Robotics
LOCALIZATION AND NAVIGATION

Submitted to :

Dr. Ralph SEULIN

Submitted by :

Mohamed Adel Mohamed Ali

Jaafer Wesam Al-Tuwayyij

Contents

Introduction

Localization and navigation are the two most important tasks for mobile robots. We want to know where we are, and we need to be able to make a plan for how to reach a goal destination. Of course these two problems are not isolated from each other, but rather closely linked. If a robot does not know its exact position at the start of a planned trajectory, it will encounter problems in reaching the destination.

for our task we will use the ROS actionlib package to define simple action client, MoveBaseAction that communicate with the move_base package to preform our Navigation for one goal, the move_base package implements an action server that accepts a goal pose for the robot (position and orientation) and attempts to reach that goal by publishing Twist messages while monitoring odometry and kinect scan data to avoid obstacles, also we use SMACH package to build state machine that include the loading and unloading area as state for the robot to be in and move from one area to another based on the state of the goal Robot .

in this report we go throw the main algorithms and method that been used for solve the Localization and navigation problem in Robotics, and also ROS packages that implemented them , and we mention which one of them we will use for our project, in the first section we give short description of the Robot Localization problem and what is the main method and algorithms that been us d to solve them, in the second section we describe the task of build a map as a global map also the local cost map, in the third section we consider the Robot Navigation problem and also path planning and the different between local planner and global planner and the ROS implementation, the last section we give over view of the task management in ROS and the use of "actionlib" and "SMACH" package the build state machine for our Robot.

Mobile Robot Localization

The first Question for any mobile robot - or mobile human - is where am I?!

To localize, the robot might use its on-board sensors (ultrasonic, range sensor, vision) to make observations of its environment. The information provided by the robot's odometry and gyroscope, plus the information provided by such observations of the environment, can be combined to enable the robot to localize as well as possible with respect to its map, this task carried by two sub-tasks that provide the main building blocks for Robot Localization " SEE ", "ACT" cycle, we consider a mobile robot moving in a known environment, as the robot starts to move, say from a precisely known location, it can keep track of its motion using wheel odometry and gyro data, due to odometry/gyro uncertainty, after some movement the robot will become very uncertain about its position.

The "ACT" task which is action update represents the application of some action model that drives the robot, in this step the model combines the movement information - the wheels encoder's - with the previous position believe, to yield a new belief state representing the robot's belief about its current position.

The "SEE" task which is Perception update represents the application of some perception model that is provided by the sensor input, in this step the model combines the sensor inputs and updated belief state from the "ACT" task, to yield a refined belief state representing the robot's current position.

there are different probabilistic models that combines the two tasks to preform the Robot localization task some of the most well known algorithms[1].

1. Markov localization

   represents the robot's belief state is usually represented as separate probability assignments for every possible robot pose in its map, this model allows for localization starting from any unknown position which can be used for kidnapped Robot problem because the robot can track multiple possible positions, but its computationally expensive due to the fact that we need to calculate the probability for every cell in the map.

2. Kalman filter localization

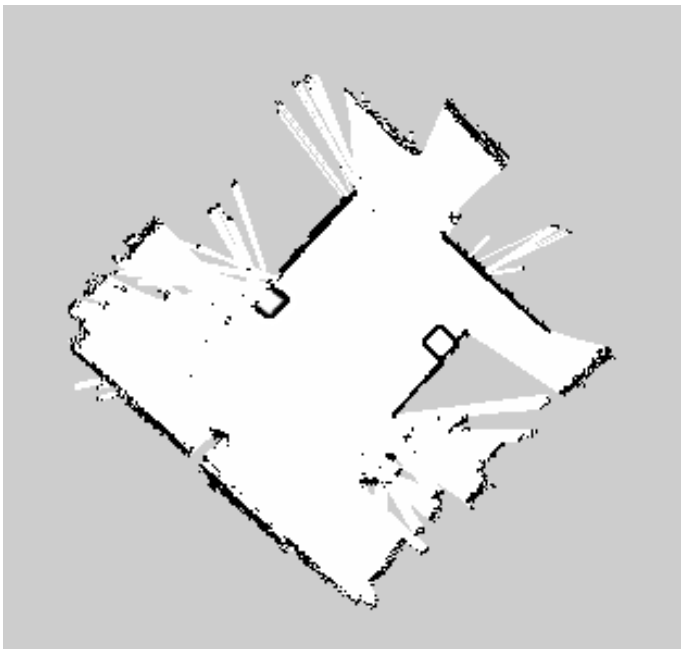
   represents the robot's belief state using a single, Gaussian probability density function just a μ and σ parameterization of the robot's belief about position with respect to the map, Updating the parameters of the Gaussian distribution is all that is required, the Kalman filter is used in many different tasks for estimation and localization, especially Extended Kalman Filter, that doesn't use the linear assumption like the normal Kalman filter, in ROS "robot_pose_ekf" package is used to estimate the pose of a robot.

3. Monte Carlo Localization

   represents the robot's belief state by particles which is samples from the motion model distribution function, using particles from present belief as starting points. The algorithm is called Monte Carl Localization "MCL" it is already become one of the most popular localization algorithms in robotics. It is easy to implement, and tends to work well across a broad range of localization problems. in ROS "amcl" package which stand for adaptive Monte Carlo localization is one of the main algorithms that is used for Localization and Navigation, and the one that we will use for our task [2].

Map Building

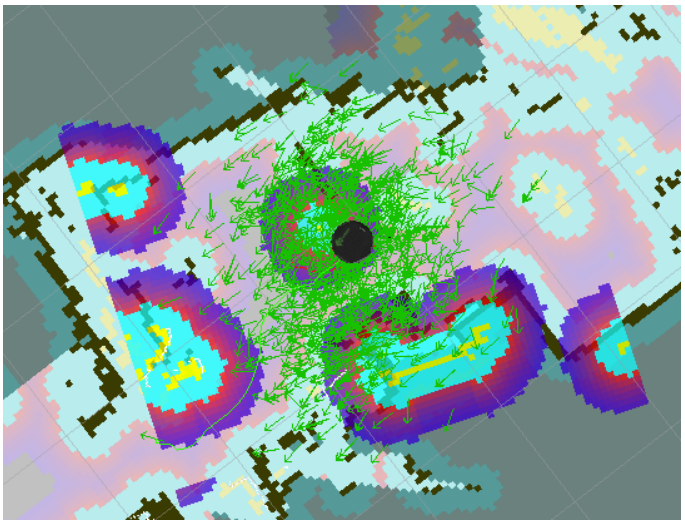
For the Robot to estimate its Location or to preform navigation or planning task , it mast have a model about its environment that embedded information about available space for movement , the obstacles and the boundaries of the environment , most of the localization and the navigation algorithms assume that that the robot was given a map in advance. This assumption is legitimate in quite a few real-world applications , for our application a map need to build prior to the begging of the navigation task , A map in ROS is simply a bitmap image representing an occupancy grid where white pixels indicate free space, black pixels represent obstacles, and gray pixels stand for "unknown" , in ROS gmapping package contains the $slam_gmapping$ node that combine the data from laser scans and odometry into an occupancy map.



ROS Static map

Costmap

Costmap 2D uses the kinect sensor information to create a local Costmap in front of the Robot. The Costmap values represent the proximity of the Robot to an obstacle using the geometry of the turtlebot, global costmap, is a data structure that says how good or bad it is for the robot to be in a particular place in the static map, the resulting map has the obstacles inflated by a security perimeter where the Robot should not allow entering. Additionally, this map is used for local planner in order to modify the local path based on the distance from the Robot to the obstacles [3].



ROS Static and Cost map combined

costmap is composed of static map layer, obstacle map layer and inflation layer. Static map layer directly interprets the given static SLAM map provided to the navigation stack. Obstacle map layer includes 2D obstacles and 3D obstacles (voxel layer). Inflation layer is where obstacles are inflated to calculate cost for each 2D costmap cell. Besides, there is a global costmap, as well as a local costmap. Global costmap is generated by inflating the obstacles on the map provided to the navigation stack. Local costmap is generated by inflating obstacles detected by the robot's sensors in real time[4].

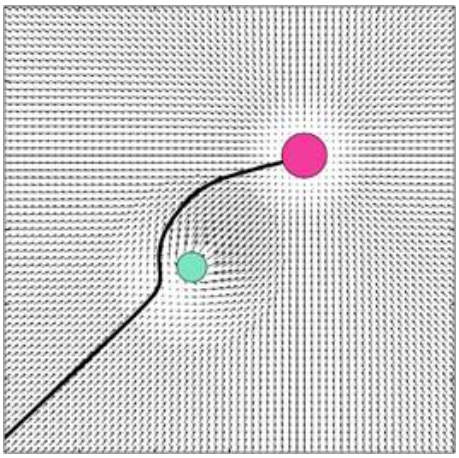
Navigation and path planning

Path planning is a well known problem in robotics and it plays an important role in in the navigation of autonomous mobile robots. In general there is three problems in navigation, these problems are: localization, path planning and motion control. One can argue that path planning is the most important of them. Path planning is the process of finding and selecting the most suitable path for the mobile robot to travel to its desired destination, in ROS "global_planner" package include some of the algorithms to perform path planning.

Here are some of the path planning algorithms[5]:

- **Artificial Potential Field**

  Artificial Potential Field Planning places values over the map with the goal having the lowest value. Obstacles are defined to have an incredibly high value. The robot then simply moves to the lowest potential value adjacent to it, which should lead it to the goal. However this technique often gets trapped in local minima. Certain techniques can be used to avoid this, such as wavefront potential field planning. Artificial potential fields can be achieved by direct equation similar to electrostatic potential fields or can be drive by set of linguistic rules.



Example of a Potential Fields

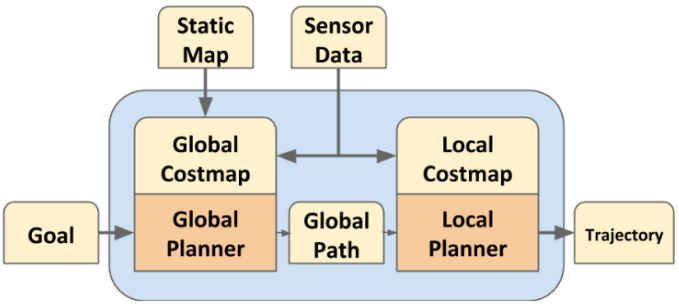
- **Grid Based Planning**

  Grid Based planning overlays a grid on the map. Every configuration then corresponds with a grid pixel. The robot can move from one grid pixel to any adjacent grid pixels as long as that grid pixel is free. Then a search algorithm such as A* or Dijkstra's can be used to find a path to get from start to the goal. One potential trade off with this method is with a lower resolution grid(bigger pixels) the search will be faster, however it may miss paths through narrow spaces of the free regions . In addition as the resolution of the grid increases memory usage increases exponentially, therefore in large areas using another path planning algorithm may be necessary.

- **Reward-Based Planning** Reward-Based Algorithms assume that robot in each state (position and internal state include direction) can choose between different action (motion). However, the result of each action is not definite. In other words, outcomes (displacement) are partly random and partly under the control of the robot. Robot gets positive reward when it reach to the target and get negative reward if collide with obstacle. These Algorithms try to find a path which maximized cumulative future rewards. Markov decision processes

(MDPs) is a popular mathematical framework which is used in many of Reward-Based Algorithms. Advantage of MDPs over other Reward-Based Algorithms is that it generate optimal path. Disadvantage of MDPs is that it limit robot to choose from a finite set of action; Therefore, the path is not smooth (similar to Grid-based approaches).

Navigation module is divided into a global planner and a local planner, where the first one finds the optimal path with a prior knowledge of the environment and static obstacles using the given map, and the second one recalculates the path to avoid dynamic obstacles.



Overview of ROS Navigation[6]

Global Planner

The global planner requires a map of the environment to calculate the best route. Depending on the analysis of the map, some methods are based on finding the path with minimum cost between the starting position and the goal position. Some examples are the Dijkstra algorithm, and A^*[5] in ROS the package "global_planner" include the different algorithms for global planner

Local Planner

The local planner creates new way points in order to transform the global path into suitable way points, taking into consideration the dynamic obstacles and the robot constraints. the main idea behind the local planner is to calculate what is the best command velocity that will be published to the robot , using a local search space for the best command velocity using a Trajectory Rollout and Dynamic Window Approach (DWA) algorithms.

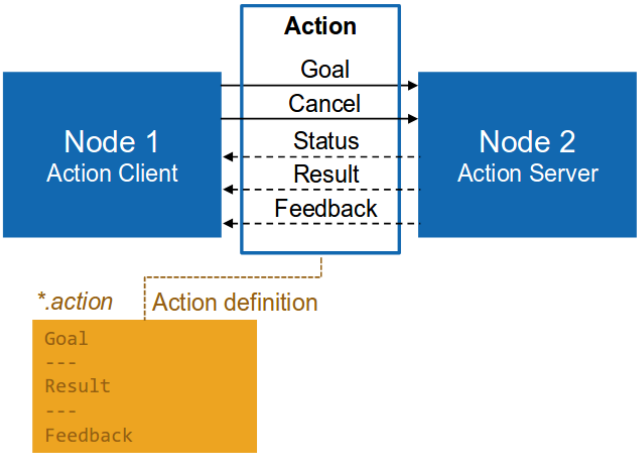
Task Management

Building Fully autonomous Robot must combine different models for Localization in environment and Navigation toward goal but also must include a behavioral model that take control of the robot actions based on different states, this behavioral paradigm work in parallel with the Robot Localization/Navigation map based model, for example to add more safety to the robot we can include specific task priorities, or even to stop the task totally if specific event occurs.

Also to task management important to preform a useful high level abstracted goal for example moving objects from one point to another or organize a scenario for the robot to preform, for our purpose we will use the ROS "actionlib" and "SMACH" packages.

"Actionlib"

provides a standardized interface for interfacing with preemptable tasks, ROS actions are the best way to implement interfaces to time-extended, goal-oriented behaviors like goto_position , actions are asynchronous.an action uses a goal to initiate a behavior and sends a result when the behavior is complete, but also action further uses feedback to provide updates on the behavior's progress toward the goal and also allows for goals to be canceled. Actions are themselves implemented using topics. An action is essentially a higher-level protocol that specifies how a set of topics (goal, result, feedback, etc.) should be used in combination [8].

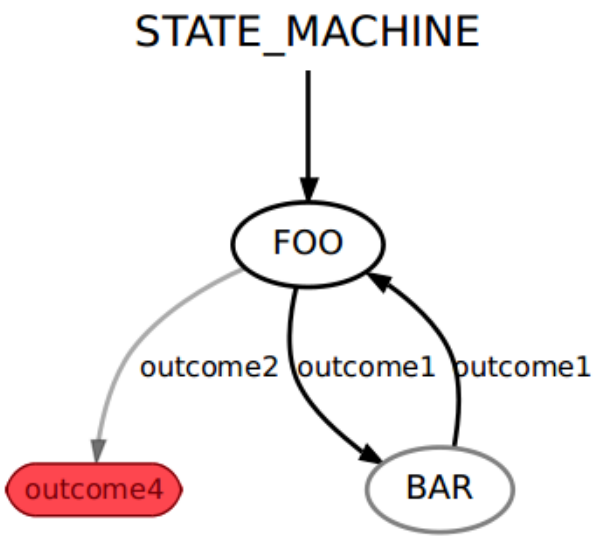


Ros Action communication diagram

"SMACH"

that enable us to build a complex state machine that control the behavior of the Robot based on its state of the given goal.

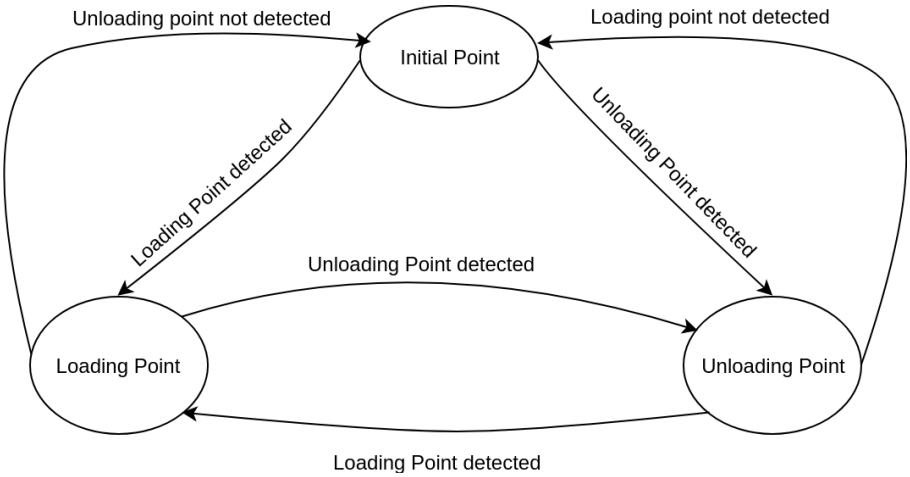
using SMACH we can decompose our scenario to smaller task well defined like move from point A to point B in the map, wait for 1 minute and after move to specific position.



State machine Using SMACH

Current state of our Project

At this moment we achieve a to combine the navigation stack with the SMACH to give the turtlebot task to loop between two positions in our Map, in the next weeks we will add more perception ability to our navigation method, to enable the robot to look for specific object or mark in the environment and preform different tasks based on which mark the robot found so we can search the room for the LOAD AREA and the UNLOAD AREA so that the Robot can move to their position and wait for the arm to load/unload.



State machine of the navigation part of the scenario

Bibliography

[1] [Intelligent Robotics and Autonomous Agents] Roland Siegwart, Illah R. Nourbakhsh - Introduction to Autonomous Mobile Robots (2004, The MIT Press)

[2] [Intelligent Robotics and Autonomous Agents] Sebastian Thrun, Wolfram Burgard, Dieter Fox - Probabilistic Robotics (2005)

[3] Layered Costmaps for Context-Sensitive Navigation - David V. Lu, Dave Hershberger, and William D. Smart

[4] ROS Navigation Tuning Guide - Kaiyu Zheng

[5] `https://en.wikibooks.org/wiki/Robotics/Navigation/Trajectory_Planning`

[6] Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles Pablo Marin-Plaza et al. 2018

[7] Navigation Illumination - Shedding Light on the ROS Navstack David V. Lu ROSCON2014 .

[8] Programming Robots with ROS - Morgan Quigley, Brian Gerkey, and William D. Smart