Universidad de Alcalá

Práctica 2

Navegación local y global

Patricia Cuesta Ruiz

Daniel Alonso Arza

Salvador Sobrino Solórzano

CONTENIDO

1. INTRODUCCIÓN

A lo largo de este documento, los alumnos Salvador Sobrino Solórzano, Patricia Cuesta Ruiz y Daniel Alonso Arza, van a llevar a cabo una explicación detallada y concisa del desarrollo y aplicación, tanto en simulación como con el robot real, los conceptos de navegación local y global estudiados a lo largo del Tema 3 de la asignatura.

En primer lugar, comenzaremos hablando de la planificación local (VFH), comenzando por la definición del algoritmo VFH, describiremos detalladamente los parámetros a configurar en el algoritmo mencionado y acto seguido, hablaremos de los parámetros de ajuste y las técnicas de depuración de este algoritmo. Una vez finalizada esta parte, mostraremos cómo ha sido el desempeño del algoritmo en simulación y con el robot real, aportando en ambos casos archivos de vídeo.

Tras esto, repetiremos este mismo proceso con la planificación global (PRM), comenzaremos definiendo el algoritmo PRM, los parámetros a configurar y los parámetros de ajuste y los métodos de depuración disponibles para este algoritmo. Como hicimos con el VFH, una vez acabemos de describir lo citado anteriormente, mostraremos el desempeño de este algoritmo en el simulador y también con el robot real, mostrando en ambos casos archivos de vídeo en los que mostraremos el funcionamiento.

A continuación, describiremos los problemas a los que nos hemos ido enfrentando durante el transcurso de la práctica, los cuales, fueron todos y cada uno de ellos solventados correctamente.

Por último, y para finalizar con este documento, hemos incluido la bibliografía que hemos empleado y que nos ha resultado muy útil para lograr un correcto funcionamiento de todo lo requerido para la realización de esta práctica.

2. PLANIFICACIÓN LOCAL (VFH)

2.1. 2.1. ESTUDIO DEL ALGORITMO

2.1.1. Describir el funcionamiento del algoritmo VFH disponible en la Robotics Toolbox de Matlab.

El algoritmo de histograma de campo vectorial (VFH) calcula las direcciones sin obstáculos para un robot en función de las lecturas del sensor. Estas se utilizan para calcular histogramas de densidad polar para identificar la ubicación y proximidad de obstáculos. Según los parámetros y umbrales especificados, estos histogramas se convierten en histogramas binarios para indicar las direcciones válidas para el robot.

El algoritmo VFH tiene en cuenta el tamaño del robot y el radio de giro para generar una dirección con la cual el robot evite obstáculos y siga una dirección objetivo.

2.1.2. Describir los parámetros que se pueden configurar en dicho algoritmo.

Algunos de los parámetros, que consideramos más importantes, que se pueden configurar son:

- **RobotRadius**: especifica el radio del círculo más pequeño que puede rodear todas las partes del robot. Este radio asegura que el robot evite obstáculos en función de su tamaño.
- **SafetyDistance**: distancia que agrega un factor de seguridad al navegar.
- **MinTurningRadius**: radio de giro mínimo para el robot que se desplaza a la velocidad deseada.
- **DistanceLimits**: rango de distancia que desea considerar para evitar obstáculos. Los límites son un vector de dos elementos, [inferior superior]. El límite inferior se usa para ignorar lecturas del láser que se crucen con partes del robot, la poca precisión del sensor a distancias cortas, o ruido del sensor. El límite superior es el rango efectivo del sensor o la máxima distancia que se quiera considerar.

2.1.3. ¿Cuáles son los parámetros de ajuste de la función de coste?

Los parámetros de ajuste de la función de coste son:

- **CurrentDirectionWeight**: peso de la función de coste que indica la dirección objetivo.
- **PreviousDirectionWeight**: peso que indica la dirección previa.
- **TargetDirectionWeight**: peso que indica la dirección actual.

Cambiar estos pesos afecta la capacidad de respuesta del robot y cómo reacciona a los obstáculos. Para hacer que el robot se dirija hacia su ubicación objetivo, establezca TargetDirectionWeight más alto que la suma de los otros pesos.

2.1.4. *¿Qué método está disponible para ayudar en la depuración del funcionamiento del algoritmo?*

Podemos visualizar las propiedades y parámetros del algoritmo utilizando la función *show*. Este método muestra el gráfico de densidad polar y el histograma binario enmascarado. También muestra los parámetros del algoritmo y la dirección de dirección de salida para el VFH. Permitiéndonos ajustar los parámetros para crear un prototipo de la aplicación para evitar obstáculos.

2.2. PRUEBAS EN SILUMACIÓN

2.2.1. *Comportamiento tipo wander. Script wander_vfh.m*

https://youtu.be/CVf9il97E0s

Vídeo wander_vfh

2.2.2. *Comportamiento wander y localización. Script wander_localiza.m*

https://youtu.be/KtSUO3KYNyM

Vídeo wander_localiza

2.3. PRUEBAS CON EL ROBOT REAL

No se ha realizado pruebas en concreto del funcionamiento del VFH en robot real, todo se ajusta, explica y muestra en el apartado 3.3 junto con lo pedido en el apartado 3.

3. PLANIFICACIÓN GLOBAL (PRM)

3.1. ESTUDIO DEL ALGORITMO

Para comenzar a trabajar con el planificador global, se pide:

3.1.1. Describir el funcionamiento del algoritmo PRM disponible en la robotics toolbox de Matlab.

El algoritmo de planificación PRM (Probabilistic Roadmap Method) solventa el problema de planificación en entornos limitados, crea un camino entre una configuración inicial del robot y una configuración final a la par que esquiva colisiones. Básicamente, este algoritmo está basado en coger muestras aleatorias del entorno del robot, clasificar las zonas como libres u ocupadas y usar un planificador local para conectarlas. Se añaden las configuraciones de inicio y objetivo y se emplea un algoritmo de búsqueda gráfica al gráfico resultante para crear una ruta entre las dos configuraciones.

El algoritmo PRM consta de dos fases: una primera de construcción y la segunda de búsqueda.

En la primera fase, se construye el mapa, aproximando los movimientos que se podrán realizar en el entorno. En primer lugar, creamos una configuración aleatoria. Después, se conecta a algunos vecinos o configuraciones cercanas, normalmente a los k nodos más próximos o a todos los que se encuentran a una distancia inferior de una distancia predeterminada. Se continúan añadiendo nodos y conexiones hasta que el mapa es lo suficientemente denso.

En la fase de búsqueda, los nodos inicial y final son conectados al gráfico y se obtiene el camino.

3.1.2. Describir los parámetros que se pueden configurar en dicho algoritmo.

Los parámetros que se pueden configurar son:

- **ConnectionDistance**: distancia máxima entre dos nodos conectados.
- **Map**: representación del mapa.
- **NumNode**: número de nodos en el mapa.

3.1.3. ¿Cuáles son los métodos que sirven para actualizar y recalcular la planificación? ¿Cuándo resulta interesante emplearlos?

- **mobileRobotPRM**

Cuando se utiliza el objeto y se modifican las propiedades, con cada nueva llamada de función, el objeto desencadena los puntos de hoja de ruta y las conexiones que se van a volver a calcular

- **findpath**

Dado que volver a calcular el mapa puede ser intensivo desde el punto de vista informático, puede reutilizar la misma hoja de ruta llamando a diferentes ubicaciones inicial y final.

- **simpleMap.mat**

Se carga el mapa, desde un archivo como una matriz lógica y cree una cuadrícula de ocupación.

```
load('exampleMaps.mat')
map = binaryOccupancyMap(simpleMap,2);
```

Se crea una hoja de ruta. Los nodos y las conexiones pueden tener un aspecto diferente debido a la colocación aleatoria de nodos.

```
prm = mobileRobotPRM(map,100);
show(prm)
```

- **update**

Llama o cambia un parámetro para actualizar los nodos y las conexiones.

```
update(prm)
show(prm)
```

El algoritmo PRM vuelve a calcular la ubicación del nodo y genera una nueva red de nodos.

3.2. PRUEBAS EN SIMULACIÓN

3.2.1. Comportamiento wander_localiza_planifica

https://youtu.be/7U7dGp-TKpE

Vídeo wander_localiza_planifica

3.2.2. Comportamiento wander_localiza_planifica_controla

https://youtu.be/WKh-oSk2klc

Vídeo wander_localiza_planifica_controla

3.2.3. Comportamiento navegación_total

https://youtu.be/dM93Hq5j_dg

Vídeo navegación_total

3.3. PRUEBAS CON EL ROBOT REAL

Una vez validados los algoritmos en simulación, se pide:

3.3.1. Emplear la mejor configuración de parámetros obtenida en el apartado de simulación para comprobar el funcionamiento en modo real.

3.3.2. ¿Es suficientemente realista la simulación como para validar dichos parámetros en modo real?

No es del todo realista la simulación, ya que en la simulación nos encontramos en unas condiciones ideales y a la hora de llevar a cabo las pruebas con el robot real, estas condiciones no son las óptimas, ya sea por problemas de conexión vía Wi-Fi, por obstáculos, tales como personas caminando por el pasillo... Lo cual, se nos presentaba una casuística bastante favorable para las pruebas con el robot real, pero no tan ideales como las que encontrábamos

en el entorno de la simulación, lo cual, nos llevó a ajustar la configuración de parámetros para lograr el mejor funcionamiento posible con el robot real.

3.3.3. En caso de no ser suficiente, ajustar los parámetros en modo real.

Respecto a las pruebas hechas en la simulación ajustamos las propiedades tanto del VFH como del PurePursuit.

Tras muchas pruebas en el laboratorio con el robot real, iba muchísimo mejor que lo hecho en simulación e introdujimos algunos de los cambios hechos en el script para el robot real en el script de la simulación y mejoraron mucho los resultados (apartado 3.2.3).

Uno de los mayores cambios que hicimos fue que el robot es capaz de dar la vuelta y volver a la posición inicial sin problemas y esquivando obstáculos (vídeos del apartado 3.3.4), cosa que con los valores anteriores no era posible.

3.3.4. Vídeos

https://youtu.be/2_zA5M2GnGQ

Vídeo navegación_total.m (MATLAB)

https://youtu.be/vGHh_qpd19g

Vídeo navegación_total.m (Robot real)

4. PROBLEMAS

Nos surgió un problema con la ruta trazada ya que no se nos mostraba en una de las ventanas emergentes (*figure*). Esto era debido a que el parámetro *ConnectionDistance* tenía un valor demasiado pequeño, por lo que al aumentar dicho valor se solucionó el problema y se mostraba en el *figure* correspondiente la ruta planificada.

Tuvimos otro problema a la hora ejecutar el código de 'navegación_total.m' tanto en la simulación como en el robot real, ya que este oscilaba de un lado a otro. Esto lo resolvimos aumentado el valor del parámetro *LookAHeadDistance*. A pesar de haber aumentado dicho valor, seguía haciendo oscilaciones, lo cual era debido a la "importancia" que le dábamos a las velocidades angulares del VFH o del PurePursuit. Tras muchas pruebas, llegamos a la conclusión de que la velocidad angular resultante del PurePursuit debe tener más importancia que la del VFH.

Por último, tuvimos otro percance al utilizar el robot real en la zona inicial del pasillo ya que la red Wi-Fi no llegaba bien a esa zona y generaba retrasos de unos 5 segundos aproximadamente a la hora de enviar los datos de los sensores. Nos planteamos también que pudiera ser problema de las conexiones de Windows 11 porque en el ordenador en el que se ejecutaba todo se instaló la última versión de Windows y no se sabía si era eso lo que provocaba los problemas. Finalmente, se solucionó ejecutando todo desde el laboratorio y no desde el pasillo como hicimos inicialmente, por lo que dedujimos que el problema era de la conexión Wi-Fi.

5. BIBLIOGRAFIA

https://es.mathworks.com/help/robotics/ug/vector-field-histograms.html

https://es.mathworks.com/help/robotics/ref/robotics.vectorfieldhistogram-systemobject.html?searchHighlight=robotics.vectorFieldHistogram&s_tid=doc_srchtitle

https://es.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html

https://es.mathworks.com/help/robotics/ref/robotics.prm-class.html