ESCUELA DE INGENIERÍA INDUSTRIAL

Área de Conocimiento: Guiado y Navegación de Robots

Trabajo Apolo:
Navegación a través de una mina abandonada

Autores: Javier Fernández Pintor M24065 / Héctor Gordillo García M24193 / Martín Loring Bueno M24119

Titulación: Máster Universitario en Automática y Robótica

9 de enero de 2025

1. Introducción

El desarrollo de robots móviles capaces de desplazarse de manera autónoma en entornos diversos ha cobrado una gran relevancia en los últimos años debido a sus aplicaciones en sectores como la logística, la exploración, y los servicios. Este trabajo aborda el análisis y la simulación de un robot móvil en el entorno dado en esta asignatura, Apolo (entorno de simulación para Windows desarrollado en la UPM), y MATLAB.

El estudio se centrará en los componentes esenciales de un sistema de navegación robótica, incluyendo el sistema de locomoción, los sensores, los algoritmos de estimación del estado, los algoritmos de control de movimiento y los algoritmos de planificación de trayectorias. Cada uno de estos elementos será analizado y evaluado en un entorno simulado, proporcionando una base sólida para el diseño y la implementación de sistemas robóticos autónomos más complejos.

2. Objetivos

El objetivo de este trabajo es desarrollar una aplicación de navegación autónoma para un robot terrestre en un entorno definido, enfrentando desafíos asociados con la incertidumbre, control y planificación. Donde se buscará:

- Familiarizarse con los conceptos clave de navegación robótica, como sistemas de locomoción, selección y calibración de sensores, control de movimiento y planificación de trayectorias.
- Plantear y resolver un problema de navegación robótica mediante sistemas de locomoción diferenciales con modelos sujetos a incertidumbres.
- Seleccionar y diseñar los mejores sensores en cada momento para la localización y control reactivo.
- Simular diferentes algoritmos de localización y control, observando su funcionamiento y sus efectos.
- Integrar calibración y control a través de un entorno simulado utilizando las herramientas de Apolo y MATLAB.

3. Distribución de roles

La distribución de los roles asignada a los integrantes del equipo basada en sus contribuciones específicas y responsabilidades adicionales es la siguiente:

Héctor - Diseñador del Entorno, control PD y Responsable del Mapeado:

- Diseña el mapa y define el entorno en Apolo.
- Implementa el sistema de mapeado del entorno.
- Coordina las tareas del equipo para garantizar que las diferentes etapas del proyecto se completen en los plazos establecidos.
- Programa un control PD que permite el movimiento del robot
- Redacción informe y puesta en común con el equipo.

Javier - Desarrollador de Algoritmos y Analista:

- Implementa los algoritmos de odometría, estimación del estado y los controladores básicos para la navegación del robot (Kalman).
- Programa algoritmos de control reactivo para evitar obstáculos.
- Realiza el análisis gráfico de los resultados obtenidos, evaluando el comportamiento de los algoritmos y sistemas implementados, y genera las conclusiones finales del proyecto.
- Redacción informe y puesta en común con el equipo.

Martín - Responsable de Sensores, planificación y ejecución final:

- Selecciona y calibra los sensores más adecuados para las tareas de localización y control reactivo del robot.
- Modela la incertidumbre de los sensores y analiza su impacto en el desempeño del sistema.
- Diseña e integra el planificador de trayectorias, permitiendo que el robot siguiera una secuencia de puntos para alcanzar sus objetivos de manera eficiente.
- Realización de un archivo de ejecución final que enlace todo lo anteriormente realizado.
- Redacción informe y puesta en común con el equipo.

4. Elección del mapa

En la elección del entorno de trabajo, existen diversas áreas en las que los robots móviles tienen aplicación, ya sea porque su uso está plenamente establecido, se encuentra en una etapa emergente, o representa una posibilidad a futuro. Ejemplos de sectores donde los robots móviles han sido empleados durante años incluyen la exploración espacial, la agricultura o ámbito militar entre otros.

En este estudio, hemos considerado de especial interés el uso de robots móviles en minas abandonadas, una práctica que ya está siendo implementada actualmente debido a la creciente demanda de minerales específicos requeridos para la fabricación de diversos productos.

El uso de robots móviles en estas minas se justifica debido a los riesgos asociados a la exploración de estos entornos, que incluyen la presencia de obstáculos, derrumbes o cambios en el terreno causados por el abandono prolongado. Todos estos factores hacen que la intervención humana en esas áreas sea muy peligrosa, por lo que el uso de sistemas autónomos es cada vez más necesario para garantizar la seguridad y la eficiencia de las operaciones.

Figura 1: Ejemplo robot exploración de minas

En este contexto, el presente trabajo se enfoca en la utilización de un robot móvil autónomo que pueda desempeñar tareas de exploración de minas abandonadas. Este robot utilizará un mapa histórico de la mina generado a partir de datos previos como base para realizar la exploración. Durante todo el recorrido, el robot actualizará el mapa continuamente con información sobre posibles obstáculos, alteraciones o caminos estructurales, mejorando y actualizando en todo momento el mapa, haciéndolo más preciso.

4.1. Entorno generado

Para poder simular de manera eficiente el movimiento del robot en Apolo a lo largo de la mina, se ha creado un mapa del entorno a través de un lenguaje XML, simulando el mapa real sobre el cual se realizarán todas las pruebas y estudios. Este mapa tiene en cuenta todas las

características y dimensiones del terreno original de la mina, incluyendo pasillos, habitaciones vacías y obstáculos.

Con este mapa se busca ofrecer un entorno controlado y representativo de las condiciones reales de la mina, asegurando que las simulaciones sean los más realistas posibles. En la Figura 2 se aprecia el mapa en Apolo, donde se ha tratado de imitar los pasillos largos y estrechos de una mina, así como las galerías más grandes y voluminosas.

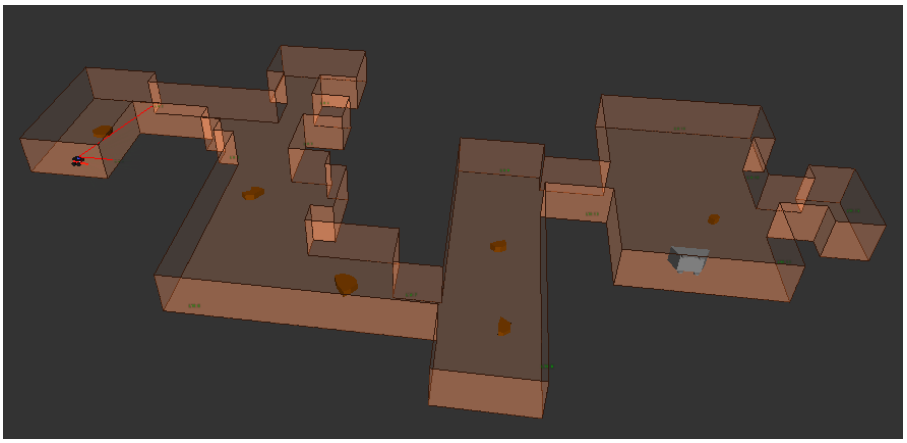


Figura 2: Mapa generado en el entorno de Apolo para simular la mina

Además de recorrer este terreno, el robot minero también debe esquivar obstáculos, como la piedra de la Figura 3. Se han añadido estos obstáculos simulando aquellos que en la realidad un robot de estas características se podría encontrar en una mina abandonada, como pueden ser piedras, o antiguos carros. Estos obstáculos se deberán esquivar mediante control reactivo, como se explicará con más detenimiento en el apartado sobre planificación, puesto que en una antigua mina los obstáculos aparecerán improvisadamente y no con planificación.

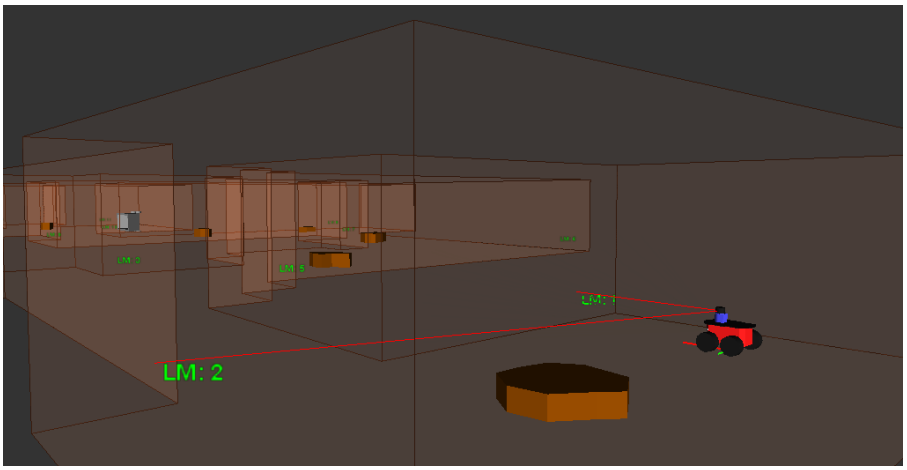


Figura 3: El robot debe esquivar obstáculos mediante control reactivo

5. Resolución del problema

En este proyecto se enfrentan varios desafíos técnicos para la navegación del robot como la estimación de la posición, control de movimiento o la planificación de la trayectoria, entre otros. A continuación se detallan las soluciones adoptadas y su implementación para lograr un desplazamiento efectivo del robot desde una posición inicial hasta su punto final objetivo.

5.1. Incertidumbre Odometría

La odometría es una técnica utilizada en la navegación de robots móviles para estimar la posición y orientación del robot en función de las mediciones de los motores y sus desplazamientos. Es una técnica muy utilizada, pero hay que tener mucho cuidado con ella, ya que es muy propensa a los errores debido a imprecisiones de las medidas.

Para solucionar esto, se estudia la incertidumbre de la odometría, realizando un gran número de mediciones y calculando la diferencia entre esas medidas, y la posición real del robot. Todo esto está recogido en el archivo *OdometriaInc.m* , diferenciando varias partes:

1. **Error de posición y orientación:**
   - Se busca calcular en todo momento la diferencia existente entre la odometría y los valores reales en términos de distancia y giro. Esto permite calcular los errores de la posición en el espacio bidimensional y en los ángulos de orientación del robot.
   - Los errores en x e y se calculan utilizando la fórmula euclídea de la distancia, lo que nos aporta un punto de vista de cuanto de alejadas están las estimaciones de la odometría de las posiciones reales.
   - El error angular se pretende ajustar dentro de un rango de [0, pi] calculado de igual manera que los errores de posición, a través de la diferencia entre los ángulos estimados con los reales.
2. **Análisis de resultados:**
   - Para cuantificar la incertidumbre en las mediciones odométricas, es necesario calcular la media (nos ofrece una idea de lo cerca, en promedio, que se encuentra la estimación de la realidad) y la desviación estándar (refleja la dispersión de los errores).
   - Se calcula también la varianza, que indica la medida más precisa de la incertidumbre asociada a cada dimensión y a los desplazamientos.
3. **Interpretación y uso de los resultados**
   - A partir de la medida del error se puede conocer si las estimaciones tienden a sobrestimar o subestimar la posición real del robot para corregir.
   - La desviación estándar y varianza nos ofrecen un conocimiento de lo dispersos que están los errores. Una desviación estándar baja indica que las estimaciones son muy consistentes y precisas

Inicialmente, se hacen una serie de pruebas para estudiar el movimiento y ver la presencia de errores en la odometría que presenta.

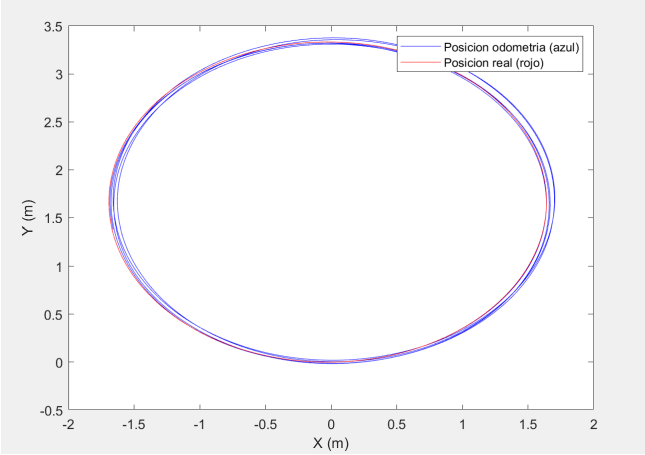


Figura 4: Simulación odometría vs realidad

En esta imagen aparece representada en azul la trayectoria que describe el robot según la odometría y en rojo la trayectoria real. Como se puede observar, aunque deberían ser lo más próximas posible, no lo son debido a errores en las mediciones.

Estos errores obtenidos, como la media, desviación estándar y la varianza, no solo nos ofrecen una visión de las mediciones con sus errores, sino que también tienen una aplicación crucial en la estimación de la posición y orientación del robot mediante el filtro de Kalman estudiado posteriormente.

Este filtro de Kalman es una herramienta de estimación que se utiliza para predecir la posición del robot a partir de medidas pasadas y de los errores obtenidos. Para ello se utiliza una matriz de covarianzas que representa la incertidumbre de las mediciones. Los valores de varianza calculados en este apartado de Incertidumbre de la odometría se emplearán pera definir esta matriz de covarianzas. Una baja incertidumbre de la odometría, indicará que el filtro confía más en las predicciones del modelo.

5.2. Incertidumbre y calibración de los sensores

El conjunto de sensores extereoceptivos presentes en el modelo también ha sido modelado para ajustar su incertidumbre de una manera correcta, todo presente en el archivo *SensoresInc.m*. La calibración de los sensores es crucial para mejorar la precisión y fiabilidad de las mediciones. Este proceso pretende medir y cuantificar las variaciones en las mediciones y adaptar el sistema para reducir estos errores.

Para satisfacer todas las necesidades para el correcto desplazamiento y detección de objetos se han utilizado los siguientes sensores con sus respectivos códigos para la calibración:

1. **Calibración Telémetro Láser:** El telémetro Láser 'LMS100' del sistema Apolo ha sido utilizado para obtener medidas de distancia y ángulos a partir de las balizas situadas en el entorno. Las medidas tomadas pueden variar notablemente de manera aleatoria, por lo que es importante realizar un gran número de medidas para obtener una vista adecuada del comportamiento de los sensores y evaluar su incertidumbre.

   Este proceso permite acumular suficientes datos de distancias y ángulos para calcular en todo momento las varianzas asociadas (varianza de la distancia y varianza de los ángulos) a cada medición. Las varianzas calculadas de las mediciones por parte de los sensores proporcionan una medida de la dispersión de los datos con respecto al valor ideal.

2. **Calibración sensores Ultrasónicos:** Aparte del telémetro láser se emplean unos sensores ultrasónicos para medir distancias a objetos. Para captar la variabilidad inherente al sensor se realiza un elevado número de mediciones, la acumulación de estos datos es importante para caracterizar la incertidumbre asociada a las mediciones ultrasónicas.

Tras obtener todas las mediciones con ambos sensores se busca obtener la varianza de las distancias, que ofrece una medida de la precisión de los sensores, lo cual es muy importante para garantizar que las medidas obtenidas sean fiables, utilizadas más adelante en el filtro de Kalman.

La combinación de estos sensores permite cubrir en todo momento y de manera eficiente un gran rango de aplicaciones en diferentes entornos. Mientras el láser permite realizar medidas a largas distancias de manera muy precisa, detectando las distintas balizas, los ultrasonidos aseguran una detección efectiva en distancias cortas, lo que es esencial en tareas críticas de navegación segura y detección de obstáculos (control reactivo).

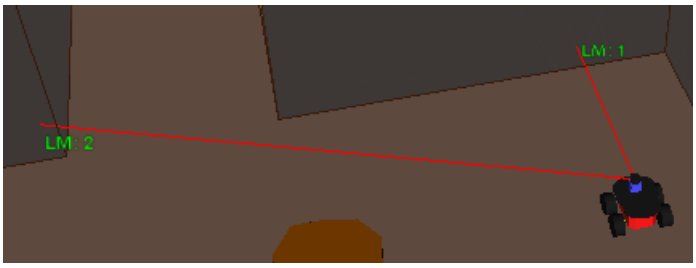


Figura 5: Con el telémetro láser el robot puede localizar balizas y orientarse

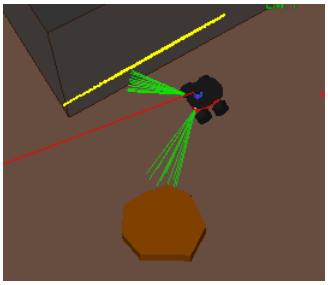


Figura 6: Los sensores ultrasónicos permiten al robot esquivar obstáculos cercanos

5.3. Estimación del estado (Kalman)

El filtro de Kalman es un algoritmo que permite mejorar la precisión de la posición y orientación del robot teniendo en cuenta las mediciones de ruido de los sensores y las predicciones del movimiento. En este apartado se explicará en que consiste principalmente este filtro y como se ha aplicado a nuestro robot para lograr un movimiento mucho más preciso.

El principal objetivo de este filtro es, a través de diferentes predicciones del movimiento del robot y las medidas de los sensores, corregir en cada momento la estimación de la posición y orientación, logrando un movimiento y localización mucho más eficiente.

Este filtro se inicializa con una serie de variables iniciales (archivo *KalmanEXT.m*)

- **Xk1:** Estado estimado
- **Pk1:** Matriz de covarianzas inicial
- **Qk:** Ruido del proceso
- **Rk:** Matriz de covarianzas de la medición de sensores.
- **LM:** Posiciones balizas

5.3.1. Etapa de predicción

La primera etapa del filtro de Kalman consiste en la predicción del estado del robot a través de su odometría, calculando la posición actual a través de las medidas pasadas, obteniendo X-k(estado nuevo), Ak(matriz de transición de estados) y Bk(matriz de control).

$$X_k = \begin{bmatrix} X_{k-1}(1) + \text{avance} \cdot \cos\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ X_{k-1}(2) + \text{avance} \cdot \sin\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ X_{k-1}(3) + \text{giro} \end{bmatrix} \tag{1}$$

$$A_k = \begin{bmatrix} 1 & 0 & -\text{avance} \cdot \sin\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ 0 & 1 & \text{avance} \cdot \cos\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$B_k = \begin{bmatrix} \cos\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) & -0{,}5 \cdot \text{avance} \cdot \sin\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ \sin\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) & 0{,}5 \cdot \text{avance} \cdot \cos\left(X_{k-1}(3) + \frac{\text{giro}}{2}\right) \\ 0 & 1 \end{bmatrix} \tag{3}$$

Una vez obtenidos estos valores, se calcula la nueva matriz de covarianzas Pk cuyo valor indica la incertidumbre y confianza sobre la predicción del estado obtenida.

$$P_k = A_k P_{k-1} A_k^T + B_k Q_k B_k^T \tag{4}$$

Este filtro se aplicará cuando se detecte más de una baliza, esto es debido a que la etapa de corrección se lleva a cabo en función de diferentes observaciones, si se quiere realizar una corrección fiable es necesario que haya al menos más de una baliza o medida. Si se detecta más

de una baliza, el filtro dispondrá de información suficiente para realizar una buena corrección. Cuanto más datos y medidas, más fiable será el sistema.

Se calcula la medida esperada (Zk') que representa un vector con tres elementos que representan el conjunto de mediciones esperadas basadas en la estimación actual del estado (x-k). Estas predicciones serán utilizadas para comparar con las mediciones reales y para actualizar la estimación del estado en etapas futuras.

5.3.2. Etapa observación

Esta etapa es la responsable de ajustar todas las estimaciones que se han realizado del estado a través de observaciones de los sensores, comparando las predicciones con las reales.

Se obtendrán dos valores:

- **Zk:** Vector de medidas del sensor láser (distancia y ángulo)
- **Hk:** Matriz jacobiana que describen la relación entre el estado del robot y las mediciones.

5.3.3. Comparación

En esta etapa se realizará el cálculo de la diferencia entre las medidas actuales en el instante k, y las predicciones, ajustándolas dentro de un rango [-pi,pi] obteniendo Yk.

$$Y_k = Z_k - Zk' \tag{5}$$

Además, se busca obtener una matriz que represente la incertidumbre de la diferencia entre la predicción y la medida real (Sk), y la ganancia de Kalman (Wk), representando cuanto confía el modelo en las predicciones.

$$S_k = H_k P_k H_k' + R_k \tag{6}$$

$$W_k = P_k H_k' S_k^{-1} \tag{7}$$

5.3.4. Corrección

Esta etapa consiste en ajustar la matriz de covarianza y la estimación del estado de manera definitiva, centrándose en los resultados de Yk y Wk.

$$X_k = X_k + W_k \cdot Y_k \tag{8}$$

$$P_k = (I - W_k \cdot H_k) \cdot P_k \tag{9}$$

5.4. Controlador

Para el correcto funcionamiento del movimiento del robot se implementaron diferentes estrategias de control para garantizar la navegación plenamente autónoma a través del entorno. Para ello se ha utilizado un controlador PD para controlar el movimiento y estabilidad y un controlador reactivo para la detección y evasión de los obstáculos.

5.4.1. Controlador PD

El controlador PD se utiliza para lograr que el robot se desplace correctamente hasta su objetivo, ajustando y corrigiendo velocidades y orientaciones, buscando la trayectoria más óptima, obteniendo en cada momento la velocidad angular y lineal más recomendable.

El controlador se utiliza para calcular el error entre el estado objetivo y el actual real, siendo el error principal la diferencia angular entre la orientación del robot y la dirección hacia el objetivo calculada a través de la tangente.

El código (ContPD.m) se puede dividir en:

- **Cálculo del error angular:** Inicialmente, se busca calcular el ángulo deseado objetivo (*anguloObjetivo*) para ser comparado con la orientación que ocupa el robot en ese momento (*estadoActual (3)*) y de esta manera obtener el error angular.

$$\text{ángulo\_deseado} = \arctan 2\left(\text{objetivo}(2) - \text{estado\_actual}(2), \text{objetivo}(1) - \text{estado\_actual}(1)\right) \tag{10}$$

$$\text{error\_angular} = \text{ángulo\_deseado} - \text{estado\_actual}(3) \tag{11}$$

- **Controlador PD para la velocidad:** Se ajusta la velocidad a través de:

$$\text{vel\_angular} = K_{\text{proporcional}} \cdot \text{error\_p} + K_{\text{derivativo}} \cdot \text{error\_d} \tag{12}$$

  El proporcional (*kP * errorPoporcional*) corrige el error actual, y el derivativo usa (*errorDerivado*) para anticiparse y suavizar las oscilaciones en la trayectoria.

  La velocidad angular obtenida tras este proceso se ajusta a través de unos límites definidos (*limite-w*) para evitar y suavizar los comportamientos bruscos.

- **Ajustar velocidad lineal calculada:** Tener en cuenta que cuando el robot se encuentra perfectamente alineado con su objetivo, la velocidad lineal debe ser la máxima y si hay algún error angular (robot no alineado con objetivo) se utiliza una velocidad más reducida, dando prioridad a la corrección de su posición para lograr un movimiento más eficiente.

Como resultado final y más importante se calcula si el robot ha llegado a su destino, comparando la distancia al objetivo calculada con el umbral establecido devolviendo un operador booleano indicando si se ha logrado llegar al punto deseado, terminando la simulación en caso afirmativo a través del (*EjecutableMina.m*)

5.4.2. Controlador reactivo

El control reactivo en este trabajo se ha implementado para lograr gestionar el movimiento del robot de forma autónoma utilizando la información que capta del entorno a través de sensores ultrasónicos.

A través de la función creada en *ContReactivo* se determinan las velocidades de giro y avance del robot basándose en las mediciones de los propios sensores en cada momento y detecta obstáculos presentes. Pudiendo dividirse en las siguientes partes:

- **Lectura inicial de los sensores e inicialización:** A través de funciones de apolo se obtienen las distancias medidas por cada uno de los sensores ultrasónicos (derecho, izquierdo y frontal) y se inicializan también las variables de las velocidades.

- **Búsqueda de obstáculos:** En todo momento se verifica si hay algún obstáculo detectado por los sensores. Un sensor detectará un obstáculo si la distancia medida es menor que 1 en el sensor frontal o menor de 0.5 en los laterales.

- **Evitar obstáculos:** En el caso de haber detectado un obstáculo se evita. Si el obstáculo se encuentra muy próximo al robot (medida menor que 0.2) se reduce la velocidad de avance (0.05 en nuestro caso)y se modifica la velocidad de giro dependiendo de donde este el objeto (derecha o izquierda) para poder evitarlo. Si el sensor izquierdo detecta el objeto muy próximo se gira hacia la izquierda con velocidad de giro negativa para esquivarlo (en este caso velocidadAngular = -0.4 / sensorIzquierdo)

  En caso de no detectar ningún obstáculo se sigue con la velocidad de giro y avance continua a 1.

5.5. Planificador

Para conseguir hacer las planificaciones en el mapa se debe comenzar por tener un fichero de mapa adecuado para realizar las mismas en MATLAB. Para ello, se ha comenzado tomando el mapa en dos dimensiones de la mina, que se puede ver en la Figura 7, en el que se pueden ver las limitaciones del mapa mediante los vértices y paredes del mismo.

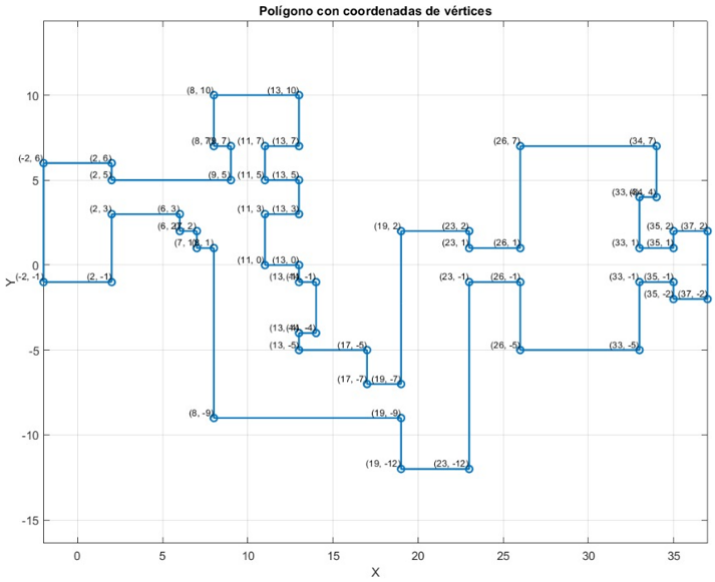


Figura 7: Visualización del mapa de la mina en 2D mediante polígonos

Con este mapa, se ha generado un mapa de celdillas de ocupación (Figura 8), que es una imagen formada por píxeles blancos y negros, en el que las celdillas blancas indican espacios libres, y las celdillas negras espacios con obstáculos. Para realizar esta conversión, se ha utilizado el programa de MATLAB *generarMapa*, el cual se entrega junto con los códigos de esta trabajo y cuyos comentarios explican en mayor detenimiento lo realizado en cada caso.

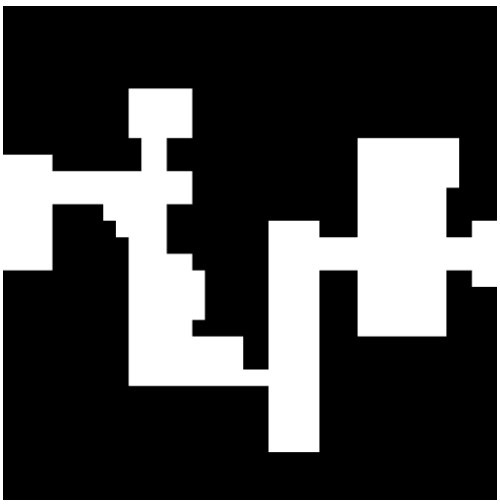


Figura 8: Mapa de celdillas generado para el mapa de la mina

En nuestro caso, y como se puede ver en la imagen previamente comentada, se ha tomado todo el interior de la mina como una zona libre de obstáculos. Esto se ha hecho así, puesto que se ha tratado de buscar la máxima realidad de nuestra aplicación, en la que se tenga un mapa antiguo de una mina que se quiere explorar, pero no se sepa su estado actual, por lo que es posible que existan diversos obstáculos que no sean previamente conocidos y que tenga que

esquivar fuera de la planificación, con el control reactivo explicado en el apartado anterior.

Para continuar con la planificación, se ha pasado este mapa a un mapa binario de ocupación, como el que se ve en la Figura 9, en el cual se han añadido ejes y rejillas. Sin embargo, este mapa no está en la misma escala que el mapa de Apolo, ni en el mismo eje de coordenadas, por lo que ese debe realizar una traslación y una reconversión de las coordenadas para poder mover el robot en MATLAB. Para esto, se ha utilizado otro fichero, en este caso $transformToMap1$, el cual tiene la función de convertir los puntos del mapa de ocupación de rejillas a coordenadas de Apolo. Es decir, se calcularán los puntos por los que debe pasar el robot en el mapa de MATLAB, y después se convertirán a puntos de Apolo.

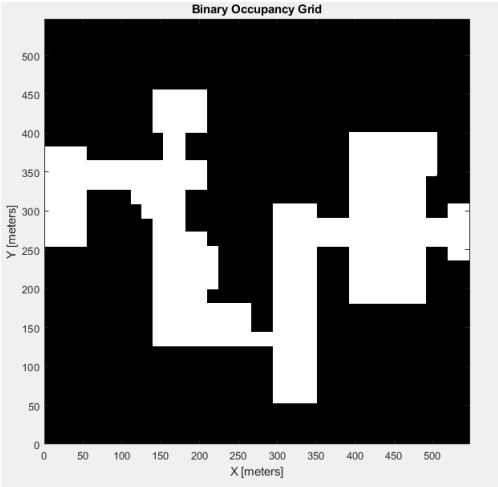


Figura 9: Mapa binario de ocupación de la mina

Una vez conocemos cómo funcionan los mapas con MATLAB y las conversiones necesarias, falta describir el algoritmo que nos permitirá deducir los puntos por los que debe pasar el robot para conseguir dirigirse desde el punto inicial al punto final, con el mapa dado. En nuestro caso, se ha utilizado el algoritmo RRT Estrella, que es uno de los vistos en esta asignatura durante el curso.

El algoritmo RRT ($Rapidly-exploringRandomTree$, por sus siglas en inglés) es un método de planificación de rutas. Es muy utilizado en robótica y simulaciones debido a su capacidad para explorar rápidamente un espacio complejo. Su funcionamiento se puede resumir en:

1. Se inicia el árbol, 2. Se genera un punto aleatorio. 3. Se encuentra el punto más cercano del árbol, 4. Se expande hacia ese punto, 5. Se comprueba si hay colisiones, 6. Si no hay colisión se añade, sino que se añade el punto más cercano sin colisión.

En el caso del algoritmo RRT Estrella, se hace de la misma forma, con la única diferencia de que es un algoritmo estrella, es decir, busca la solución óptima. Para la búsqueda de esta solución óptima se reorganiza continuamente el árbol, de forma que se busca si existen otras combinaciones de puntos en el árbol que hagan que los puntos se conecten con menor distancia.

Este es el algoritmo que se ha utilizado en este trabajo, y el cual permite encontrar rutas óptimas entre dos puntos indicados. En el caso de la prueba final que se ha enviado grabada junto con este trabajo, el robot se desplaza por toda la mina siguiendo la ruta indicada en la Figura 10.

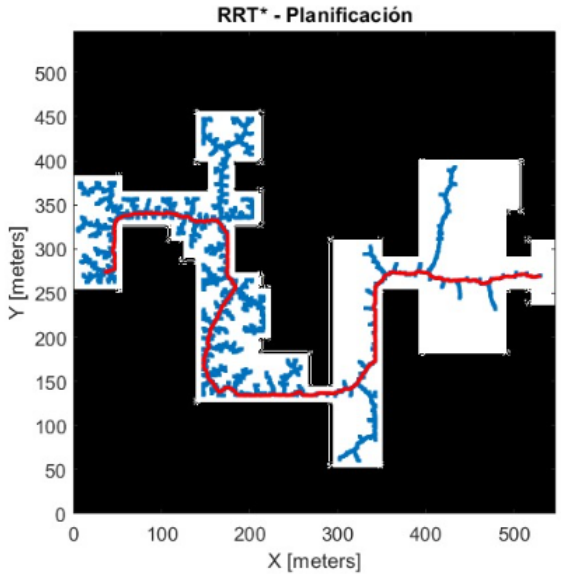


Figura 10: Ruta planificada mediante algoritmo RRT* para recorrer la mina

5.6. Ejecutable final

La última y más importante parte del código se centra en la creación de un ejecutable final que integra todos los componentes previamente programados, con el objetivo de simular el movimiento del robot en un entorno controlado y visualizar el resultado final de su navegación. Este ejecutable es resultado de combinar todas las funciones necesarias para lograr que el robot se mueva de forma autónoma en el entorno creado en Apolo, simulando su desplazamiento y alcanzando su objetivo.

Este ejecutable final (*EjecutableMina.m*) se ha estructurado de la siguiente manera:

- **Configuración parámetros iniciales:** Se establecen los parámetros iniciales más importantes como la frecuencia de actualización, tiempo máximo de simulación, las velocidades máximas lineares y angulares o las coordenadas iniciales del robot o los valores de Kalman entre otros parámetros.

  Con estos parámetros se posiciona el robot en las coordenadas deseadas iniciales y se inicializa la matriz de covarianzas y estados.

- **Planificación de la trayectoria:** Se determina cuál es mejor camino que debe seguir el robot para desplazarse desde su posición inicial a la posición objetivo. Esto se logra a través un algoritmo de planificación (*PlanificadorRRT.m*) a través de mapas generados que crea una secuencia de waypoints que debe seguir el robot para lograr llegar a su

destino. Si la planificación determinada no es viable, se transforman las coordenadas a un mapa nuevo para garantizar el desplazamiento.

En el momento que se necesite transformar las coordenadas a un mapa nuevo se utiliza la función *transforToMap1.m*, permitiendo la conversión de coordenadas ente mapas en función de un sentido determinado (de mapa 1 a 2 o inversa). Esta transformación de coordenadas se basa en función de una relación escalar entre X e Y (*scaleX* y *scaleY*) predefinidos, en nuestro caso:

$$\text{scale}_x = \frac{39}{547} \tag{13}$$

$$\text{scale}_y = \frac{32}{547} \tag{14}$$

Si *sentido==1*, la función convertirá las coordenadas de *xNew* y *yNew* del mapa 2 en las coordenadas originales de X e Y del mapa 1 (*xOriginal, yOriginal*).

$$x_{\text{original}} = \frac{x_{\text{new}} + 2}{\text{scale}_x} \tag{15}$$

$$y_{\text{original}} = \frac{y_{\text{new}} + 16}{\text{scale}_y} \tag{16}$$

En caso de *sentido==2* el proceso será inverso.

- **Movimiento del robot:** Se realiza y bucle de navegación en el que el robot se desplaza por el entorno detectando y siguiendo los waypoints mientras va estimando su posición. Esta es una de las partes más importante, ya que en él se utiliza el filtro de Kalman (*KalmanEXT.m*) que irá corrigiendo las estimaciones del robot en función de las medidas que van obteniendo los sensores a medida que se desplaza. Se utiliza también un control del PD (*ContPD.m*) para ajustar su velocidad y realizar el desplazamiento a través de la trayectoria calculada con la mayor precisión posible. En caso de detectar algún obstáculo, se activa el control reactivo (*ContReactivo.m*) para evitar colisionar, basado en el uso de los sensores láser.

Durante toda la simulación el entorno se va actualizando en tiempo real, visualizando el comportamiento del robot en el mapa creado, permitiendo observar claramente como se va desarrollando el proceso.

6. Comprobaciones y resultados

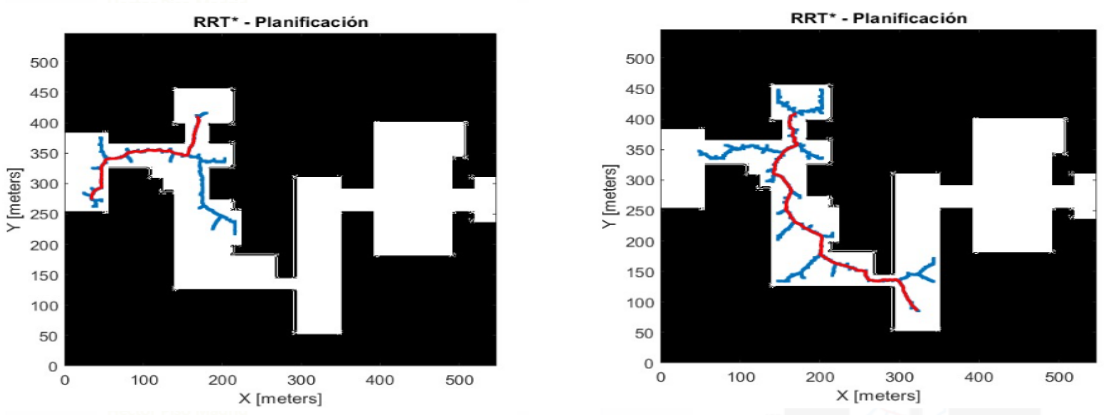
Para la comprobación de resultados se han realizado diferentes simulaciones en los que el robot ha recorrido diversas rutas según fuera conveniente.

Como se ha reflejado durante el desarrollo de esta memoria, nuestro robot es un robot minero, el cual debe entrar en minas abandonadas, recorrerlas y comunicar al grupo de ingenieros

responsables las condiciones de la mina. Algunas de estas condiciones fundamentales pueden ser la presencia de obstáculos, lo que puede ayudar a los operarios a la hora de recorrer la mina una vez que el robot ha compartido las regiones con objetos. Además, con el robot en la mina, se podrían medir otras condiciones, como el oxígeno en las diferentes zonas de la misma, para asegurar la seguridad de los operarios que fueran a entrar. Otras mediciones de seguridad del robot podrían ser relativas a los niveles de temperatura o radiación.

Para hacer esta primera medición de las condiciones de la mina, se ha decidido llevar al robot desde el punto inicial, que se puede considerar como punto de entrada a la mina, hasta el punto final, según se tiene en los planos. Este recorrido es el que se comentó previamente en la Figura 10. y en este el robot es capaz de recorrer todos los pasillos y galerías hasta llegar al final, esquivando los obstáculos mediante control reactivo. Esta simulación se encuentra en uno de los vídeos enviados en la entrega del trabajo.

Se ha hecho otras dos simulaciones, en las cuales el robot se ha desplazado desde puntos iniciales a otros, según se le indicaba. Esto se puede ver en la imagen 11, y son las simulaciones de los otros dos vídeos entregados. En este caso, se ha tratado de probar el movimiento del robot por la mina en otras circunstancias, probando que podría dirigirse a los puntos que necesitáramos. Por ejemplo, en el primer caso (Figura 11a) se ha dirigido el robot a la galería primera superior, puesto que puede ser interesante estudiar primero las condiciones de esta. Tras esto, desde este punto anterior, se ha calculado la ruta hacia la galería inferior. De esta forma, se podría recorrer la mina no solo de principio a fin, sino pasando por puntos estratégicos donde medir condiciones, o incluso esperar a que una vez se asegure la seguridad, pasen los operarios.



(a) Primera ruta (b) Segunda ruta

Figura 11: Rutas generadas mediante algoritmo RRT* en diversas simulaciones

7. Conclusiones y posibles mejoras

El desarrollo e implementación de un sistema de navegación autónoma para robots desarrollado en este proyecto ha permitido abordar diversos desafíos relacionados con el control y estimación de la posición en entornos dinámicos. A partir de diversas técnicas como el filtro de

Kalman para la estimación de los estados, el control reactivo para la detección de obstáculos o un controlador PD, se ha logrado un desempeño satisfactorio en la movilidad del robot.

El filtro e Kalman ha demostrado ser una herramienta muy eficaz para mejorar la precisión de las estimaciones de la posición en cada momento a pesar de la presencia de ruidos. A través del controlador PD se ha logrado ajustar de manera adecuada la velocidad y dirección del robot en cada momento, garantizando un comportamiento fiable y eficiente a lo largo de su trayectoria. El control reactivo ha permitido evadir con perfección los obstáculos del entorno, consiguiendo que el robot tome decisiones en poco tiempo en función de las medidas de los sensores.

En resumen, este proyecto ha demostrado que la gran eficiencia de combinar múltiples técnicas de planificación, control y estimación para lograr un desplazamiento eficiente y autónomo en la navegación de robots.

Una mejora significativa sería integrar las rocas en el mapa de ocupación directamente, lo que optimizaría la navegación. Sin embargo, en entornos como una mina abandonada donde puede haber desprendimientos constantes de rocas, el número de obstáculos es impredecible y varían constantemente. Para poder solucionar esto de una manera más eficiente se podría incorporar sensores adicionales o algoritmos de aprendizaje automático, que permitan al robot actualizar su mapa y trayectoria constantemente, lo que garantizaría una navegación mucho más segura y eficiente.

Una posible mejora de este proyecto, pero que no podemos abordar por falta de tiempo, sería un algoritmo que genere un mapa nuevo conforme el robot se desplaza, de forma que al mapa inicial se le añadan los obstáculos que encuentra. Este mapa se podría generar fuera de la mina y permitiría a los operarios tener una idea más clara de la mina abandonada.

También se podrían haber añadido más sensores ultrasónicos en el robot, lo que mejoraría la calidad del control reactivo, esquivando los obstáculos con mayor previsión y rapidez. Esta es una línea que se podría estudiar, si bien con los tres sensores utilizados se consiguen buenos resultados.

Por último, se podrían estudiar otros algoritmos de planificación para la ruta que sigue el robot móvil a través de la mina. Si bien el campo de la planificación es muy amplio, como se ha visto durante la asignatura, sería posible probar con otros algoritmos y ver las mejoras que ofrecen. Algunos de estos algoritmos vistos en la asignatura pueden ser: *Fast Marching*, PRM o algoritmos discretos como A*.