Esercitazione 3 - ROBOTICA

Esercizio 1 - Pratica

Implementare una strategia di navigazione con i campi di potenziale per un movimento del robot nello stage dato:

- un Goal posizionato nella mappa.
- una posizione globale del robot.

Provare a stimare la posizione utilizzando la tecnica del "Dead Reckoning" anzichè sottoscrivere il topic Pose.

Ragionamento:

Per lo svolgimento dell'esercizio, io e la mia collega abbiamo inizialmente sottoscritto al topic Pose, solo una volta, per prendere le coordinate iniziali del robot. Successivamente il topic viene distrutto.
L'unica sottoscrizione che andremo a considerare sarà il topic LaserScan. Questo topic ha una funzione per l'utilizzo dei laser che rileva la distanza dagli ostacoli (per un massimo di distanza di 30 cm). La funzione "ranges" consta di un vettore di 1081 celle, dove in ognuna di esse vi è una lettura del laser. Le letture di laser sono scostate l'una dall'altra di un valore pari a "$angle\_increment = 0.00436736317351$".
Le coordinate del Goal vengono passate tramite argomento alla funzione main e il robot verrà "attratto" proprio da quella posizione.
Per il "Dead Reckoning", considerata inizialmente la posizione del robot, si utilizza una semplice funzione fisica per la quale lo spazio percorso è uguale alla velocità moltiplicata per il tempo: $s = v * t$. Come tempo consideriamo la frequenza delle sottoscrizioni, mentre la velocità è quella impostata dal topic "geometry_msgs::Twist".

Andremo ora ad analizzare il codice più in dettaglio.

Analisi del Codice:

Include e define di comodità che utilizzeremo per il codice:

```
1  #include <ros/ros.h>
2  #include <geometry_msgs/Twist.h>
3  #include <signal.h>
4  #include <stdio.h>
5  #include <string>
6  #include <fstream>
7  #include <iostream>
8  #include <nav_msgs/Odometry.h>
9  #include <geometry_msgs/Vector3.h>
10 #include <sensor_msgs/LaserScan.h>
11 #include <tf/transform_datatypes.h>
12 #include <math.h>
13
14 #define angle_increment 0.00436736317351
15 #define SOGLIA 0.2
16 #define ANGULAR_V 0.5
17 #define LINEAR_V 0.5
```

Le variabili globali da noi utilizzate saranno le seguenti:

```
// VARIABILI GLOBALI
double goal_x;
double goal_y;

double robot_pose_x = 0.0;
double robot_pose_y = 0.0;
double robot_pose_orientation = 0.0;

const double k_obstacle = 2.0;
const double k_goal = 25.0;

// questa variabile verra' utilizzata per risolvere il
// problema dei minimi locali
double f_tot;

bool check = true;
bool linear_movement = true;
bool angular_movement_left = true;
bool angular_movement_right = true;
```

Sottoscrizione al topic Pose per la posizione iniziale del robot. la variabile booleana (check) indica che la posizione è stata letta correttamente e, impostata a false, ci consente di uscire dalla chiamata ros::spinOnce() per successive chiamate ai topic.

```
void subPose(const nav_msgs::Odometry::ConstPtr& pos)
{
   // setto le posizioni iniziali del robot
   robot_pose_x = pos->pose.pose.position.x;
   robot_pose_y = pos->pose.pose.position.y;
   robot_pose_orientation = tf::getYaw(pos->pose.pose.orientation);

   printf("posx posy posw: [%f] [%f] [%f]\n", robot_pose_x, robot_pose_y,
           robot_pose_orientation*180/M_PI);

   check = false;
}
```

Passiamo adesso alla funzione principale che comanderà il movimento del robot tramite potenziali: la callBack del laser.

```
void laserCallBack(const sensor_msgs::LaserScan::ConstPtr& scan)
{
    double f_repx, f_repy;
    double f_attrx, f_attry;
    double angle_from_goal, distance_from_goal;
    double angle_tot;

   // questo mi serve per rilevare gli ostacoli e calcolare
   // la forza repulsiva data da essi e fare la somma per
   //  calcolarmi le componenti repulsive totali.
   for(int i=0;i<1080;i++) {
      f_repx += cos((i*angle_increment)-(3*M_PI/4))/(k_obstacle*
         (pow(scan->ranges[i], 2)));
      f_repy += sin((i*angle_increment)-(3*M_PI/4))/(k_obstacle*
         (pow(scan->ranges[i], 2)));
   }

   //calcolo la distanza del robot dal goal
   distance_from_goal = sqrt(pow(goal_x-robot_pose_x, 2)+
      (pow(goal_y-robot_pose_y, 2)));


```

```
75      //calcolo l'angolo tra la posizione frontale del robot e il goal
76      angle_from_goal = atan2(goal_y-robot_pose_y, goal_x-robot_pose_x) -
77         robot_pose_orientation;
78
79      if(angle_from_goal < -M_PI)
80         angle_from_goal += 2*M_PI;
81      else if (angle_from_goal > M_PI)
82         angle_from_goal -= 2*M_PI;
83
84      printf("-----------------------------\n");
85      printf("distance_from_goal: %f angle_from_goal: %f\n", distance_from_goal,
86         angle_from_goal*180/M_PI);
87      printf("-----------------------------\n");
88
89      f_attrx = cos(angle_from_goal)*k_goal*distance_from_goal;
90      f_attry = sin(angle_from_goal)*k_goal*distance_from_goal;
91
92      printf("FORZA ATTRATTIVA X: %f\n", f_attrx);
93
94      angle_tot = atan2(f_attry-f_repy, f_attrx-f_repx);
95      if(angle_tot < -M_PI)
96         angle_tot += 2*M_PI;
97      else if (angle_tot > M_PI)
98         angle_tot -= 2*M_PI;
99
100     f_tot = sqrt(pow(f_attrx-f_repx, 2)+pow(f_attry-f_repy, 2));
101     angle_tot = atan2(f_attry-f_repy, f_attrx-f_repx);
102
103     printf("forza totale: %f angolo totale: %f\n", f_tot, angle_tot*180/M_PI);
104
105     printf("-----------------------------\n");
106     printf("angolo totale: %f angolo robot: %f\n", angle_tot*180/M_PI,
107        robot_pose_orientation*180/M_PI);
108     printf("differenza: %f\n", angle_tot);
109     printf("-----------------------------\n");
110
111     if (fabs(angle_tot) > SOGLIA) {
112        linear_movement = false;
113        if (angle_tot > 0) {
114           angular_movement_left = true;
115           angular_movement_right = false;
116        } else {
117           angular_movement_right = true;
118           angular_movement_left = false;
119        }
120     } else {
121        linear_movement = true;
122        angular_movement_left = false;
123        angular_movement_right = false;
124     }
125 }
```

Il main non fa altro che inizializzare il nodo, i topic, settare il movimento del robot e calcolare lo spazio percorso tramite Dead Reckoning.

GRUPPO BACK:
Antonino Buscetta (0610591)
Chiara Capobianco (0609919)