Práctica 1 - Entrega 2: Navegación local sin y con mapa

Antonio Álvarez Caballero
Alejandro García Montoro
5º Doble Grado Ingeniería Informática y Matemáticas
analca3@correo.ugr.es agarciamontoro@correo.ugr.es

Índice

1. Resumen

Esta memoria describe brevemente el trabajo realizado para la segunda entrega de la primera práctica.

Se han desarrollado únicamente las tareas de la navegación local sin mapa:

- Se ha aumentado el campo de visión del escáner láser del robot analizando el código y modificándolo convenientemente.
- Se ha implementado una técnica para evitar la detención del robot cerca del objetivo controlando el módulo del vector atractivo.
- Se ha corregido el comportamiento suicida del robot controlando el módulo de la velocidad lineal.
- Se ha diseñado una técnica de mínimos locales con aplazamiento de objetivos y control local de la posición.
- Se ha reducido el comportamiento oscilatorio del robot con técnicas aritméticas que suavizan el movimiento.

La siguiente sección describe con detalle cada una de estas técnicas.

2. Ejercicio 1

2.1. Aumento del campo de visión

Para el aumento del campo de visión se analizó primero la definición del escáner láser en el archivo *.world* que describe al robot.

Nos aseguramos entonces de que el campo *fov* de la definición *sensor* fuera al menos 270.

Sin embargo, esto define únicamente el rango físico del láser, no los datos que de él consideramos. Así, recalculamos las constantes *MIN_SCAN_ANGLE_RAD* y *MIN_SCAN_ANGLE_RAD* definidas en *myPlannerLite.h* para que reflejaran nuestras necesidades: definimos la primera a -135 grados y la segunda a 135, teniendo así un total de 270 grados.

2.2. Detención más rápida cerca del objetivo

El módulo del componente atractivo de la fuerza tiende a cero cuando el robot se acerca al objetivo; esto ocurre porque depende inversamente de la distancia a la que se encuentra de él.

La solución a este problema pasa entonces por controlar este módulo, comprobando su valor y acotándolo por debajo: no queremos que sea menor que un valor prefijado. Así conseguimos que el robot no deje nunca de moverse a un ritmo aceptable.

La idea es entonces cambiar las fórmulas del componente atractivo del vector:

$$\Delta x = \alpha(d - r)cos(\theta)$$
$$\Delta y = \alpha(d - r)sen(\theta)$$

por las siguientes:

$$\Delta x = \alpha c_{d,r} cos(\theta)$$
$$\Delta y = \alpha c_{d,r} sen(\theta)$$

donde $c_{d,r} = max(d - r, 1)$.

Así, la componente atractiva del vector será siempre, como mínimo, la siguiente:

$$\Delta x = \alpha cos(\theta)$$
$$\Delta y = \alpha sen(\theta)$$

Esto es, el módulo mínimo de la componente atractiva del vector será el módulo del vector anterior, esto es, α^2.

2.3. Corrección del comportamiento suicida

La causa del comportamiento suicida es de nuevo debida al módulo del vector de dirección; en concreto de la componente repulsiva. Al ser la velocidad lineal el módulo

del vector de dirección —que surge de sumar las componentes atractiva y repulsiva—, esta tenderá a infinito cuando la distancia entre el robot y un obstáculo tienda a cero.

Así, aunque el sentido del vector resultante sea contrario al obstáculo, la velocidad lineal será muy grande y no dejará tiempo al robot para girar.

Por tanto, para solucionar este comportamiento simplemente hemos acotado la velocidad máxima del robot. Para determinar qué valor poner como cota, hemos analizado la ecuación de la velocidad lineal:

$$v = \sqrt{\Delta_x^2 + \Delta_y^2} \tag{1}$$

donde Δ_x y Δ_y son las componentes del vector resultante —suma de las componentes atractiva y repulsivs—:

$$\Delta = \Delta_{atractivo} + \Delta_{repulsivo}$$

En la ecuación 1 podemos que la velocidad lineal máxima del robot en ausencia de obstáculos —esto es, cuando $\Delta_{repulsivo} = 0$— es el máxima del módulo de la componente atractiva. Este máximo se alcanza cuando el robot está fuera del radio y la extensión del campo atractivo; es decir, el máximo está regido por las ecuaciones

$$\Delta_{atractivo_x} = \alpha \cdot s \cdot cos(\theta)$$
$$\Delta_{atractivo_y} = \alpha \cdot s \cdot sen(\theta)$$

Evidentemente, el módulo máximo de este vector será $(\alpha \cdot s)^2$, así que esta parece una opción razonable para acotar la velocidad máxima cerca de un obstáculo. Sin embargo, después de pruebas empíricas decidimos tomar como valor máximo la raíz cuadrada de este valor; es decir, tomamos como velocidad lineal el siguiente valor:

$$v = min(\sqrt{\Delta_x^2 + \Delta_y^2}, \alpha \cdot s)$$

2.4. Salida de mínimos locales

Lo primero ha sido determinar la siguiente cuestión: ¿Cómo sabemos que estamos atascados en un mínimo local? Nosotros lo determinamos si no salimos de un radio de 0,5 unidades de distancia durante 5 segundos —25 iteraciones a 5Hz—. Esto lo hacemos almacenando la posición anterior del robot. La mantenemos siempre que la distancia a dicha posición sea menor que 0,5. En cuanto hay un cambio mayor de dicho umbral, actualizamos la posición anterior para en la siguiente iteración volver a comprobar la diferencia de distancia.

Una vez se ha comprobado que el robot está *encerrado* en una región muy pequeña del mapa, le mandamos un objetivo con una posición aleatoria del mapa, gestionando la cancelación del objetivo anterior en el servidor y la creación del nuevo en el cliente. Este nuevo objetivo lo mantenemos durante 10 segundos, que es un tiempo viable para salir de una esquina, y después de eso volvemos a mandarle el objetivo original.

Si se diera el caso de que la posición aleatoria del mapa está demasiado cerca, o bien el tiempo de salida se ha establecido demasiado corto, no hay problemas: al volver a

comprobarlo, se volvería a mandar un objetivo aleatorio y se volvería a intentar. Como nuestro objetivo es salir de un mínimo local, esto nos asegura que en algún momento saldrá y será capaz de llegar al objetivo.

2.5. Reducción de oscilaciones

La causa de las oscilaciones en el movimiento del robot se debe a la componente repulsiva: que sufre variaciones bruscas en la dirección al alejarse y acercase a los obstáculos.

Para resolver este problema hemos ideado lo siguiente: como queremos suavizar el movimiento, aplicamos la media a las dos últimas componentes repulsivas calculadas

$$v_{final} = \frac{v_{anterior} + v_{actual}}{2}$$

Así se reduce la diferencia entre las iteraciones anteriores y la actual, uniformizando el movimiento errático del robot al evitar obstáculos.