CSE591 Perception in Robotics, Final Project Report
Team 14 - RoboHelper

Anik Jha **Saurabh Goyal**

Matthew Cavorsi **Jae Hyo Jeong**

Abstract

The objective of the project is to create a complete product, which could replace the exhaustive task of monitoring the inventory for large stores with minimal human intervention. In the process, bot automatically or manually navigates through the store to identify items on the shelves, capture shelf image, and process image to set an out-of-stock flag for an item. A sincere approach has also been made to make it ready for real-life scenario, wherein new items are constantly being added to the repository. Overall, with certain assumptions and limitations, the product was successfully able to categorize the items in the repository as In-Stock and Out-of-stock items.

1 Introduction

In a retail store usually a store employee monitors inventory, noting in or out of stock items, an action called facing the shelves. This labour intensive task is expensive and time consuming, if delayed; it can result in losses of sales and customers to the store and companies providing products. This project aims to explore applications of the robots to check the status of the inventory for mega stores including the likes of Walmart, Target among others. This project has been inspired by the advancements in Amazon Pick-up challenge and signicant improvements by Bossa Nova Robotics for achieving a similar objective.

In order to accomplish the objective, the project tries to make use of multiple computer vision modules including Facial recognition for authenticating the system for verified users, Gesture recognition system for enabling the user to move the robot to a specified location, SLAM to autonomously move the robot to capture the inventory of the entire store with minimal human intervention, and Object recognition system to accurately detect and recognize the objects in the vicinity and flag them accordingly based on their availability.

The sequence of actions associated with the project is depicted as a flow chart in figure 1. Further, section 2 describes the implementation of different modules of the project. Subsequently, the experimental procedure along with system requirements are briefly described in section 3. For the ease of understanding both sections 2 and 3 are split further according to their functionality. Finally, section 4 discusses the conclusion along with the possible scopes of improvement for the project.

2 Approach

The overall approach of the project and implementation of different modules are explained in this section.

New user authentication: This module provides a functionality to enable new users to add their faces to the face recognition module through a default password.

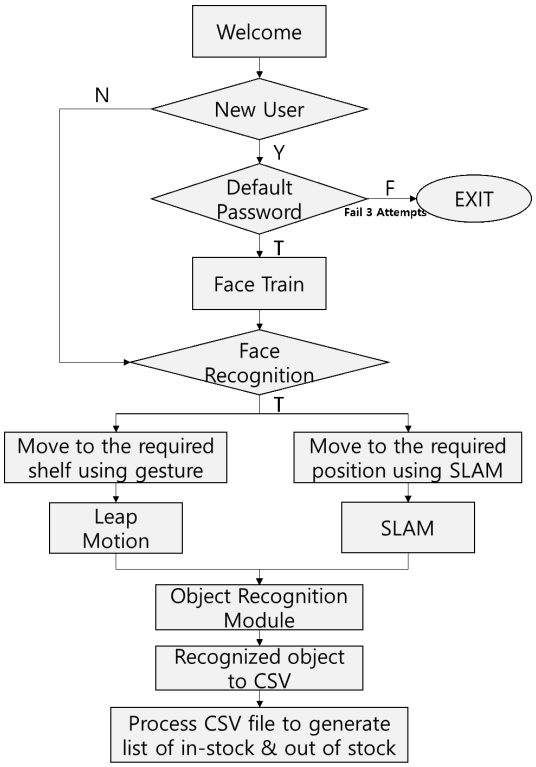

Figure 1: Project flow chart

Face Recognition: This module is used as a security means. More secure than a password that someone could figure out or learn. Using a user's face to unlock and control the system ensures that only those allowed to control the system can do so. The facial recognition was accomplished by using the ROS package 'face_recognition'. The package works by acquiring training images of a users face. A Haar-Cascade classifier detects faces in a live video stream and captures images of the face. The training images are compiled into an Eigen faces database for later use. When the module is run, the package loads the trained images and compares them to what it sees in the current video stream. If the face presented to it matches one of the faces, with at least 88% confidence, it will conclude that it is a certified user and allow them access to the system. Faces are compared to the trained images by horizontal and vertical features that it finds are similar between the images and face on screen. Figure 2 shows an example of the facial recognition module finding a face, and having enough confidence to conclude it as 'anik', one of the certified users.

SLAM: To map the room, the ROS package 'gmapping' can be used, along with the ROS Turtlebot keyboard teleop node. To map the room, the gmapping package is launched. Gmapping will take distances measured by the two on board front facing cameras and convert them into walls and obstacles in the robots surroundings. A user can drive the robot around using the keyboard teleop node (or hand gestures) to allow the cameras to see different parts of the room. After driving around enough, eventually the gmapping can put together a fairly complete map of the room. After saving that map, the robot can navigate through the room autonomously with the help of the preinstalled Turtlebot amcl_demo launch file linked with the newly acquired map.

Autonomous Navigation: The robot has the ability to move by user input through hand gestures and leap motion, or to move autonomously. In order to move autonomously, the robot must first have a map of the room, such as the one generated by the SLAM module. It is necessary to pull the map up in RVIZ to start in order to give the robot an initial estimate of its pose in the room. After clicking relatively where the robot is in the map, it can use camera feedback to further and more precisely estimate its exact location in the space. Once localized, the robot can plan a path to any Cartesian location given to it, while avoiding obstacles, as long as the location given to it is defined within

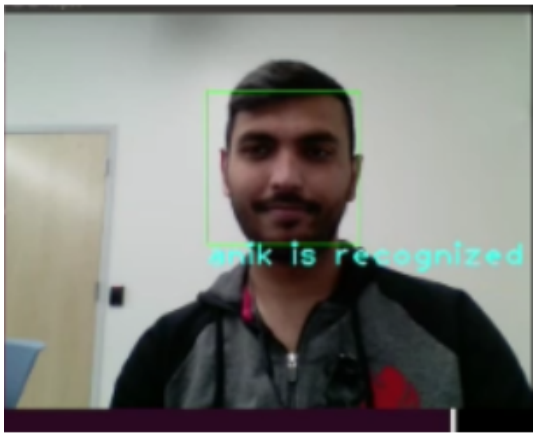

Figure 2: System is able to recognize a certified user

the map. To achieve the autonomous navigation for the purpose of this project, a python code is executed that gave the robot successive points to go to (points A, B, C, etc.). In terms of this project, point A and B are two endpoints with minimal distance from the shelf. Bot navigates from point A to B simultaneously scans the shelf and update the inventory database.Then point C would be the end where it would essentially return to home, or wherever it needs to be next. A larger grocery store would require many more points, but all can still easily be reached autonomously through this same process. Figure 3 is an image of the robot, localized in its map on RVIZ, planning a route and moving to point A autonomously.

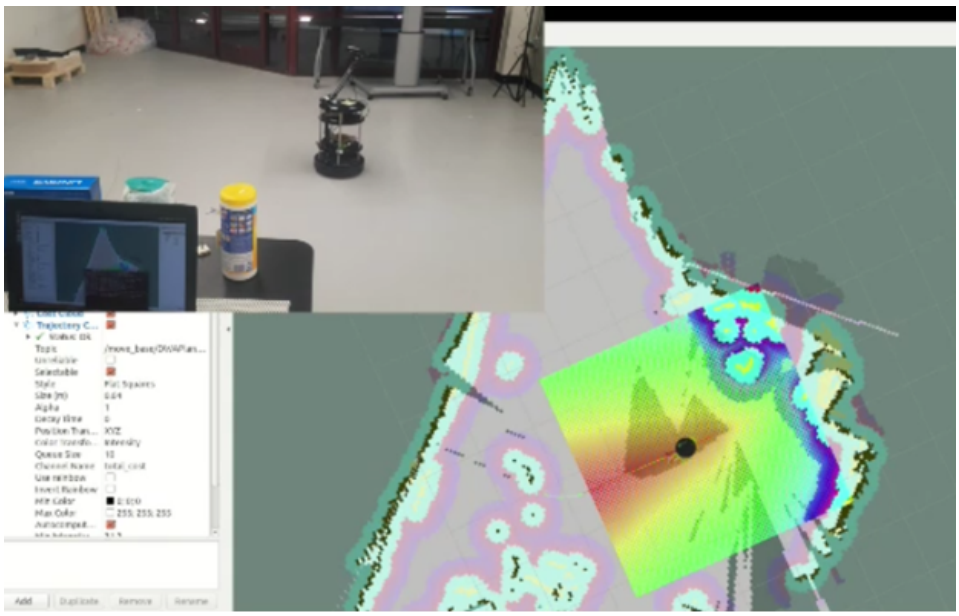

Figure 3: Moving the robot autonomously in a given space

Leap Motion: A Leap Motion sensor is used by the user to intuitively control the robot using hand gestures. It tracks hand positions and movements and then collects data through the sensor. The collected data is visualized as virtual hands in Diagnostic Visualizer. The hand positions(pitch, yaw, roll) are converted into ROS Twist messages which publishes the topic called \vel_mux\input\teleop which in turn controls the linear and angular movement of bot. According to the range of hand positions, the python code publish a particular Twist value (command velocity). The robot is operated by simple hand gestures. 1) If the hand is pitched down in 30°, the robot moves forward, and if

the hand is pitched up in 30°, the robot moves backward. 2) If the hand rotates anticlockwise in 30°, the robot rotates anticlockwise and subsequently if the hand rotates clockwise in 30°, the robot rotates clockwise. 3) The robot can be stopped either by placing the hand parallel to the surface of the sensor or by taking it out of the range of the sensor. Figure 4 shows an example of using hand gestures for moving the robot. Due to simultaneous live feed video from the camera it becomes very convenient for users to operate the turtlebot.

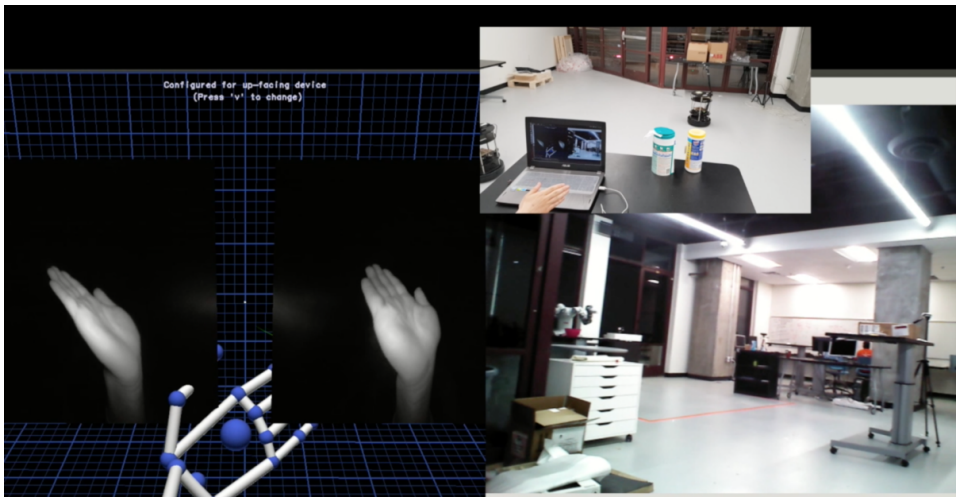

Figure 4: Using gesture recognition to move the robot

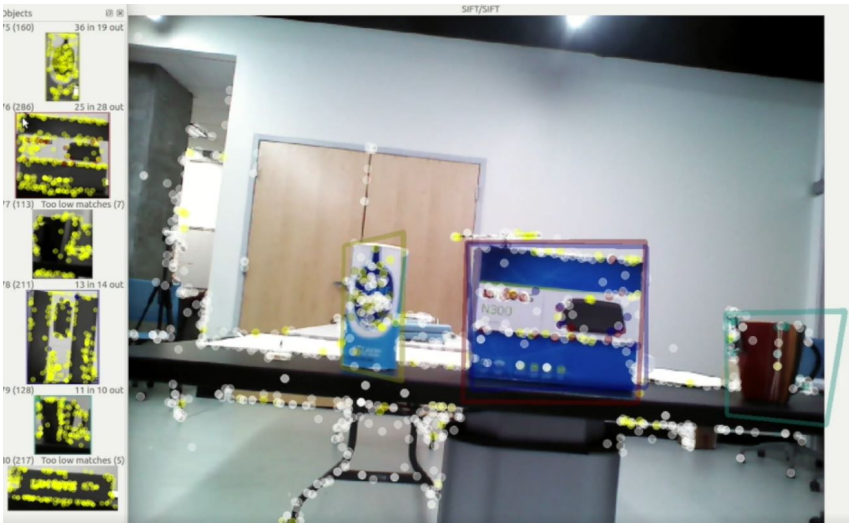

Figure 5: System is able to recognize pre-trained objects

Object Recognition: This module is activated/launched once the robot reaches the vicinity of the shelves with the items. The launch can be autonomous by associating it with the location of the shelves, as in the case of autonomous bot movement. It can also be launched manually when the motion of bot is controlled using hand gestures. For the core object recognition task, the system makes use of the find_object_2d ROS package. This package uses 'Feature2D' module of the OpenCV 2.4.17 to provide enhanced accuracy to the system. It also provides an option to use different detectors and descriptors alternatively, compare their performances and then select the best possible combination for the task in hand. In this project, after comparing the performances of different descriptors, SIFT is preferred due to its robustness to scale and rotational variations. Figure 5 depicts the process of detecting and recognizing the pre-trained objects.

3 Experiments

This section deals with the data acquisition steps for different modules, system requirements to implement this project as well as the experimental results associated with individual modules.

3.1 Data Collection

Face Recognition: The only data that needed to be collected for the facial recognition module is the trained images for each certified user of the system so that the module can correctly identify the right faces.

Autonomous Navigation and SLAM: Data collected for the autonomous navigation are the map and points of interest on the map. The map is mostly an arranged set of data points a certain distance from the robot. The points make up walls and other obstacles the robot needs to avoid. After the map is built, driving the robot to specific points that we would want it to later navigate to autonomously is necessary in order to get the Cartesian coordinates of those points, which will be shown on the RVIZ screen, to be put into the python code.

Leap Motion: The data for hand position and orientation(pitch, yaw, roll) is collected by Leap Motion sensor and, the data is converted into messages so that user can control robot using hand gesture.

Object Recognition: For the objects to be successfully recognized by the system, it is required to add the images of the objects in the repository of images. This is be achieved at the initial stage after the user is verified by the face recognition unit. Post verification, the user is asked to add multiple images of each object in sub-folders, named after the items themselves, in the root folder. It is extremely important to add the images in the aforementioned specified manner for the system to work accurately. This task is performed every time one or more new objects are added to the list.

3.2 ROS System Setup

The project is built on Ubuntu 14.04 with ROS version Indigo. It makes use of in-built ROS packages including face_recognition, gmapping, leap_motion and find_object_2d. It also uses Turtlebot 2 with Asus camera installed over it.

3.3 Experimental Results

Face Recognition: The facial recognition module worked good enough for its purpose. It struggled sometimes to recognize a trained users face with more than 88% confidence, but after a few seconds of moving the face around it was able to recognize it. Because it sometimes struggled to even recognize the trained face with enough confidence, it is safe to say it will not recognize untrained faces confidently enough to allow them access, which is the main security purpose of the facial recognition module.

Autonomous Navigation and SLAM: The results of the autonomous navigation varied based on the accuracy of the initial pose estimate given to the robot by the user. The robot was not very good at correcting its pose when driving around the room, so a poor initial estimate led to poor navigation. It is recommended that the robot builds the map of the room, and be localized during this process, and then autonomous navigation run from there without ever having to re-estimate its pose, to ensure the highest accuracy.

Leap Motion: , As leap motion is unstable on Ubuntu due to missing dependencies installation was really difficult. Moving the hand back and forth was easy for the robot to recognize and most accurately worked. However, as pitch and roll angle changes with hand movement sometimes range of these angles overlap resulting both linear and angular movement. To minimize this error the optimum range of pitch and yaw angles was obtained, which was adapted to be pitch = -30 to 30 and roll = 120 to -130. To obtain more accurate results, neuro-gesture package was implemented by providing more than 100 training datasets per gesture

Object Recognition: The system was successfully able to recognize pre-trained objects, provided the lighting conditions and the distance between the robot and the items were adequate. The perfor-

mance of module was also impacted by the changing backgrounds of the images of items. Hence, for maintaining a higher accuracy of module, it was also required to add multiple pictures of the same object with varying backgrounds, preferably plane, and from varying viewpoints. Following the detection and recognition of objects, this unit creates a list of objects detected during its operation. This list is further used to determine the items that are not present in the inventory and needs to be replenished. This final list is crated with a timestamp for a better log of the check.

4 Conclusion and Future Scope

In this project, we addressed the problem of labour intensive task of monitoring inventory on shelves in large retails stores. The main focus of our work is done to develop a complete product to check the status of the inventory for mega stores, with minimum human interventions. To a large extent it was a successful venture to incorporate different modules to enhance the usability of the system. We were able to navigate autonomously in the store to scan the shelves and generate product database to flag items which are out of stock. Manual control system through leap sensor is also embodied to remotely navigate the robot thereby facilitating in generating inventory map and location of shelves. We were also able to deal with scenarios such as frequently adding new items and/or removing old items from the store repository. Since the project has numerous modules interconnected with each other, it was important to take into account the errors due to each of these modules. Considering individual modules, for facial recognition, the confidence threshold was kept at 88% to make sure that authorization is not given to unrecognized user. Similarly to cope up with the position estimation error when going with autonomous/manual motion of the robot, object recognition module is launched slightly earlier. The images of the objects are also taken in such a way that even if the robot is at slightly larger distance, they are recognized accurately.

Although, we were successful in launching a few interconnected modules sequentially without human intervention, we were unable to extend it for all modules and the system still needs external support to launch last couple of modules. This would be something to look forward to in follow-up projects. We also tried to include the detection of low-stocks for items into the project by considering different scenarios using the reference from the internet, We, however were unable to include the functionality. We believe it would be interesting to consider this in future.

5 Acknowledgement

This project would not have been possible without the guidance of Dr.Yezhou Yang. We would also like to thank Varun Jammula and Shashank Tadepalli for their constant support throughout the project.

6 References

1. Planogram Assisted Inventory System and Method, a patent by Bossa Nova Robotics IP Inc,https://patents.google.com/patent/US20170286773A1/en
2. Leap-motion module in ROS, http://wiki.ros.org/leap_motion
3. 2-D object recognition module in ROS, http://wiki.ros.org/find_object_2d
4. Mapping the space using gmapping module in ROS, http://wiki.ros.org/gmapping
5. ROS Robotics Projects by Lentin Joseph