Homework 3

Mobile Robot Obstacle Avoidance with DWA

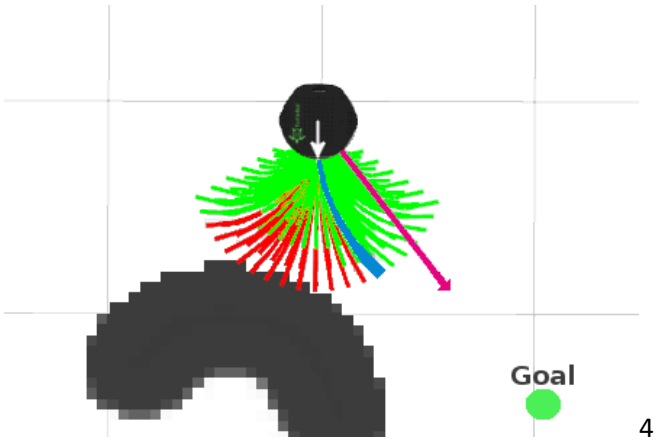
Introduction:

Obstacle avoidance methods designed to provide reactive navigation behaviors to the robots. These methods try to convey the robot to the goal point by using obstacle coordinates within the sensor range. Because these methods depend on sensory information, obstacles can be detected and avoided in a fast and efficient way.

The Dynamic Window Approach is a velocity space search method which takes robot dynamics into consideration. This method is a three stage process. The first stage eliminates unreachable velocities coming from the acceleration limits of the robot. In the second stage, all velocity pairs which are not able to stop before colliding with obstacles are eliminated. In the third stage, DWA evaluates an admissible velocity set by maximizing its objective function shown in the following equation. DWA predicts the results of each velocity pair candidates in terms of final heading angle, minimum distance to obstacles and linear velocity values and chooses the optimum velocity pair by maximizing the objective function.

$$G(V, w) = \sigma[\alpha head(V, w) + \beta dist(V, w) + \gamma vel(V, w)]$$

The heading function head(V, w) represents the approximation to goal angle and its value increases when the robot's heading approaches to target location. The aim of the distance function dist(V, w) is to promote safe navigation. It calculates the minimum distance values to obstacles on the trajectory obtained from velocity pair. The velocity function vel(V, w) calculates the linear velocity values in the velocity set. Coefficients α, β and γ are weights of these functions and σ is a smoothing operator. Maximizing this objective function results in safe trajectories to reach target as fast as possible. Possible outputs of a DWA implementation are illustrated in the following figure.



In this figure, green and red trajectories indicate admissible trajectories and inadmissible trajectories, respectively. Goal heading angle is shown as the magenta arrow and best trajectory is the blue trajectory. We created a code ("dwa_simulation.py") and a simulator to implement a basic version of DWA algorithm. You need to do following instructions to doing simulation of the DWA algorithm.

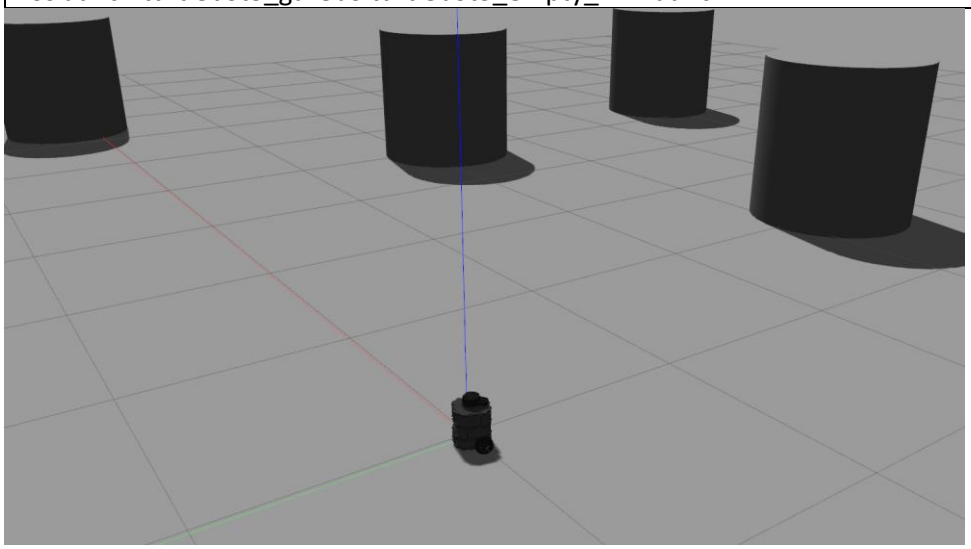
Setup Instructions:

1) Prepaing of Gazebo simulation of Turtlebot3

Download the “turtlebot3_simulations_new.zip” file and extract it into “/catkin_ws/src” folder. Replace it with old version if you installed before.

Run the following commands in terminal:

```
cd catkin_ws
catkin_make
roslaunch turtlebot3_gazebo turtlebot3_empty_rviz.launch
```



The last command will open a Gazebo window as seen in the figure above, and Rviz visualization tool automatically.

2) Programming environment

In new terminal, cd into “pycharm..../bin” folder and execute “./pycharm.sh” command. This will open your Pycharm IDE. Create a Python 2.7 Project and make sure all ROS packages is appeared in packages in System Interpreter menu in Project settings.

3) Download the base DWA script named as “dwa_simulation.py” and copy into your project folder. Our base script is almost complate, we are expecting from you to make small modifications in code. We already created subscriber functions for robot position, goal position, obstacle coordinates and made comments about code segments. DWA specific functions are also created and commented (“distance_to_obstacles”,” find_best_velocity”). We also write the DWA algorithm steps into main function. Our expectations are:

Requirements:

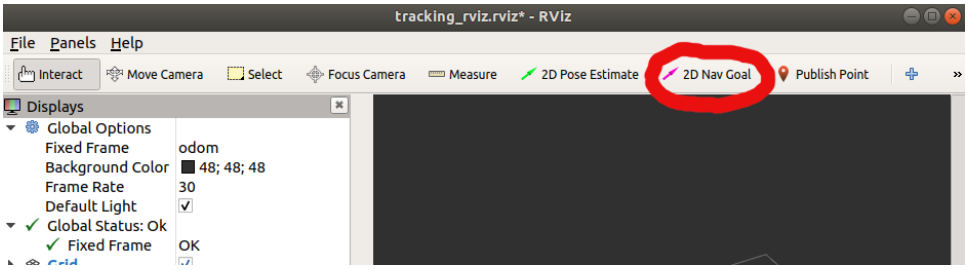
1) (40pt) Fill the “robot_model” function with following requirements:

- This function inputs are (linear velocity (V), angular_velocity (w) and total time (T)). We will simulate this velocities for a given time T.
- We have time interval variable which set to 0.2 seconds.
- You will find the next x,y positions and theta angles for every interval in time T
- This function must return with trajectory xy points in [[x1,y1],[x2,y2],...,[xn,yn]] form (as a python list) and last angle of the trajectory (as a single float).
- You should update these poses using differential drive kinematic equations (an example update shown in the following figure).

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} + \begin{bmatrix} cos(\theta + \frac{w*\Delta t}{2}) * V * \Delta t \\ sin(\theta + \frac{w*\Delta t}{2}) * V * \Delta t \\ w * \Delta t \end{bmatrix}$$

PS: Please explain your motion model and “robot_model” function code in your report.

2) (30pt) Run your script, if everything is correct your algorithm will work when a goal position given. You can set a goal position by sing following tool in RViz.



PS: Share a link about video demonstration of the algorithm in your report. Discuss potential reasons if there are errors occured.

3) (30pt) Change objective function constants (in “find_best_velocity” function) and discuss the results of the differences. You can add video if its required.

Please inform me if you have questions. Best regards.

Res. Assist. Aykut ÖZDEMİR

Mail : ozdemirayk@itu.edu.tr