# Consenso Distribuido para la Navegación y Evasión de Obstáculos en Robots Móviles

Angie Katherinne Beltrán Lamus, Gabriela Maria Castro Beltrán. Universidad Nacional de Colombia, anbeltranl@unal.edu.co, gcastrob@unal.edu.co

**Resumen** - Este trabajo presenta la implementación de un sistema de consenso distribuido para la coordinación de múltiples robots móviles en un entorno conocido. Los agentes ajustan sus trayectorias en función de la información local de sus vecinos, logrando converger a un punto objetivo sin necesidad de un controlador centralizado. El sistema integra la cinemática diferencial de los robots, asegurando movimientos suaves y evitando colisiones mediante interacciones locales. La combinación de reglas de consenso y evasión de obstáculos permite que los robots mantengan el objetivo de grupo mientras navegan en el entorno.

***Abstract*— This work presents the implementation of a distributed consensus system for the coordination of multiple mobile robots in a known environment. The agents adjust their trajectories based on local neighbor information, converging to a target point without requiring a centralized controller. The system integrates the differential kinematics of mobile robots, ensuring smooth movements and collision avoidance through local interactions. The combination of consensus rules and obstacle avoidance enables the robots to maintain group cohesion while navigating in the environment.**

***Keywords: Consensus, navigation, multiple mobile robots, differential kinematics, collision avoidance***

## I. INTRODUCCIÓN

La coordinación distribuida de robots móviles es un problema clave en la robótica multiagente, con aplicaciones en exploración, formación de enjambres y misiones colaborativas. Un desafío fundamental en estos sistemas es lograr que un grupo de robots alcance un punto de consenso de manera autónoma, considerando restricciones cinemáticas y la necesidad de evitar colisiones con obstáculos en el entorno.

Esta implementación se basa en técnicas de optimización distribuida para diseñar un algoritmo de control para lograr un consenso que permita a los robots ajustar dinámicamente su trayectoria en función de mediciones locales y la interacción con sus vecinos. En lugar de depender de una planificación centralizada o del intercambio de grandes cantidades de datos, cada robot tomará decisiones en tiempo real utilizando un modelo basado en retroalimentación, asegurando un desplazamiento eficiente y coordinado hacia el punto de consenso.

El enfoque propuesto considera la cinemática de los robots móviles y la implementación de estrategias de evasión de obstáculos, asegurando que la navegación sea segura sin comprometer la convergencia al consenso. A través de simulaciones, se evaluará el desempeño del sistema en términos de tiempo de convergencia, estabilidad y robustez frente a perturbaciones. Con este artículo, se busca demostrar la efectividad de la optimización distribuida en la coordinación multiagente y su potencial aplicación en tareas de navegación autónoma y colaboración robótica.

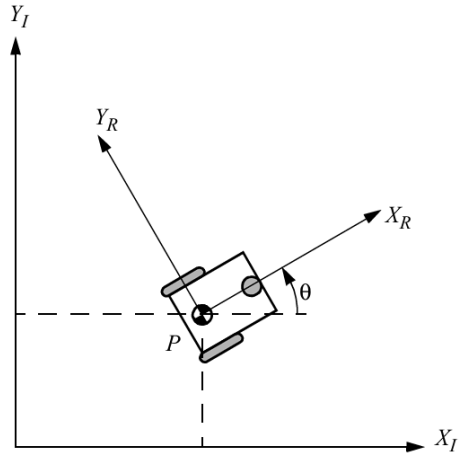


Fig. 1. Parámetros de un robot diferencial

## II. CONTROL Y EVASIÓN DE OBSTÁCULOS

### A. Cinemática Diferencial

Un robot móvil es un sistema autónomo capaz de desplazarse libremente en su entorno. No es posible determinar su posición de manera instantánea y directa, por lo que se debe calcular su desplazamiento a lo largo del tiempo. Además, la presencia de errores en la estimación del movimiento, como los causados por el deslizamiento, hace que obtener una ubicación precisa del robot sea un desafío considerable.

Para comprender cómo se mueve un robot, es fundamental analizar la función de cada una de sus ruedas en su desplazamiento. Cada rueda contribuye al movimiento general del robot, pero al mismo tiempo impone ciertas limitaciones, como la restricción lateral.

Un robot diferencial tiene dos ruedas motrices separadas por una distancia d `(wheel_distance = 0.5)`. Las velocidades de cada rueda son $v_L$ y $v_R$, y la velocidad del centro del robot v y su velocidad angular $\omega$ están dadas por:

$$v = \frac{v_R + v_L}{2} \tag{1}$$

$$\omega = \frac{v_R - v_L}{d} \tag{2}$$

Adicional, la posición del robot en coordenadas (x,y) y su orientación $\theta$ evolucionan según

$$\dot{x} = v\cos(\theta) \tag{3}$$

$$\dot{y} = v\sin(\theta) \tag{4}$$

$$\dot{\theta} = \omega \tag{5}$$

Estos últimas tres ecuaciones describen cómo cambia la posición y orientación del robot en función de su velocidad y giro.

### B. Control por consenso

Para lograr la integración con el consenso se da por la velocidad angular $\omega$ que se asigna proporcionalmente a $\theta_{diff}$, permitiendo que el robot gire más rápido cuando la diferencia de orientación es mayor. Al mismo tiempo, la velocidad lineal v disminuye a medida que el giro requerido es más pronunciado, evitando así cambios bruscos de dirección. Así las velocidades de las ruedas se determinan en función de v y $\omega$, asegurando un movimiento suave y estable.

$$\theta_{diff} = atan2(sen(\dot{\theta} - \theta), cos(\dot{\theta} - \theta)) \tag{6}$$

Donde $\dot{\theta}$, es el ángulo calculado de la dirección hacia el consenso, y la función $atan2$ se usa para asegurar que el resultado esté en el rango adecuado $[\pi, \pi]$, evitando problemas de discontinuidad al calcular la diferencia de ángulos.

## III. Modelo de Consenso Multiagente

Para desarrollar esta implementación basada en consenso distribuido en necesario que los robots ajusten sus posiciones y orientaciones para converger hacia un punto objetivo. Esto se logra mediante reglas locales que cada agente sigue:

1) Atracción al punto de consenso: Cada robot calcula una dirección de movimiento hacia el punto objetivo.Todos los robots comparten la misma ecuación de actualización basada en la diferencia de orientación $\dot{\theta}$ con respecto al punto de consenso. El consenso puede si bien ser el promedio de las posiciones, tener cualquier punto deseado siempre y cuando no esté dentro de los obstáculos.
2) Evasión de obstáculos: Cuando un robot detecta un obstáculo cercano, genera una fuerza de repulsión que modifica su trayectoria.
3) Cinemática diferencial: Se usa un modelo de robot diferencial, lo que implica que la orientación juega un rol clave en el movimiento. La velocidad lineal disminuye cuando hay un gran giro requerido, evitando movimientos bruscos.
4) Parámetro de parada: Si un robot ya ha alcanzado el punto de consenso, su velocidad se reduce a 0. Esto evita que siga moviéndose innecesariamente y permite que otros robots continúen su trayectoria sin interferencias.
5) El tiempo de convergencia: El consenso está influenciado por los obstáculos, puesto que estos desvían a cada robot de su trayectoria óptima hacia el consenso, el tiempo total para que todos lleguen puede aumentar.

Adicionalmente, se puede destacar en este sistema el comportamiento de control basado en potenciales, donde las fuerzas de atracción (hacia el consenso) y repulsión (obstáculos) generan trayectorias suaves y adaptativas. Cabe resaltar que el comportamiento colaborativo se basa en converger todos al mismo punto y alcanzar el consenso, cumpliendo las reglas mencionadas.

En un espacio o entorno no estructurado, un sistema como este podría ser clave en la coordinación colaborativa al compartir información de los obstáculos que va detectando, si un robot detecta un obstáculo, puede enviar su ubicación a los demás para que ajusten su trayectoria antes de llegar a él.

## IV. Implementación del consenso en Matlab

### A. Consenso en el promedio de los agentes

El código simula el sistema de agentes móviles que buscan alcanzar un acuerdo en sus posiciones mientras evitan obstáculos en un entorno bidimensional. La simulación se desarrolla en una cuadrícula de tamaño especificado por `scene_size`, donde los nodos se desplazan durante un número máximo de iteraciones definido por `max_iter`. Cada nodo comienza en una posición inicial y está conectado a otros nodos mediante una matriz de adyacencia, en el caso del consenso por promedio, que establece las relaciones de vecindad. Los nodos intentan moverse hacia una posición de consenso, calculada a partir de las posiciones de sus vecinos, mientras evitan colisiones con obstáculos ubicados en posiciones específicas y con radios determinados.

Inicialmente el código configura varios parámetros clave para la simulación. Se establece que el sistema consta de $n = 4$ nodos, lo que significa que hay cuatro agentes moviéndose en el espacio bidimensional. Este espacio está representado por una cuadrícula de 20 por 20 unidades, definida por el parámetro `scene_size`.

La simulación se lleva a cabo durante un máximo de 150 iteraciones, como se indica en `max_iter`. Cada iteración representa un intervalo de tiempo en el que los nodos pueden ajustar sus posiciones. La magnitud de estos ajustes está regulada por la tasa de aprendizaje, `alpha` = 0.1, que determina el grado de movimiento de un nodo en respuesta a las fuerzas calculadas. Para asegurar que los nodos eviten colisiones con los obstáculos, se define una distancia de seguridad de 1 unidad alrededor de cada obstáculo, especificada por `safe_distance`. Además, se aplica una fuerza de repulsión de 5 unidades (`repulsion_strength`) cuando los nodos se acercan demasiado a un obstáculo, empujándolos hacia una zona segura. Simultáneamente, los nodos son atraídos hacia una posición de consenso con una fuerza de 0.5 unidades (`attraction_strength`), lo que facilita su convergencia hacia una posición común.

La escena se inicializa como una cuadrícula vacía, y los obstáculos se colocan en posiciones específicas, cada uno definido por sus coordenadas centrales y un radio. Las posiciones iniciales de los nodos se configuran en las esquinas de la cuadrícula, proporcionando un punto de partida variado para la simulación. Estas posiciones se especifican en una

matriz donde cada fila representa las coordenadas $(x, y)$ de un nodo.


Fig. 2. Figura de los obstaculos definidos en el espacio de 20x20.

Finalmente, se utiliza una matriz de adyacencia para describir las conexiones entre los nodos. Esta matriz es crucial para determinar las relaciones de vecindad, indicando qué nodos pueden influenciarse mutuamente. Por ejemplo, si la entrada $(i, j)$ en la matriz es 1, significa que el nodo $i$ está conectado al nodo $j$. Con estos parámetros y configuraciones, el sistema está listo para simular el movimiento de los nodos hacia un consenso, mientras navegan de manera segura alrededor de los obstáculos en el espacio definido. La ecuación del consenso se basa en calcular el promedio de las posiciones de los nodos vecinos. Para un nodo $i$, la posición de consenso $\mathbf{c}_i$ se calcula como:

$$\mathbf{c}_i = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbf{x}_j \tag{7}$$

donde:
- $\mathbf{x}_j$ es la posición del nodo vecino $j$.
- $\mathcal{N}_i$ es el conjunto de nodos vecinos de $i$ (definido por la matriz de adyacencia).
- $|\mathcal{N}_i|$ es el número de vecinos del nodo $i$.

En el código, esta ecuación se implementa calculando el promedio de las posiciones de los nodos vecinos para cada nodo en cada iteración. Si un nodo no tiene vecinos, su posición de consenso es simplemente su propia posición.

Para ajustar el valor de consenso y evitar que caiga dentro de un obstáculo, primero se calcula la distancia entre el valor de consenso y el centro del obstáculo. Esto se realiza mediante la función `norm`, que calcula la distancia euclidiana entre dos puntos en el espacio bidimensional:

$$dist\_to\_obs = \|consensus\_value - obs\_center\| \tag{8}$$

A continuación, se verifica si el valor de consenso está dentro del obstáculo. Esto se determina comprobando si la distancia calculada (`dist_to_obs`) es menor que el radio del obstáculo (`obs_radius`). Si esta condición se cumple, significa que el consenso está dentro del área del obstáculo. Si el consenso está dentro del obstáculo, se ajusta para que esté fuera de él. Se calcula un vector de dirección de repulsión (`repulsion_dir`) que apunta desde el centro del obstáculo hacia el valor de consenso. Este vector se normaliza dividiendo por `dist_to_obs` para obtener un vector unitario:

$$repulsion\_dir = \frac{consensus\_value - obs\_center}{\|consensus\_value - obs\_center\|} \tag{9}$$

El valor de consenso se ajusta moviéndolo a lo largo de la dirección de repulsión hasta una distancia igual al radio del obstáculo más una distancia de seguridad (`safe_distance`):

$$consensus\_value = \tag{10}$$
$$obs\_center + (obs\_radius + safe\_distance) \times repulsion\_dir \tag{11}$$

Para calcular la dirección de movimiento hacia el consenso, se determina la diferencia entre el valor de consenso ajustado y la posición actual del nodo (`node_pos`):

$$move\_direction = consensus\_value - node\_pos \tag{12}$$

Si la dirección de movimiento no es un vector nulo, se normaliza para convertirlo en un vector unitario, asegurando que tenga una magnitud de 1. Finalmente, se inicializa la fuerza total que se aplicará al nodo. Esta fuerza se calcula multiplicando la dirección de movimiento normalizada por la fuerza de atracción (`attraction_strength`). Esta fuerza representa el impulso inicial del nodo hacia el consenso, antes de considerar cualquier ajuste por repulsión de obstáculos:

$$total\_force = attraction\_strength \times move\_direction \tag{13}$$

Esta fuerza se suma a la fuerza total:

$$\text{total\_force} = \text{total\_force} + \text{repulsion\_force} \times \text{repulsion\_dir} \tag{14}$$

Donde:

$$repulsion\_dir = \frac{node\_pos - obs\_center}{dist} \tag{15}$$
$$repulsion\_force = repulsion\_strength * (obs\_radius - dist) \tag{16}$$

La fuerza total se normaliza si no es un vector nulo:

$$\text{total\_force} = \frac{\text{total\_force}}{\|\text{total\_force}\|} \tag{17}$$

Finalmente, se propone un nuevo movimiento para el nodo, actualizando su posición con base en la fuerza total escalada por la tasa de aprendizaje (`alpha`):

$$\text{new\_pos} = \text{node\_pos} + \alpha \times \text{total\_force} \tag{18}$$

La posición del nodo se actualiza con esta nueva posición, asegurando que los nodos se muevan de manera segura y eficiente hacia una posición de consenso común.

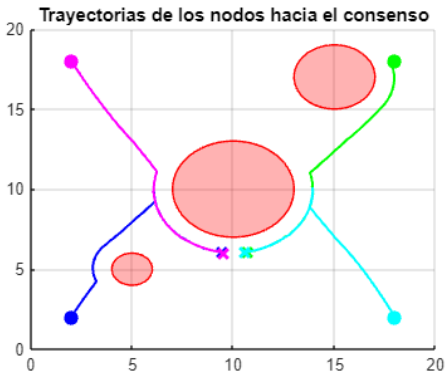


Fig. 3. Trayectorias de cada robot hacia el consenso evitando los obstáculos.

### B. *Consenso en un punto determinado*

Se quiso implementar cómo lograr el consenso en un punto que no fuera necesariamente el promedio de los agentes. Para esto se eligió el punto $(16, 2)$ como se muestra con el circulo rojo en la Figura 4, donde los robots tienen adicionalmente una orientación diferente.

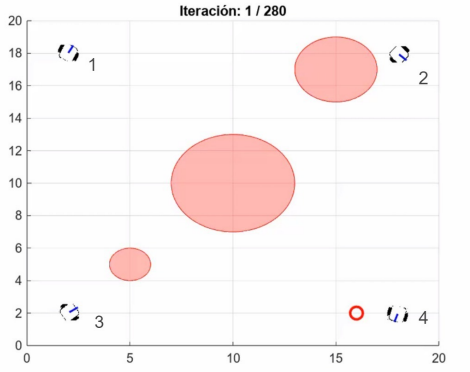


Fig. 4. Robots en su posición inicial

Los agentes al empezar a iterar, similar al consenso por promedio, ajustan sus trayectorias y orientaciones hacia este nuevo destino. Sin embargo, al tener el agente 4 muy cerca al punto seleccionado, en la iteración 26 ya este ha logrado el consenso por lo cual aplica el criterio de parada a través de un condicional en el código para que este deje de moverse y quedar oscilando.

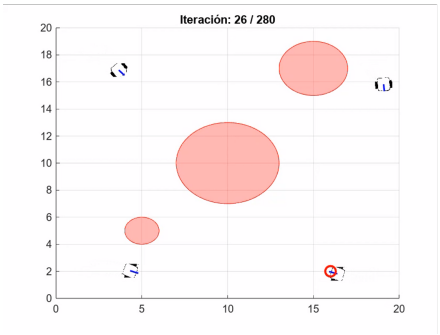


Fig. 5. El agente 4 es el primero en lograr llegar al consenso

El consenso es alcanzado por todos los agentes en la iteración 127, a excepción del agente 1 que era quien estaba más alejado y tuvo que evitar más obstáculos. No obstante, esto lo logro sin colisionar, lo cual era el objetivo principal. En la Figura 7, se aprecia el consenso final en la iteración 260.

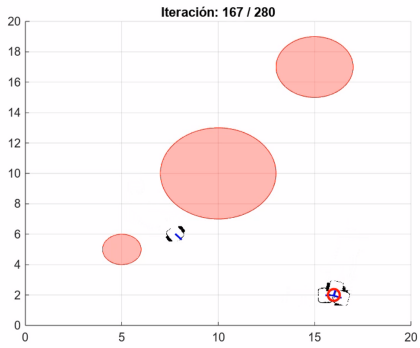


Fig. 6. Agente en proceso del llegar al consenso

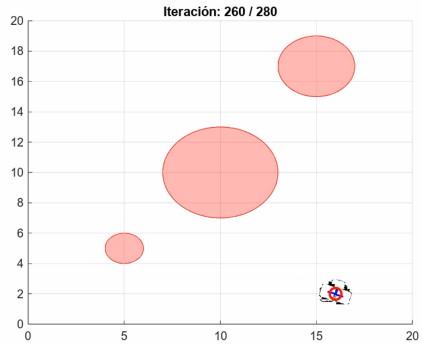


Fig. 7. Agentes con el consenso logrado

## V. Conclusiones

La implementación desarrollada permite la coordinación de múltiples agentes móviles en un entorno conocido, utilizando principios de consenso distribuido. A través de la interacción local, los robots logran alcanzar un punto objetivo, sin depender de un controlador centralizado. Este tipo

de simulaciones es común en el estudio de sistemas multi-agente, donde se busca entender cómo agentes individuales pueden colaborar para lograr un objetivo común mientras navegan en un entorno con restricciones. La combinación de fuerzas de atracción hacia el consenso y de repulsión para evitar obstáculos es un enfoque típico en robótica y redes de sensores, siendo de esta manera aplicada a la cinemática de cualquier robot en busca del consenso.

Si bien el sistema permite la sincronización de los agentes y la evasión de obstáculos de forma reactiva, no incorpora una optimización explícita de trayectorias. Las decisiones de movimiento se basan en reglas locales y no en la minimización de una función de costo global. Además, la ausencia de un mecanismo de planificación anticipada implica que los robots ajustan su trayectoria en tiempo real sin prever rutas óptimas. Esto puede generar caminos subóptimos y tiempos de convergencia mayores en comparación con enfoques que emplean técnicas de optimización de trayectorias.

En futuras mejoras, se podría integrar un esquema de optimización que permita a los agentes encontrar trayectorias eficientes, considerando tanto la distancia como la energía consumida. Asimismo, la incorporación de estrategias de aprendizaje o adaptación al entorno permitiría mejorar la robustez del sistema frente a dinámicas más impredecibles.

## REFERENCES


[1] Mojica-Nava, Optimización y control en grafos, Editorial Universidad Nacional de Colombia, Bogotá D.C, Primera edición, 2023. Disponible: https://www.researchgate.net/publication/376682016_Optimizacion_y_control_en_grafos

[2] D. Yanguas, "Control of Heterogeneous Robot Networks for Assistance in Search and Rescue Tasks," Tesis de maestría, Facultad de Ingeniería, Universidad Nacional, Bogotá, Colombia, 2018.

[3] Siegwart, Roland "Introduction to autonomous mobile robots" MIT, 2004.