Proyecto 1

Planificación de Caminos Geométricos, Ejecución (Modo Autónomo) y Relocalización

Sebastian Guerrero – 202021249

Nicolás Rincón – 202021963

Método de Planificación

Para resolver el problema, implementamos un método de planificación basado en el algoritmo de frente de onda con 4 vecinos. Este enfoque permite encontrar un camino desde la configuración inicial (q_0) hasta la configuración final (q_f) dentro del escenario definido. Para comenzar, se importaron los escenarios desde archivos .txt, leyendo y organizando los datos en un diccionario que almacena las dimensiones, configuraciones inicial y final, obstáculos, y distancias para su fácil acceso posteriormente.

El espacio del escenario fue discretizado en una matriz con el doble de resolución respecto a las dimensiones originales, de modo que cada celda representa un área de 0.5x0.5 unidades. En esta matriz, los obstáculos se marcaron como áreas no transitables, mientras que q_0 y q_f se ubicaron en sus respectivas celdas. Esta discretización asegura precisión en la planificación y compatibilidad con el algoritmo de frente de onda.

El algoritmo comienza en la celda correspondiente a q_f, asignándole un costo inicial de 0. Desde este punto, se maraca iterativamente los cuatro vecinos. Cada celda vecina recibe un costo que es igual al costo de la celda actual más 1. Este proceso continúa hasta llenar la matriz con los costos asociados a cada celda, creando un mapa que permite determinar el costro relativo desde q_f a cualquier otra posición en el espacio.

Con la matriz de costos completa, se inicia en q_0 para construir el camino hacia q_f. En cada paso, se selecciona la celda vecina con el menor costo, lo que asegura que la trayectoria sigue la ruta más eficiente. Los índices de cada celda recorrida se registran para formar el camino, asegurando precisión en la representación del trayecto.

El camino generado se traduce a coordenadas en el espacio global tomando el centro de las celdas correspondientes. A continuación, se calculan los ángulos necesarios que debe tener el robot para que gire y se oriente correctamente entre cada par de puntos consecutivos, permitiendo movimientos en línea recta hacia el siguiente destino. Esto se realiza tomando la información de dos puntos, y representándolo como un triángulo con rectángulo, usando matemática básica es posible encontrar el ángulo deseado. Este resultado se formatea en el formato especificado y se exporta a un archivo .txt.

Además, se genera una visualización gráfica que muestra el escenario, los obstáculos, y el camino encontrado.

Aunque el método podría emplear el espacio de configuración (C-Space), en este caso lo simplificamos operando directamente en el espacio de trabajo (W-Space), ya que el robot, representado como un cuadrado que encaja en las celdas de 0.5X0.5 no requiere cálculos adicionales para manejar colisiones. Las restricciones se limitaron a los movimientos en las celdas adyacentes (4 vecinos), evitando diagonales para prevenir posibles colisiones en estos movimientos.

El resultado final del algoritmo incluye las coordenadas y orientaciones necesarias para que el robot navegue de q_0 a q_f, evitando obstáculos y respetando las restricciones del espacio de trabajo. Estas instrucciones permiten al robot moverse de forma eficiente, ajustándose a las condiciones del escenario y garantizando la seguridad durante la navegación.

Para mejorar el método, se podrían explorar varias optimizaciones y extensiones. Una opción es aumentar la resolución de la discretización, lo que permitiría una planificación más precisa en escenarios complejos, aunque a costa de un mayor consumo computacional. También se podría incorporar un modelo dinámico del robot para planificar trayectorias más realistas, considerando su velocidad y aceleración. Otra mejora sería la implementación del algoritmo con 8 vecinos, evaluando métodos para evitar colisiones en movimientos diagonales mediante un análisis más detallado de los obstáculos en el C-Space. Además, integrar métricas para priorizar celdas que se aproximan más rápido al objetivo o que se alejen de los obstáculos. Por último, incluir detección y manejo de cambios en tiempo real, como la aparición de nuevos obstáculos, haría el método más robusto para aplicaciones dinámicas y en entornos desconocidos.

	$q_f = (x, y, \theta)$	$q_{f-est} = (x, y, \theta)$	Diferencia posición (m)	Diferencia angular (°)
E1	(3.25,4.25,90.0)	(3.2807,4.177,89.814)	(0.0307,0.73)	0.186
E2	(3.25,4.25,90.0)	(3.2798,4.208,89.4723)	(0.0298,0.042)	0.5277
E3	(3.25,4.25,90.0)	(3.2687,4.2548,89.8992)	(0.0187,0.0048)	0.1008
E4	(3.25,4.25,90.0)	(3.2464,4.165,89.3399)	(0.0036,0.085)	0.6601
E5	(3.25,4.25,90.0)	(3.2667,4.2211,89.8501)	(0.0167,0.0289)	0.1499
E6	(3.25,4.25,90.0)	(3.2475,4.1858,89.6396)	(0.0025,0.0642)	0.3604

De la tabla se puede ver que el error es bastante bajo y el robot logra llegar al objetivo en todos los escenarios, el error en posición no supera los $0.03\,m$ y en los ángulos no supera los 0.7 °. En ambos casos este error es más que aceptable y se puede ver que el robot sigue el camino planeado de forma correcta.

Método de Relocalización

Para el método de relocalización, se asumió inicialmente que el robot estaba correctamente alineado con las paredes o superficies finales del entorno. A partir de esta suposición, se midieron las distancias hacia los obstáculos al frente y a la derecha utilizando los sensores del robot. Estas mediciones fueron comparadas con los valores teóricos obtenidos del modelo del escenario. La diferencia entre estas medidas, denominada error de localización, se utilizó para ajustar la posición estimada del robot. Este ajuste se realizó aplicando las variaciones detectadas en las distancias directamente sobre la posición teórica inicial, permitiendo una corrección incremental que mejora la precisión de la localización en tiempo real.

Entre las causas más comunes de errores en la localización del robot se encuentran las inexactitudes en los sensores debido a condiciones del entorno, como reflejos o interferencias, la acumulación de errores en la estimación de posición durante movimientos previos y las discrepancias entre el modelo teórico del entorno y el escenario real. En cuanto a la relocalización en nuestro caso, podemos ver que dependemos directamente de que el robot este orientado correctamente, si este tiene alguna desviación en este ángulo el método puede no ser muy exacto. Por otro lado, también esperamos que estemos realmente en un punto cercano al final y midiendo la distancia de las paredes deseadas. El método se basa en que las estimaciones son medianamente correctas y se llegó a un punto cercano al esperado.

	$q_{f-est} = (z, y, \theta)$	$q_{f-act} = (z, y, \theta)$	Diferencia posición (m)	Diferencia angular (°)
E1	$(3.2807, 4.177, 89.814)$	$(3.2813, 4.174, 90.0)$	$(0.0006, 0.003)$	0.186
E2	$(3.2798, 4.208, 89.4723)$	$(3.2812, 4.2035, 90.0)$	$(0.0014, 0.0045)$	0.5277
E3	$(3.2687, 4.2548, 89.8992)$	$(3.2689, 4.2548, 90.0)$	$(0.0002, 0.0)$	0.1008
E4	$(3.2464, 4.165, 89.3399)$	$(3.2477, 4.1599, 90.0)$	$(0.0013, 0.0051)$	0.6601
E5	$(3.2667, 4.2211, 89.8501)$	$(3.2671, 4.22, 90.0)$	$(0.0004, 0.0011)$	0.1499
E6	$(3.2475, 4.1858, 89.6396)$	$(3.2485, 4.182, 90.0)$	$(0.001, 0.0038)$	0.3604

Los resultados obtenidos muestran que el método de relocalización es altamente preciso, con diferencias en la posición que oscilan entre **0.0002 m y 0.0014 m**, lo cual es un margen extremadamente pequeño y adecuado para aplicaciones controladas. En términos angulares, las desviaciones varían entre **0.1008° y 0.6601°**, siendo también muy reducidas. Sin embargo, se observa que los casos con mayor error angular, como en E4, tienden a mostrar una ligera desviación adicional en posición, lo que sugiere una

correlación entre estos errores. En general, el método ajusta adecuadamente la posición estimada del robot, logrando una alta precisión tanto en ubicación como en orientación, aunque podrían surgir acumulaciones de errores en aplicaciones más prolongadas o en escenarios con mayor complejidad.

Por otro lado, en las simulaciones se puede ver que los robots llegan a posiciones bastante cercanas a las esperadas y se sigue el camino correctamente. A pesar de que en todas las simulaciones se logró una simulación exitosa, se tuvo algunos problemas en el escenario 4 ya que la rotación del robot no es realmente sobre su eje z, esta discrepancia en la rotación hacía que se tuviera una ligera discrepancia en la alineación al girar, por lo que el robot se desfasaba unos centímetros, en este caso específico este ligero cambio fue suficiente para que el robot quedara atrapado detrás de un obstáculo, sin poder seguir, esto solo sucedía algunas veces y en este escenario especifico, puede ser necesario arreglar el giro del robot para no tener estas discrepancias o ajustar el método de planeación para alejarse un poco más de los obstáculos así disminuyendo el impacto de este error.