Universidad De Granada

Grado ingeniería informática (2016 – 2017)

Técnicas Sistemas Inteligentes

Práctica 1 | Encontrar y enviar al goal

Trabajo realizado por: Antonio Miguel Morillo Chica

---

1. Introducción.

En la anterior practica vimos como hacer que un robot deambulador por un mapa evitando los obstáculos, lo que vendría siendo una materialización de un agente reactivo. Ahora nos proponemos crear un agente deliberativo, es decir, un robot que use la información del mundo para poder decidir, y planificar, cual será su próxima ubicación.

Esta meta se considera goal, u objetivo, aquel punto que según nuestra valoración haremos que el robot vaya a el.

2. Como funciona.

El protocolo normal de actuación es muy simple. He aplicado un procedimiento muy simple, usando las funciones que hemos implementado para los ejercicios.

- Al cargarnos en el mapa lo primero que haremos es girar 360 grados para que el escáner pueda reconocer el terreno y elaborar un mapeo de la región.
    - En esta parte hacemos varias cosas, con la lectura del mapa decidimos que celdas están no libres a la ver que marcamos los obstáculos, muros en nuestro caso, dándoles un tamaño grande, en nuestro caso, una casilla de obstáculo suponen 20 más de anchura y altura. Esto ayuda al robot a que no se choque pudiendo así girar de forma correcta. Para el robot el inicio de un muro, casillas a 100, en el mundo real no existirá hasta 10 casillas después.
- Tras leer nuestro mundo elegimos los puntos que son frontera, es decir aquellos que siendo casillas libres tienen casillas desconocidas en cierta posición. Estos nodos son guardados en memoria generando una lista de puntos que serán nuestra frontera.
    - La frontera nos sirve para decidir a donde viajar ya que estos puntos son interesantes, poseen cerca suyo casillas inexploradas siendo nuestro

objetivo explorarlas todas.

- Por último nuestro robot selecciona el punto de la frontera más lejano en linea recta desde su posición. Este punto es nuestro goal, la futura posición a la quedemos ir. Tras haber decidido el nodo lo que hacemos es publicarlo para que el planlificador a largo plazo sepa el punto y el planificador a corto tiempo le dirija a este punto.
  - Si llega al punto vuelve a girar, decidir que es frontera y selecionar la más lejana.
  - Si no se llega al objetivo, en vez de girar como hacemos a llegar al objetivo lo que hago es de la vista actual, seleccionar la frontera, la diferencia es que no giro.

3. Explicación del código.

Para quierar 360 grados lo que hago es calcular con la velocidad angular el tiempo necesario para girar, la velocidad es el espacio / tiempo por lo que el tiempo actual es lo que decide si se ha recorrido un espacio determinado a una velocidad determinada.

```
void FrontierExplorer::gira360(){
  ROS_INFO("Entro en girar 360");

  // angular.z es radianes / segundos
  double VELOCIDAD_ANGULO_GIRO = V_ANGULAR_CTE,
         tiempo_total = 4*PI / VELOCIDAD_ANGULO_GIRO,
         inicio = ros::Time::now().toSec(),
         ahora = inicio;

  double tiempo_empleado = 0;

  while(tiempo_total > tiempo_empleado){
    ahora = ros::Time::now().toSec();
    tiempo_empleado = ahora - inicio;
    geometry_msgs::Twist msg;
    msg.angular.z = VELOCIDAD_ANGULO_GIRO;
    commandPub.publish(msg);
    ros::spinOnce();
  }
}
```

Para seleccionar los nodos de la frontera buscamos en el mapa en memoria theGlobalCm. Será un nodo si la casilla en cuestion está libre, es decir, su valor es 0 y a la vez esté rodeado de casillas desconocidas.

```
void FrontierExplorer::labelFrontierNodes(){
  // Frontera es un vector de nodosFront, nodo es un struct con
x e y.
  double  resolucion = cmGlobal.info.resolution;
  double  origen_x = cmGlobal.info.origin.position.x,
          origen_y = cmGlobal.info.origin.position.y;

  for (int y = 0; y < theGlobalCm.size(); y++) {
    for (int x = 0; x < theGlobalCm[0].size(); x++) {
      if (someNeighbourIsUnknown(x, y) && theGlobalCm[y][x] ==
0) {
        double wx = x * resolucion + origen_x,
               wy = y * resolucion + origen_y;

        nodeOfFrontier nuevoNodo;
        nuevoNodo.x = wx;
        nuevoNodo.y = wy;

        frontera.push_back(nuevoNodo);
      }
    }
  }

  eraseFrontier(nodoPosicionRobot.x, nodoPosicionRobot.y);
}
```

Dentro de los puntos que son frontera selecciono el punto más lejano en distancia en linea recta desde la posición del robot. He probado a elegir el más cercano intentando descartar los puntos que se quedan en la base que generalmente son desconocidos pero me es imposible ya que el robot continuamente piensa que ya está sobre el nodo objetivo. Hay que tener en cuenta que siempre acabo eliminando los punto de la frontera que distan a menos de 50 cm que es la propia anchura del robot.

```
void FrontierExplorer::selectNode(nodeOfFrontier &selectednode)
{
  double aux_x, aux_y;
  double maxDistance = 0;
```

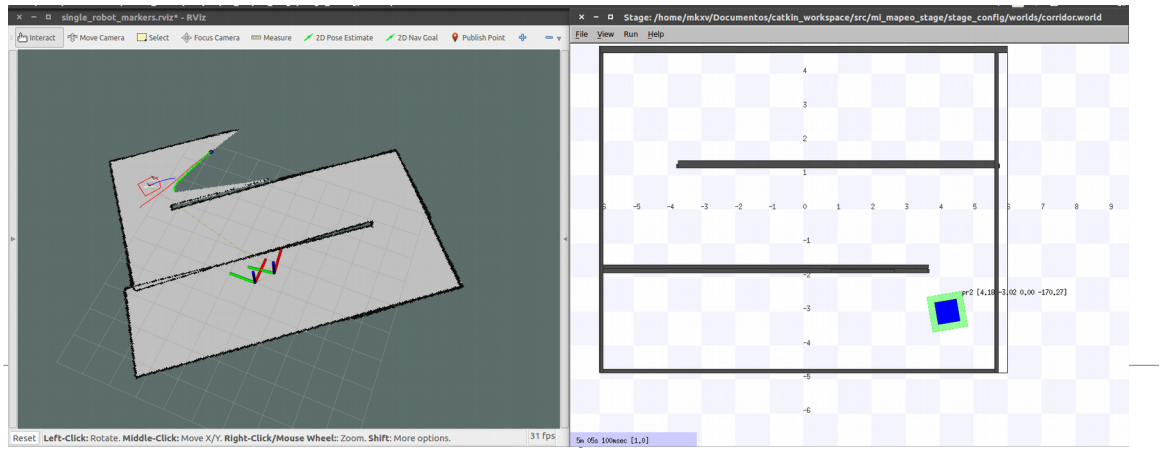
```
  bool sigue = true;
  // Se supone que el que tenga mayor X e Y será el más lejano?
  for (int i = 0; i < frontera.size(); i++) {
    if (distancia(nodoPosicionRobot.x, nodoPosicionRobot.y,
frontera[i].x, frontera[i].y) > maxDistance) {
      maxDistance = distancia(nodoPosicionRobot.x,
nodoPosicionRobot.y, frontera[i].x, frontera[i].y);
      selectednode.x = frontera[i].x;
      selectednode.y = frontera[i].y;
    }
  }
}
```

4. Experiencia en Corridor y Willow

Puedo decir que tengo una navegación estable en corridor, sin embargo en willow por algún motivo me da una violación de segmento. No he sido capaz de depurar y entender porque esto se produce.

Sin embargo en corridor funciona bien. En algunos momentos se choca pero debido a que el planificador a corto plazo actua de forma extraña. Por otro lado la experiencia de llegada a veces es tope ya que no consigue posicionarse de forma natural sobre el nodo. Mi pequeña solución a esto ha sido que la orientación de llegada sea de 180 grados para que no tenga que girar pero a veces lo hace.

Aquí podemos ver una ejecución en corridor donde ha descubierto la gran parte del mapa.



Nota: He de mejorar que no se choque en ningún punto del papa.