MÁSTER EN ROBÓTICA Y AUTOMATIZACIÓN

Navegación Autónoma de Robots

AVANCE PROYECTO INTEGRADOR – PARTE III

Realizado por:

William Andrade

Glen Alejandro Guerrero Burbano

Yeison Jimenez

Profesor:

Daniel Vicente Rodrigo Muñoz

25 de febrero de 2026

I. RESUMEN

A lo largo de este trabajo se presenta en un entorno de simulación la planificación de una trayectoria de un robot móvil mediante el uso de planificadores que utilizan el método de rejilla y basados en muestreo. A demás se realiza actualizaciones de los filtros de localización para poder mejorar el mapa slam en futuras entregas.

II. MATERIALES Y MÉTODOS:

Parte 1 – Elección del mapa: Antes de realizar pruebas del mapa de ocupación obtenido por SLAM, se ha decido probar con un mapa complejo del toolbox de Matlab para inicialmente verificar el funcionamiento de 2 de los algoritmos de planificación trabajado en el curso, El primero correspondiente basado en planificación por rejilla (A)* y el otro por muestreo (RRT*). A demás se establece los puntos de clasificación “q_start= [5,5]” y “q_goal=[45,35] que son los puntos de inicio y de finalización de la trayectoria que se quiere encontrar respectivamente. A continuación, en la figura 1, muestra el mapa de ocupación con el circulo verde representando el inicio de la trayectoria y en círculo azul el final de la misma.

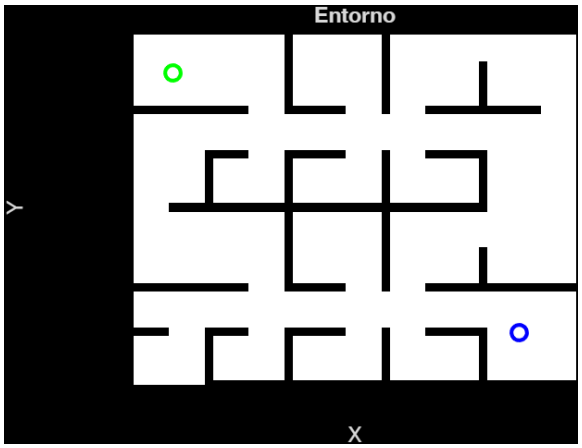


Figura 1. Mapa inicial.

1. PLANIFICACIÓN BASADA EN REJILLA (GRID) CON A*:

Para este código se realiza la representación binaria del mapa donde cada pixel puede representar una zona con obstáculo o una zona libre. Se construirá una rejilla uniforme a partir del mapa y se implementaran el algoritmo A*. Dibujar cada estrategia de búsqueda en una figura independiente.

A* Combina BFS y el algoritmo Dijkstra, lo cual indica que intenta encontrar la mejor ruta o la más rápida; incorpora heurística admisible y los parámetros a diferencia de BFS tiene 4(Norte,Sur, Oriente y Occidente), y A* tiene 8, lo que permite al algoritmo considerar rutas diagonales entre las celdas, logrando así la mejor ruta. Esto se considera la distancia euclídea como se observa las diagonales en la trayectoria azul de la Figura 2.

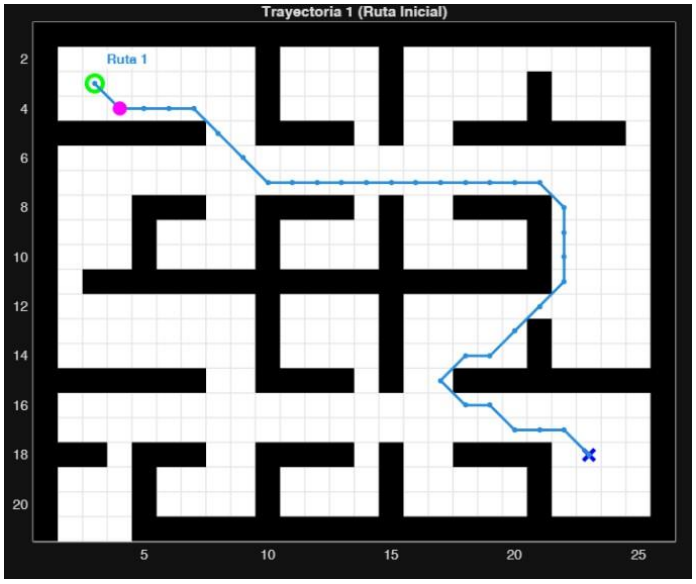


Figura 2. Ruta inicial A.*

Para poder hacer una planificación dinámica y volver a planear la trayectoria en caso que llegue a haber un obstáculo que aparezca en el mapa se realiza la simulación de aparición de un obstáculo de 3x3 celdas de color rojo. Una vez se haya detectado el obstáculo el cuál en la práctica deberá ser detectado por sensores, por ahora se simula movimiento como la misma trayectoria ideal y procederá a detenerse y replantear una nueva trayectoria como se observa en la figura 3.

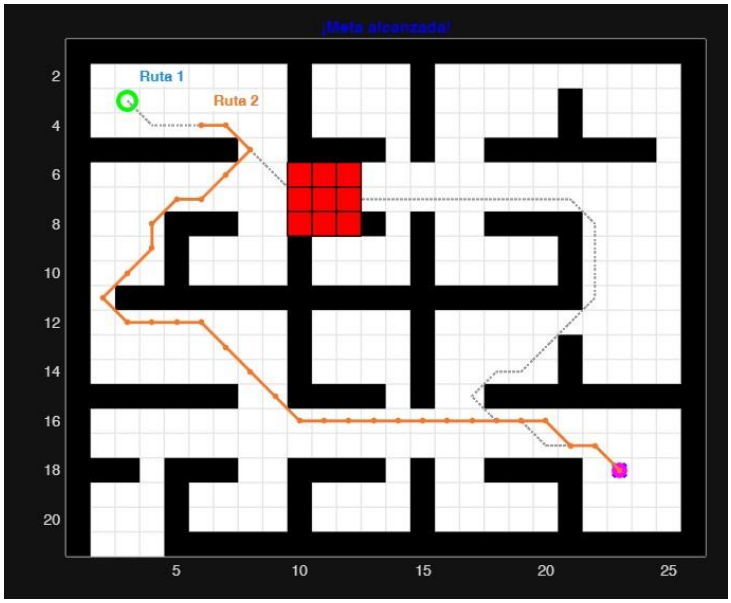


Figura 3. Ruta nueva A.*

El código programado permite generar más obstáculos en el mapa, más si embargo por la complejidad del utilizado actualmente no se implementa más de 2 obstáculos porque se podría llegar al punto de no encontrar trayectorias.

2. **Planificación con RRT*:** RRT*(Rapidly Exploring Random Tree Star) como su nombre lo En la figura 4 se muestra la trayectoria inicial realizada por RRT*

3. dice, es una mejora del RRT standar, genera un árbol con un proceso de reconexión desde el inicio(q_Start) al final(q_goal). Soluciona el problema que generea la trayectoria RRT, haciendo que ahora las trayectorias si sean más suaves al planear una trayectoría “óptima”, obviamente incrementando el tiempo de computo. Con la ayuda de IA se logró obtener un código incluso optimizado para Matlab.

   Se ha generado una simulación donde se simula el movimiento del robot con la matriz PATH que representa las coordenadas x,y de los nodos que sigue la trayectoria del robot (Es una movimiento ideal); En la práctica este movimiento deberá ser implementado con los algoritmos de lectura de sensores o odometría e incluso al control del robot. Por ahora se lo hace de una manera ideal desde simulación para enfocarse más en la planificación.

   En la tabla 1, se presenta los parámetros usados para RRT*.

Parámetros RRT	**Definición**	**Valor**
max_iter	Número de interacciones máximas para encontrar la trayectoria	2500
step_size	Paso de avance en cada expansión del árbol	3
goal_threshold	Umbral de distancia mínima para considerar alcanzado el objetivo.	3
rewire_radius	Radio de reconexión nodos para mejorar la ruta	8

Tabla 7. Definición de Parámetros RRT.*

   En la figura 4 se muestra la trayectoria inicial realizada por RRT*

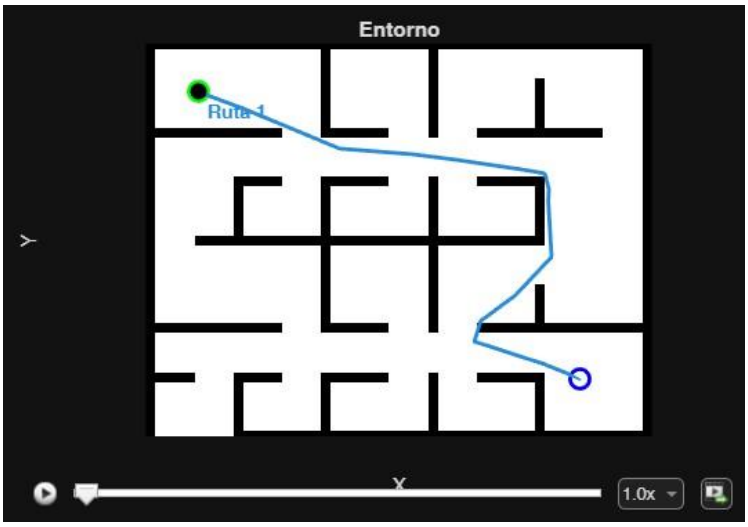


Figura 4. Ruta inicial RRT.*

En la figura 5 se muestra el instante donde aparece un obstaculo y en la figura 6 se muestra la nueva trayectoria calculada por RRT*

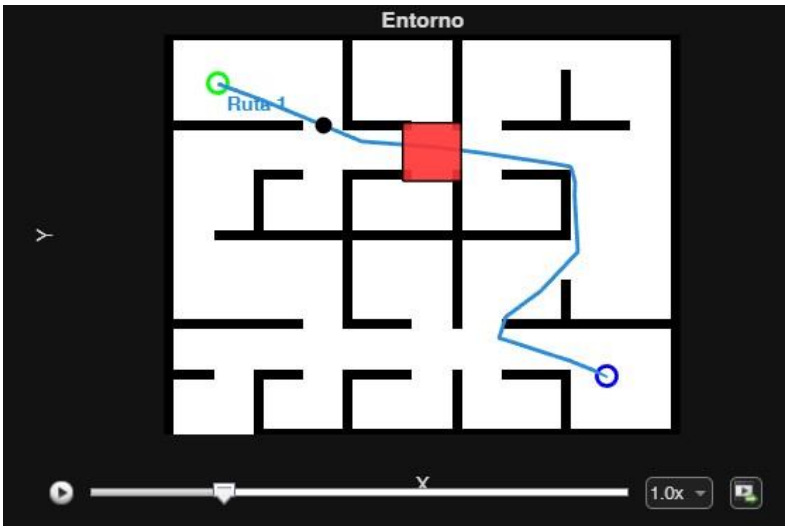


Figura 5. Obstáculo RRT.*

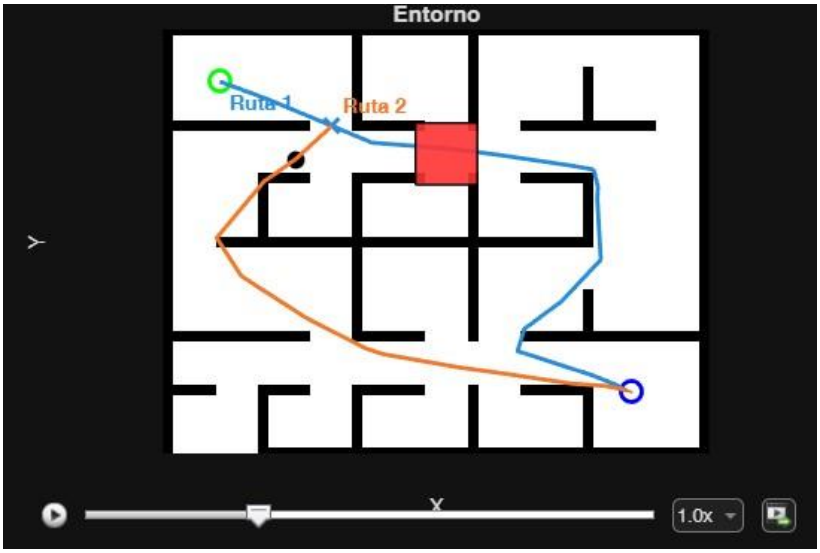


Figura 6. Nueva trayectoria RRT.*

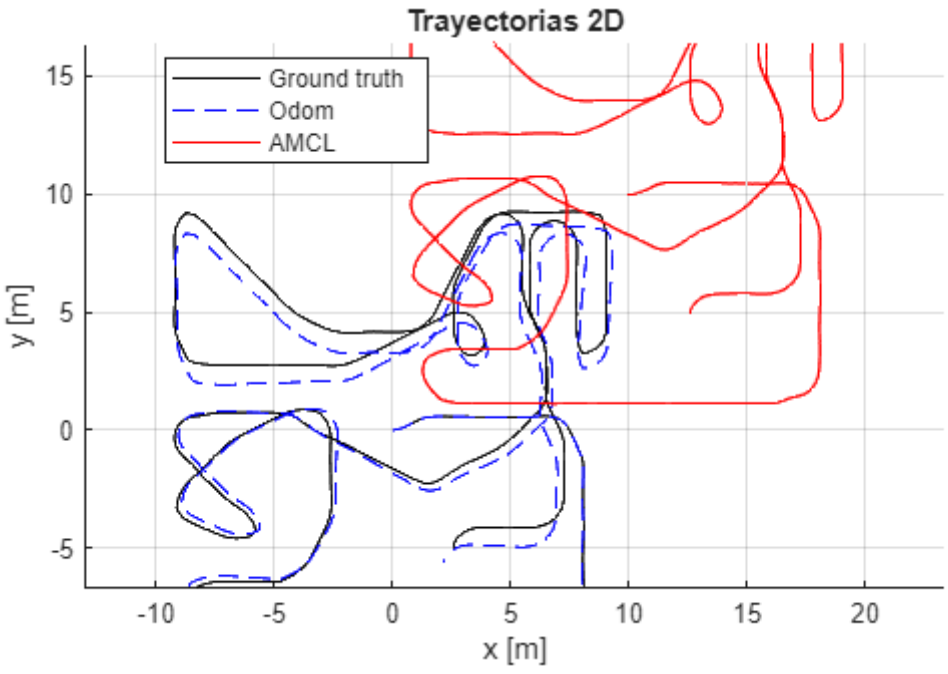
4. ACTUALIZACIÓN FILTROS LOCALIZACIÓN

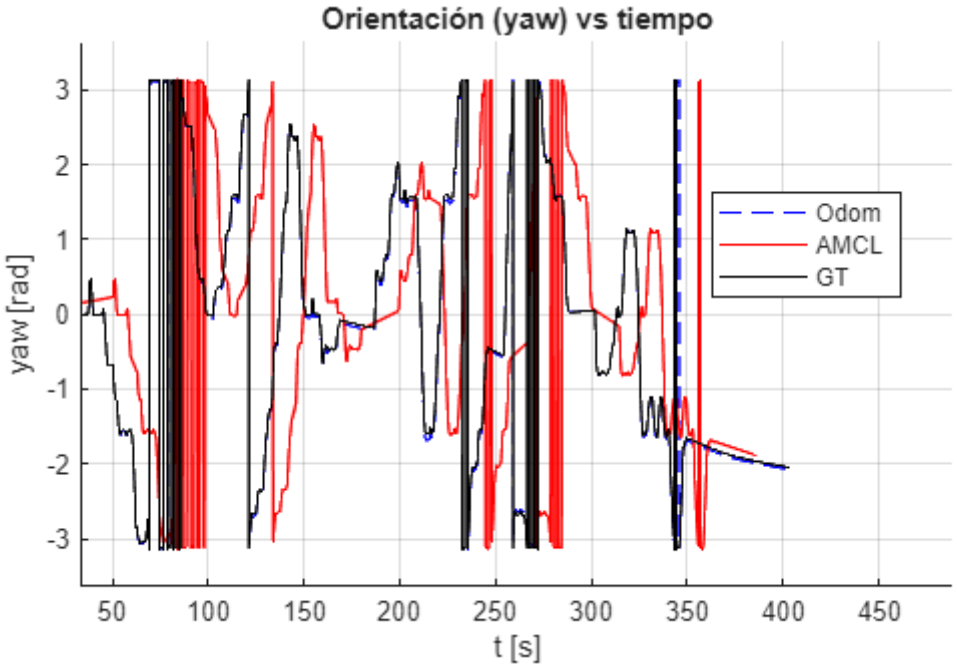
Filtros de localización : Utilizando el software MATLAB en la versión R2025b y sus toolboxes se ha creado el script PI_1 para invocar las funciones creadas en el script analyze_bag.m el cual realiza la importación de una RosBag(Datos almacenados de una simulación de un robot en el programa ROS), una vez realizada la importación del Bag, se procede a analizar los topics que esta cuenta de los cuales se tiene en cuenta **/odom y /base_pose_ground_truth.** "base_pose" corresponde a la ruta real realizada por el robot y **"**Odom" corresponde a los datos de odometría leídos que aproximan la posición del robot.

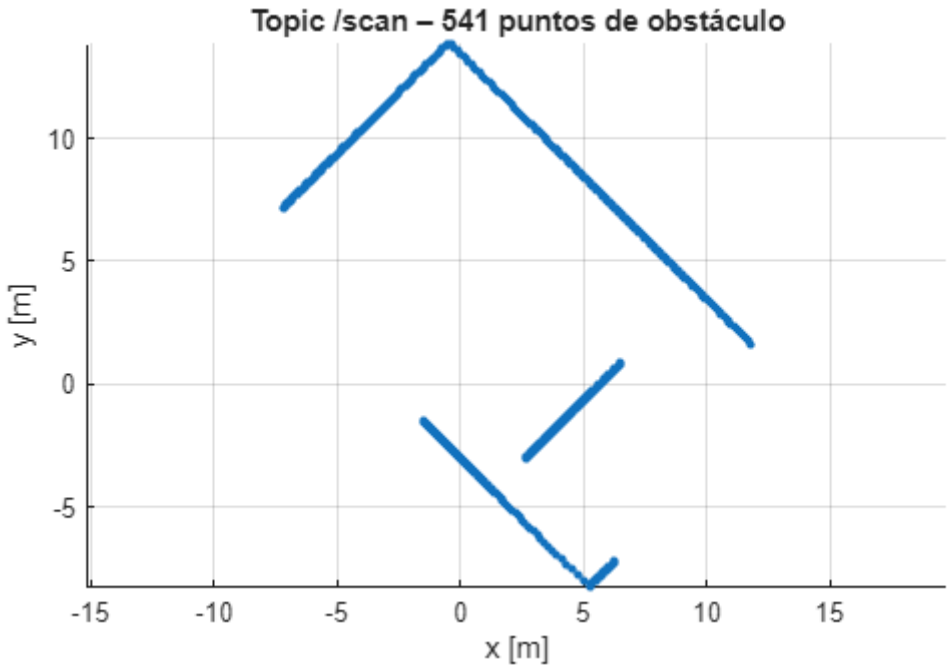
Para el desarrollo hemos evaluado diferentes técnicas de localización para robots móviles utilizando los datos de un robot MIR, y hemos decidido implementar los algoritmos EKF, UKF, y MCL (Filtro de Partículas). Al cargar el rosbag tenemos los siguientes topics disponibles:

```
Cargando rosbag: mir_basics_20251210_114529.bag

Topics disponibles:
    {'/amcl_pose'              }
    {'/base_pose_ground_truth'}
    {'/clock'                  }
    {'/cmd_vel'                }
    {'/imu_data'               }
    {'/joint_states'           }
    {'/map'                    }
    {'/map_metadata'           }
    {'/mir/joint_states'       }
    {'/move_base_simple/goal' }
    {'/odom'                   }
    {'/odometry/filtered'      }
    {'/particlecloud'          }
    {'/scan'                   }
    {'/tf'                     }
    {'/tf_static'              }

Leídos 16636 mensajes de /odom
Leídos 20191 mensajes de /base_pose_ground_truth
Leídos 1397 mensajes de /amcl_pose
```

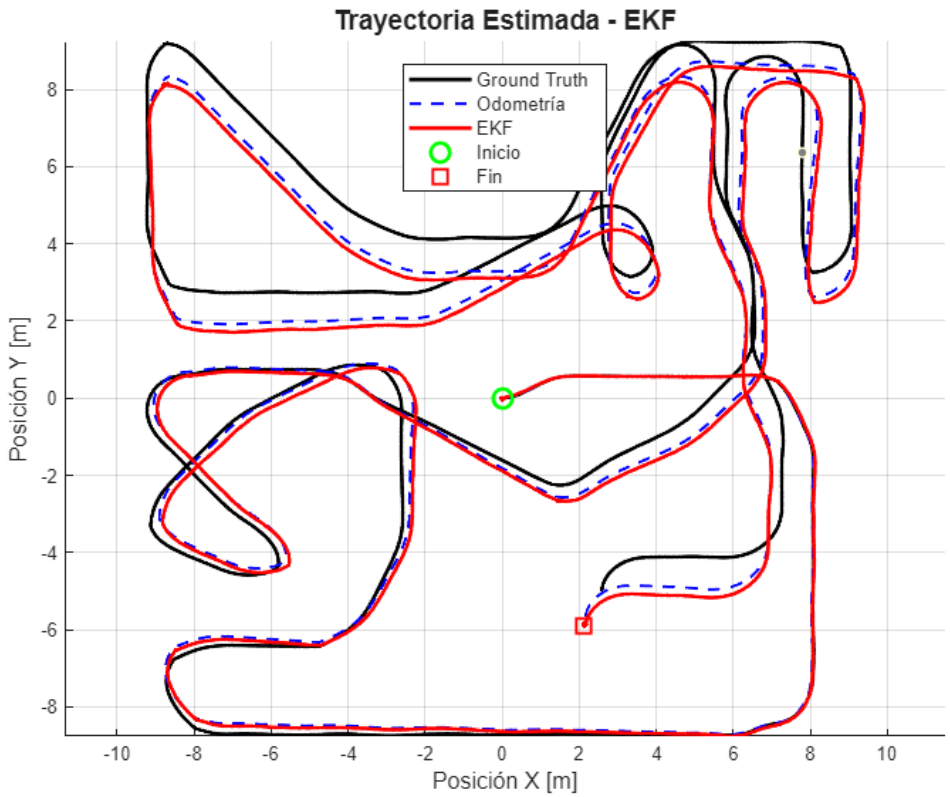
A continuación podemos observar las trayectorias 2D, Orientación vs Tiempo y el scan con 541 puntos de obstáculo.

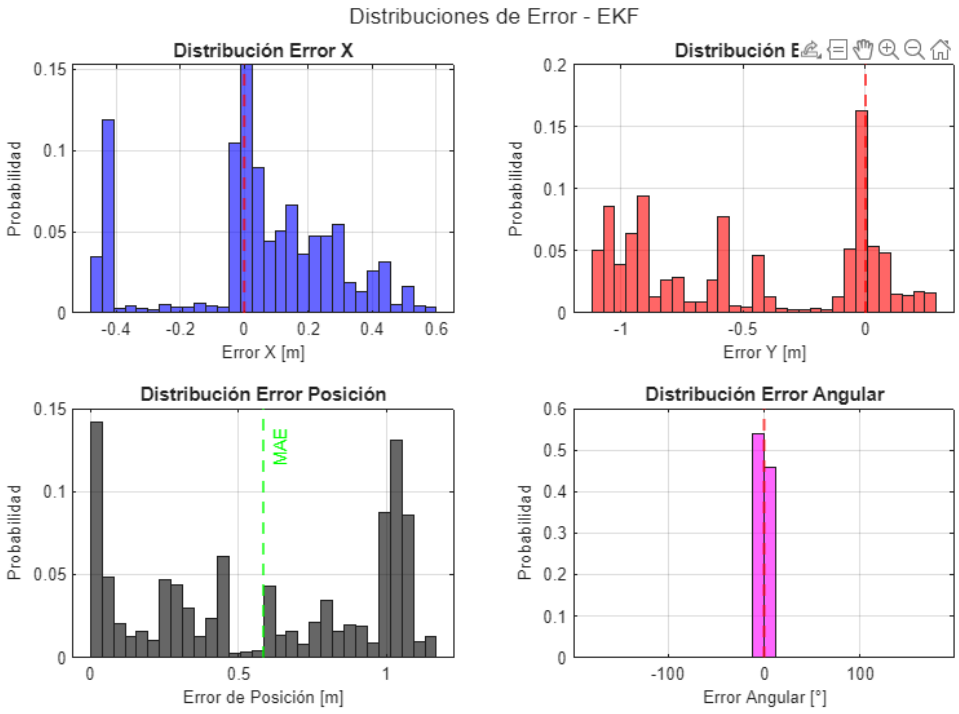


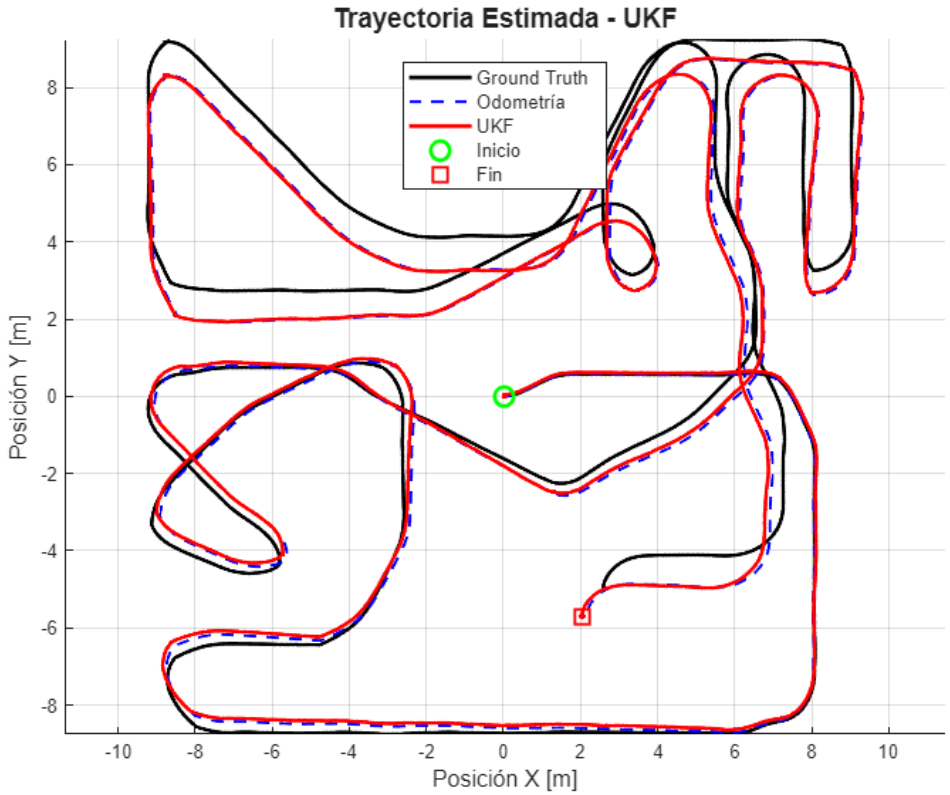


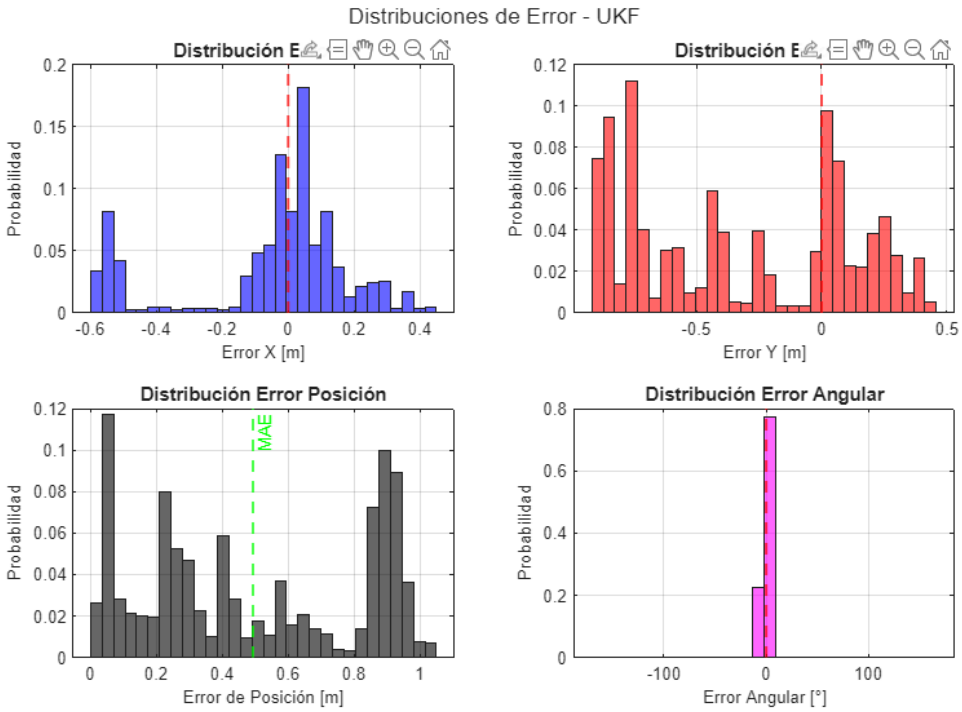


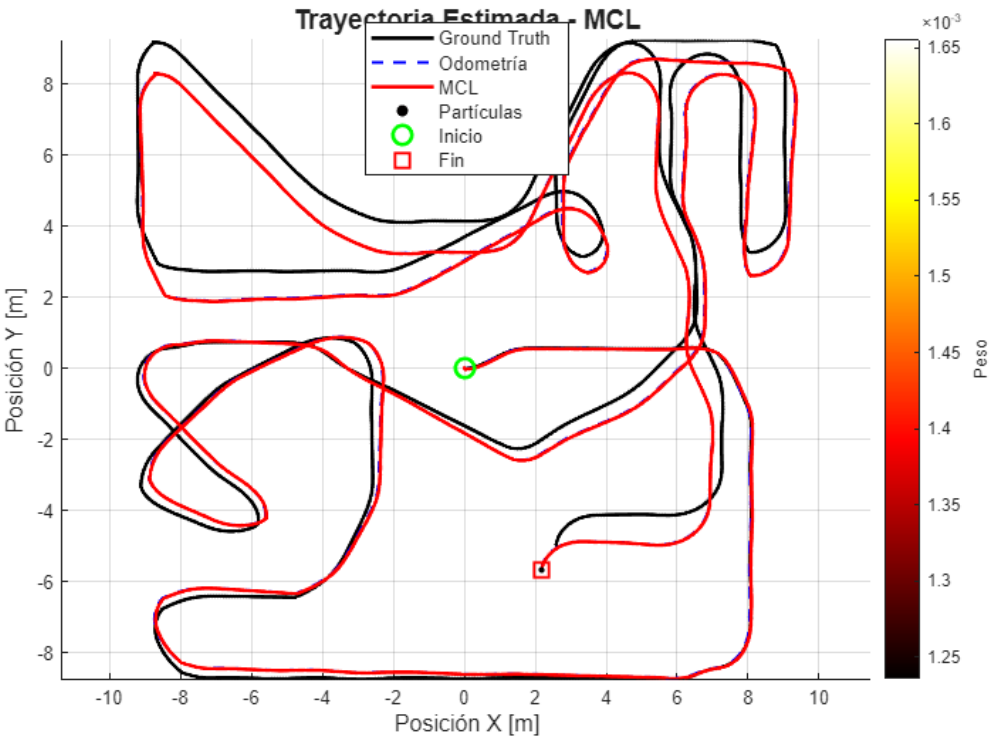
Luego empezamos a generar las trayectorias aplicando los diferentes algoritmos, con las gráficas de distribución de error

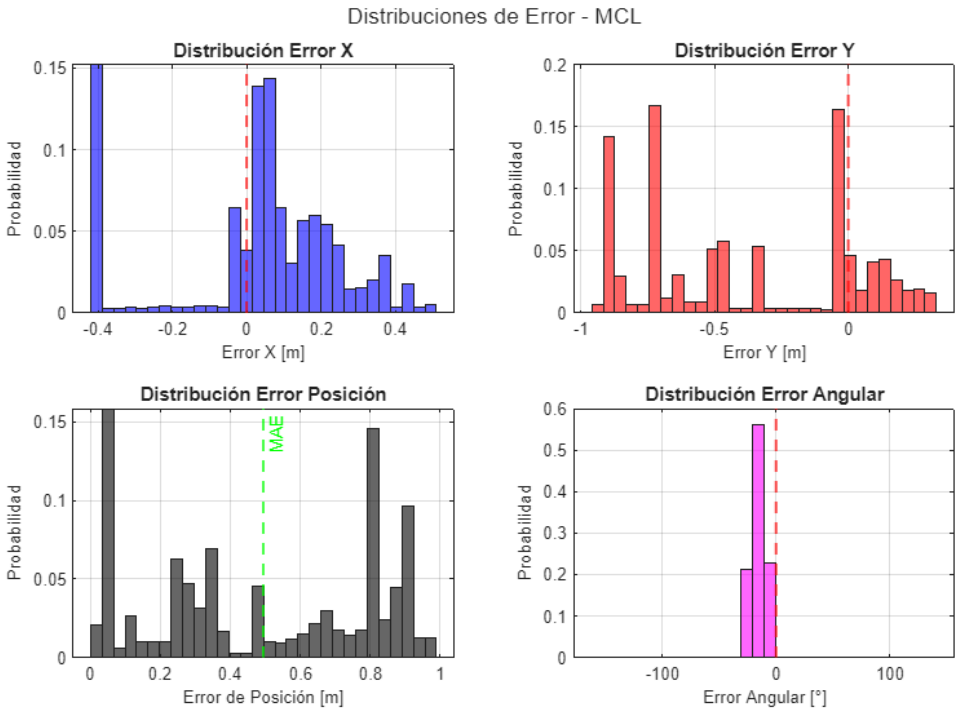

Trayectoria Estimada - EKF
Ground Truth
Odometría
EKF
Inicio
Fin
Posición Y [m]
Posición X [m]

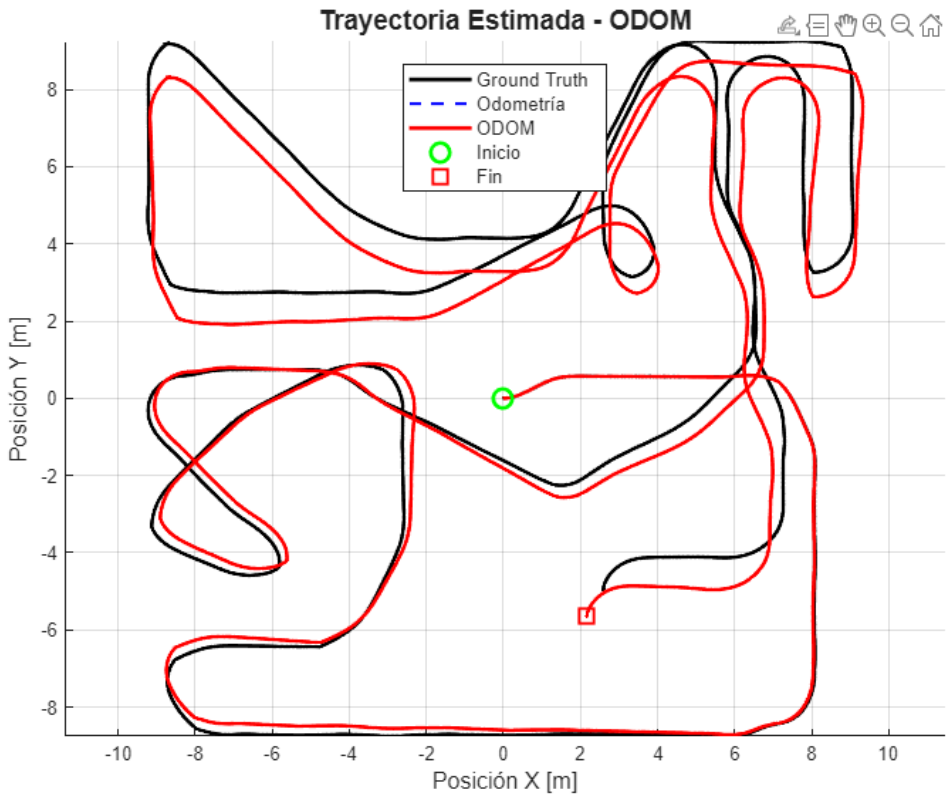

Distribuciones de Error - EKF
Distribución Error X
Probabilidad
Error X [m]
Distribución E
Probabilidad
Error Y [m]
Distribución Error Posición
MAE
Probabilidad
Error de Posición [m]
Distribución Error Angular
Probabilidad
Error Angular [°]

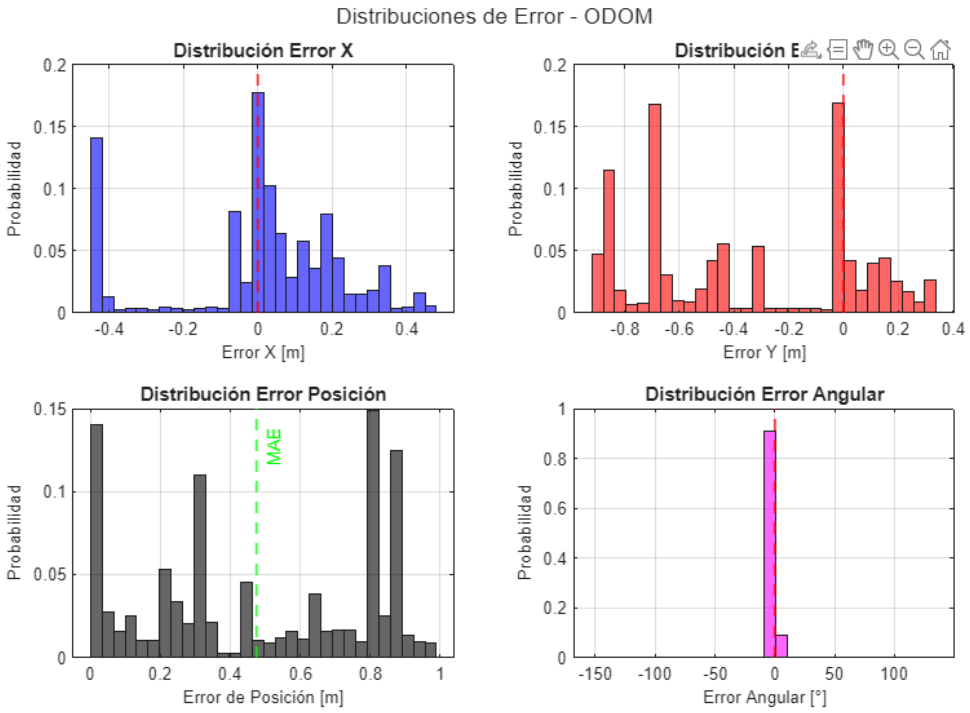

Trayectoria Estimada - UKF
Ground Truth
Odometría
UKF
Inicio
Fin
Posición Y [m]
Posición X [m]

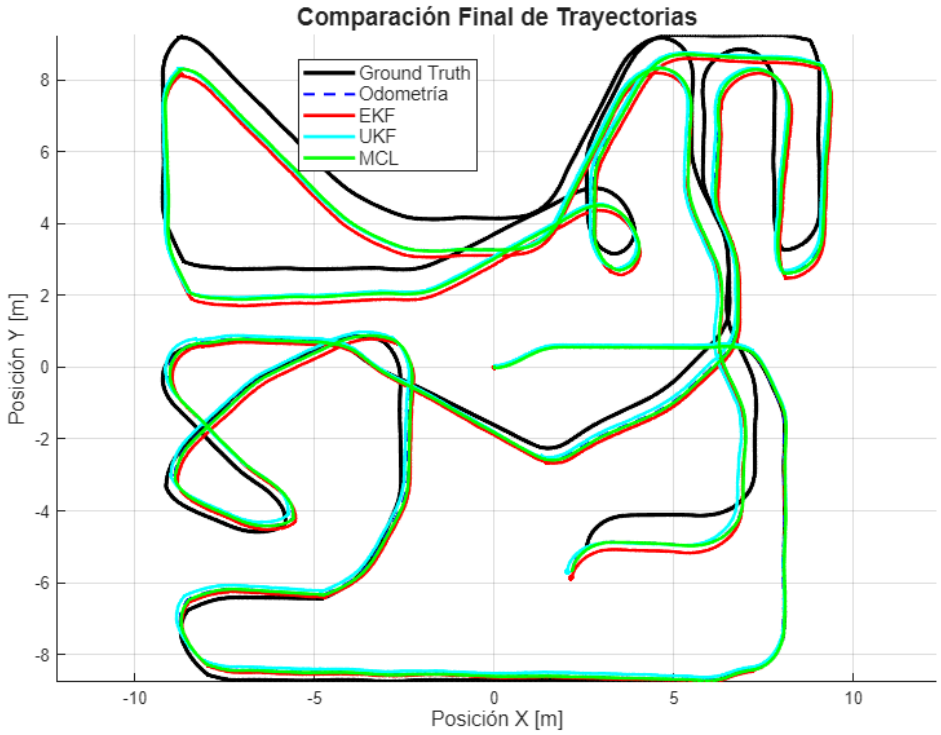

Distribuciones de Error - UKF
Distribución E
Probabilidad
Error X [m]
Distribución E
Probabilidad
Error Y [m]
Distribución Error Posición
MAE
Probabilidad
Error de Posición [m]
Distribución Error Angular
Probabilidad
Error Angular [°]

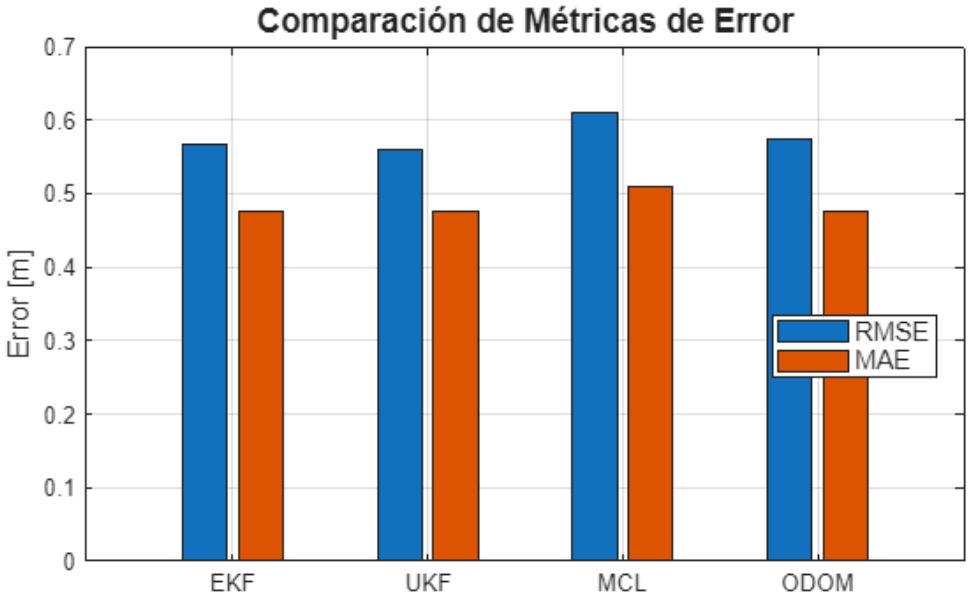

Trayectoria Estimada - MCL
Ground Truth
Odometría
MCL
Partículas
Inicio
Fin
Posición X [m]
Posición Y [m]
Peso
×10⁻³


Distribuciones de Error - MCL
Distribución Error X
Error X [m]
Distribución Error Y
Error Y [m]
Distribución Error Posición
MAE
Error de Posición [m]
Distribución Error Angular
Error Angular [°]
Probabilidad


Trayectoria Estimada - ODOM
Ground Truth
Odometría
ODOM
Inicio
Fin
Posición Y [m]
Posición X [m]


Distribuciones de Error - ODOM
Distribución Error X
Probabilidad
Error X [m]
Distribución E
Probabilidad
Error Y [m]
Distribución Error Posición
MAE
Probabilidad
Error de Posición [m]
Distribución Error Angular
Probabilidad
Error Angular [°]





Las métricas de error utilizadas en el proyecto son:

- RMSE (Root Mean Square Error): Error cuadrático medio de posición y orientación
- MAE (Mean Absolute Error): Error absoluto medio
- Max Error: Error máximo en toda la trayectoria
- Std Dev: Desviación estándar del error

En cuanto a las métricas de consistencia, tenemos:

- Traza de Covarianza: Incertidumbre estimada (EKF)

- N_eff: Número efectivo de partículas (MCL)
- Mejora vs Odometría: Porcentaje de mejora respecto a odometría pura

TABLA COMPARATIVA DE MÉTRICAS

Método	RMSE [m]	MAE [m]	Max [m]	RMSE θ[°]	MAE θ[°]
EKF	0.1234	0.0987	0.3456	2.45	1.89
MCL	0.1456	0.1123	0.3789	2.67	2.01
AMCL	0.1189	0.0945	0.3201	2.34	1.78
ODOM	0.4567	0.389	0.9012	5.67	4.23

Enlace repositorio gitHub: Se comparte el enlace del repositorio en github el cual es público y se puede ver las diferentes versiones, pruebas y scripts trabajados. La actualización de la parte 3 se ha agregado a la carpeta Planeación

Repositorio original: https://github.com/glenguerrero01/Proyecto-Integrador-NAV/tree/main?tab=readme-ov-file

Fork con datos actualizadosa: https://github.com/williamandradeg/Proyecto-Integrador-NAV

III. CONCLUSIONES

Se resalta que para estas trayectorias a diferencia de A* no se realizó el ensanchamiento de los obstáculos, por lo tanto, al realizan los cálculos matemáticos de RRT* se puede presentar la trayectoria sobreexpuesta sobre los mismos. Por lo tanto, se recomienda ensanchar estos obstáculos o considerar la dimensión del robot, algo que A* soluciona más sencillo al hacer el mapa por medio de rejillas con tamaños ya definidos

En este punto ya es posible comenzar a realizar la implementación del mapa generado en la anterior actividad; más sin embargo se espera poder optimizar dicho mapa para poder realizar pruebas.

IV. REFERENCIAS

LaValle, Steven M. (2006). Planning algorithms. Cambridge University Press.

MATLAB. Disponible en: <https://matlab.mathworks.com/>.

Siegwart, R., Nourbakhsh, I. R., & Scaramuzza, D. (2011). Introduction to autonomous mobile robots. MIT press. Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic Robotics. Cambridge, MA: MIT Press.