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 para llevar a cabo los primeros pasos de 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 la detección de una frontera y seleccionar un punto de la frontera siguiendo algún criterio de selección. 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_stage.zip** en PRADO. Es recomendable **leer en primer lugar las instrucciones** en PRADO sobre como instalar y ejecutar el paquete ***mi_mapeo_stage***. Una vez leídas estas instrucciones de instalación y ejecución, leer las descripción del problema de exploración basada en fronteras y **explicación del código fuente** en el documento "Explicación_codigo_fuente.pdf".

Ejercicios

1. *Giro de 360º.*

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

2. Inflación de zonas de obstáculos

Modificar la función *getmapCallBack* (implementada en el fichero "*frontier_explorer_lite.cpp")* para que, cada vez que encuentre una celda que representa un obstáculo en el mapa recibido en el argumento ***msg*** (el valor de la celda en este caso sería ***100***), rellene con obstáculos un entorno de 1 metro cuadrado centrado en esa celda. 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*.

Para comprobar y depurar el etiquetado de la frontera se recomienda hacer uso de las funciones de visualización proporcionadas en la implementación y descritas en el documento ***Explicacion_codigo_fuente.pdf.***

4. Seleccionar un nodo (celda) de la frontera

Implementar la función selectNode (nodeOfFrontier &selectedNode) para que devuelva en el argumento *selectedNode* 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, …).

Para comprobar y depurar la selección del nodo se recomienda hacer uso de las funciones de visualización proporcionadas en la implementación y descritas en el documento ***Explicacion_codigo_fuente.pdf.***