Definido el controlador adecuado para el agente robótico; el siguiente es paso es definir la formación del sistema multi-agente. Para este trabajo, como se explicó en capitulo anterior, se utiliza una formación basada en líderes y seguidores.

Líderes

En base a [3], cuando no se cuenta con una estructura real que pueda ejercer de líder, se puede hacer uso de líderes virtuales que son puntos referencia que dependen de la

posición x y y de la trayectoria de seguimiento. Por lo tanto, los líderes virtuales se consideran como agentes con dinámicas ideales, y son capaces de proporcionar información de posición y velocidad, por lo que pueden intervenir en el consenso. Además, estos agentes cumplen la función de formar la superficie de contención en forma de polígono, por lo que mínimo se requiere de 3 líderes de este tipo. La ubicación de los líderes es obtenida mediante las ecuaciones (2.3).

$$i = 1,2,\dots,l$$

$$\begin{aligned} x_l(i) &= x_d + d\cos\left(\frac{2i\pi}{n_l} + \phi\right), \\ y_l(i) &= y_d + d\sin\left(\frac{2i\pi}{n_l} + \phi\right) \end{aligned} \tag{2.3}$$

donde, x_d, y_d es la posición de la trayectoria de seguimiento del sistema; n_l, es el número de líderes virtuales; ϕ, la orientación de la formación multi-agente; d, es la distancia del punto de referencia hacia los líderes virtuales; y i, define el número de agente líder y su ubicación en la superficie de restricción.

Para este trabajo se consideran 3 agentes líderes virtuales, que forman un triángulo, como se ve en la Figura 2.3; el área de contención es constante en forma; la orientación por otro lado es constante a lo largo de toda la trayectoria.

Seguidores

Los agentes seguidores permanecerán dentro del área de restricción, determinada por los agentes líderes, los cuales determinan el movimiento del sistema, por lo que las referencias de los seguidores se las obtiene en base al consenso. Para este trabajo se plantea trabajar con formaciones en base al número de seguidores robóticos de uno, dos y hasta tres. Una vez definida la formación y las ponderaciones del grafo representativo, el siguiente paso es encontrar el modelo algebraico de toda la formación. A continuación, se detalla el análisis para el caso más simple con el cual se pueda comprender el procedimiento de diseño.

Para el caso 1, el sistema cuenta con 4 agentes involucrados de los cuales tres son líderes y uno de ellos corresponde a un seguidor robótico, tal como se presenta en el grafo de la Figura 2.3.

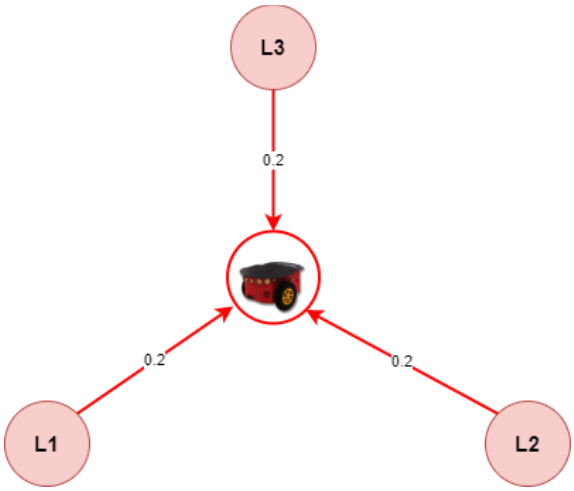


Figura 2.3. Formación 3 líderes y 1 seguidor, resaltando el árbol de expansión dirigido [Autoría propia]

Para que los agentes estén dispuestos físicamente en el plano XY como en la Figura 2.3, su ponderación debe ser igual entre cada agente líder y el único seguidor, recordando que el valor de ponderación indica el grado de influencia de un agente sobre el otro dependiendo de la dirección de la arista. Del grafo dirigido, se obtienen sus dos matrices representativas, las cuales se presentan en la Tabla 2.2.

Tabla 2.2. Matrices representativas sistema multi-agente 3 líderes y 1 seguidor

Matriz de Adyacencia	Matriz de Laplaciana
$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.2 & 0.2 & 0.2 & 0 \end{bmatrix}_{4x4}$	$L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -0.2 & -0.2 & -0.2 & 0.6 \end{bmatrix}_{4x4}$

ANÁLISIS DE LA FORMACIÓN: Árbol de expansión dirigido

La Figura 2.3, muestra como de cada agente líder tiene al menos un camino directo, hacia el agente seguidor lo cual asegura que todos los estados de interés lleguen hacia el agente seguidor Pioneer, por tanto, asegura el consenso de la formación multi-agente.

Mediante el mismo análisis descrito para el sistema con un seguidor, se determinan las matrices representativas de las formaciones con dos y tres agentes robóticos dados en la Figura 2.4, cuyas matrices representativas se resumen en la Tabla 2.3. Para estos sistemas, de manera similar al primer caso, los líderes forman un área triangular de restricción para la interacción de los agentes seguidores.

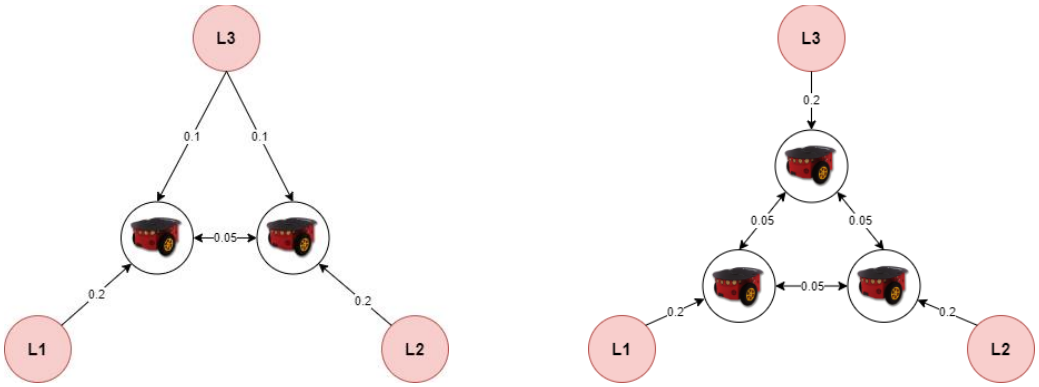


Figura 2.4. Grafos de sistemas multi-agente de líderes y seguidores, a) con 2 seguidores b) con 3 seguidores. [Autoría propia]

Tabla 2.3. Tabla de matrices de Adyacencia y Laplaciana de un sistema con dos y tres agentes robóticos

	Matriz de Adyacencia	**Matriz Laplaciana**
Con 2 seguidores	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0.1 & 0 & 0.05 \\ 0 & 0.2 & 0.1 & 0.05 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -0.2 & 0 & -0.1 & 0.35 & -0.05 \\ 0 & -0.2 & -0.1 & -0.05 & 0.35 \end{bmatrix}$
Con 3 seguidores	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0 & 0 & 0 & 0.05 & 0.05 \\ 0 & 0.2 & 0 & 0.05 & 0 & 0.05 \\ 0 & 0 & 0.2 & 0.05 & 0.05 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -0.2 & 0 & 0 & 0.3 & -0.05 & -0.05 \\ 0 & -0.2 & 0 & -0.05 & 0.3 & 0.05 \\ 0 & 0 & -0.2 & -0.05 & -0.05 & 0.3 \end{bmatrix}$

CENTROIDE DE LA FORMACIÓN:

El centroide de la formación permite representar en una única coordenada (x_c, y_c), toda la formación, lo que permite analizar toda formación. Este parámetro se lo encuentra en base a geometría analítica, para lo que se utilizan las expresiones de la ecuación (2.4).

$$x_c = \frac{x_{Si} + \cdots + x_{Sn}}{ns}$$
$$y_c = \frac{y_{Si} + \cdots + y_{Sn}}{ns} \qquad (2.4)$$

donde, x_{Si}, y_{si}, responden a las posiciones x, y del agente seguidor i; ns, es el número de agentes seguidores en la formación.

Una vez conseguidas las formaciones de los sistemas propuestos, ahora se analiza el algoritmo de consenso aplicado al sistema multi-agente propuesto.

2.3. ALGORITMO DE CONSENSO

El algoritmo de control de la ecuación (1.15), proporciona el consenso cuando se trabaja con agentes líderes y seguidores. Para este caso los estados de interés son las posiciones x, y dentro de plano cartesiano, las cuales determinan la posición de cada uno de los agentes robóticos.

Este algoritmo ha sido estrictamente analizado en [4], donde se analiza todo el algoritmo mediante el criterio de Lyapunov, de donde se resumen los criterios de selección de las constantes α, β, γ.

$$\alpha > 0 \tag{2.5}$$

$$\beta > \frac{\|M^{-1}\Psi_F\|}{1-c} \tag{2.6}$$

donde, M, es una matriz asociada a la matriz Laplaciana L, está definida como $M = [m_{ij}] \in \mathbb{R}^{m\times m}$ con $m_{ij} = \ell_{ij}$ si $i \neq j$ y $m_{ii} = \sum_{k\neq i} \ell_{ik}$ [4]; Ψ_F, es el vector aceleraciones; y c, una constante positiva entre 0 y 1.

$$\gamma > \sqrt{\lambda_{\min}(M)} \tag{2.7}$$

donde, λ_{min}, es el valor propio mínimo de la matriz M.

Para este trabajo los valores seleccionados, en base a los criterios de las referencias citadas [3], [4], se presentan en la Tabla 2.4.

Tabla 2.4. Constantes del algoritmo de consenso

Constante	Símbolo	Valor
Alfa	α	2.5
Beta	β	2.5
Gama	γ	0.25

La estabilidad del algoritmo, ha sido profundamente analizado en los trabajos [4], [9], [20], [21].

2.4. IMPLEMENTACIÓN DEL SISTEMA MULTI-AGENTE

Basado en los capítulos anteriores, el siguiente paso es la implementación de la aplicación propuesta para este trabajo, que se resume gráficamente en la Figura 2.1; donde se observa una distribución por bloques la implementación de la aplicación. Todos los bloques descritos se implementan dentro de MATLAB-Simulink, con excepción de la plataforma robótica la cual esta implementada dentro del ambiente de CoppeliaSim diferenciada con color distinto.

2.4.1. IMPLEMENTACIÓN EN SIMULINK

La primera etapa, corresponde a la generación de trayectorias donde se implementan en Simulink las ecuaciones parametrizadas de las 4 trayectorias de seguimiento de la Tabla 2.1 en función del tiempo t, como se aprecia en la Figura 2.5.

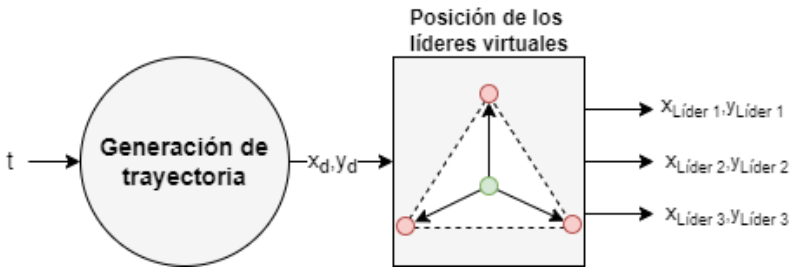


Figura 2.5. Generación de trayectorias [Autoría propia].

En función del punto de trayectoria generado (x_d, y_d), el bloque denominado "Posición de los líderes virtuales" genera las posiciones $(x_{Líder\ 1}, y_{Líder\ 1})$, $(x_{Líder\ 2}, y_{Líder\ 2})$, $(x_{Líder\ 3}, y_{Líder\ 3})$, por medio de las ecuaciones (2.3), con una distancia entre el punto de referencia y las posiciones de los líderes virtuales de $d = 1.5m$ y una orientación de $\phi = 90°$.

Posteriormente se realiza la aplicación del algoritmo de consenso de acorde a la ecuación (1.15) y el controlador de movimiento del robot móvil diseñado anteriormente, siguiendo una estructura como se observa en la Figura 2.6. Esta implementación está dispuesta bajo una arquitectura distribuida es decir que cada robot cuenta con su controlador de movimiento y la aplicación del algoritmo de consenso individual, utilizando los de bloques y funciones que proporciona Simulink.

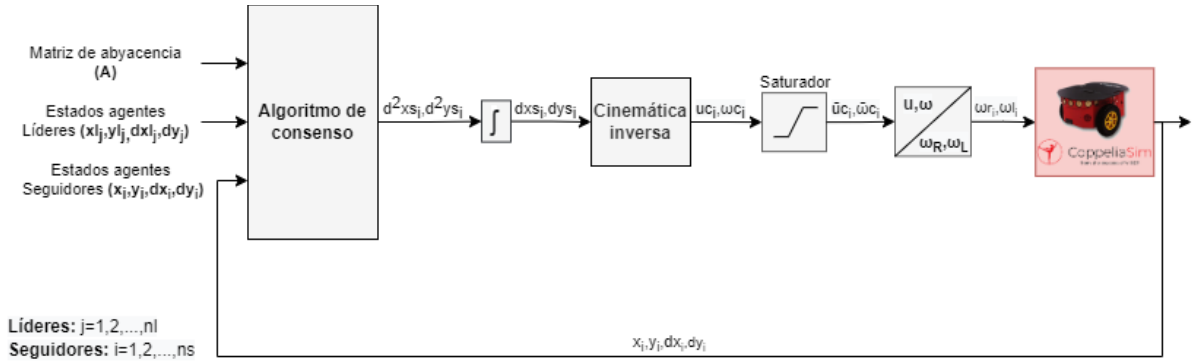


Figura 2.6. Consenso y Cinemática inversa [Autoría propia].

Para el consenso, se requiere la matriz representativa (Matriz de Adyacencia) de la formación del multi-agente obtenida a partir del grafo dirigido; el estado de información y su derivada, para el caso de esta aplicación la posición en x, y, que son las variables de interés; la cinemática inversa y los saturadores descritos anteriormente. Además, las relaciones de velocidad lineal angular con las velocidades a cada una de las ruedas del robot Pioneer dentro de CoppeliaSim, mediante las ecuaciones (2.8),(2.9).

$$\omega_r = \frac{2u + \omega l}{2R} \tag{2.8}$$

$$\omega_l = \frac{2u - \omega l}{2R} \tag{2.9}$$

donde, ω_r, ω_l, son las velocidades de las de las dos ruedas derecha e izquierda; u, ω, son las las señales de control en forma de velocidades lineal y angular. Y l, R, son características del robot virtual Pioneer de la plataforma de simulación las cuales se resumen en la Tabla 2.5.

Tabla 2.5. Características del Robot Virtual Pioneer 3-DX

Característica del robot Pioneer	Descripción	Valor
l	Distancia entre ambas ruedas	0.331
$R = D/2$	Radio de las ruedas	0.195/2

Finalmente, para enlazar el ambiente de Simulink y CoppeliaSim, se utiliza como interface el bloque denominado *2-Level MATLAB s-function*, donde se implementará la comunicación con CoppeliaSim, definiendo características como parámetros de salida y entrada, siguiendo la metodología descrita en el ANEXO II.

Los datos de entrada corresponden a las señales de control del robot móvil, posiciones de los líderes virtuales, y la referencia de la toda la formación; estas dos últimas permitirán representar toda la formación multi-agente dentro del ambiente de CoppeliaSim. Por otro

lado, están las salidas que contienen los estados actuales de los agentes robóticos; esto permitirá cerrar el lazo de control de todo el sistema.

Figura 2.7. Bloque Level-2 MATLAB s-function [Autoría propia].

Las entradas y salidas de la Figura 2.7. corresponden a:

- Referencia de seguimiento x_{ref}, y_{ref}
- Posiciones de los lideres virtuales $x_{L_1}, y_{L_1}, x_{L_2}, y_{L_2}, x_{L_3}, y_{L_3}$
- Señales de control de los agentes robóticos $uc_i, \omega c_i$

Las salidas corresponden a:

- Estado actual del Pioneer 3-DX $x_i, y_i, \psi_i, \dot{x}_i, \dot{y}_i, \omega_i$

2.4.2. IMPLEMENTACIÓN COPPELIASIM

Esta implementación consiste en la construcción del escenario de simulación del sistema multi-agente robótico. El bloque representado en color rojo de la Figura 2.7, indica que el Pioneer 3-DX se encuentra dentro del entorno de CoppeliaSim. Además, se emplean otros objetos representativos que mejoran la representación de la formación sistema multi-agente.

2.4.2.1. Agente seguidor Pioneer 3-DX

Esta implementación resulta sencilla, debido a la existencia de la plataforma robótica dentro de la biblioteca de robots disponible en CoppeliaSim, como se observa en la Figura 2.8. Para agregar este modelo, basta con arrastrarlo hacia el escenario y soltarlo.

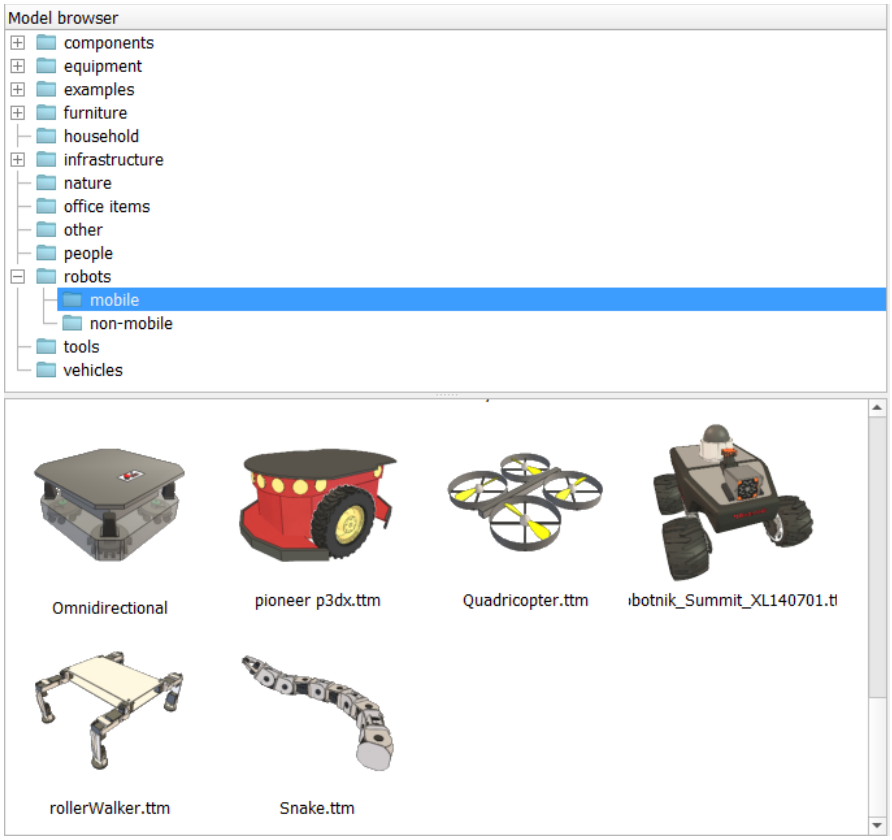


Figura 2.8. Librería de robots móviles en CoppeliaSim [Autoría propia]

Para este trabajo las señales de control son obtenidas desde MATLAB-Simulink, por esto se debe deshabilitar el script asociado al robot, que contiene un código por defecto, tal como se ve en la Figura 2.9.

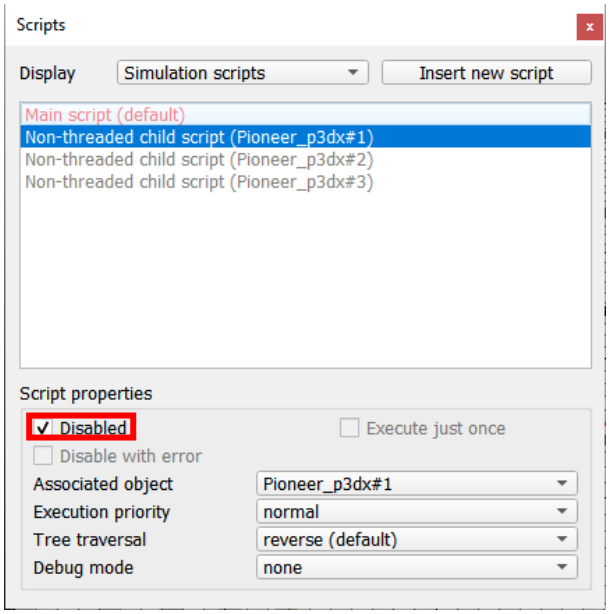
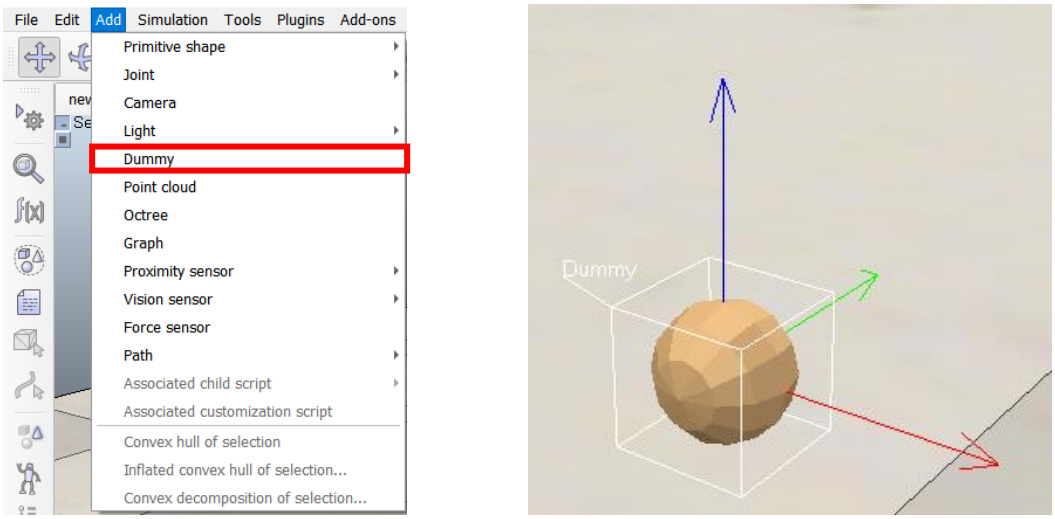


Figura 2.9. Deshabilitación del script Pioneer3-DX [Autoría propia].

Para obtener algunas propiedades gráficas que permitan observar la trayectoria de seguimiento de la plataforma robótica, es necesario añadir un objeto multipropósito denominado como *Dummy*; este procedimiento se resume en la Figura 2.10.



a) b)

Figura 2.10. Objeto multipropósito Dummy; a) inserción de objeto, b) objeto dentro de la escena. [Autoría propia]

Además de agregar este objeto a la escena de simulación se requiere asociarlo al robot. Este paso se realiza fácilmente arrastrando y soltando el objeto, dentro de la ventana de jerarquía de escena, como se observa en la Figura 2.11, donde se aprecia el objeto *Dummy* renombrado como Pioneer_1.

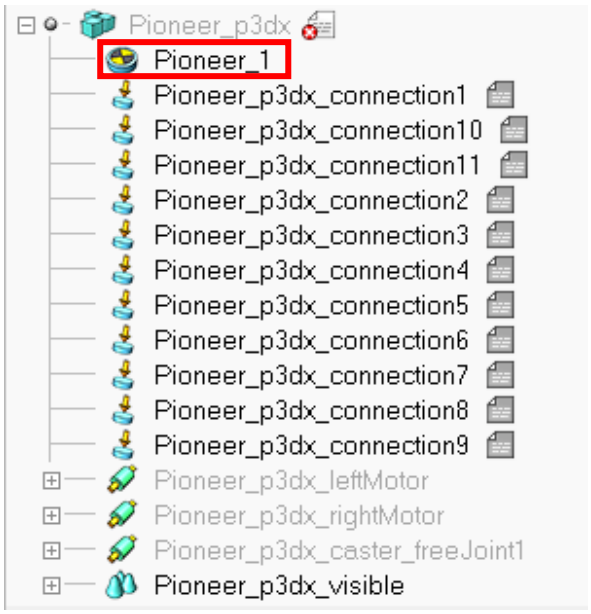


Figura 2.11. Jerarquía de escena del robot Pioneer [Autoría propia]

Con este paso se tiene el robot Pioneer 3-DX, estructuralmente completo. Adicional a los agentes robóticos, también los agentes líderes virtuales requieren una presentación dentro de la escena de simulación, para esto se utilizan objetos adicionales, cuyas implementaciones se describen a continuación.

2.4.2.2. Inserción de objetos: Primitive shapes

La posición de los agentes líderes virtuales, son representados mediante un objeto tipo primitive disc disponible en CoppeliaSim. Las propiedades de los 3 objetos que representarán a los agentes líderes aparecerán descritas en la Tabla 2.6, cuyas características son configuradas en la ventana de propiedades como el que se observa en la Figura 2.12. Adicionalmente, se agrega un objeto adicional que representa la posición de la referencia de todo el sistema multi-agente durante el seguimiento de trayectoria.

Tabla 2.6. Descripción de los objetos *Primite Shape* tipo disco

Objeto	Radio	Color
Líder 1	10 mm	Rojo
Líder 2	10 mm	Rojo
Líder 3	10 mm	Rojo
Referencia	5 mm	Gris

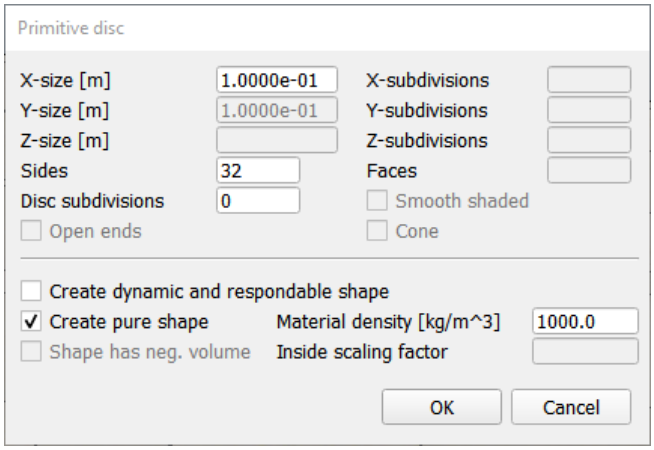


Figura 2.12. Objeto tipo disco – propiedades [Autoría propia]

Toda esta implementación permite visualizar un ambiente de simulación, más representativo de la aplicación propuesta. Estas implementaciones se pueden observar en la Figura 2.13.

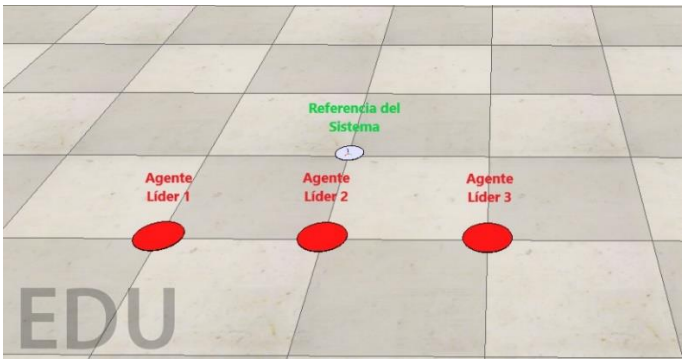


Figura 2.13. Objetos representativos: 3 líderes virtuales, Referencia del sistema

Todas las implementaciones descritas, del sistema multi-agente, sobre el entorno de simulación, Se puede verificar dentro de la jerarquía de escena de la aplicación desarrollada en se aprecia en Figura 2.14.

Figura 2.14. Arquitectura de objetos final del escenario de simulación [Autoría propia]

De donde:

Agentes líderes: Lider_1, Lider_2 y Lider_3, que corresponden a las posiciones de los agentes virtuales.

Agentes seguidores: Son robots móviles Pioneers 3-DX.

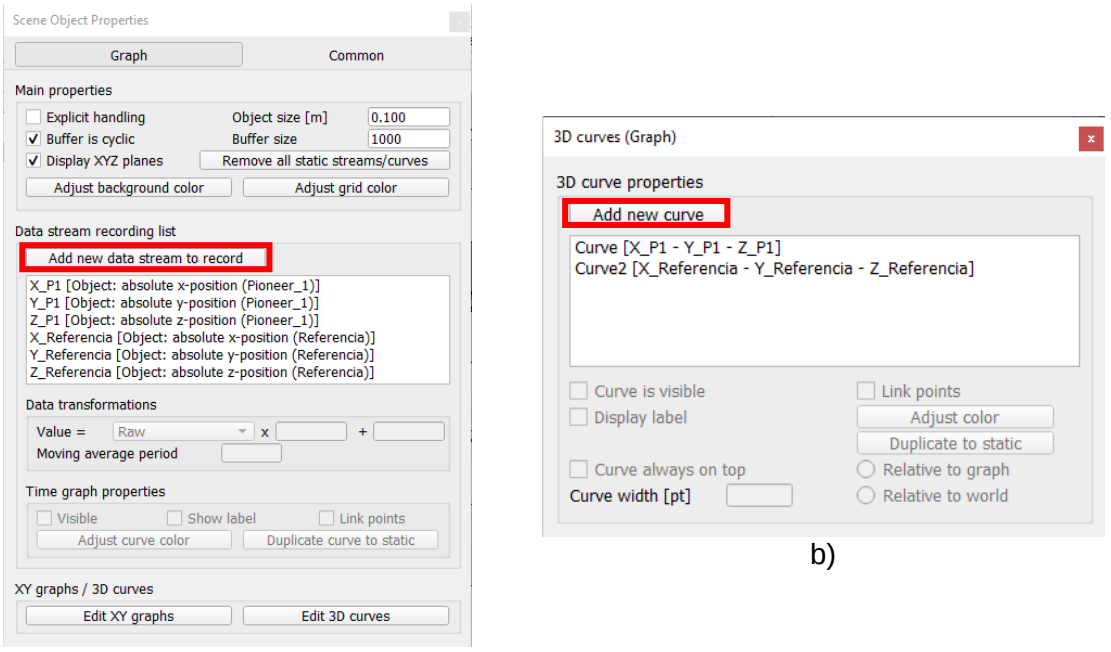
Referencia de seguimiento: Referencia de seguimiento para todo el sistema multi-agente.

Se han descrito la creación de todos objetos necesarios, para la representación del sistema multi-agente. El siguiente paso consiste en la personalización del objeto *Graph* que permitirá la visualización de la trayectoria recorrida por los agentes seguidores y la referencia del sistema multi-agente; este paso se describe a continuación.

2.4.2.3. Objeto gráfico: *Graph*

El objeto *Graph* al igual que todos los objetos se deben de insertar a la escena de simulación utilizando la barra de tareas en la sección *add,* donde se puede encontrar todos

los objetos disponibles. Este objeto permitirá la visualización un registro de las posiciones de objetos dentro de la escena mediante marcas de tiempo. Para esto se seleccionan los identificadores cartesianos que asocien, las coordenadas absolutas x, y y z del objeto, mediante la opción *add new data stream to record* marcado en rojo, como se observa en la Figura 2.15 (a).



a) b)

Figura 2.15. Configuración del objeto Graph, a) Selección de indicadores xyz, b) Adición de curvas 3D [Autoría propia]

El siguiente paso es la asignación de indicadores xyz, presionando Edit ED curves, de donde aparecerá una nueva ventana de dialogo denominada como 3D curves (Graph), en esta ventana se podrá editar personalizar la visualización de las marcas de tiempo. Para esta aplicación los objetos seleccionados son los agentes seguidores robóticos y la referencia de seguimiento del sistema; para esto se debe agregar los indicadores seleccionados anteriormente de las posiciones absolutas en x, y y en z, utilizando la opción *Add new curve* marcado en rojo, como se ve en la Figura 2.15 (b). Adicional, a esto se debe de iniciar la comunicación con MATLAB, lo cual se describe a continuación.

2.4.2.4. Puerto de comunicación entre MATLAB-Simulink

Para la asignación del puerto existen dos opciones: asociar un nuevo script a un objeto existente o utilizar cualquier script de algún objeto existente, dentro de la escena de simulación. Una vez seleccionado el script, dentro de su estructura de código, se asigna el

puerto *19999* mediante una función API dentro de la sección *sysCall_init*, tal como se ve en la Figura 2.16; esto una asegura intercambio sincronizado de datos.

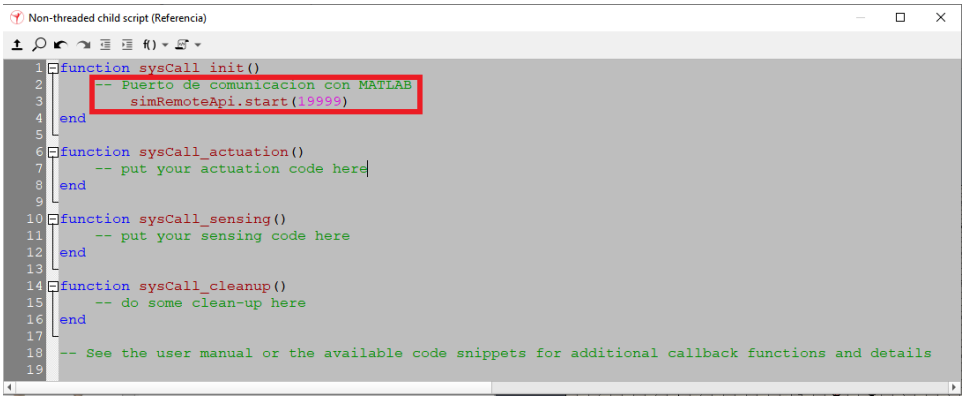


Figura 2.16. Declaración del puerto de comunicación con MATLAB [Autoría propia]

La implementación final se lo puede apreciar en la Figura 2.17. Para iniciar con la simulación, se debe iniciar la simulación dentro de la plataforma CoppeliaSim, y posteriormente el programa implementado en MATLAB-Simulink.

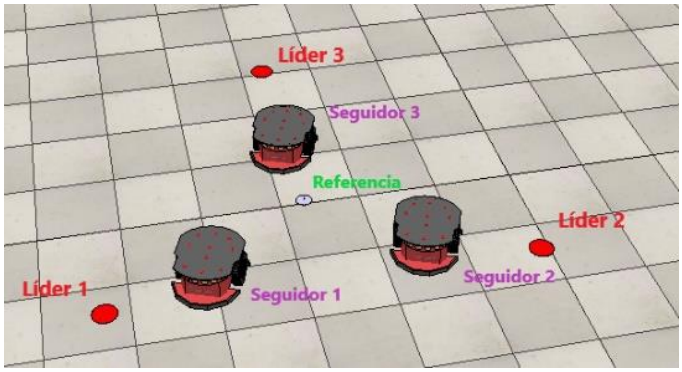


Figura 2.17. Vistas del escenario simulación [Autoría propia]

Para mejorar la operación de la aplicación desarrollada, se cuenta con una interfaz gráfica que permita su manejo, la cual se describe a continuación.

2.5. INTERFAZ GRÁFICA

A más del manejo de la aplicación, esta interfaz de usuario está diseñada para mostrar resultados como: la trayectoria recorrida, señales de control y errores de posición. Su diseño se realiza utilizando la herramienta de MATLAB *AppDesigner* de la versión R2021b, debido a sus características y sencillo entorno de diseño.

La interfaz gráfica consta de 3 ventanas, identificadas de la siguiente manera:

- Ventana de presentación
- Ventana de simulación
- Ventana de resultados

La primera ventana describe las características de la aplicación, la segunda permite al usuario definir algunos parámetros antes de empezar la simulación de toda la aplicación; la interfaz permite iniciar o parar todo en el caso de ser necesario. Finalmente, con la simulación culminada se mostrará la trayectoria recorrida por los agentes seguidores. Además, el usuario podrá encontrar una nueva ventana que muestra resultados adicionales de la simulación. Todas estas ventanas se describen en detalle a continuación.

2.5.1. VENTANA DE INICIO

Esta ventana tiene el objetivo de presentar una breve descripción de la aplicación. Además, se incluye un botón de ayuda, como se ve en la Figura 2.8, que permitirá abrir el manual de usuario de la interfaz gráfica.

Figura 2.18. Pantalla principal [Autoría propia]

Al presionar el botón “SIGUIENTE”, esta ventana se cerrará y se abrirá la siguiente denominada de Simulación.

2.5.2. VENTANA DE SIMULACIÓN

En esta ventana el usuario podrá seleccionar las características de la simulación.

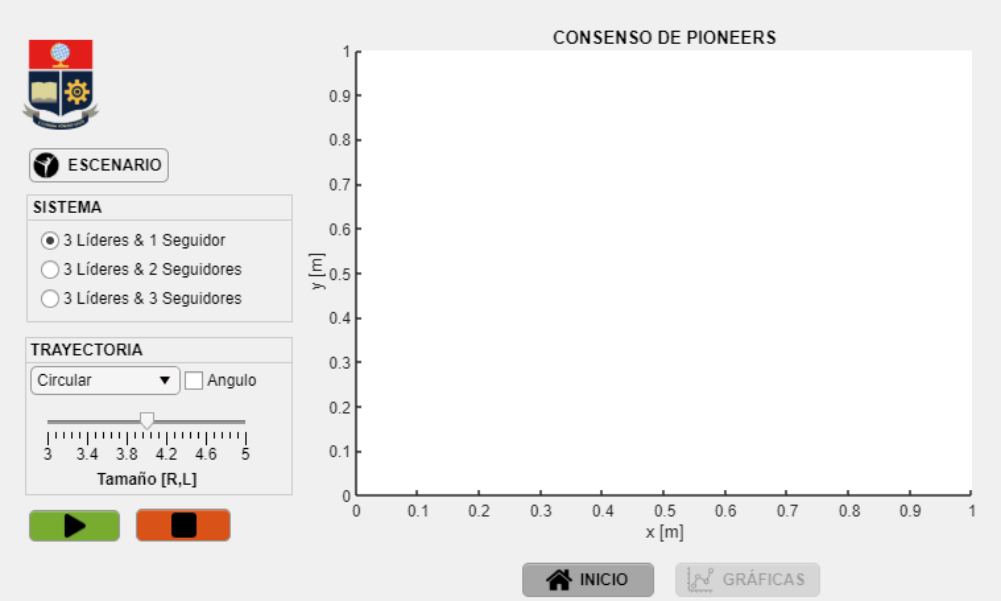


Figura 2.19 Ventana de simulación [Autoría propia]

Para iniciar la Simulación se debe seleccionar sus características como el tipo de sistema, la trayectoria de seguimiento; el tamaño de la trayectoria y la inclusión de un ángulo de trayectoria en el caso de ser necesario; caso contrario la simulación puede iniciar con los datos por defecto. Además, se tiene incluido un botón denominado “Escenario” con el cual, el usuario podrá abrir el escenario de simulación de manera directa. Todo esto se aprecia de mejor manera en la Figura 2.19.

Terminada la simulación se presenta la trayectoria recorrida por el sistema multi-agente y se habilitará el botón denominado “GRÁFICAS”; este botón, permite visualizar más resultados en una siguiente ventana.

2.5.3. VENTANA DE RESULTADOS

Esta ventana denominada de Resultados, se la denomina así porque permite la visualización de los resultados de la simulación con las características de seleccionadas en la ventana de simulación. Los resultados corresponden a las señales de control u, ω , y errores en las coordenadas y, x. La presentación gráfica puede ser grupal o individual, todo esto se podrá personalizar en la ventana presentada en la Figura 2.20. Con todas las características seleccionadas y después de presionar el botón “MOSTRAR” inmediatamente aparecerán los resultados gráficos seleccionados. Esta ventana está diseñada para mostrar las veces que sea necesaria los gráficos, una vez seleccionado las características de la gráfica y presionando nuevamente el botón de “MOSTRAR”. El botón “CERRAR”, cierra esta ventana, y con ayuda de la ventana de anterior podrá realizar una nueva simulación.

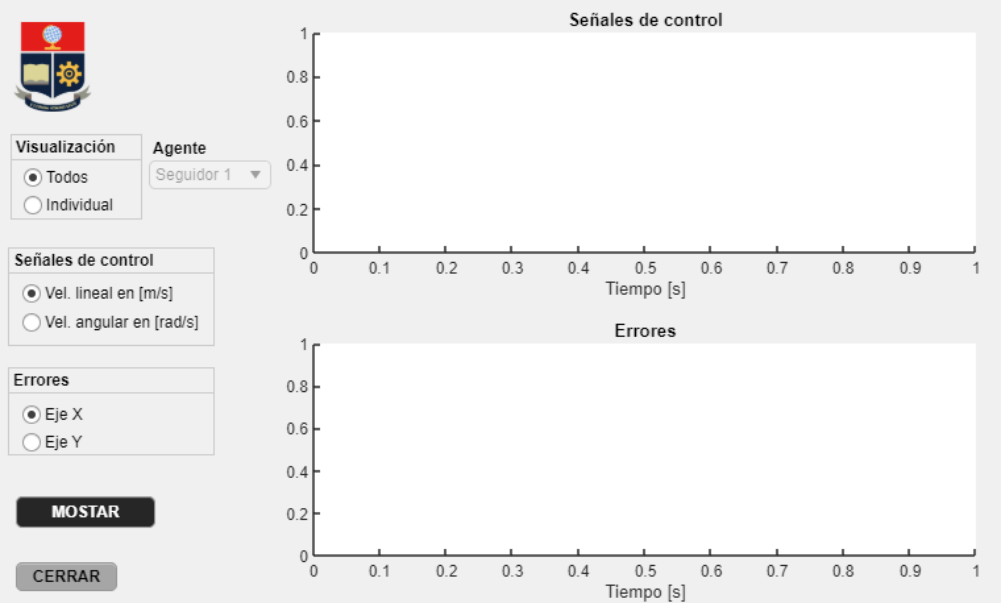


Figura 2.20 Ventana de resultados [Autoría propia]

Todos los resultados obtenidos de la aplicación resultante son analizados de manera detallada en el siguiente capítulo.

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

Este capítulo presenta los resultados obtenidos de la aplicación desarrollada, mediante gráficas como: la trayectoria recorrida, las señales de control y errores de posición; que permitan verificar y analizar el funcionamiento y desempeño. También, se mencionan varias conclusiones y recomendaciones obtenidas a partir del desarrollo de este Trabajo de Integración Curricular; esto con el fin de complementar el análisis de los resultados; y motivar el desarrollo de próximos trabajos relacionados con esta aplicación.

3.1. PRUEBAS Y RESULTADOS

Las pruebas están divididas en dos secciones, en primer lugar, se describen los resultados del desempeño del controlador de movimiento en base a la cinemática inversa de manera que se verifique el seguimiento de trayectoria para el robot móvil Pioneer 3-DX. En segundo lugar, se tiene las pruebas realizadas a toda la aplicación ya implementada, donde se verifique el su correcto funcionamiento.

3.1.1. PRUEBA DE SEGUIMIENTO DE TRAYECTORIAS PIONEER 3-DX

Para esta prueba, se realizan simulaciones durante un tiempo 60 segundos para las 4 para trayectorias propuestas en este trabajo. La plataforma robótica tendrá que realizar el seguimiento de trayectoria en los 4 casos, tal como se puede verificar en la Figura 3.1.

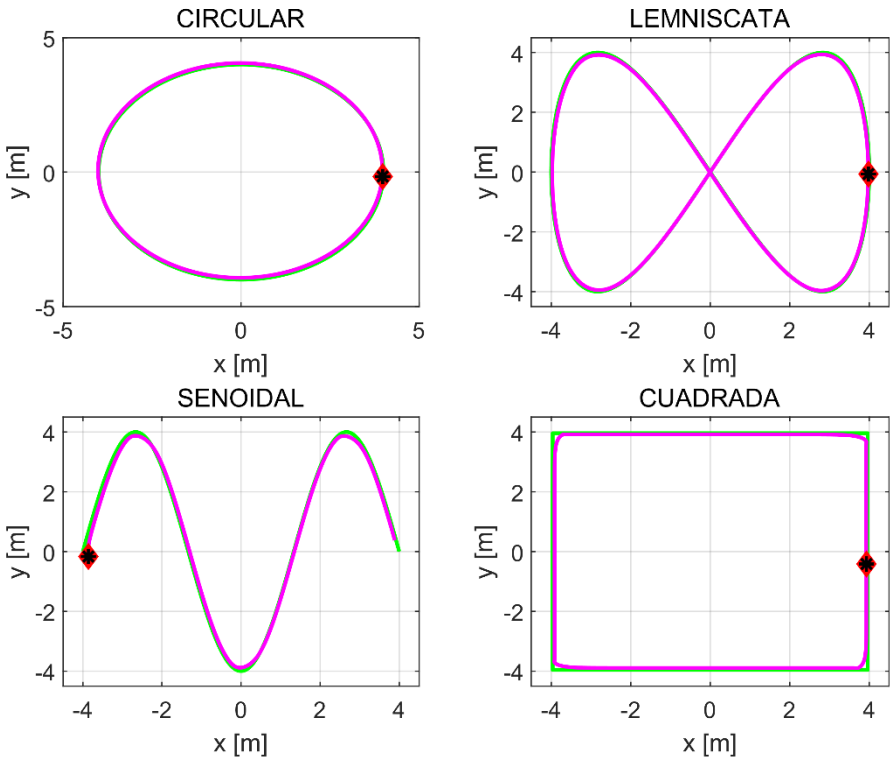


Figura 3.1. Pruebas de seguimiento de trayectoria a las 4 trayectorias

Estas cuatro trayectorias seleccionadas poseen características particulares que ponen a prueba la cinemática inversa, como por ejemplo en las zonas curvas y vértices que pueden ser tan cerradas que provocan variaciones instantáneas en la orientación del robot, que para en los 4 casos el seguimiento es satisfactorio con errores de posición mínimos, como se pueden apreciar en la Figura 3.2.

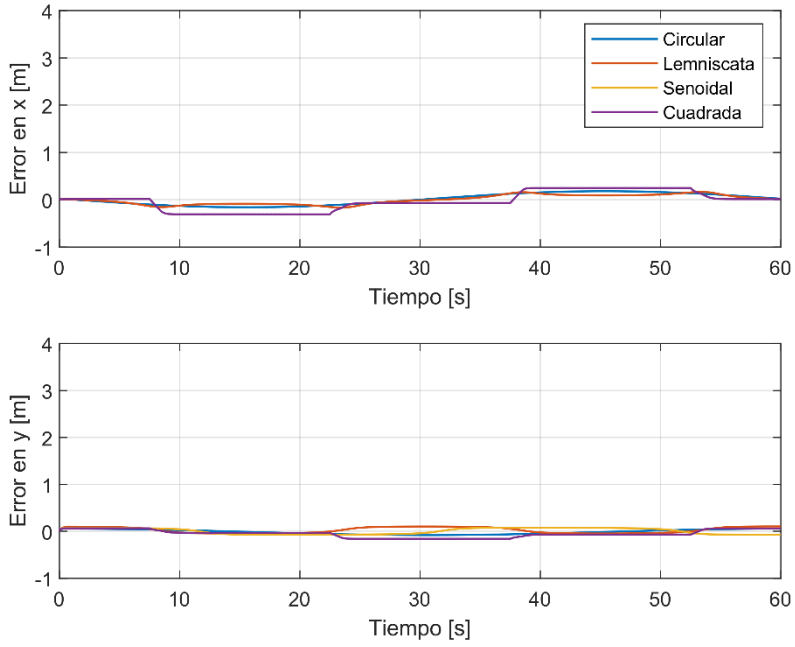


Figura 3.2. Errores XY de las cuatro trayectorias

Todas estas pruebas, como se mencionó al inicio de este capítulo, tienen el fin de comprobar el buen desempeño de seguimiento de trayectoria para un único robot Pioneer; sin embargo, el objetivo principal de esta aplicación, es ponerlo a prueba dentro de una formación multi-agente, cuyos resultados se describen a continuación.

3.1.2. PRUEBAS REALIZADAS AL SISTEMA MULTI-AGENTE EN FORMACIÓN LIDER-SEGUIDOR

Para estas pruebas se proponen 2 sistemas multi-agentes, que constan de dos y tres agentes robóticos Pioneers. Los cuales estarán a prueba para las cuatro trayectorias, descritas en el capítulo anterior, en un tiempo de simulación de 60 segundos.

Las características de los 2 sistemas se describen a continuación:

Sistema 1: cuenta con 2 agentes robóticos, que interactúan dentro de una superficie de contención formada por 3 agentes líderes virtuales, de manera que forman un triángulo.

Sistema 2: cuenta con 3 agentes robóticos, que interactúan dentro de una superficie de contención formada por 3 agentes líderes virtuales, de manera que forman un triángulo.

Antes del consenso, existe un periodo denominado de ubicación inicial de todos los agentes seguidores, en el cual deben ubicarse en la posición y orientación inicial, ya que, dependiendo de la trayectoria seleccionada su posición de partida es distinta, luego de este periodo descrito se inicia con el consenso.

De la simulación se obtienen las trayectorias recorridas en el plano XY, de todos los agentes seguidores; además las señales de control de las velocidades lineal y angular de la simulación. Todos estos resultados gráficos permiten determinar el correcto funcionamiento de la aplicación.

3.1.2.1. Pruebas realizadas al sistema multi-agente conformado por 3 líderes y 2 seguidores.

Prueba 1: Trayectoria circular

La trayectoria circular es una de las más simples y comunes al momento de probar el desempeño de controladores y algoritmos de seguimiento de trayectoria. Para esta simulación el radio de la trayectoria circular es de 4 m, como se observa en la Figura 3.3, junto con el recorrido de los 2 seguidores, donde se puede notar el punto de apartida de ambos agentes.

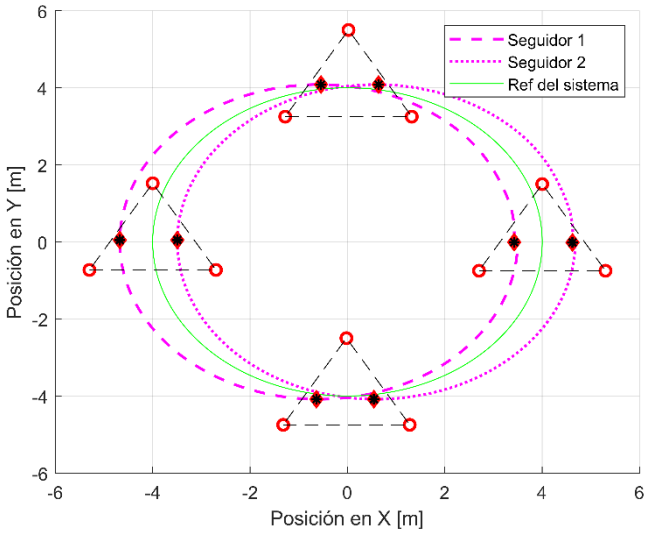


Figura 3.3. Seguimiento de trayectoria circular

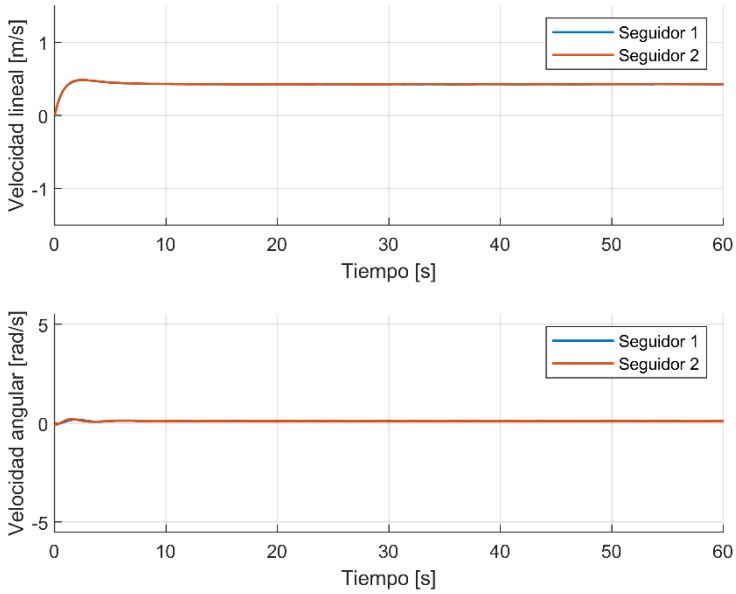


Figura 3.4. Acciones de control de los agentes seguidores

Además de la trayectoria recorrida es gran importancia revisar las señales de control, para este caso se muestran en Figura 3.4, donde se nota como al inicio existe un leve esfuerzo en las velocidades lineal y angular, por un tiempo de 3 segundos, luego de este tiempo ambas señales de control se estabilizan para mantenerse constantes a lo largo de la trayectoria.

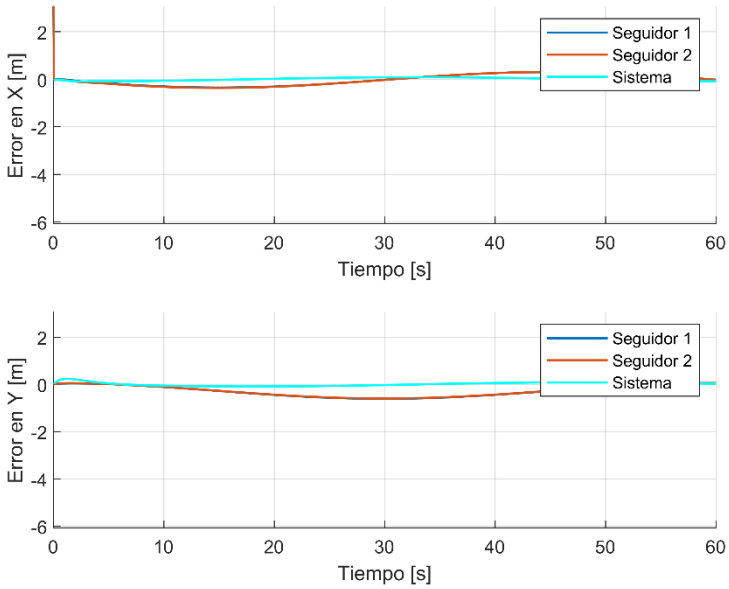


Figura 3.5. Errores de posición de los agentes seguidores para una trayectoria circular.

De la Figura 3.5, se puede notar como a lo largo de la trayectoria, el error en los dos ejes xy son bajos. Además, la curva de color celeste muestra gráficamente el error todo el sistema, permitiendo determinar el desempeño en conjunto de toda la formación.

Prueba 2: Trayectoria lemniscata

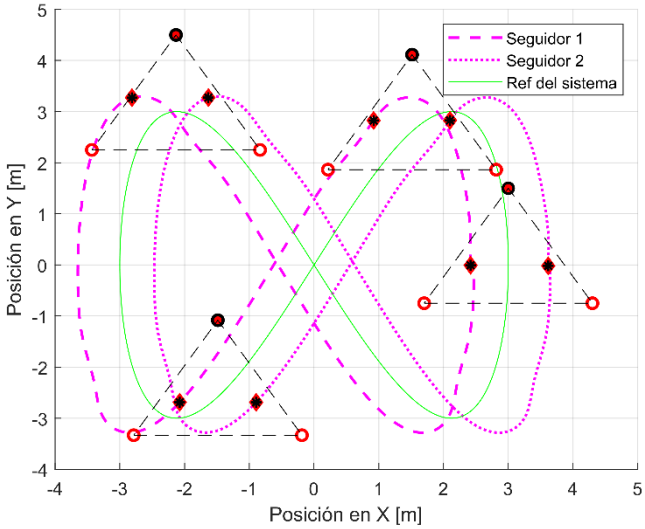


Figura 3.6. Seguimiento de trayectoria lemniscata

Esta trayectoria en forma de infinito debido a su forma característica tiene curvas cerradas en 4 puntos como se ve en la Figura 3.6. Dependiendo del tamaño de la trayectoria y las dimensiones del robot estas curvas van pueden ser muy cerradas, de manera que generarán un cambio progresivo en la velocidad lineal y angular.

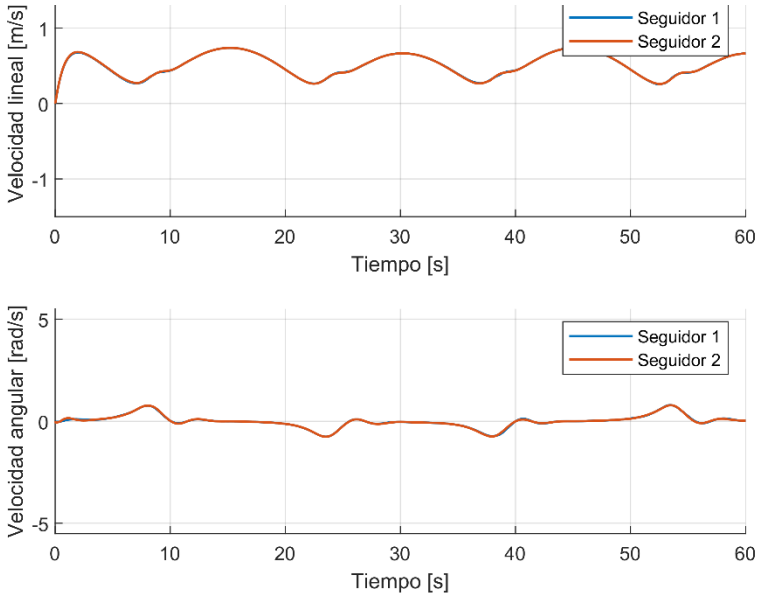


Figura 3.7. Acciones de control de los agentes seguidores

Como se menciona anteriormente estas 4 curvas generan acciones de control como respuesta, tratando de llevar al robot de nuevo a la trayectoria.

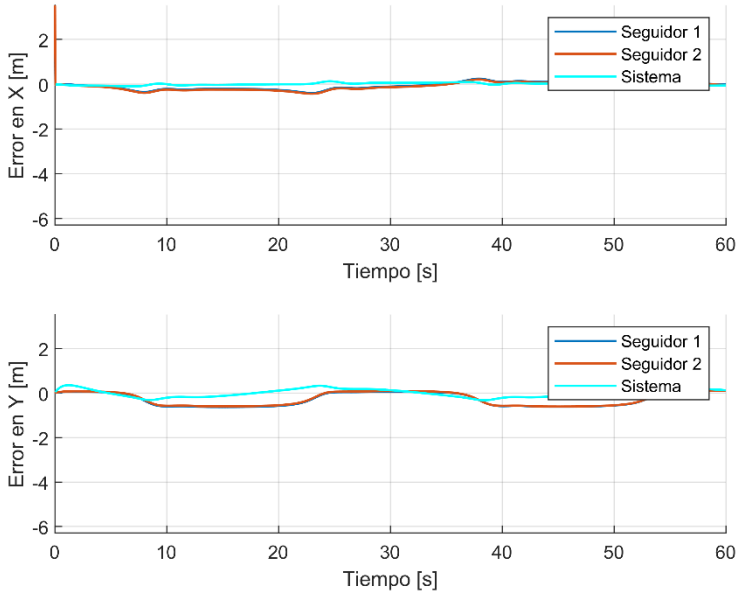


Figura 3.8. Errores de posición de los agentes seguidores para una trayectoria lemniscata

De la Figura 3.8, se aprecia que en esta trayectoria se exige mucho más que la trayectoria anterior, por tanto, el error de posición que existe es mayor, especialmente en el eje y que se encuentra dentro rango aproximado de -0,5 a 0,5 m. Esto debido a que para esta trayectoria el eje y tiene 2 veces la frecuencia de oscilación que el eje x, por tanto, tiene una mayor variación; esto se puede verificar en la Tabla 2.1.

Prueba 3: Trayectoria senoidal

Esta trayectoria a diferencia de las otras, el punto final es diferente al punto de partida, además, se tienen curvas más pronunciadas y tramos semi-rectos, como se ve en Figura 3.9, lo cual genera nuevas condiciones de pruebas.

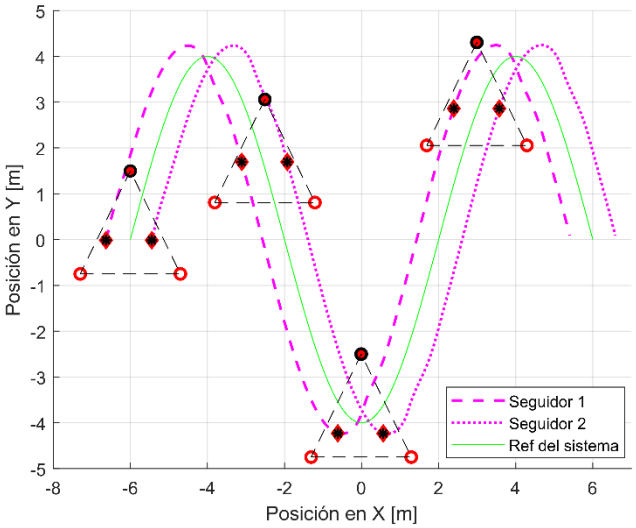


Figura 3.9. Seguimiento de trayectoria senoidal

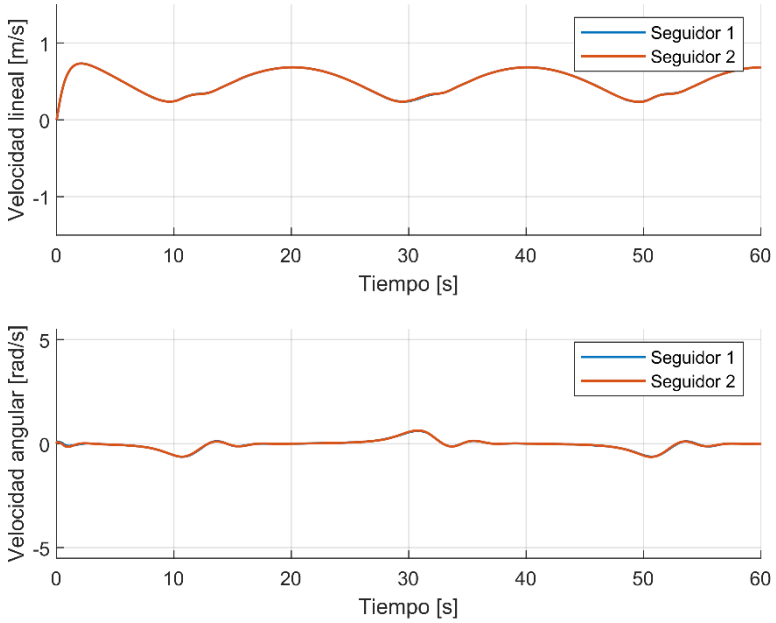


Figura 3.10. Acciones de control de los agentes seguidores trayectoria senoidal

En mayor medida existe un esfuerzo en la velocidad lineal variable en todo momento, cuya forma se parece a la trayectoria de seguimiento. Sin embargo, no sobrepasa los límites de las velocidades lineal y angular, por tanto, los saturadores no actúan.

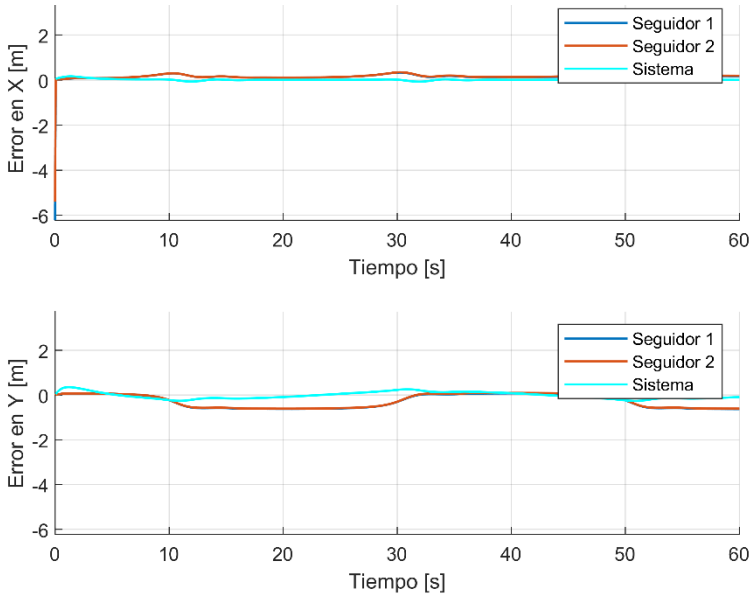


Figura 3.11. Errores de posición de los agentes seguidores para una trayectoria senoidal

De las señales de control que se ven en la Figura 3.10, se pude apreciar como a pesar de tener una forma similar a la anterior trayectoria, el error de todos robos es menor y en consecuencia del sistema. Para esta trayectoria se tiene un gran desempeño de seguimiento de toda la formación multi-agente.

Prueba 4: Trayectoria cuadrada

Esta trayectoria a comparación de las anteriores cuenta con tramos rectos de 4 m; además de un cambio repentino en la orientación de 90º en cada vértice, como se observa en la Figura 3.12. Esto provoca un esfuerzo instantáneo, cuyo tiempo de recuperación hasta regresar a su ubicación correspondiente, dura aproximadamente 5 segundos.

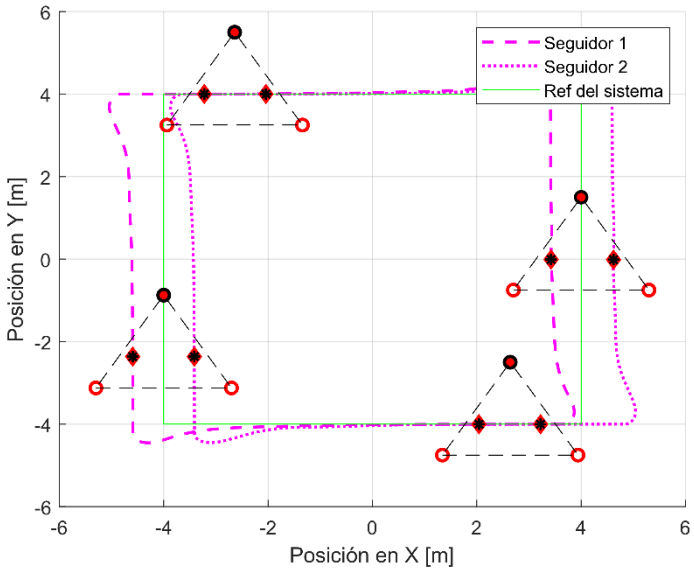


Figura 3.12. Seguimiento de trayectoria cuadrada

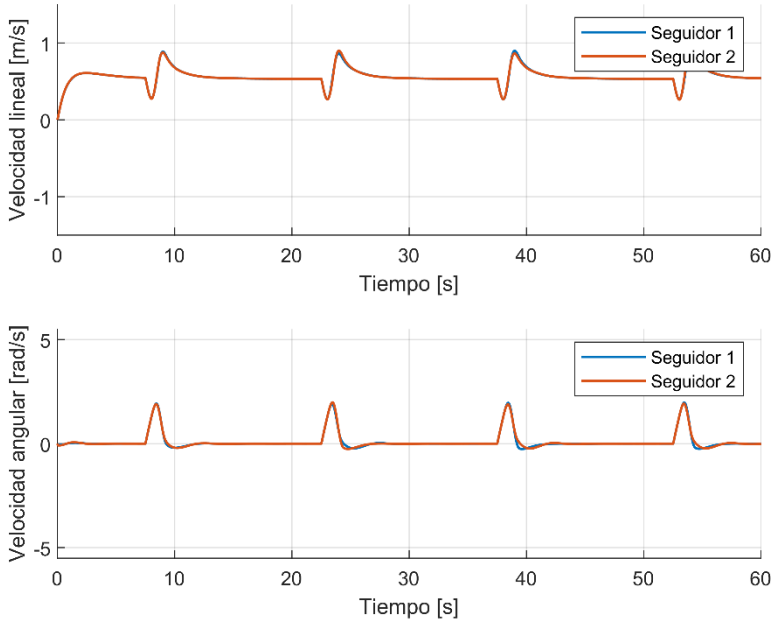


Figura 3.13. Acciones de control de los agentes seguidores

Como se describió anteriormente, las características de esta trayectoria exigen en mayor medida a la velocidad angular debido al cambio repentino en su orientación. Aun así, no es lo suficiente para alcanzar los límites máximos permitidos por el fabricante, pues en la Figura 3.13 no se ven limitaciones en ambas señales de control.

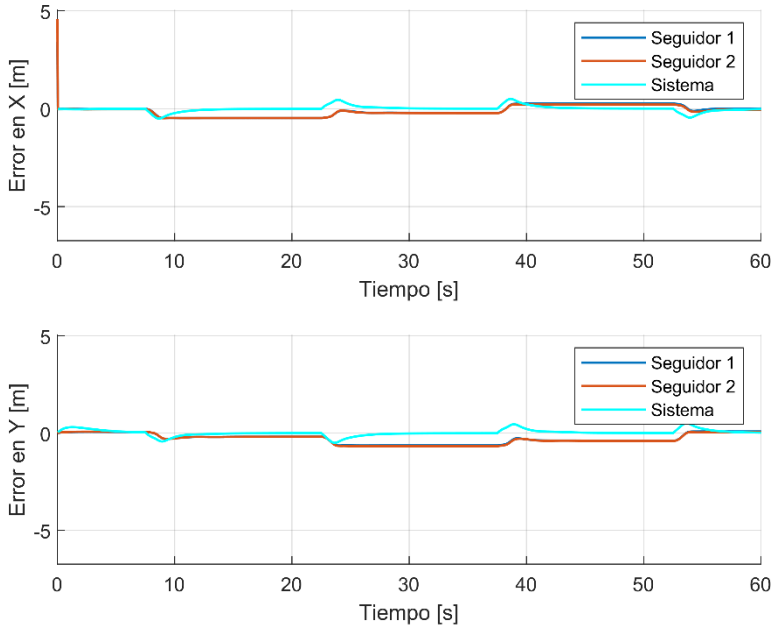


Figura 3.14. Errores de posición de los agentes seguidores para una trayectoria cuadrada

Analizando el error de todo el sistema identificado se mantiene muy bajo, incluso en los vértices de la trayectoria, es decir que el tiempo de respuesta es muy pequeño, por tanto,

todo el sistema tiene el desempeño deseado; no es exacto, pero si tiene una tendencia más apegada al mundo real, debido a las características de simulación de CoppeliaSim.

3.1.2.2. Pruebas realizadas al sistema multi-agente conformado por 3 líderes y 3 seguidores.

A diferencia del sistema anterior, en este caso al contar con 3 agentes robóticos, mediante la asignación de las ponderaciones en el grafo dirigido, se puede formar un triángulo con los 3 Pioneers, que tratarán de mantenerse de la misma manera por toda la trayectoria.

Prueba 1: Trayectoria circular

Adicional al seguimiento de trayectoria para esta prueba se agregará pequeñas perturbaciones en las formaciones, manipulando la posición de cada uno de los agentes robóticos durante el seguimiento de trayectoria de la formación multi-agente.

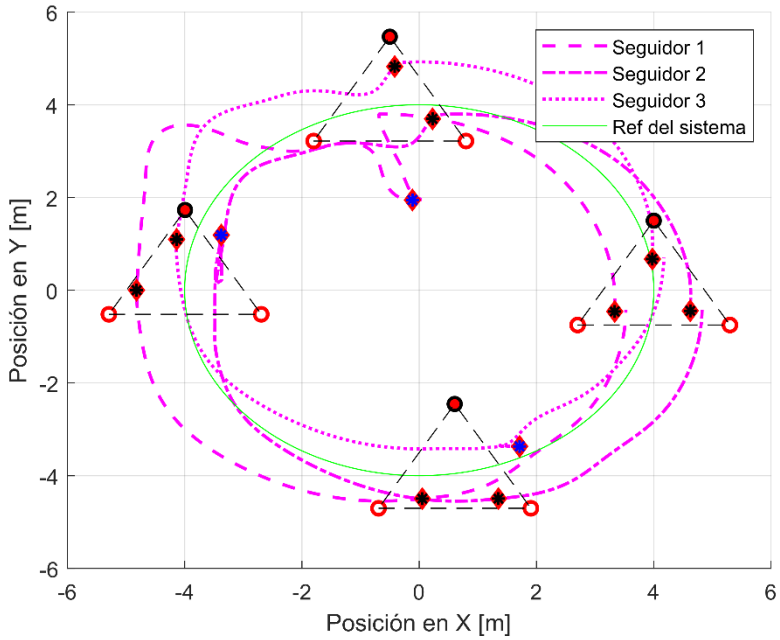


Figura 3.15. Seguimiento de trayectoria circular

Las perturbaciones mencionadas, se pueden realizar aprovechando las características del simulador robótico, el cual permite mover y girar los objetos durante la simulación, lo cual se acerca al mundo real en donde los robots pueden ser retenidos de manera manual por una persona. A continuación, se detalla los movimientos realizados a los robots móviles dentro de CoppeliaSim, durante la simulación.

Movimiento 1: A los 16.20 seg del tiempo de simulación, el agente seguidor 1, es separado 2 m de la formación.

Movimiento 2: A los 29.45 seg del tiempo de simulación, el agente seguidor 2, es arrastrado con un movimiento hacia atrás de 1 m de la formación.

Movimiento 3: A los 46.95 seg del tiempo de simulación, el agente seguidor 3, es arrastrado con un movimiento hacia delante de 1 m de la formación.

Todos estos movimientos se pueden verificar en la Figura 3.15, en donde se muestran con una marca con azul, el agente manipulado. Producto de estos movimientos, se observa que al manipular cualquier robot el resto de los robots también sufren un cambio inusual en sus movimientos, como se ve en la Figura 3.16, esto debido al consenso, pues la posición es la variable de interés la cual pasa procesamiento algoritmo, del cual se obtiene la posición de todos los agentes involucrados.

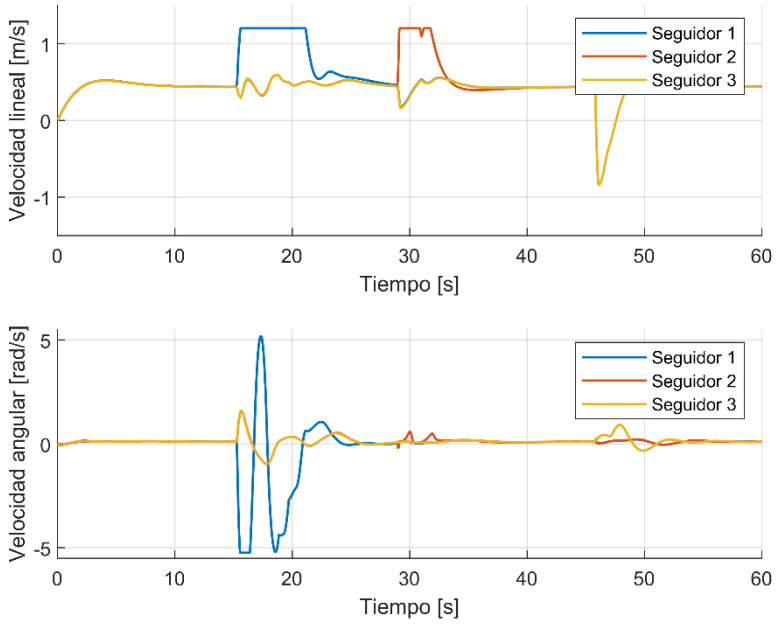


Figura 3.16. Acciones de control de los agentes seguidores

De las acciones de control de la Figura 3.16, se identifica comportamientos inusuales producto de los movimientos provocados, en donde los saturadores se ven obligados actuar pues las señales de control sobrepasan de ±1.2 m/s y ±5.24 rad/s para las velocidades lineal y angular respectivamente, especificados por el fabricante del modelo real en [19].

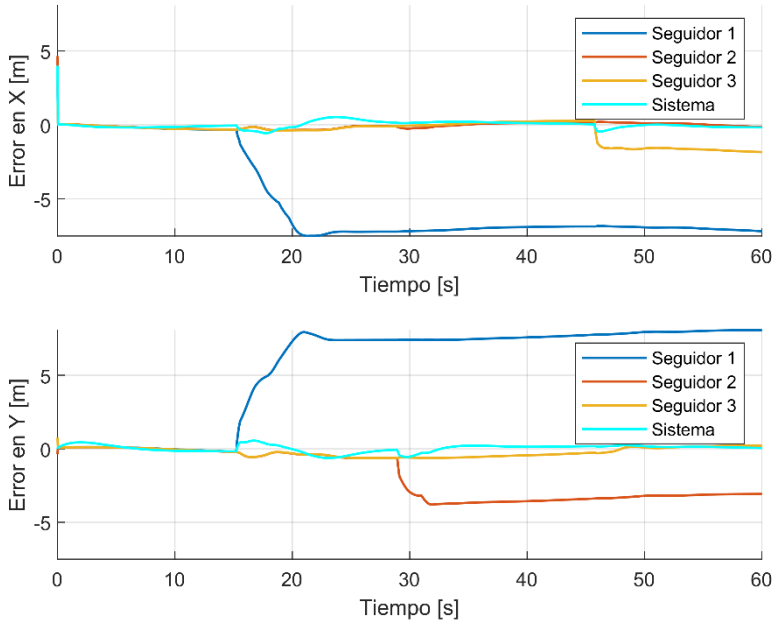


Figura 3.17. Errores de posición de los agentes seguidores para una trayectoria circular

Como se evidencia en la Figura 3.17, estos movimientos se ven reflejados en los errores de posición, de donde se nota que el movimiento del agente seguidor 1, tuvo una mayor influencia, lo cual también concuerda con la gráfica de las señales de control. Sin embargo, luego de retornar a su ubicación correspondientes en la formación, el comportamiento sigue siendo de las mismas características.

Adicionalmente, se ha buscado características de simulación que impliquen un comportamiento inusual en la simulación. De donde se agregó un ángulo de trayectoria a la formación mientras de mueve por la trayectoria, esto añadiendo $\phi = dx/dy$ a la ecuación (2.3), lo cual implica mayor esfuerzo de movimiento en su orientación.

Las 4 trayectorias propuestas en este trabajo siguen con un comportamiento normal, los cuales se muestran en el ANEXO I. Además, se pone a prueba con una nueva trayectoria en forma de un corazón de la cual se describe la parametrización en x y y, en las ecuaciones (3.1)(3.2), respectivamente.

$$x(t) = Ksin^3(\omega_f t) \tag{3.1}$$

$$y(t) = K\left[\frac{13}{19}cos(\omega_f t) - \frac{5}{16}cos(2\omega_f t) - \frac{1}{8}cos(3\omega_f t) - \frac{1}{16}cos(4\omega_f t) + \frac{1}{2}\right] \tag{3.2}$$

donde, K, es la amplitud de la trayectoria, y $\omega_f = \frac{2\pi}{T}$; T, es el tiempo de simulación.

Esta nueva trayectoria generada, tiene un comportamiento normal para una simulación en la cual no se incluye ángulo de trayectoria como se observa en la Figura 3.18 (a); por otro

lado, incluyendo un ángulo de trayectoria ocurre un evento inesperado dado en Figura 3.18 (b).

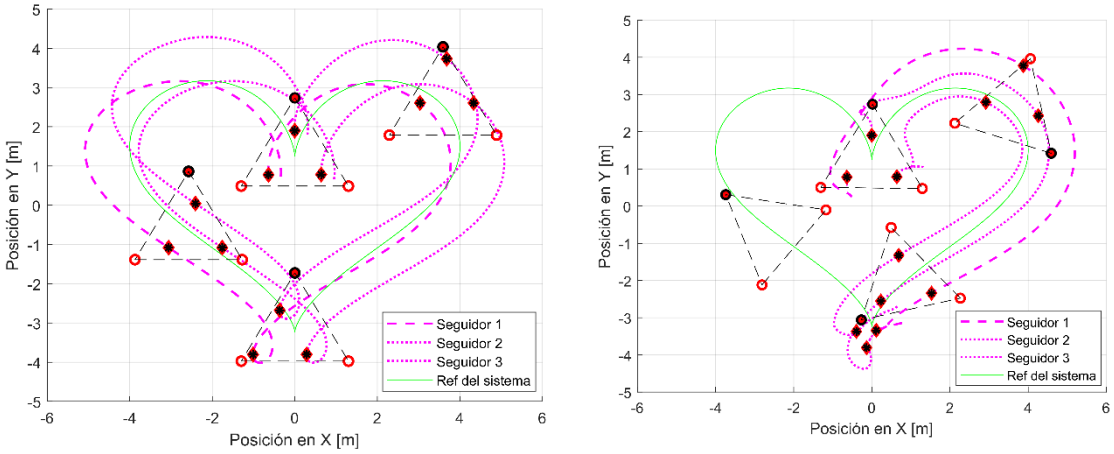


Figura 3.18. Seguimiento de trayectoria acorazonada; a) Sin ángulo de trayectoria, b) Con ángulo de trayectoria.

El sistema sufre una condición de colisión entre los 3 agentes robóticos, lo cual impide continuar el seguimiento de trayectoria. Esto debido a un cambio repentino en su orientación de unos 300º aproximados, lo cual provoca señales de control que llevan a pasar por el punto central de la formación, a los 3 robots al mismo instante; provocando una colisión que ninguno de los agentes puede evadir.

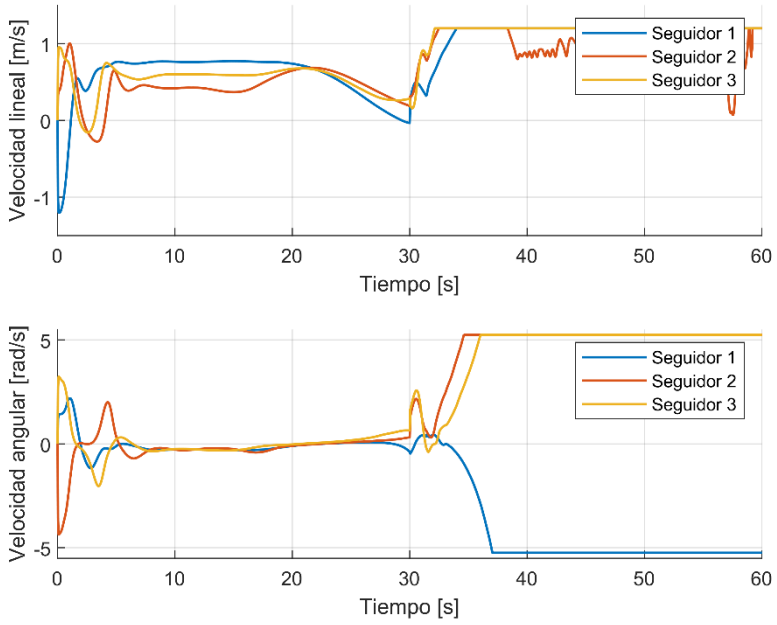


Figura 3.19. Señales de control trayectoria acorazonada con ángulo de trayectoria

Como se puede verificar en la Figura 3.19, el consenso se ve interrumpido debido a este obstáculo, por tanto, que el algoritmo seguirá generando las posiciones por consenso para cada uno de los agentes involucrados; sin embargo, debido al estancamiento de todos agentes seguidores las magnitudes de las señales de control se incrementan alcanzando sus límites de ±1.2 m/s y ±5.24 rad/s; debido a las mismas características físicas de las tres plataformas robóticas, se mantendrán en colisión mientras culmina el tiempo de simulación. Así mismo el error entre la posición actual y la posición esperada por consenso se incrementa dependiendo de la posición en la trayectoria de seguimiento como se ve en la Figura 3.20.

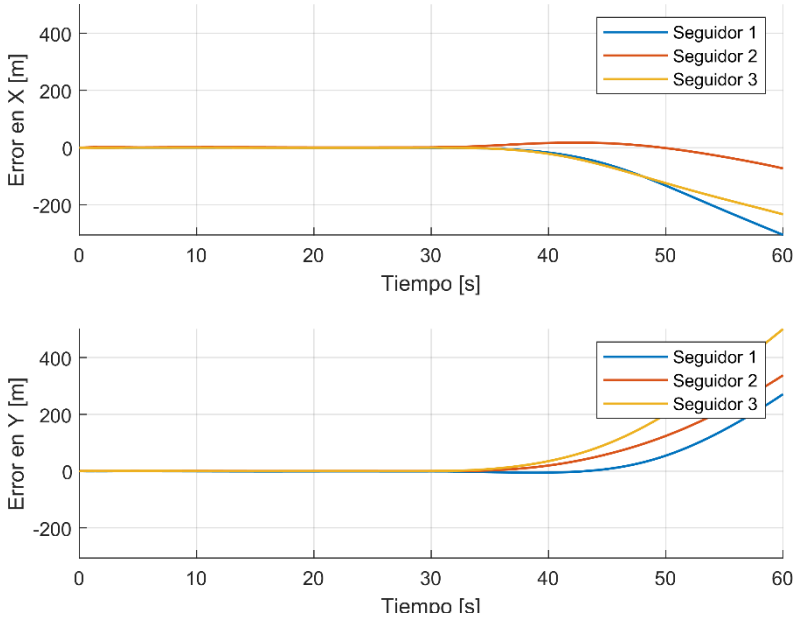


Figura 3.20. Errores de posición de los agentes seguidores trayectoria acorazonada con ángulo de trayectoria

Las acciones adicionales incluidas en estas pruebas son eventos fortuitos que en una implementación pueden afectar inevitablemente. Sin embargo, estos salen del alcance de este trabajo de titulación y pueden ser analizados en posteriores nuevos trabajos.

3.2. CONCLUSIONES

Existen muchos estudios y desarrollos investigativos sobre consenso de agentes, aun así, pocas aplicaciones e implementaciones de estos sistemas sobre simuladores robóticos como CoppeliaSim, donde se puede encontrar buenas prestaciones debido a su amplia biblioteca de herramientas disponibles y facilidad de diseño de objetos personalizados; además de la opción de importar objetos creados en otros softwares y la capacidad de comunicación con otras aplicaciones.

El desempeño de un sistema multi-agente depende en gran medida de su diseño y análisis, para esto existen muchas técnicas, una de ellas es la teoría de grafos revisada a lo largo de este trabajo, en el cual se presentan tres ejemplos de sistemas multi-agentes.

El algoritmo de consenso seleccionado para esta aplicación tiene un gran desempeño en el trabajo cooperativo designado a la formación multi-agente, manteniendo la formación en gran parte del seguimiento de trayectoria. En las zonas curvas, todo el sistema requiere un esfuerzo mayor de control, sin embargo, tanto el algoritmo y el controlador de movimiento trabajan eficientemente.

Mediante la interfaz gráfica diseñada, resulta sencillo la manipulación y el análisis del desempeño de todo el sistema multi-agente. Para esta aplicación, el operar junto a CoppeliaSim, es de gran importancia contar con esta interfaz, donde se puedan seleccionar características de la simulación y adicionalmente mostrar resultados de la simulación dependiendo de las características de simulación.

Se ha logrado comprobar el funcionamiento correcto de la aplicación implementada, a partir de la cual se presenta en este documento un análisis detallado del desempeño del trabajo cooperativo de la formación multi-agente.

3.3. RECOMENDACIONES

Debido a todo el tiempo que conllevo el desarrollo de esta aplicación se utilizó las versiones de CoppeliaSim disponibles, sin embargo, se recomienda utilizar las prestaciones de las últimas versiones de CoppeliaSim, las cuales traen mejoras considerables como, por ejemplo: scripts directos en lenguaje Python, procesamiento de matrices y vectores, entre otras características que mejorarán en gran medida el desarrollo de nuevas aplicaciones.

A la aplicación desarrollada se le podrían añadir características adicionales como: resiliencia, evasión de obstáculos, planeación de rutas, entre otras, que permitan disponer de un sistema multi-agente más robusto.

En futuras aplicaciones enfocados a esta área, se recomienda utilizar sistemas heterogéneos los cuales involucren otros modelos de robots disponibles en muchas plataformas de simulación. Además, podría comprobar físicamente el desarrollo de los sistemas multi-agente utilizando las características de MATLAB o CoppeliaSim de enlazarse con plataformas robóticas en un ambiente real.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] MobileRobots Inc, *Pioneer 3 Operations Manual*. MobileRobots, 2006.

[2] Xtec, “Programa Quantum Robòtics » Robòtica Industrial,” 2011. https://blocs.xtec.cat/quantumrobotics/category/robotica-industrial/ (accessed Jun. 05, 2022).

[3] Cando Cruz Patricio E., “Aplicación de un algoritmo de resiliencia para la identificación de agentes no cooperativos y la reestructuración de formaciones en sistemas multiagente heterogéneos,” Quito, 2020., 2020.

[4] Vizuete Haro Renato S., “Diseño y Simulación de Algoritmos de Control Distribuido para la Formación y Seguimiento de Trayectoria de Robots Móviles Tipo Uniciclo y para Manipuladores Móviles de 3 Grados de Libertad,” Quito, 2017., 2017.

[5] G. M. Andaluz Ortiz, “Modelación, Identificación y Control de Robots Móviles,” BIBDIGITAL, El Repositorio Digital Institucional de la Escuela Politécnica Nacional, QUITO, 2011.

[6] Á. Soriano Vigueras Tutores and Á. Valera Fernández Marina Vallés Miquel, “Control distribuido y coordinación de robots mediante sistemas multiagente.”

[7] M. Rodríguez Prieto, “Teoría espectral de grafos en la formación de redes : mínimo valor propio,” Universidad del Rosario, 2019.

[8] Y. Cao, D. Stuart, W. Ren, and Z. Meng, “Distributed containment control for multiple autonomous vehicles with double-integrator dynamics: Algorithms and experiments,” *IEEE Trans. Control Syst. Technol.*, vol. 19, no. 4, pp. 929–938, Jul. 2011, doi: 10.1109/TCST.2010.2053542.

[9] Y. Cao, D. Stuart, W. Ren, and Z. Meng, “Distributed containment control for double-integrator dynamics: Algorithms and experiments,” *Proc. 2010 Am. Control Conf. ACC 2010*, pp. 3830–3835, 2010, doi: 10.1109/ACC.2010.5531204.

[10] J. Velasco Seguido-Villegas, “Análisis y comparación de las principales plataformas de simulación robótica y su integración con ROS,” E.T.S.I. Industriales (UPM), 2019.

[11] L. Nogueira, “Comparative Analysis Between Gazebo and V-REP Robotic Simulators,” Accessed: Dec. 06, 2021. [Online]. Available: https://www.dca.fee.unicamp.br/~gudwin/courses/IA889/2014/IA889-02.pdf.

[12] “CoppeliaSim Features - Coppelia Robotics.” https://www.coppeliarobotics.com/features (accessed May 21, 2022).

[13] “MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink.” https://la.mathworks.com/products/matlab.html?s_tid=mlh_so_learn (accessed Apr. 03, 2022).

[14] “Programmatic Model Editing — Functions.” https://la.mathworks.com/help/simulink/referencelist.html?type=function&category=programmatic-modeling&s_tid=CRUX_topnav (accessed May 07, 2022).

[15] L. T. Vivar, “Desarrollo de app en Matlab para rehabilitación de espasticidad con ayuda del robot colaborativo KUKA LBR IIWA.” .

[16] Coppelia Robotics, “Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics.” https://www.coppeliarobotics.com/ (accessed Dec. 23, 2021).

[17] “Remote API functions (Matlab).” https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsMatlab.htm (accessed Jan. 17, 2022).

[18] L. Morales, M. Herrera, O. Camacho, P. Leica, and J. Aguilar, “LAMDA Control Approaches Applied to Trajectory Tracking for Mobile Robots,” *IEEE Access*, vol. PP, p. 1, Feb. 2021, doi: 10.1109/ACCESS.2021.3062202.

[19] “PIONEER 3-DX,” Accessed: Mar. 28, 2022. [Online]. Available: www.mobilerobots.com.

[20] W. Ren and Y. Cao, *Distributed coordination of multi-agent networks: Emergent problems, models, and issues*, no. 9780857291684. Springer International Publishing, 2011.

[21] R. Vizuete, J. A. Torres, and P. Leica, “Trajectory Tracking based on Containment Algorithm Applied to a Formation of Mobile Manipulators.,” in *ICINCO (1)*, 2017, pp. 122–131.

5. ANEXOS

El listado de los anexos del presente Trabajo de Integración Curricular es:

ANEXO I: PRUEBAS ADICIONALES

ANEXO II: INSTRUCCIONES PARA LA COMUNICACIÓN DE MATLAB y COPPELIASIM

ANEXO III: MANUAL DE USUARIO

ANEXO IV: ENLACES VIDEOS DEMOSTRATIVOS DE LA APLICACIÓN

ANEXO II

PRUEBAS ADICIONALES

En este anexo se muestran las pruebas adicionales realizadas al sistema multi-agente de 3 líderes virtuales más los agentes seguidores Pioneers 3-DX utilizando un ángulo de trayectoria el cual permite que toda la formación adquiera una orientación distinta a lo largo de trayectoria; sus resultados de funcionamiento se muestran a continuación.

FORMACIÓN MULTI-AGENTE DE: 3 líderes y 2 seguidores con ángulo de trayectoria

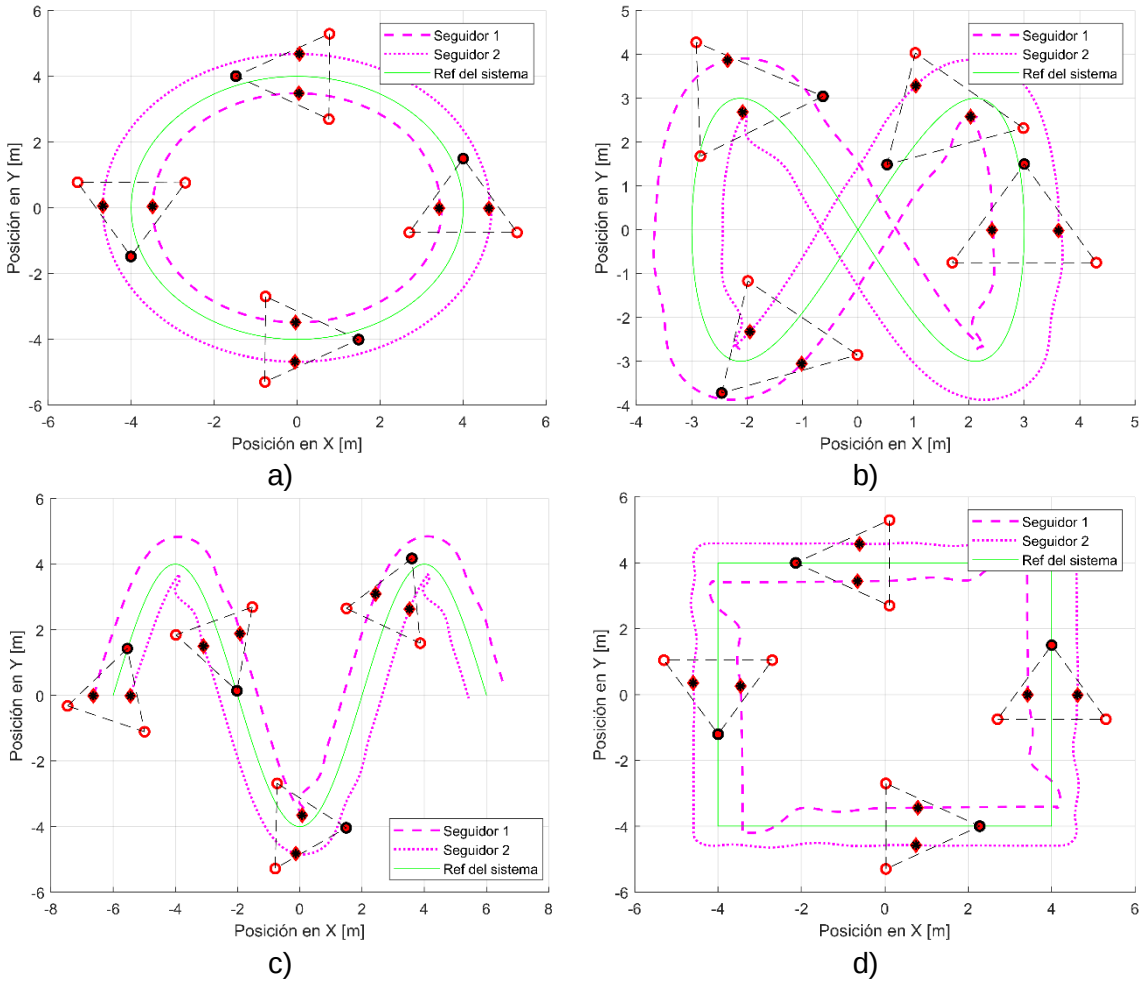


Figura I.1. Trayectorias recorridas para un sistema con 2 agentes seguidores, a) Circular, b) Lemniscata, c) Senoidal, d) Circular

Mediante esta prueba se nota como el agregar un ángulo de trayectoria generan una ruta de seguimiento diferente para cada agente seguidor, siempre manteniéndose dentro del área de interacción envuelta por los agentes líderes. Debido a tamaño de la envoltura de la formación en cada curvatura o vértice, la orientación de la formación cambia súbitamente provocando cambios rápidos en las posiciones de los agentes seguidores; sin embargo, por medio consenso, los robots móviles retornan a sus posiciones correspondientes de

manera autónoma para continuar con el seguimiento, de manera siempre existe una interacción entre agentes.

FORMACIÓN MULTI-AGENTE DE: 3 líderes y 2 seguidores con ángulo de trayectoria

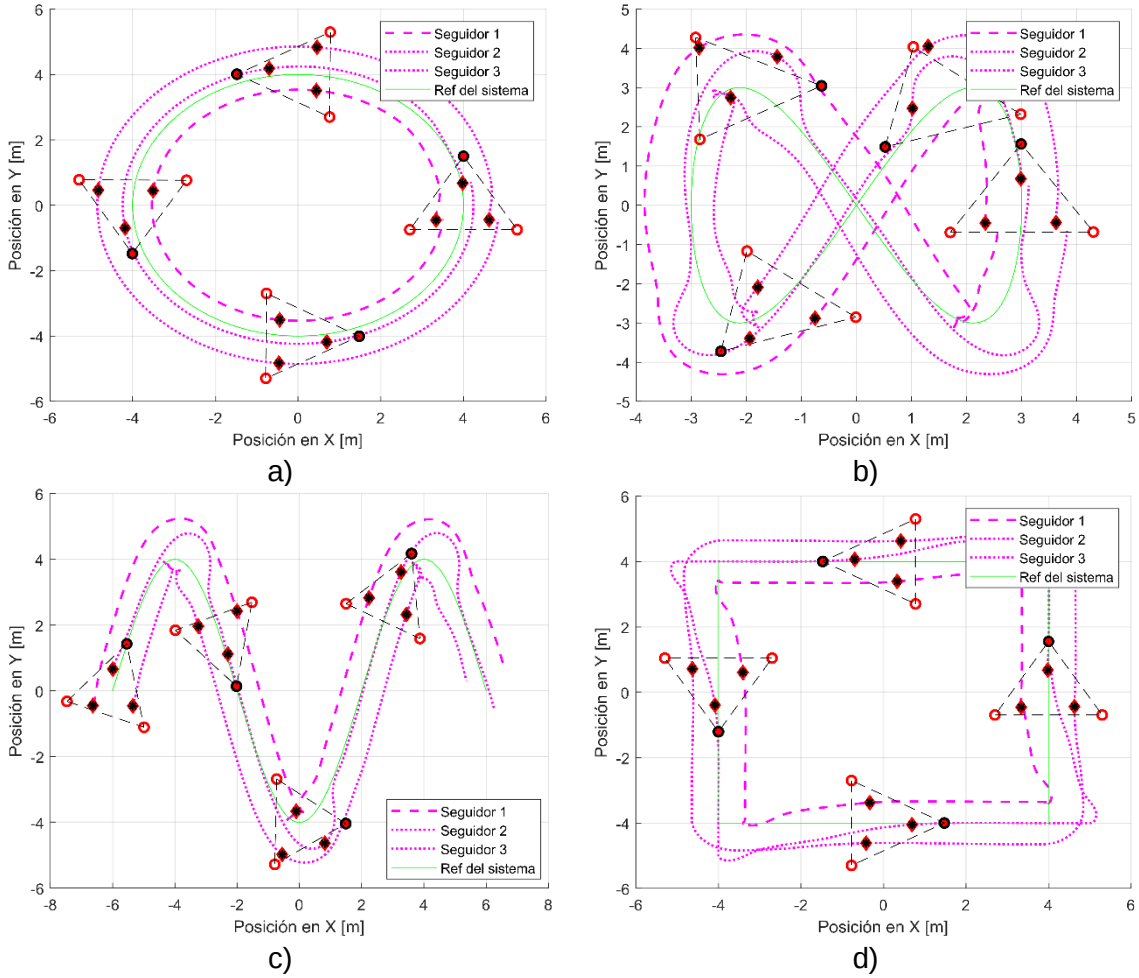


Figura I.2. Trayectorias recorridas para un sistema con 3 agentes seguidores, a) Circular, b) Lemniscata, c) Senoidal, d) Circular

De igual manera para este sistema con 3 agentes seguidores para las 4 trayectorias como se ve en la Figura I.2, el ángulo de trayectoria genera ruta distinta para cada uno de ellos. Sin embargo, mediante el consenso los agentes seguidores actúan de manera autónoma hasta regresar a sus posiciones correspondientes.

ANEXO III

INSTRUCCIONES PARA LA COMUNICACIÓN DE MATLAB y COPPELIASIM

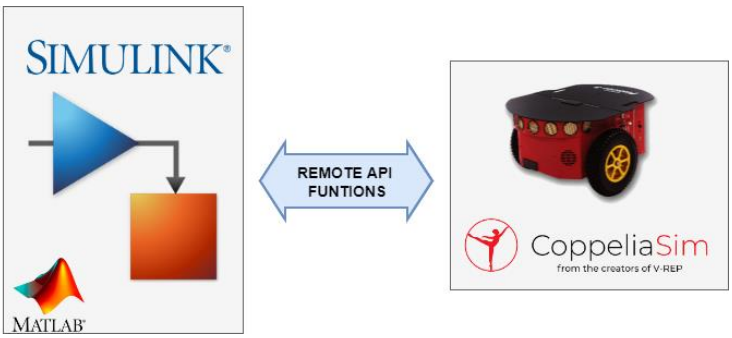


Figura II.1 CoppeliaSim y Matlab [Autoría propia]

Debido a la poca información clara y precisa encontrada respecto al procedimiento para establecer la comunicación entre MATLAB y CoppeliaSim utilizando Simulink, se ha creado este documento, donde se describe la implementación para la comunicación utilizada en este trabajo. A lo largo de este documento se podrán encontrar una serie de recomendaciones para comunicar las dos plataformas.

En primer lugar, se debe verificar la existencia de todos los archivos dentro del mismo directorio de trabajo, de MATLAB:

ARCHIVOS DE COPPELIASIM:

- remApi.m
- remoteApi.dll
- remoteApiProto.m

Los cuales se obtienen de manera automática del proceso de instalación de CoppeliaSim. Ubicados en las siguientes ubicaciones:

C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\matlab\matlab

C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\lib\lib\Windows

- Escena_CoppeliaSim.ttt

Este archivo contiene la escena de simulación.

ARCHIVOS DE MATLAB:

- archivo_simulink.slx
- scritp_MATLAB_CoppeliaSim.m

Una vez verificados todos los requisitos, a continuación, se describe la metodología:

La implementación para la comunicación entre ambos softwares se resume gráficamente mediante la Figura II.2, de donde se aprecia la existencia de un script.m, implementado en en MATLAB en un archivo (.m), el cual contiene las características de la simulación de Simulink según la estructura del bloque S-Function de Simulink. El bloque *2-level MATLAB S-Funtion* de MATLAB-Simulink cumple la función de interface de comunicación entre ambos softwares; y el escenario de CoppeliaSim, el cual se enlaza mediante una serie de instrucciones creadas por CoppeliaSim para interactuar directamente con cada uno de los objetos del escenario de simulación las cuales se conocen como Remote API Functions.

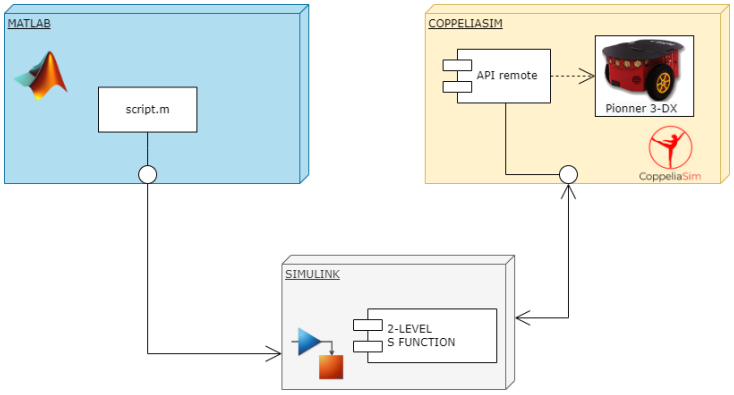


Figura II.2. Resumen gráfico de la comunicación MATLAB - CoppeliaSim

Para la comprensión de esta metodología se debe entender la función del bloque de Simulink denominado 2-level MATLAB S-Funtion el cual se resume a continuación.

Esta función es parte de las amplias librerías de Simulink, la cual permite definir algunas características de la simulación, como, por ejemplo: inicialización, datos de salida, actualización de datos, finalización y particularmente describir características antes de iniciar la simulación en el cual se pueden agregar parámetros iniciales. Todas estas condiciones deben estar dentro de un archivo script.m de acorde a la plantilla disponible en MATLAB, el cual podrá obtener ingresando en el Command Window `edit([matlabroot,'/toolbox/simulink/blocks/msfuntmpl_basic.m'])`, cuya

estructura se describe de mejor manera con la ayuda de un ejemplo, el cual se describe a continuación:

Este ejemplo consiste en controlar el movimiento de un robot móvil Pioneer 3-DX, disponible en CoppeliaSim; para esto se debe personalizar el script.m con las siguientes características:

a. En este primer segmento del script, se coloca el nombre de la función principal, que debe coincidir con el nombre del archivo.m, para este caso el nombre seleccionado es script_MATLAB_CoppeliaSim. Entonces el nombre del archivo seria *script_MATLAB_CoppeliaSim.m*

```
function script_MATLAB_CoppeliaSim(block)
setup(block);
function setup(block)
```

b. En el siguiente segmento del script, se coloca el número de entradas y salidas necesarias, ya pueden ser señales de control y por otro lado la retroalimentación del estado del robot móvil, para esto se requieren de 2 entradas y 1 salida. Luego, se personalizan cada una de las entradas y salidas requeridas para él envió y recepción de datos. Para este caso las dos entradas son las señales de control de la velocidad lineal y angular, u y ω, respectivamente. La única salida consiste un array de datos de dimensión 3 con la posición (x,y) y orientación φ, del robot.

```
% Registro de número de entradas y salidas
block.NumInputPorts  = 2;   %2 entradas
block.NumOutputPorts = 1;   %1 salidas
% Setup port properties to be inherited or dynamic (sin cambios)
block.SetPreCompInpPortInfoToDynamic;
block.SetPreCompOutPortInfoToDynamic;

% Propiedades de la entrada 1 (u)
block.InputPort(1).Dimensions   = 1;
block.InputPort(1).DatatypeID   = 0; % double
block.InputPort(1).Complexity   = 'Real';
block.InputPort(1).DirectFeedthrough = false;

% Propiedades de la entrada 1 (w)
block.InputPort(2).Dimensions   = 1;
block.InputPort(2).DatatypeID   = 0; % double
block.InputPort(2).Complexity   = 'Real';
block.InputPort(2).DirectFeedthrough = false;


% Propiedades de la salida 1 (Pioneer Status)
block.OutputPort(1).Dimensions       = 3;
block.OutputPort(1).DatatypeID  = 0; % double
block.OutputPort(1).Complexity  = 'Real';
```

```
% Register parameters
block.NumDialogPrms     = 0;

% Register sample times
%  [0 offset]            : Continuous sample time
%  [positive_num offset] : Discrete sample time
%
%  [-1, 0]               : Inherited sample time
%  [-2, 0]               : Variable sample time
block.SampleTimes = [-1 0];
% Specify the block simStateCompliance. The allowed values are:
%    'UnknownSimState', < The default setting; warn and assume DefaultSimState
%    'DefaultSimState', < Same sim state as a built-in block
%    'HasNoSimState',   < No sim state
%    'CustomSimState',  < Has GetSimState and SetSimState methods
%    'DisallowSimState' < Error out when saving or restoring the model sim
state
block.SimStateCompliance = 'DefaultSimState';

%% -----------------------------------------------------------------
%% registered methods within this s function:
%% -----------------------------------------------------------------

block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
block.RegBlockMethod('Start', @Start);
block.RegBlockMethod('Outputs', @Outputs);     % Required
block.RegBlockMethod('Update', @Update);
block.RegBlockMethod('Terminate', @Terminate); % Required
%end setup

%% SetInpPortFrameData
%% Importante para que funcione
function SetInpPortFrameData(block, idx, fd)
  block.InputPort(idx).SamplingMode = fd;
  block.OutputPort(1).SamplingMode = 'Sample';
%% PostPropagationSetup (deleted)
%% InitializeConditions (deleted)
```

c. En este segmento del script denominado “Start:”, se detallan los estados iniciales de la simulación como ejemplo: la inicialización de la comunicación entre ambas plataformas para lo cual se requiere las Remote API Functions disponibles en el siguiente enlace; además, la inicialización de variables y constantes. También, se declaran los conocidos *handle’s* o manejadores, que permitirán tener un control directo con cada uno de los objetos dentro de CoppeliaSim; estos manejadores se deben guardar en variables globales que permitan ser utilizadas en otras secciones. Para este caso los manejadores necesarios serán: el robot Pioneer 3-DX y las ruedas izquierdas & derecha.

```
%% Start:
function Start(block)

    global sim clientID REF left_motor right_motor Pioneer
    sim =remApi('remoteApi');
    sim.simxFinish(-1);
```

```
     clientID=sim.simxStart('127.0.0.1',19997,true,true,5000,5);
     if(clientID > -1)
     %Aviso de conexión establecida
     sim.simxAddStatusbarMessage(clientID,'Start communication con MATLAB-
Simulink',sim.simx_opmode_blocking);
     %activar modo síncrono
     sim.simxSynchronous(clientID,true);
     %Run simulación en coppelia
     sim.simxStartSimulation(clientID,sim.simx_opmode_blocking);
     end

%Declaración handles:

% Robot:
[returnCode, Pioneer]=sim.simxGetObjectHandle(clientID, 'Pioneer_p3dx',
sim.simx_opmode_blocking);

% Objetos del robot: Motores derecho y izquierdo;
[returnCode,left_motor]=sim.simxGetObjectHandle(clientID,'Pioneer_p3dx_leftMoto
r',sim.simx_opmode_blocking);
[returnCode,right_motor]=sim.simxGetObjectHandle(clientID,'Pioneer_p3dx_rightMo
tor',sim.simx_opmode_blocking);

%end Start
```

d. En este segmento del script denominado “Outputs:”, se definen las características de las salidas, es decir los datos obtenidos desde CoppeliaSim. El gran número de funciones API permitirán la obtención de datos requeridos de posición y orientación de manera directa, para esto se requerirá de los manejadores de la anterior sección.

```
%% Outputs:
%% salidas x, y, phi
function Outputs(block)
global sim clientID Pioneer
%Comando para obtener la posición (absoluta)
[returnCode,position]=sim.simxGetObjectPosition(clientID, Pioneer, -1,
sim.simx_opmode_blocking);
[returnCode,orientation]=sim.simxGetObjectOrientation(clientID, Pioneer, -1,
sim.simx_opmode_blocking);
[returnCode,lineal,angular]=sim.simxGetObjectVelocity(clientID, Pioneer,
sim.simx_opmode_blocking);%% Obtener velocidad
x = double(position(1));
y = double(position(2));
Phi = double(orientation(3));

%Envio de datos
Pioneer_status=[x y Phi];
block.OutputPort(1).Data = Pioneer_status;

%end Outputs
```

e. En este segmento del script denominado “Update”, se actualizan los estados dentro de la plataforma de simulación como, por ejemplo, las señales de control del robot. Para

este caso se reciben las entradas u_c y ω_c, y se actualizan las acciones de control en las ruedas en base a las ecuaciones descritas en el siguiente segmento del programa.

```
%% Update:
%% Ejecutar cada vez: envio de u y w
function Update(block)
global sim clientID REF left_motor right_motor

%cálculo de velocidades
R = 0.195/2; % Radio de las ruedas del motor
L = 0.331; % diantancia entre los centros de ambas ruedas
uc = block.InputPort(1).Data;
wc = block.InputPort(2).Data;

% Calculo de velocidad de cada rueda
wr = (2*uc + wc*L)/(2*R);
wl = (2*uc - wc*L)/(2*R);

%Enviar todo a CoppeliaSim
[returnCode]=sim.simxSetJointTargetVelocity(clientID,right_motor, wr,
sim.simx_opmode_oneshot);   %Rueda derecha
[returnCode]=sim.simxSetJointTargetVelocity(clientID,left_motor, wl,
sim.simx_opmode_oneshot);    %Rueda izquierda

%TRIGGER
sim.simxSynchronousTrigger(clientID);
%end Update
```

f. En este segmento del script denominado “Terminate”, se definen algunas características finales de la simulación, para este caso se apagan los motores del robot y finalizar la comunicación con CoppeliaSim.

```
%% Terminate:
%% Desconexión CoppeliaSim
function Terminate(block)
global sim clientID left_motor right_motor
%Detener
[returnCode]=sim.simxSetJointTargetVelocity(clientID,right_motor, 0,
sim.simx_opmode_oneshot);
[returnCode]=sim.simxSetJointTargetVelocity(clientID,left_motor, 0,
sim.simx_opmode_oneshot);

%Terminar comunicación
sim.simxAddStatusbarMessage(clientID, 'Comunicación con MATLAB-Simulink
finalizada', sim.simx_opmode_blocking);
sim.simxStopSimulation(clientID,sim.simx_opmode_blocking);
sim.simxFinish(-1);
sim.delete();
%end Terminate
```

Con el script creado según las descripciones mencionadas, se debe se asignar al bloque level-2 MATLAB S-Function como en la Figura II.3; posteriormente se deben aplicar los cambios.

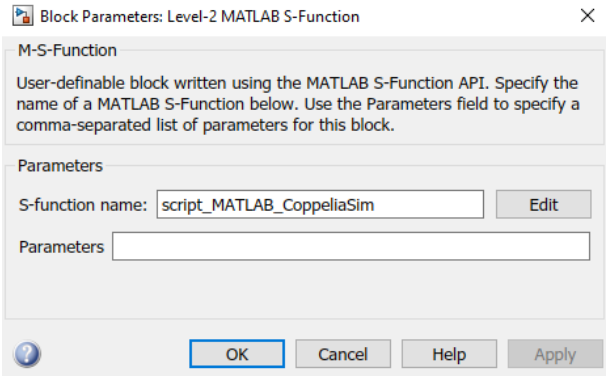


Figura II.3. Propiedades del bloque level-2 MATLAB S-Function

Una vez aplicado los cambios en las propiedades del bloque, automáticamente el número de entradas y salidas se reflejarán en el bloque como se ve en Figura III.4. Lo siguiente es asignar las entradas: señal de control velocidad lineal (u_c), Señal de control velocidad angular (ω_c); y salidas: vector estado robot (x, y, φ),

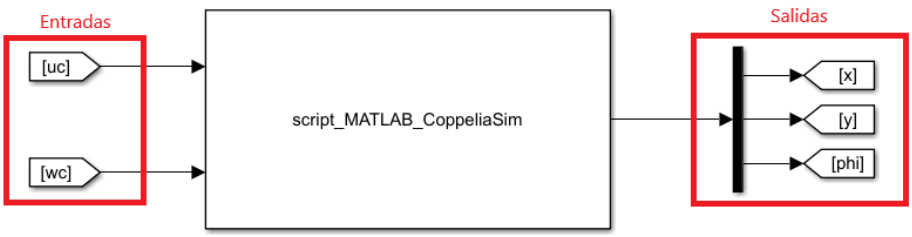


Figura II.4. Bloque Level-2 MATLAB S-Function, asignado el script.m

Si todo ha realizado según las instrucciones indicadas, y corre el archivo Simulink, el resultado debe ser similar al indicado en la Figura III.5., mismo que dependerá del controlador del robot.

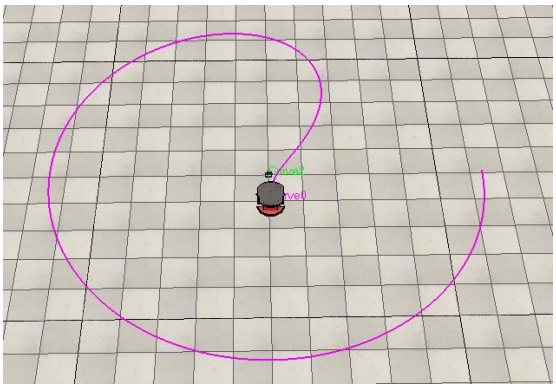


Figura II.5. Resultado ejemplo de comunicación CoppeliaSim y MATLAB-Simulink

ANEXO III

MANUAL DE USUARIO

Generalidades

La aplicación ha sido desarrollada en conjunto de los softwares CoppeliaSim; MATLAB y sus herramientas. Para manejar la aplicación se ha creado una interfaz gráfica, en el cual el usuario dispondrá de 3 pantallas: 1ra, Presentación; 2da, Características de la simulación; y 3ra, Resultados. La interfaz cuenta con opciones de selección del tipo de sistema, trayectoria a seguir, ángulo de trayectoria y dimensión de la trayectoria. La ventana de los resultados consiste en graficas que muestran el desempeño de los agentes robóticos durante la simulación donde se podrá elegir entre mostrar los datos de la simulación de los seguidores robóticos individuales o todos.

Para la simulación de la aplicación, es necesario contar con los siguientes requisitos:

REQUERIMIENTOS

Requerimientos para la comunicación

- remApi.m
- remoteApi.dll
- remoteApiProto.m

MATLAB

- Consensus_Algorithm_3L1S.slx
- Consensus_Algorithm_3L2S.slx
- Consensus_Algorithm_3L2S.slx
- Coppelia_3L1S.m
- Coppelia_3L2S.m
- Coppelia_3L2S.m
- f_sgm.m
- f_inicial_formation.m
- f_trayectoria.m
- Interfaz_Portada.mlapp
- Interfaz_Ventana_1.mlapp
- Interfaz_Ventana_2.mlapp

CoppeliaSim

- Escenario_Coppelia.ttt

Imágenes

- Logo_EPN.png
- Marco.png
- CoppeliaSim.png
- Graphic.png
- Home.png
- Run.png
- Stop.png

PDF

- Manual_Usuario.pdf

Una vez verificados todos los requisitos necesarios para la correcta ejecución de esta aplicación, se detalla los pasos necesarios para la simulación.

Figura III.1. Ejecutable CoppeliaSim

Antes de comenzar a utilizar el interfaz gráfico, el usuario deberá ejecutar el programa de simulación que se muestra en la Figura III.1. Una vez dentro del entorno de CoppeliaSim, el usuario deberá abrir el escenario de simulación, siguiendo los pasos que se muestran en las Figura III.2. y Figura III.3.

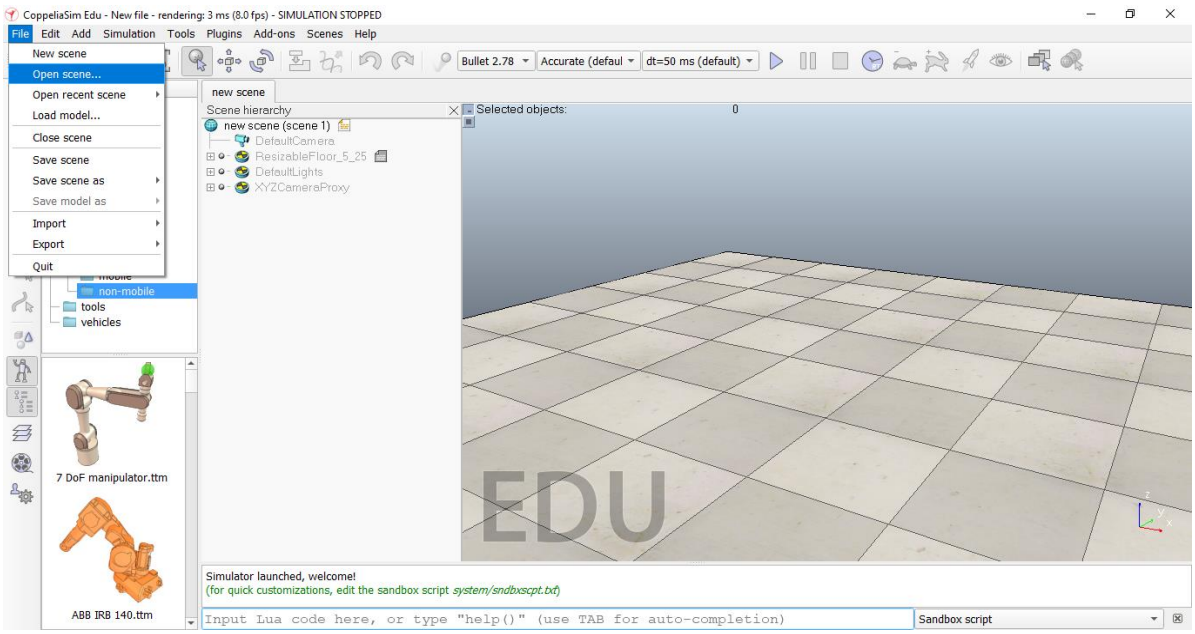


Figura III.2. Ambiente de CoppeliaSim

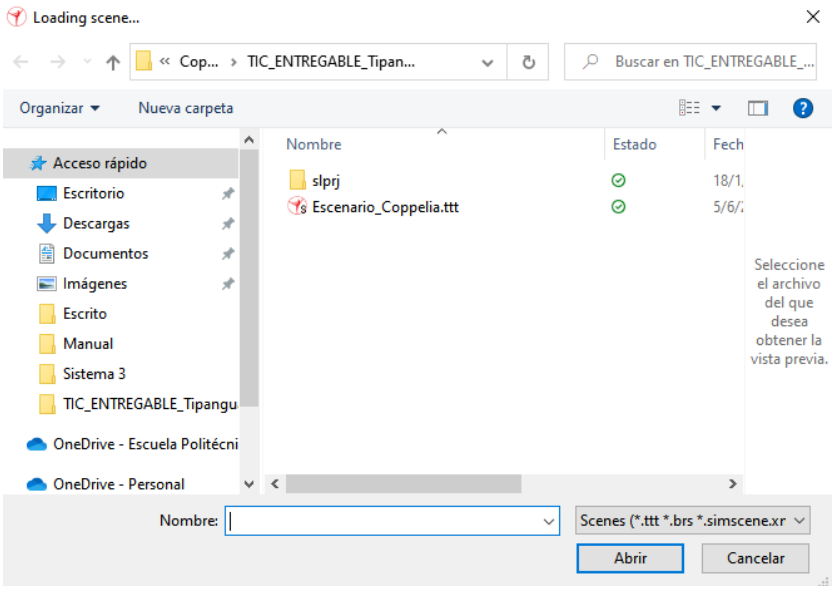


Figura III.3. Selección archivo.ttt

Una vez seleccionado el escenario de simulación, el usuario podrá observar un ambiente que se observa en la Figura III.4. A partir de este paso el usuario podrá utilizar el interfaz gráfico para comandar esta aplicación.

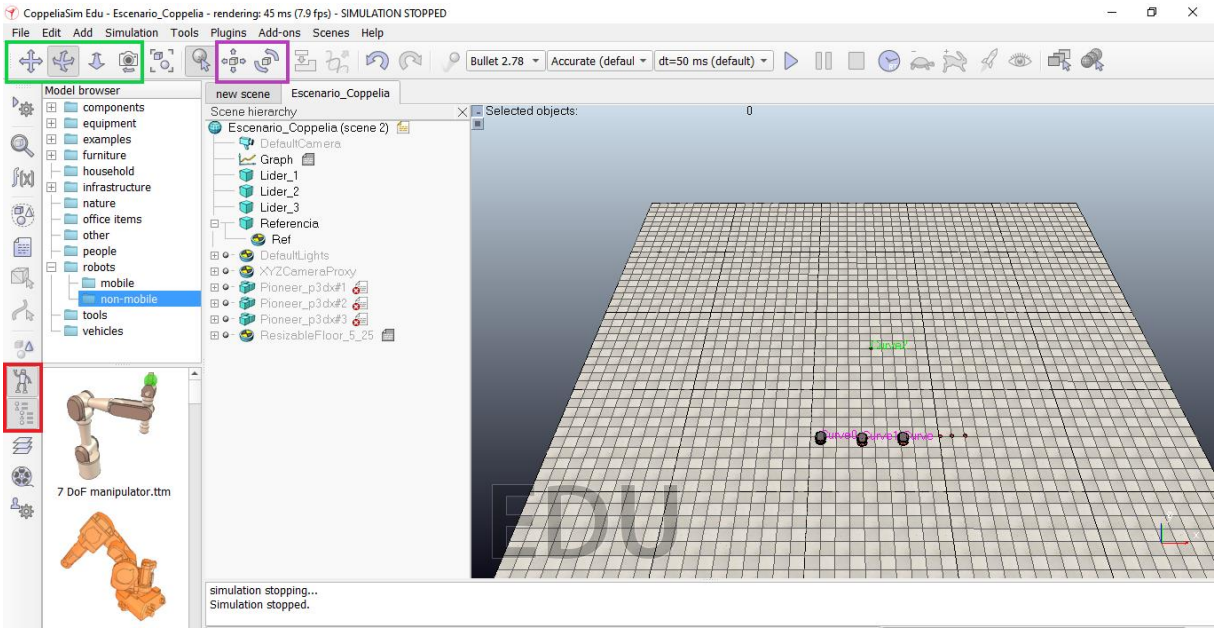


Figura III.4. Escena de simulación

Recomendaciones:

- Los botones encerrados con el color **rojo** desactivan las ventanas “Model broser” y “Scenary hieraechy”, lo cual ampliara el escenario de simulación para una mejor visualización de la simulación.
- Los botones encerrados con color **verde** permiten desplazar de manera lineal y angular el escenario de simulación. Personalizase en el caso de necesario, para tener una mejor experiencia dentro del entorno de CoppeliaSim.
- Los botones encerrados con el color **lila** permiten desplazar los elementos seleccionados del escenario de simulación. Para este caso específico, se puede movilizar los robots disponibles en la escena antes, durante y después de la simulación.

USO DEL INTERFAZ GRÁFICO

El interfaz gráfico cuenta con 3 ventanas:

- Ventana de inicio
- Ventana de simulación
- Ventana de resultados

Ventana de inicio

Esta ventana contiene la presentación principal de la aplicación desarrollada en este trabajo de titulación; donde se describe de manera resumida algunos detalles importantes. Además, cuenta con dos botones denominados de AYUDA que permite abrir el Manual de usuario de la aplicación y SIGUIENTE permite abrirá la siguiente ventana de la interfaz.

Figura III.5. Interfaz gráfico

Ventana de simulación

La ventana de simulación contiene varias opciones que permiten la personalizar la simulación, para lo cual se puede elegir el sistema multi-agente, la trayectoria de seguimiento, el tamaño de la trayectoria e incluir el ángulo de trayectoria para la formación. Personalizados las características, bastará con pulsar el botón RUN para iniciar toda la simulación; una vez culminada, la trayectoria de seguimiento será mostrada en la herramienta gráfica con la cual cuenta esta ventana, si se desean más detalles de la simulación el botón GRÁFICAS permitirá abrir la ventana de resultados.

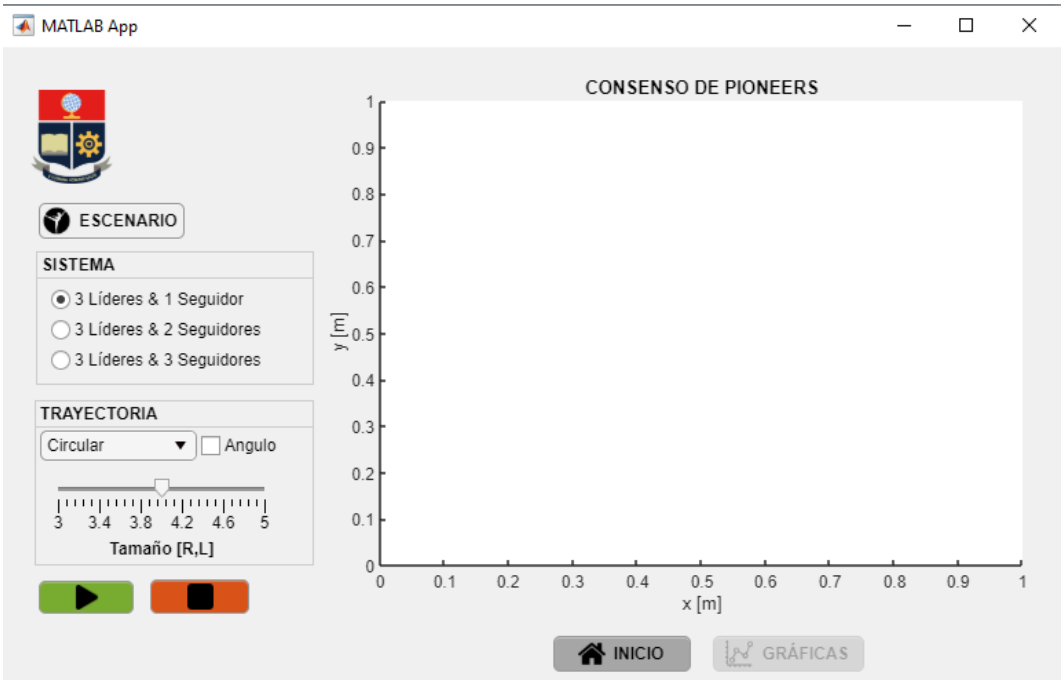


Figura III.6. Ventana de simulación

A continuación, se describen cada una de las características de la simulación:

Sistema: Formación del sistema multi-agente a utilizar.

Trayectoria: La trayectoria de seguimiento.

Botones:

“RUN”, permite iniciar la simulación utilizando los parámetros seleccionados.

“STOP", permite parar la simulación en cualquier instante de la simulación.

“ESCENARIO”, permitirá abrir directamente el escenario de simulación de CoppeliaSim.

“INICIO", permite regresar a la venta de inicio.

“GRÁFICAS”, permite avanzar a la siguiente ventana de resultados.

Angulo: permite incluir un ángulo de trayectoria a la formación.

Tamaño de trayectoria: permite personalizar el tamaño de la trayectoria en un rango de 3 – 5 metros.

Ventana de resultados

Esta ventana permite la visualización de algunos resultados del desempeño del sistema multi-agente, mostrando las señales de control lineal/angular y errores de posición x/y; estas gráficas, se pueden mostrar de manera grupal o individual.

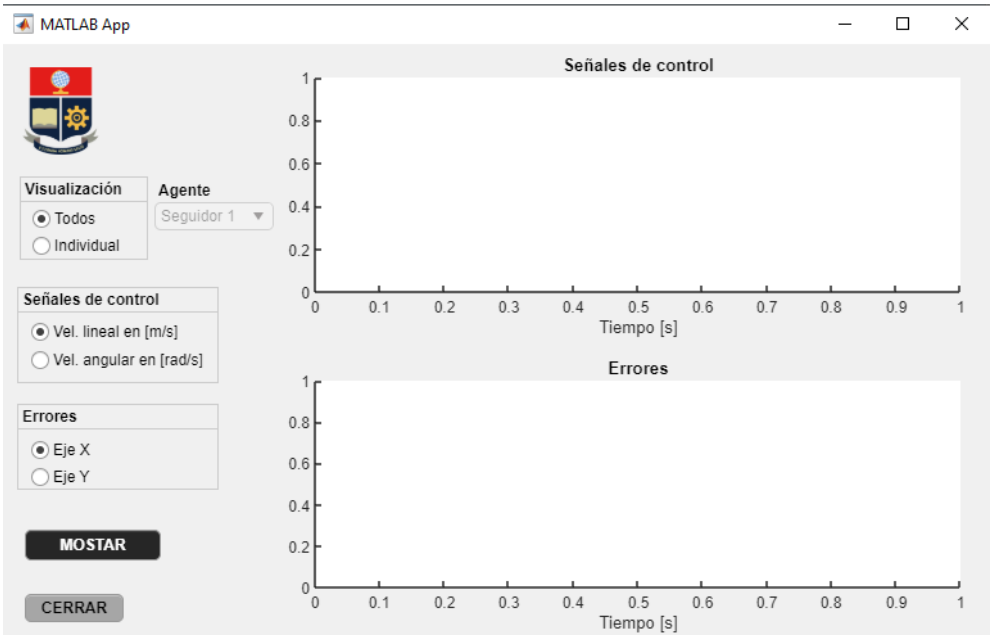


Figura III.7. Resultados

Visualización: Permite la selección del tipo de gráficas a mostrar.

Agente: Permite la selección del agente seguidor a mostrar.

Señales de control: Permite la selección del tipo de señal de control lineal o angular.

Errores: Permite la selección del error de posición en el eje x o y.

Botones:

“Mostrar”, este botón permite mostrar los resultados según las características seleccionadas.

“ATRÁS”, este botón regresará hacia la ventana de simulación.

ANEXO IV

ENLACES VIDEOS DEMOSTRATIVOS DE LA APLICACIÓN

Manipulación de la aplicación

https://youtu.be/S_uPYcTdfh8

Seguimiento de trayectorias del Sistema multi-agente:

- Prueba 1: 3 líderes y 1 seguidor

https://youtu.be/o07cshSF4TI

- Prueba 2: 3 líderes y 2 seguidores

https://youtu.be/dv075kSGqIA

- Prueba 3: 3 líderes y 3 seguidores

https://youtu.be/R899ZSoG2PA

Simulación con y sin ángulo de trayectoria

https://youtu.be/kpV9fc1owYA

Manipulación de la formación en CoppeliaSim

https://youtu.be/KmeSJK9j2aE