DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Técnicas de los Sistemas Inteligentes. Curso 2016-17. Práctica 1: Robótica

Entrega 2: Ejercicios prácticos para detectar una frontera

Objetivo

El objetivo de los siguientes ejercicios es servir como guía a la implementación de un explorador de fronteras. La exploración basada en fronteras es una técnica que permite a un robot descubrir de forma autónoma un mapa de su entorno, inicialmente desconocido. Estos ejercicios permitirán hacer las etapas básicas para implementar un explorador basado en fronteras. Se asume que los métodos y variables referenciados en los enunciados de los ejercicios se basan en los métodos y datos miembro definidos en la clase *FrontierExplorer* que se encuentra en el fichero frontier_explorer_lite.h que está en el fichero mi_mapeo.zip en PRADO. Es recomendable crear un paquete denominado mi_mapeo. Descargar el paquete mi_mapeo.zip en PRADO, compilarlo y ejecutar los launch y nodos según las instrucciones explicadas en clase de prácticas y en la propia descripción del fichero en PRADO.

Ejercicios

1. *Giro de 360º.*

Implementar el método *gira360()* para que el robot gire dos veces dos vueltas completas. Compilar el paquete, ejecutar y comprobar que funciona. Observar cómo se va modificando el mapa conforme el robot gira.

2. Inflación de zonas de obstáculos

Modificar la función *getmapCallBack* para que, cada vez que encuentre una celda en el mapa recibido con obstáculo (el valor de la celda en este caso sería ***100***), rellene con obstáculos un entorno centrado en la celda de 1 metro cuadrado. Asumir que la resolución del mapa es de 0.05, es decir, cada celda representa un cuadrado de 5 cm de lado. Implementar y utilizar para ello el método rellenaObstaculos (int cell_x, int cell_y), que modifica el mapa *theGlobalCm* rellenando obstáculos alrededor de una celda dada con coordenadas cartesianas (x,y). Tener en cuenta que las celdas de la matriz que representa el mapa se acceden con el operador [fila][col], donde *fila* representa una fila de la matriz y *col* representa una columna de la matriz.

Para probar que funciona compilar, ejecutar y observar el contenido del fichero "grid.txt" que se genera en el directorio donde se haya ejecutado el nodo.

3. Etiquetar celdas frontera

Implementar la función *labelFrontierNodes()* de manera que inserte en la lista *frontera (*que es una variable de la clase *FrontierExplorer*) las coordenadas reales de los puntos del mapa que estén rodeados de alguna celda desconocida (cuyo valor sea -**1**). Hacer uso del método someNeighbourIsUnknown(int x, int y).

Tener en cuenta que la posición (x,y) en coordenadas cartesianas del mapa está representada en la celda [y][x] de la matriz *theGlobalCm* . Además, debido al proceso de discretización del mundo real, mediante descomposición en celdas, cada posición (wx, wy) en el mundo real asociada a la posición (x,y) del mapa se puede obtener de la siguiente forma:

wx = x*resolucion – origen_x
wy = y*resolución – origen_y

donde *resolución* es la resolución del mapa y *(origen_x, origen_y)* son las coordenadas de origen del mapa. Tener en cuenta que la resolución y origen del mapa se pueden consultar en la variable cmGlobal de la clase FrontierExplorer.

4. Seleccionar un nodo (celda) de la frontera

Implementar la función selectNode (nodeOfFrontier &selectedNode) para que devuelva un nodo de la frontera seleccionado siguiendo algún criterio (distancia mínima a la posición actual del robot, o distancia máxima, o aquél que tenga la mayor zona desconocida en su entorno cercano, …).

C/ Periodista Daniel Saucedo Aranda s/n, ETSI Informatica, 18071 Granada Tlf: +34.958.244019 Fax: +34.958.243317