DOMESTIC ROBOT FOR ROOM ORGANIZATION TASKS

COMS 6731 – HUMANOID ROBOTS

Prof: Peter Allen

TAs: Neil Chen, Jingxi Xu

TEAM:

HRISHIKESH PATIL (hp2105)

ABHINANDAN SRINIVASAN (as5630)

ABHIJEET MISHRA (aam2285)

VIDEO: https://www.youtube.com/watch?v=X5z-h0o_znU

Table of Contents

ACKNOWLEDGEMENT

We would like to thank our Professor, Dr. Peter Allen, who has always been available to us and provided us with key inputs about the project. He gave us a free hand to use the resources available in his lab. We would also like to thank Jingxi Xu and Neil Chen, the TAs for the course, who assisted us whenever we needed it and were accommodating of all our requests as we worked on the project.

1) Project Overview:

Robots have long been used in industries and especially closed workspaces. They can create immense value as they can complete repetitive predefined tasks with high accuracy for extended periods time. However, even in industries, the tasks performed by robots are predefined and one needs humans to control them. Robots are also not aware of their surroundings and will not be aware of any changes in the environment unless explicitly specified. Lately, several companies are trying to make robots work autonomously in industries. This is specially seen in the field of warehouse automation where several robots move in a large warehouse and complete tasks collectively. They also communicate with each other and are aware of how the environment around them is evolving as several robots move around the facility to pick and place objects. The warehouse environment too is fairly organized as objects are mainly placed in predefined locations and the shape and grasp required for each package can be specified beforehand. The robots typically have limited interaction with humans while working in such places.

One environment where we see extremely limited use of robots is the domestic environment. A chore for a human can quickly become an extremely hard task for a robot. A typical home is extremely disorganized and the environment is rapidly changing which is hard to track. Thus, it becomes extremely important for a robot to be cognizant of its surroundings and adopt to the current environment rather than keep track of the changes continuously. Every home also has a multitude of objects placed all around the house in different poses. This can quickly confuse a robot which is trained on a limited dataset of objects. While humans can manipulate objects of several shapes easily, robots are not as dexterous. There is another factor which plays an important role in robots not being widely used for domestic purposes. In a domestic environment, the tasks a robot must perform are not predefined. Humans can understand tasks or learn to imitate them fairly quickly. It is very hard for robots to imitate tasks. For example, a human might teach another human to pour water in a container by just performing the act once. However, it is almost impossible for a human to teach a robot to perform such a task without multiple tries. For these reasons, general purpose robots have seen limited applications in house.

Our group thus chose to do a project wherein a robot works in a domestic environment. This allowed us to understand in greater depth the difficulties one has to overcome when trying program the robot to complete simple tasks in a domestic environment.

1.1) Project Tasks

We chose to complete a set of tasks as a part of our project. This involves the following steps:

1. The robot must be able to navigate a domestic environment and must be able to reach a prescribed location without colliding with objects or walls. This involves the use of mapping of an environment and then using localization by the robot so that it is aware of its position.
2. The robot must then be able to pick up objects from a table. Different classes of objects can be placed on the table and the robot must be able to classify the objects into their respective categories. For example, different objects such as bottles, toys, food cans are placed on the table. The robot must be able to classify each object into its category and then also grasp them.

3. The next step would be that the robot picks the object and then places it in its respective position. Thus, the robot might pick up the bottle, move towards a bucket which is prescribed for bottles and place it in there.

This is a simple task that a general-purpose robot must be able to complete in a domestic environment. Thus, we chose to do this.

Even harder versions of this task could involve the robot having to pick up objects from the floor all around the house rather than from a table. It could also involve the robot manipulating deformable objects such as clothes placed on the floor. This would be an extremely hard task as the robot would have to identify the clothing materials without knowing the shape as clothes can be neatly folded or just in a stack.

2) Project Distribution

As described above, the robot is navigating around a house and must perform pick and place tasks. However, to complete this task the robot must have to perform the following sub-actions which a listed below:

1. Mapping
2. Navigation
3. Movement
4. Perception
5. Grasping

The tasks are discussed in detail below.

2.1) Mapping

Any domestic multi-purpose robot must be able to navigate around the environment. To navigate independently, the robot must have a map of the workspace. It is extremely helpful, if the robot can create a map of the surrounding on its own and use it for navigational purposes. There are mapping packages available in ROS which utilize particle filters. Fetch builds upon these mapping stacks available in ROS and customizes it for the Fetch robots. The most common package that is utilized is Gmapping. A human must control the robot and move it around the entire workspace. We utilized the Gmapping package as it was well integrated into the Fetch robots and hence easier to utilize

However, there are packages available in ROS such as frontier exploration which the robot can use to map the workspace independently without any human intervention. However, we did not use this package as it requires us to integrate it with the Fetch robot separately.

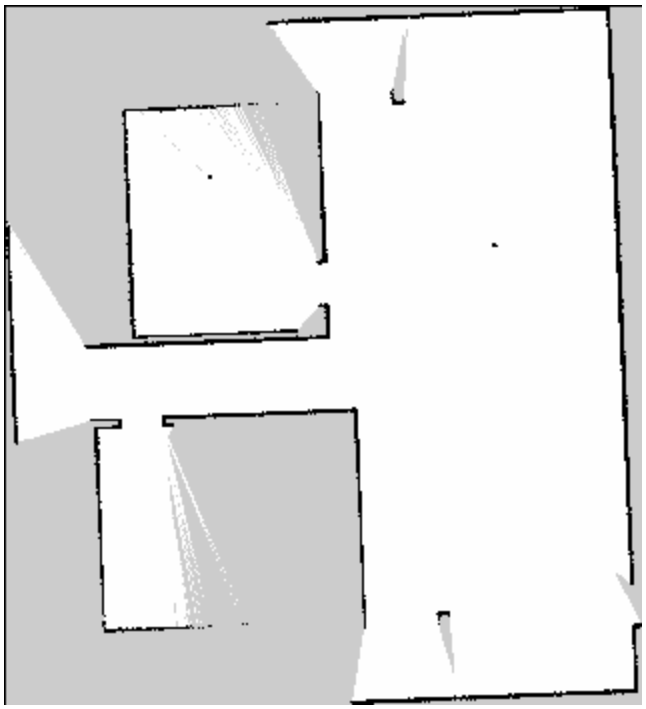


Figure: Map created by the robot

2.1.1) Gmapping

Rao-Blackwellized particle filters have been introduced as effective means to solve the simultaneous localization and mapping (SLAM) problem. This approach uses a particle filter in which each particle carries an individual map of the environment. Accordingly, a key question is how to reduce the number of particles. They present adaptive techniques to reduce the number of particles in a Rao-Blackwellized particle filter for learning grid maps. They propose an approach to compute an accurate proposal distribution taking into account not only the movement of the robot but also the most recent observation. This drastically decreases the uncertainty about the robot's pose in the prediction step of the filter. Furthermore, an approach to selectively carry out re-sampling operations which seriously reduces the problem of particle depletion. [1-3]

2.1.2) Cartographer

Cartographer is a system that provides real-time simultaneous localization and mapping (SLAM) in 2D and 3D across multiple platforms and sensor configurations. This project provides Cartographer's ROS integration. [8]

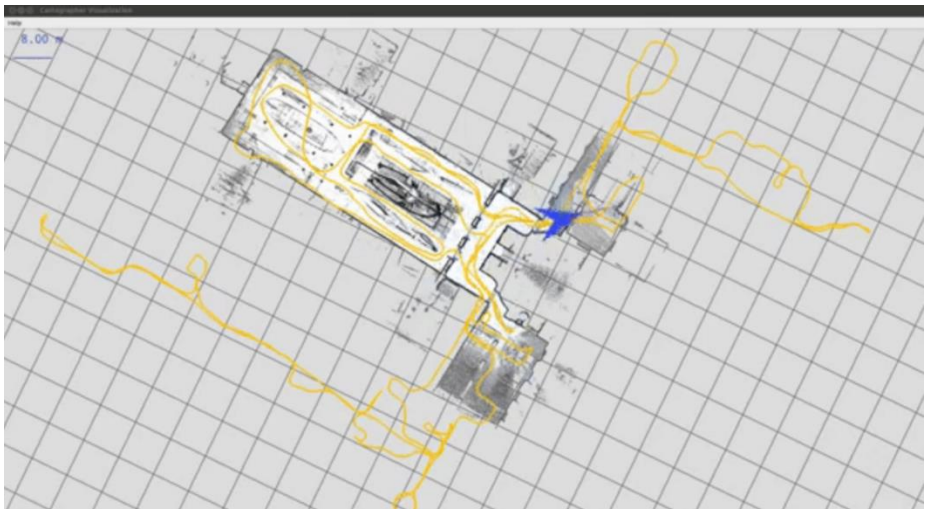


Figure: Cartograph simulated map of the robot

2.2) Navigation

Once the Robot has mapped the environment, the robot must be able to move around the environment and reach the specified location. While this can be achieved by comparing the distance travelled by the wheels or integrating the values obtained by the IMU sensors, the data can still be unpredictable and doesnt account for any outside factors such as friction, thereby causing the errors to add up over time. We thus have to rely on other methods to triangulate the location of the robot over time. A possible method to do so would be if the robot could compare the environment it sees around it at any given point with the map of the workspace it is working in. Such a method would allow it to locate itself with a high probability and that accuracy would only increase if it knew its previous states and estimations. With such kind of mapping and navigation, we would not need to incorporate extremely expensive sensors into the robot and could allow measurement errors in the sensors that we install on robot.

A method which incorporates this technique of localization and navigation is the Monte Carlo Localization. **Monte Carlo localization (MCL)**, also known as **particle filter localization**, [4] is an algorithm for robots to localize using a particle filter. Given a map of the environment, the algorithm estimates the position and orientation of a robot as it moves and senses the environment. The algorithm uses a particle filter to represent the distribution of likely states, with each particle representing a possible state. The algorithm typically starts with a uniform random distribution of particles over the configuration space, meaning the robot has no information about where it is and assumes it is equally likely to be at any point in space. Whenever the robot moves, it shifts the particles to predict its new state after the movement. Whenever the robot senses something, the particles are resampled based on recursive Bayesian estimation, i.e., how well the actual sensed data correlate with the predicted state. Ultimately, the particles should converge towards the actual position of the robot. [5]

A key problem with particle filter is maintaining the random distribution of particles throughout the state space, which goes out of hand if the problem is high dimensional. Due to these reasons it is much

better to use an adaptive particle filter which converges much faster and is computationally much more efficient than a basic particle filter.

The key idea is to bound the error introduced by the sample-based representation of the particle filter. To derive this bound, it is assumed that the true posterior is given by a discrete, piecewise constant distribution such as a discrete density tree or a multidimensional histogram. For such a representation we can determine the number of samples so that the distance between the maximum likelihood estimate (MLE) based on the samples and the true posterior does not exceed a pre-specified threshold. As is finally derived, the number of particles needed is proportional to the inverse of this threshold.

2.2.1) Adaptive filter for Localization

To use adaptive particle filter for localization, we start with a map of our environment and we can either set robot to some position, in which case we are manually localizing it or we could very well make the robot start from no initial estimate of its position. Now as the robot moves forward, we generate new samples that predict the robot's position after the motion command. Sensor readings are incorporated by re-weighting these samples and normalizing the weights. Generally, it is good to add few random uniformly distributed samples as it helps the robot recover itself in cases where it has lost track of its position. In those cases, without these random samples, the robot will keep on re-sampling from an incorrect distribution and will never recover. The reason why it takes the filter multiple sensor readings to converge is that within a map, we might have dis-ambiguities due to symmetry in the map, which is what gives us a multi-modal posterior belief. [6]

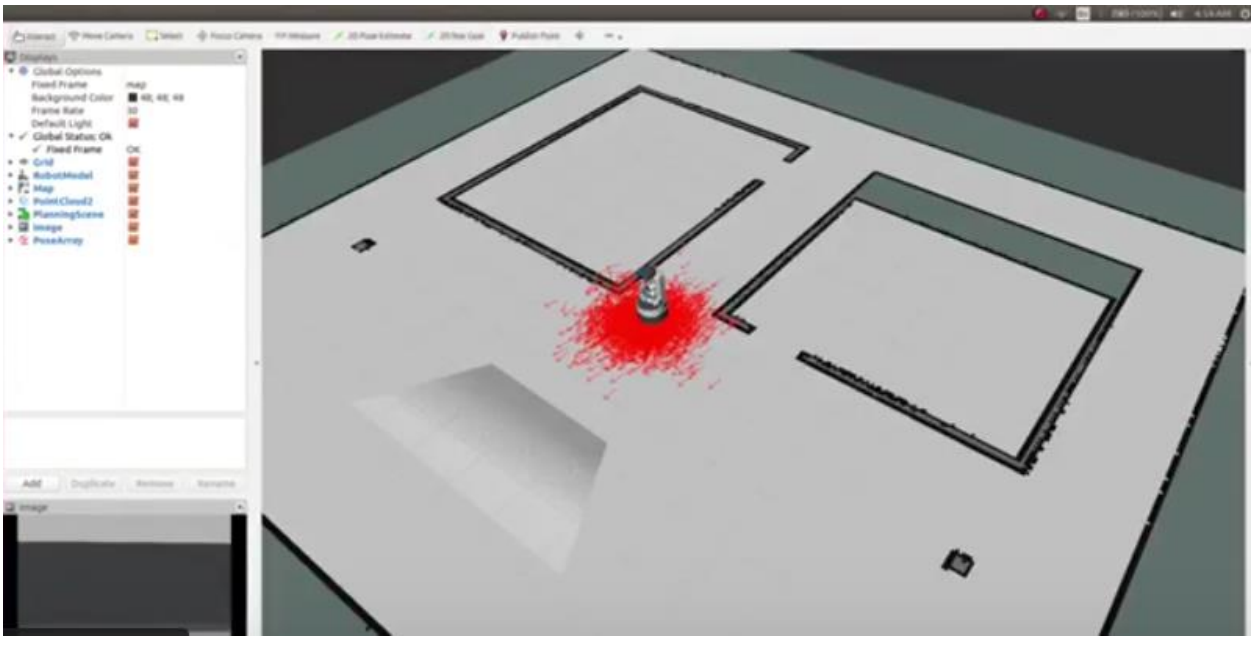


Figure: Fetch Robot Navigating in the environment

2.3) Movement

Once the robot has achieved successful implementation of localization and can identify its location, we begin to move it. Move Base is a package in ROS which supports movement of robots which can also be used with Fetch Robot. This package also allows to plan a path towards our final position if the the robot has access to the map of the workspace. We utilize this package to move the robot around the map.

Another type of movement that we must take into consideration is the movement of the movement of the end-effector and other actuators of the robot. Move-It! Is the package in ROS which is widely used to move Robots. The MoveIt! Package is integrated very well with the Fetch environment. We thus use the MoveIt! Package to manipulate the end effector of our robot. MoveIt! also has a path planning feature. Thus, given a set of target joint values, the package can create a path such that it avoids collisions. This feature is extremely beneficial; however, it comes with its drawbacks. There are multiple times when the MoveIt! Package cannot find a path to reach a set of joint values of objects are placed close together. This can create problems as the implementation of the entire task becomes extremely slow as the package tries to find a valid path to the desired object.

2.4) Perception

Perception is the process through which the robot senses and understand the surroundings around it. Perception is the quality which separates domestic general-purpose robots from industrial automation-based robots which follow predefined tasks and paths and continue to do the same task in a closed environment. The perception of a Fetch robot allows it to visually detect the surroundings and also capture the depth profile of its space. The fetch robot has RGB-D camera on its head mount and a laser scanner in its base. The laser scanner allows the robot to scan the surroundings while moving around and analyze how far are the objects around it. Since laser is more powerful, it allows the robot to scan longer distances. The data from the base laser scanner is used for SLAM purposes. It is thus helpful particularly in situations such as navigation and localization. The RGB-D camera mounted on its head mount has two purposes. The RGB camera captures images of the workspace in color and while the depth sensor captures a point cloud representing the depth of objects in different directions of the robot. The Depth Camera on the head mount has a smaller range and thus is used to objects in its immediate vicinity and classify them.

The RGBD camera is particularly useful for completing the tasks required by the robot once it has reached a specified location. The most important task once a robot reaches a given location is to recognize that there are objects in the vicinity of the robot, calculate how far are the objects, identify the class of the object and finally find the pose of the object. Once the pose of all the objects have been determined, we can identify the target joint values required to grasp an object. We can also determine a path which is collision free to the that joint state from the current joint state if we also know the poses of objects around our target object.

2.4.1) Object Detection

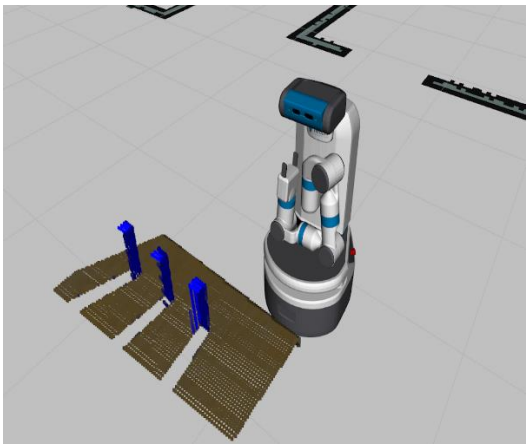
The first step in perception would thus be collecting the point cloud of the surrounding, processing it and detecting the objects in the environment. PCL was employed for the purpose of object detection and recognition.

The **Point Cloud Library** (**PCL**) is an open-source library of algorithms for point cloud processing tasks and 3D geometry processing, such as occur in three-dimensional computer vision. The library contains algorithms for feature estimation, surface reconstruction, 3D registration, model fitting, and segmentation.

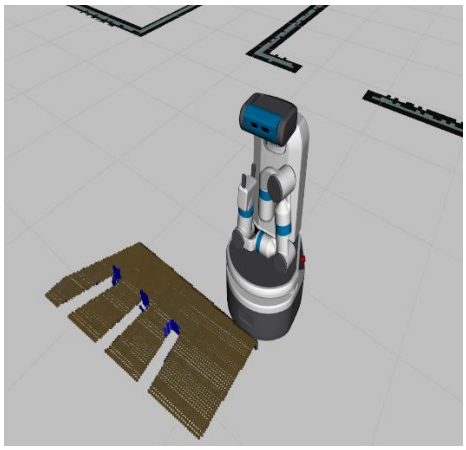
These algorithms have been used, for example, for perception in robotics to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract key points and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them.

Random sample consensus (**RANSAC**) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method. We used the RANSAC algorithm to separate the planar table from the objects.

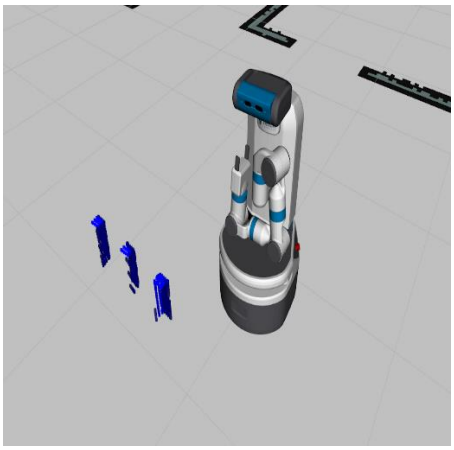
Passthrough Filter



Surface Segmentation



Object Segmentation



We then separated other objects by Euclidean distance. We also used it to fit the best shape around the object, which can be fed into graspit for grasping.
This helps us complete the task of detecting different objects, identifying pose and location.

2.5) Grasping

Once the robot has detected the objects around it, and performed segmentation, it is time to grasp the object. The Point Cloud library is gives segments the objects, gives us the location of the point cloud of the object in space. We now use this data to create a point from where the fetch robot can grasp the object. Currently, the fetch robot has a gripper which can only open and close. Since its grasp is only limited to one, we only need the pose of the object. Once the pose of the object has been determined, it is provided to MoveIt! which finds a position for the end-effector and also creates a path to the target joint states. The package then moves the end effector to the desired position, and then a command is issued to close the grippers and grasp the object.
Thus, in summary, since there is only one grasp available for the fetch robot, we do not need a separate package to find the grasp. We can directly use MoveIt! to find an ideal point to pick the object and move the end effector to that point by motion planning. However, at times, the MoveIt! Package cannot create a path to the object and that slows down the implementation of the program.

We do realize that this technique limits the kind of objects that we can pick up. Thus, we also tried another technique which is better, although a little restricted. We implemented DOPE to find the exact pose and category of objects which would allow us to create better poses for end effector. The advantage of DOPE is that we do not need a point cloud for finding the pose of the object. Only a 2D image suffices for object classification and pose. However, only a few objects have been classified in the technique. Since the technique does not require depth sensing, the selected objects have been extensively photographed and features have been mapped exhaustively. This is a long process and thus we can only train neural networks to find the pose for a few objects. However, DOPE can also separate objects hidden behind each other and find their poses. This is extremely powerful. Point CLoud Library cannot accomplish this task. However, DOPE also needs tremendous amount of CPU and GPU power, thus it cannot be used in all robots. The biggest drawback with DOPE was that it gave us access to datasets of only 6 objects. This limited our application severely.

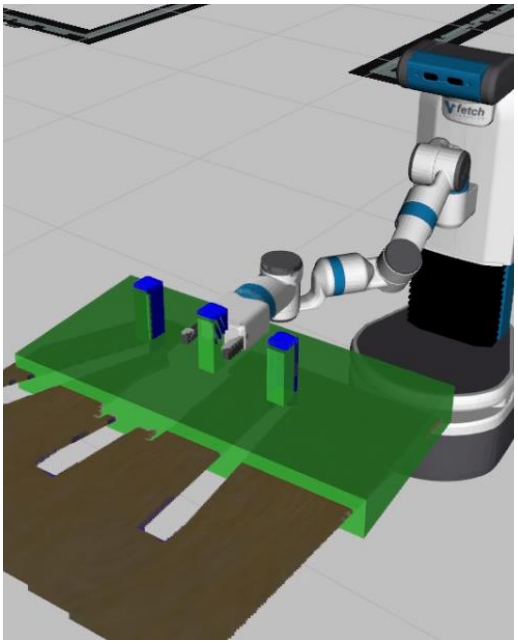


Figure: Fetch Robot Grasping objects from the scene

3) Concluding Remarks

3.1) Implementation Details

Now that we have been able to complete all the sub tasks such as mapping, navigation, movement, perception and grasping, we can attempt to complete a task which comprises of all these tasks. We thus have to integrate all the above described tasks. The final task can be divided into the following steps:

1. When left in an area, the robot will first map the area.
2. Once mapped, the robot will move to a point near the table using navigation to localize itself and move base to move the robot.
3. On reaching the table, the robot creates a point cloud of its surrounding. After processing the data, it recognizes the objects in the area. It then segments the objects and the table on which they are kept. Thus, it finally has the point clouds of the objects on the table.
4. It then creates a grasp for these objects and MoveIt! generates a path to move the end effector towards the final target joint values.
5. Once the robot has grasped the object, we can move the robot towards a bin. The robot will drop the object in the bin.

We divided the entire project into several phases and increased the complexity with each step. Our milestones during our project are detailed below:

1. The first step required us to move the robot the robot to accomplish mapping, navigation and be able to move to the table. This was accomplished in the mid-term report submission.

2. The next step was to make the robot detect a single object on the table and do segmentation on the scene.
3. This was followed by increasing the number of objects on the table to three. The robot then had to apply segmentation to this scene and also create poses for the object and be able to grasp them. The objects were simple geometrically shaped objects.
   VIDEO: https://www.youtube.com/watch?v=X5z-h0o_znU
4. This was followed by using complex objects. Thus, we began to use real life objects such as bottles and cans rather than simple cylinders or cuboids.
5. Finally, we had to place the grasped objects in a bin located in the scene. This is the last phase of the project.

3.2) Difficulties faced

1. Segmentation in PCL is a very challenging task. There are several parameters that we have to consider. While processing point clouds for segmentation, we need to provide values such as threshold values to determine the maximum distance between two data points of an object. Thus if the threshold value is too small, points from the same object might get classified as two different objects while if the threshold value is large, points from different objects might merge into one object. Moreover, the size of the object can also be limited. Thus, if an object is too large, then it will not be classified as an object only.
2. MoveIt! Fails to provide a path for the arm to move to the end-effector frame if the scene is clustered or the objects are close to each other. This acts as a limiter as the we cannot place the objects close to each other, which is very common in real life.
3. The data set provided by DOPE is severely limited (only 6 objects) and also dependent on the camera parameters which restricts the different objects that we can use.
4. Collecting data from several ROS Topics and processing it in real time and then publishing it on a single Topic was more difficult than what we first thought.
5. Map generation using Gmapping has to be done manually, i.e. we have to provide robots directions to move in manually as they create a space. It would be helpful if the robot moved independently. Frontier exploration is a technique which has this feature but we were unable to integrate it into our environment.

3.3) Future Scope

- Training on real life objects to give the robot enough data
- Object classification into categories and deciding which bin to put it in
- Extending it to perform room cleaning tasks
- Developing better recognition models for unseen models

REFERENCES:

[1] *https://openslam-org.github.io/gmapping.html*

[2] *Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard: Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters, IEEE Transactions on Robotics, Volume 23, pages 34-46, 2007.*

[3] *Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard: Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling, In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), 2005.*

[4] *Loannis M. Rekleitis. "A Particle Filter Tutorial for Mobile Robot Localization." Centre for Intelligent Machines, McGill University, Tech. Rep. TR-CIM-04-02 (2004).*

[5] *https://en.wikipedia.org/wiki/Monte_Carlo_localization*

[6] *http://roboticsknowledgebase.com/wiki/state-estimation/adaptive-monte-carlo-localization*

[7] *Data Fitting and Uncertainty, T. Strutz, Springer Vieweg (2nd edition, 2016)*

[8] GITHUB: *https://github.com/googlecartographer/cartographer*