Project Report for

Cooperating Autonomous Mobile Robots

Robot Motion Planning in Humans' Environment

Kai Zhang

Department of Electrical and Computer Engineering

Stevens Institute of Technology

Abstract

Two objectives would be achieved for this final project. The first one was motion planning. This task was to produce continuous motion to connect the initial position to the goal position. The algorithm was grid-based search. Virtual Box would be used to run Ubuntu as the default system environment through the project, which is an important composition of the GUI interface. Gmapping SLAM algorithms would be used to create the map of the environment. In addition, Move-base package played an important role which was achieved from the ROS website. It was used to solve path planning and interacting with humans. Global costmap would collect the information from the map and provide to global planner to make decision. The robot which we chose was husky robot instead of the original robot in the environment. Secondly our group solved the problem that how the robot interacted with humans. The report would show that the data from lasers on the robot would be collected and analyzed on the local costmap, and how the local planner make the trajectories decision to the robot avoiding humans. Results of the path would be shown as the process and procedure in details.

I Instruction

Robot technology is moving more and more from factories to everyday applications in households, offices and public places. Almost all such robots have to master the task of navigating through their environment. Human-aware navigation is the intersection between research on human–robot interaction (HRI) and robot motion planning. [1]

The goal of the final project was to use ROS to demonstrate robot motion planning in humans' environment. A robot motion planning algorithm in the humans' environments was developed to take into account of human-robot interaction. Also, it was implemented in the ROS simulator. In the move-base package downloaded on the ROS website. Husky robot was selected to navigate from the start position (1, -3) to (14, -1) in the given environment. Sensors on the Husky robot was used to collect information to avoid humans. The most important thing is localization and map for the mobile autonomous robot. The main objective of this project is to create a navigation. Our project utilizes the existing algorithm, gmapping SLAM algorithm to create a map for navigation. Move base package provides global planner and local planner algorithms for the husky robot to navigate in the environment and avoid people.

II Algorithm

2.1 Gmapping SLAM Algorithm

The most popular approaches in the simultaneous localization mapping are extended Kalman filters (EKFs), maximum likelihood techniques, sparse extended information filters (SEIFs), and Rao Blackwellized particle filters (RBPFs). Gmapping algorithm is a highly Rao-Blackwellized particle filter to get grid maps from laser rage data. Rao-Blackwellized particle filters has been used as an effective way to solve the simultaneous localization mapping(SLAM) problem[1]. This approach uses a particle carries an individual map of the environment. The key question is how to reduce the number of particles. Adaptive techniques can be used to reduce the numbers of Rao-Blackwellized particles for learning grid maps. This can decrease the uncertainty about the robot's pose in the prediction step of the filter. The main characteristic to gmapping is that the maximum likelihood approach can only track a single mode of the dis- tribution about the trajectory of the robot. The key idea of the Rao-Blackwellized particle filter for SLAM is to estimate the joint posterior $p(xx_1: t|z_{1:t}, u_{1:t-1})$about the trajectory $x_{1:t} = x_1, \dots, x_t$ of the robot and the map m of the environment given the observations $z_{1:t} = z_1, \ldots, z_t$ and odometry measurements $u_{1:t-1} = u_1, \ldots, u_t$, It works by using the following factorization: $p(x_{1:t}, m|z_{1:t}, u_{1:t-1}) = p(x_{1:t}, m|z_{1:t}, u_{1:t-1})p(m|x_{1:t}, z_{1:t-1})$. Localization problem relates a lot to $p(x_{1:t}, m|z_{1:t}, u_{1:t-1})$. The trajectory of the vehicle needs to be estimated. A particle filter can be performed when the observations and odometry readings processes[2]. So gmapping algorithm for slam maintain this individual map for each sample and update the map according to each movement.

2.2 Move Base Algorithm

The move_base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The move_base node links together a global and local planner to accomplish its global navigation task[3]. The details of move base package are shown in the Fig. 1.

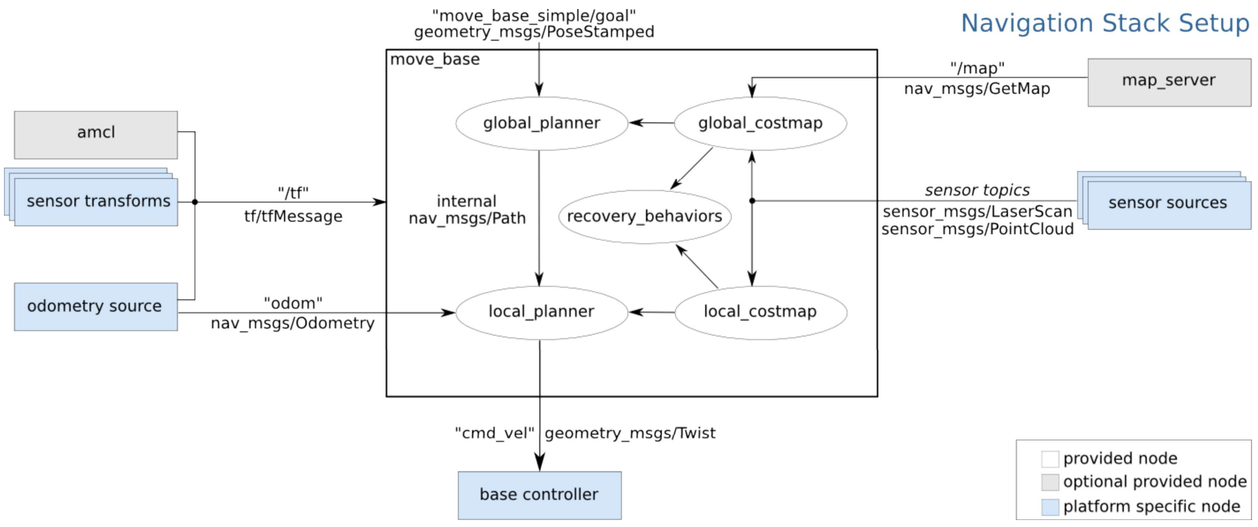

Fig. 1. Move base

The map server package help us to upload the map to the global_costmap, and the laser data receiving from the robot will give both to the local_costmap and global_costmap.

2.2.1 Global Planner

The global planner package provides an implementation of a fast, interpolated global plan for navigation. There are serval algorithms we can use for navigation, such as A* algorithm and

Dijkstra's algorithm. Here, our robot uses the Dijkstra's algorithm to make global plan. Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph[4][5].

2.2.2 Local Planner

The ros website provides our serval different local planners, but the planners we mainly use are base_local_planner and dwa_local_planner. Here, in our project, we use the dwa_local_planner to implement our experiment. The dwa_local_planner package provides a controller that drives a mobile base in the plane. This controller serves to connect the path planner to the robot. Using a map, the planner creates a kinematic trajectory for the robot to get from a start to a goal location. Along the way, the planner creates, at least locally around the robot, a value function, represented as a grid map. The controller's job is to use this value function to determine dx, dy, dtheta velocities to send to the robot. The basic idea of the Dynamic Window Approach (DWA) algorithm is as follows:

1. Discretely sample in the robot's control space (dx, dy, dtheta)
2. For each sampled velocity, perform forward simulation from the robot's current state to predict what would happen if the sampled velocity were applied for some (short) period of time.
3. Evaluate (score) each trajectory resulting from the forward simulation, using a metric that incorporates characteristics such as: proximity to obstacles, proximity to the goal, proximity to the global path, and speed. Discard illegal trajectories (those that collide with obstacles).
4. Pick the highest-scoring trajectory and send the associated velocity to the mobile base.
5. Rinse and repeat.[6]

III Simulation

3.1 Gmapping SLAM

Before we start to use move base package to implement the navigation, we need to get the map of the environment. We have already change the original robot in the environment to husky robot, so the gmapping package can simply help us to get the 2D map by executing *rosrun gmapping slam_gmapping scan:=/scan* in the terminal. At this process, we need the robot move to the every part of the environment, so we use rqt to simply publish the linear and angler velocity to the topic of cmd_vel to control the robot. Finally, we use map_server package to save the map by executing *rosrun map_server map_saver –f map*. The map is shown in Fig. 2.

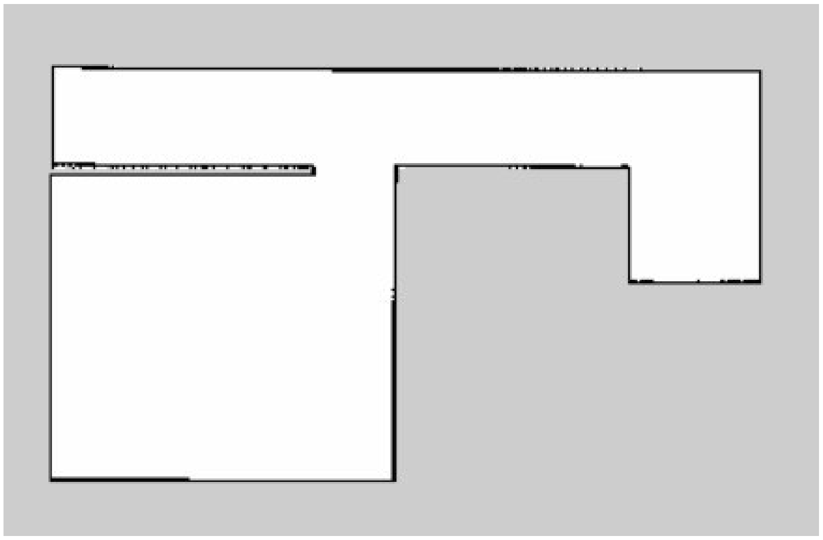

Fig. 2. Map

3.2 Navigation and Avoidance

After getting the map, now we can use the move base package to implement the navigation and to interact with people. For this part, we need to create a launch file to combine all the nodes we use here. The detail steps are shown below:

1. Creating a package and naming final
2. Writing the launch file in the package and there are four main part in the launch file
    - loading the map we get in previous part
    - including the husky robot and gazebo environment launch file
    - Setting the parameters of move_base package
    - Setting the tf

3. Starting our launch file by executing *roslaunch final husky_human.launch*, and then publishing the goal position to the robot by executing *rostopic pub /move_base_simple/goal geometry_msgs/PoseStamped '{header: {stamp: now, frame_id: "/map"}, pose: {position: {x: 13.0, y: 7.0, z: 0.0}, orientation: {w: 1.0}}}'.*

V Results and Analysis

The simulation results shows us the robot can realize our purpose in some conditions, but there are some bugs in the project. First, we can get the correct global path by the global planner algorithm in the move base package. The global path we get in Rviz is shown in Fig. 3. The green line is the global path we get.

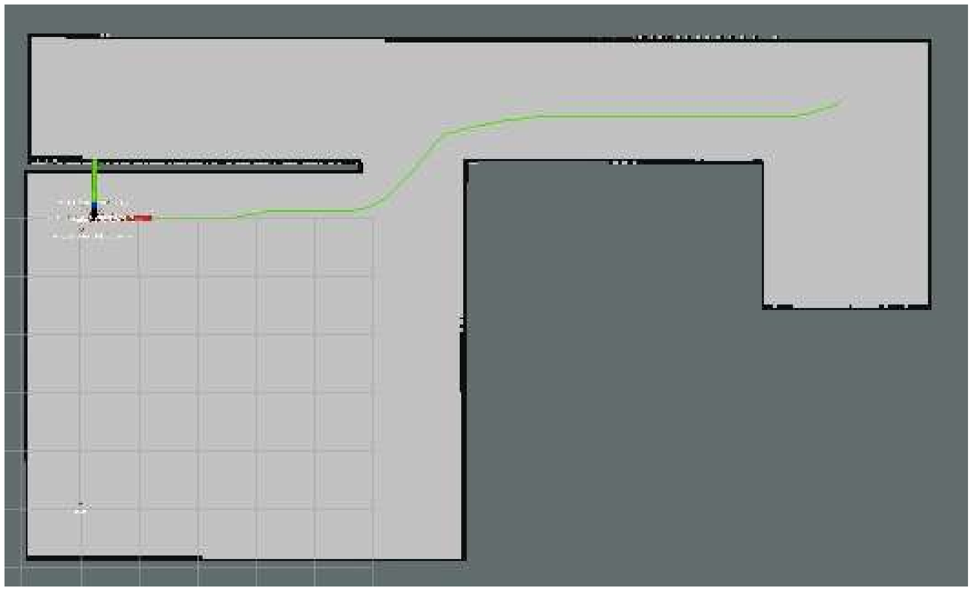

Fig. 3. The green line is the global path

After getting the global path, the robot will move to the goal position along the green line. During the process, when the robot encounters human, it will re-plan the path locally by the local planner algorithm to avoid human. The Fig. 4. shows the moving process. The initial position show in the Fig. 4. (a). The robot will encounter people while moving in Fig. 4. (b) and successfully avoid people. The Fig. 4. (c) shows the robot successfully arrive to the goal position.

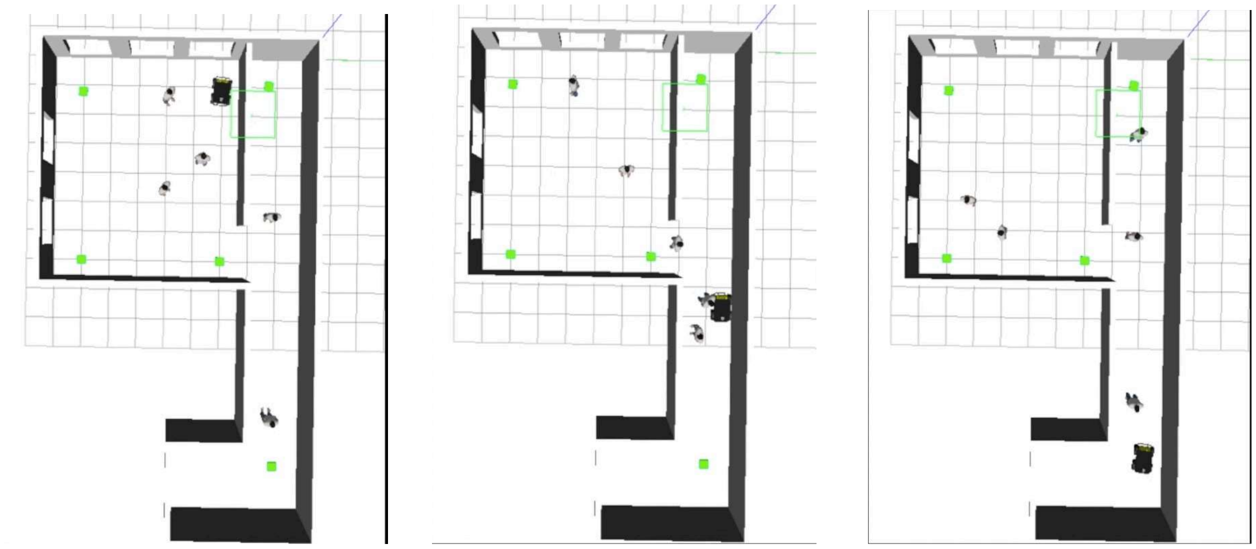
(a) initial position (b) encounter people (c) arrive destination

Fig. 4. Successful moving process

The Fig. 4. shows a successful process, but there are still some bugs for the local planner. For example, when the robot encounters too much people, the local planner will not work. The Fig. 5. shows this condition. When the robot encounter two people simultaneously, the robot cannot stop itself to wait people, and since the people block the robot, that results in the wrong odometry information of the robot. Finally, the robot cannot move to the goal position.

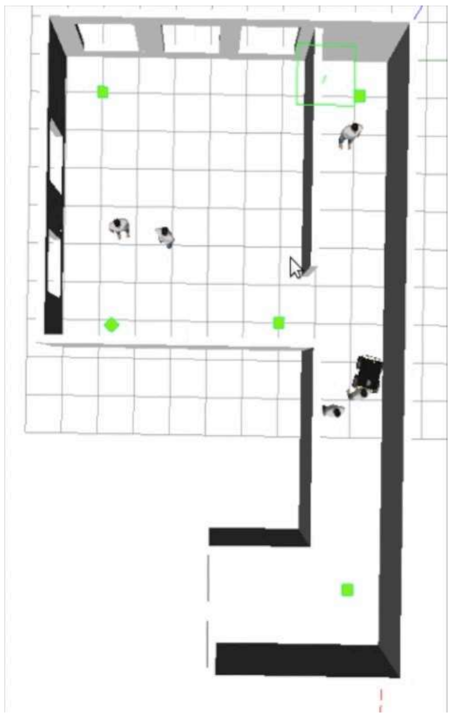
Fig. 5. Encounter two people

IV Conclusion

In this project, we successful realize our purpose of robot navigation and interaction with human. The robot can avoid people during navigation by local planner algorithm. However, there are still some bugs that we do not solve. When the robot encounter too much people, it cannot avoid people or stop itself. In the further work, we will focus on this part to solve this problem.

reference

[1] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, Alexandra Kirsch, Human-aware robot navigation: A survey, Robotics and Autonomous Systems 61 (2013) 1726–1743

[2] Fast and Accurate SLAM with Rao-Blackwellized Particle Filters, Giorgio Grisett, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, Daniele Nardi

[3] http://wiki.ros.org/move_base

[4] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

[5] http://wiki.ros.org/global_planner

[6] http://wiki.ros.org/dwa_local_planner