Autonomous Navigation and Visual Analysis with Beam Robot

Utkarsh Patel, Emre Hatay, Ghazal Zand

May 10, 2017

Abstract

We are improving the autonomous navigation of the Beam Robot, and also reporting the detected human positions using the Kinect camera and the Beam robot. In this paper, we describe the problems we faced while transforming a closed platform telepresence robot into an autonomous robot, and also explain our approach on face detection. We successfully made the Beam robot navigate from the recreation center to the student center of Cleveland State University without any human involvement. For visual analysis, we developed a program using Cob People Detection ROS package. Our future plan with this project is to have a team of fully autonomous Beam robots that can become permanent part of the Cleveland State University's Engineering building.

1 Background

We have used the Beam robot (Figure 1) for the entire project. Beam is a telepresence robot created by Suitable Technologies. Beam was originally developed by Willow Garage as Texai Remote Presence System, and spun out with most of Willow Garage team to build it as a commercial product. Our end goal with Beam is to completely transform it into a research robot for autonomous navigation and human robot interaction. Using Beam as a research platform has its own advantages and disadvantages. The advantage is that Beam has a human friendly hardware design, excellent driving control and cheaper price compared to other human-sized autonomous robots. Beam Robot has the same price as TurtleBot 2. The disadvantage is that it does not support ROS and it does not have a big user community. Beam is being used as a research robot by only few people in the world, and the company is not supporting any research projects. Beam runs on a Linux-based operating system with no source code provided. Beam does not come with autonomous functionality out of the box, but with the help of rosbeam package [3] and the open source nature of Linux, we were able to get full control of its motor. The rosbeam package makes it possible to send custom driving instruction to robot base, and to teleoperate Beam from any laptop.

2 Introduction

Our project had two primary objectives. The first objective was to make Beam Robot navigate for longer distance without any human involvement. Before we started this

Figure 1: Beam Robot

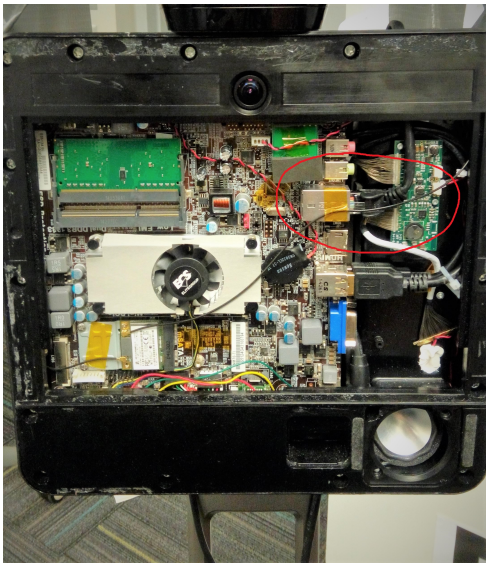

Figure 2: Beam's screen has been taken off to access the Ethernet port inside its Head. The circled areas highlights the Ethernet port.

project, we were able to teleoperate Beam using rosbeam package. We had already installed ROS navigation stack on a on-board laptop. We had also made the map of the third floor of the engineering building and used that map to test the autonomous navigation few times. Beam was able to move autonomously for 20-30 meters, and it was also able to avoid obstacles once in a while, but the navigation was unreliable in longer distances, and the Beam's movement was also unstable. It was impossible for Beam to navigate more than 30 meters without any human involvement. After observing the navigation and analyzing the problem, we concluded that in order to improve Beam's navigation, we would have to improve Beam's odometry, stabilize Beam's motion, and find a way to replace the unreliable wireless connection between the on-board laptop and Beam.

The second objective of our project was to create a visual analysis report using Beam. In order to make an autonomous robot trustworthy for humans, it should be able to move purposefully, without human intervention, and also should be able to provide a navigation report to the user to justify its decisions or actions. In the visual analysis report, we decided to include the number of humans observed by Beam during navigation and their corresponding locations. Human detection is a challenging problem because of the large variations in visual appearance. Generally, a human detector mainly has two components: a feature extraction algorithm that extracts important features from the input image frames, and a detection model that locates the target human bodies according to the computed vector. In fact, feature extraction is a fundamental process for human detection. In this project, we will use combination of visual features, such as Haar features to encode the input image frames as feature vectors. Initially, we planned to run the human detection process while Beam is navigating autonomously. But unfortunately, due to insufficient processing power, we had to run the detection process in the teleoperation mode.

We also had one secondary objective that Beam should be able to send feedback to its operator. In the case where an autonomous robot gets stuck and cannot move without

human help, the robot should be able to call for human help and notify its operator. To this end, we decided to make Beam email us whenever it gets stuck and cannot reach to its goal.

3 Setup

As stated above, Beam robot does not support ROS out of the box. The processor of the robot is also not powerful enough to run the ROS navigation stack in real time. Because of that, we attached a laptop on the robot base and installed the navigation stack on the laptop. Initially, the laptop was connected to robot via WIFI. From the navigation experiments with Beam, we found that the laptop was losing connection with the robot once in a while, which was interrupting the navigation. We also realized that the wireless connection cannot be used to navigate Beam inside elevators. To overcome these limitations, we decided to connect the laptop to robot via Ethernet cable. Using Ethernet cable was not a straightforward process since Beam robot does not have an external Ethernet port. Beam has two external USB ports, so we tried using USB to Ethernet adapters, but Beam's operating system did not let us install the drivers for USB adapters. Therefore, we decided to take out Beam's screen in order to access the Ethernet port on Beam's internal processor (Figure 2).

We are using an Asus Xtion Pro depth camera as the observation source for navigation stack. The camera is attached on the base of Beam and connected with the laptop via USB. We are using Microsoft Kinect sensor for face detection. The kinect is attached on top of the robot. Unlike Asus Xtion which is powered by USB, kinect needs external power input. To this end, we have built an adjustable regulator in order to connect any type of sensor that requires voltage more than 5V (USB). The circuitry that powers up the computer behind Beam's screen is located inside its base. Analyzing the circuitry, we found a safe port that we can use to connect our regulator circuit. Although we could only have 92.5% efficiency from our circuit, 11.2 V was enough to power up the kinect directly from beam's base.

The entire eco system of Beam's navigation is illustrated in Figure 3. One of the Beam's internal processes sends the driving commands to the motor board, and also receives the wheel encoder values from the motor board. The rosbeam package intercepts the read-write system calls between the Beam's internal process and the serial port device connected to the motor board, and provides us the way to send custom driving commands, and to get the wheel encoder values. The rosmaster and the ROS navigation nodes run on the laptop. The rosbeam node runs on Beam, which publishes the encoder values and subscribes to the cmd_vel topic. The beam_odom node runs on the laptop, which subscribes to the wheel encoder topic and publishes the actual odometry values of Beam.

4 Improving the Navigation

Beam's navigation had three major issues when we started this project. The first issue was that the on-board laptop kept getting disconnected from Beam during navigation whenever the robot entered the "dead WIFI spot" in the building. Since the navigation stack was installed on the laptop, this issue was causing the robot to not move whenever the connection gets interrupted. As explained in the Setup section, this issue was solved by replacing the wireless connection with Ethernet connection.

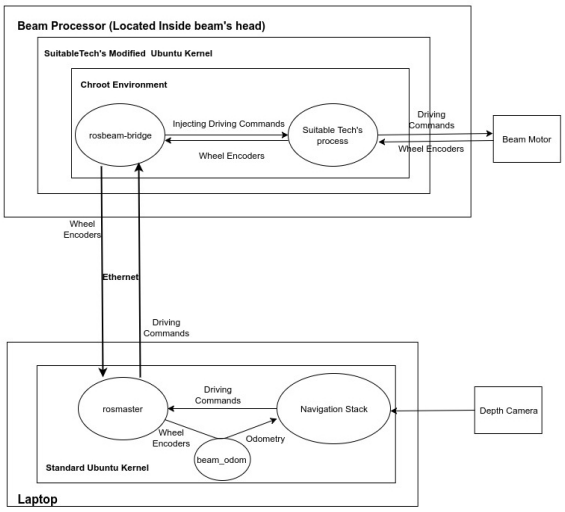

Figure 3: Navigation Setup

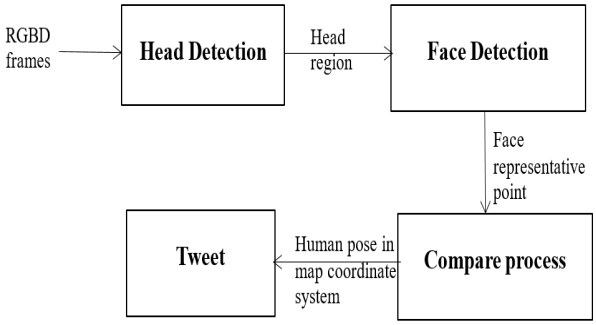

Figure 4: Face Detection Setup

The second major issue with Beam's navigation was that the odometry was inaccurate. Ideally, the localization package should fix the error in odometry and provide a corrected estimate of the robot's position with respect to a global reference frame by matching the source of odometry with the camera scan. The error in the odometry was so large that the localization package was unable to provide a good estimate of robot's position. The problem was mainly caused by two factors. The first factor was the error in the odometry calculations. We were using a ros package called differential_drive [] to calculate the odometry. The package was converting the Euler angles into quaternion the wrong way. The old code [1] and the corrected code is shown below. The second factor was the error in the odometry source (wheel encoders). Due to the fact that Beam's long neck was causing high inertia over its base, the front wheels were getting lifted from the ground during navigation. When the front wheels get lifted from the ground, the speed of the front wheels and hence the speed of the encoders increases in a short period of time. In such situations, the physical position of the robot does not change, but the odometry of the robot changes. In order to fix this issue, we have defined a threshold for the encoder speed. If the speed goes above the threshold than we stop increasing the encoder values. To further improve the odometry, we used the accelerometer instead of wheel encoders to calculate Beam's orientation. The accelerometer is located inside the base of the Beam. The rosbeam package is providing us the accelerometer reading in unrecognized unit. After observing the change in the accelerometer value with respect to the change in the physical orientation of Beam, we created a linear function to convert the accelerometer value into radians. We are still using wheel encoders for calculating the linear distance.

[1]The full source code of the original differential_drive package can be found at https://code.google.com/archive/p/differential-drive/source/default/source

Old Calculation

```
quaternion = Quaternion()
quaternion.x = 0.0
quaternion.y = 0.0
quaternion.z = sin(yaw/2)
quaternion.w = cos(yaw/2)
```

Corrected Calculation

```
th = tf.transformations
  .quaternion_from_euler(0,0,yaw)

quaternion = Quaternion(th[0],th[1],
  th[2],th[3])
```

The third major issue with Beam's navigation was that the Beam's motion was unstable and sometimes not human-friendly. Beam's body used to shake regardless of its the speed. The unstable motion was being caused by the incorrect position of robot_base frame and inadjusted parameters of dwa_local_planner [1] package. The robot_base_frame was attached to the physical center of the robot before we started this projects. After reading the documentation [4], we realized that the robot_base_frame should be attached to the rotational center, and not the physical center. Beam is a differential drive robot whose rotational center is in the front. Correcting the location of robot_base_frame dramatically improved the stabilization of Beam's motion. The dwa local planner's parameters were tuned by trial and error approach. The most significant changes in the parameters are listed below. Increasing the acceleration limits made the dwa local planner to send more consistent velocities to robot base. Increasing the number of samples taken when exploring the velocity space caused the dwa local planner to choose better trajectories. The forward_point_distance is defined as the distance from the center point of the robot to place an additional scoring point. Since, the rotational center of Beam is not same as its physical center, the default value of forward_point_distance was causing the dwa local planner to drop a lot of valid trajectories [2].

Old Parameters

```
acc_lim_x: 2.0
acc_lim_theta: 3.2
vx_samples: 6
vtheta_samples: 20
forward_point_distance: 0.325
```

Tuned Parameters

```
acc_lim_x: 8.0
acc_lim_theta: 5.0
vx_samples: 20
vtheta_samples: 40
forward_point_distance: 0.0
```

5 Experiments

We tested the autonomous navigation by navigating Beam from the recreation center to the student center through the inner link. Before we started the project, we already had the map from the recreation center to the end of engineering building (Figure 5). We built another map using the rtabmap_ros package [6] from the science building to the student center, and merged both maps using a image editor software. Beam is not always able to navigate from the one end of the map to the other end. The main reason of failure is the limitation of our sensors. We only have one RGBD sensor, which is not enough for navigating in longer distances. In our trials, we have observed that the camera does not detect objects if the object is closer than the minimum range of the camera. Despite of this limitation, Beam was able to navigate from the recreation center to the student center few times.

We have created a database to store information of each navigation trial. A separate

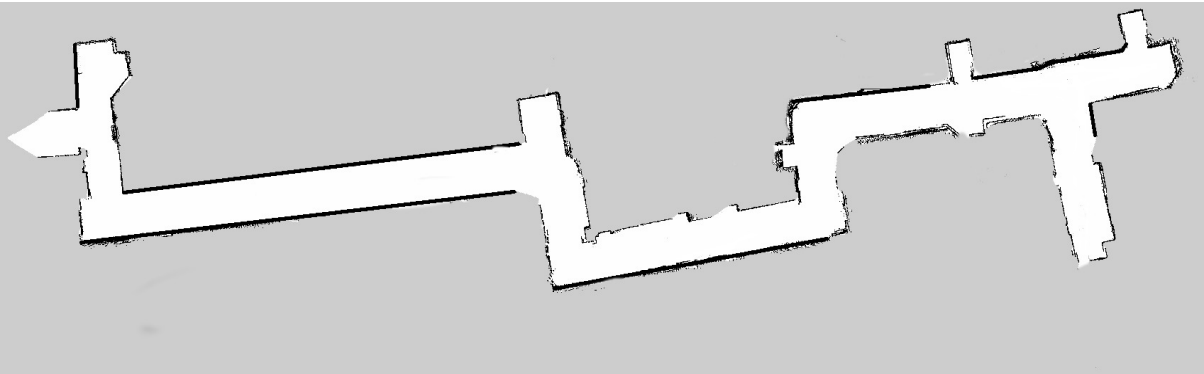

Figure 5: Map from the recreation center to the end of the third floor of engineering building

node runs on the laptop, which subscribes to the odom topic and calculates the trial information such as the total distance traveled, the average speed, the starting time, the end time, the initial position and the final position with respect to the global map frame. As of today, the beam has traveled four kilometers in fully autonomous mode.

6 Human Detection and Reporting

In the second phase of our implementation (illustrated in Figure 4), to make the robot detect humans, we utilized the Viola-Jones approach that detects the human faces [8]. We developed a program using Cob People Detection ROS package[1]. By using RGBD image frames as Head Detection block input, to do fast human face detection, it estimates the some regions as human head location. Then, in each region, to find the exact position of human face, it applies a combination of filters called Haar features [9]. As the face detection block output, we got the face representative point. After transforming the position of detected people from the camera coordinate system to the global position in the map coordinate, we needed to make sure that it does not report the same person more than once. To ignore multiple detection of one person, we made the node (the compare process block in Figure 4) to calculate the euclidean distance between every person newly found and previous ones. We compare this distance with a threshold that we found to be optimal. It means, a new human does not appear in close neighborhood of perviously found humans. If current position of the detected person is acceptable, the robot tweets the position. The Figure 6 is the visual output of human detection. It is also shown that how the map is updated by marking the human position in the map coordinate system.

The Viola-Jones object detection approach, proposed by Paul Viola and Michael Jones, is mostly used for face detection implemented in OpenCV and is an on-line approach that can process in real-time. The problem we have with this approach is the lack of independency to environmental conditions which sometimes lead to unexpected unreliability in the output . This approach extracts Haar features which can be briefly explained to be the sum of image pixels within rectangular areas with relatively shaded parts. The net score of any given feature is calculated by differences of the summations between darker and brighter parts. Although this detector is an efficient one that is invariant to scale and location (since instead of scaling the image itself, it scales the

[1]The tutorial of this ROS package can be found at http://wiki.ros.org/cob_people_detection

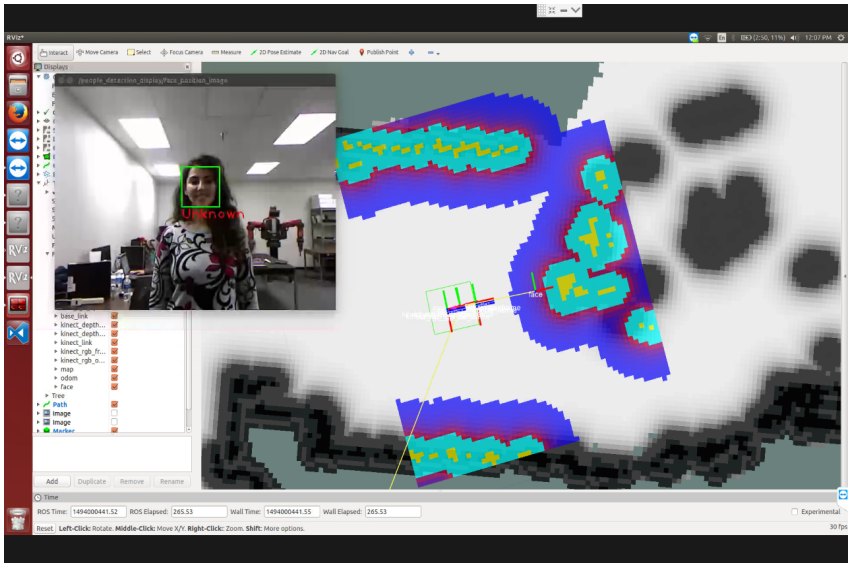

Figure 6: Face Detection using the kinect mounted on Beam. The position of the human is marked in the map.

features), it is sensitive to lighting conditions, and also is most effective only on frontal images of faces.

7 Conclusion and Future Work

Making the Beam robot navigate for longer distance and making it detect humans was the first step toward transforming Beam robot into a research robot for human-robot interaction and autonomous navigation. Our next goal is to make Beam robot navigate autonomously and continuously for the entire day in the engineering building. To this end, we are trying to make Beam dock itself to its charger whenever the battery is low. We are also going to make a scheduling application that will allow users to schedule Beam at a specific location and at a specific time. Additionally, we plan to create a human appearance frequency map, so that when the Beam robot needs human help, it knows in which position the probability of finding a human is high. For this purpose, we only have to make robot navigate for several times, while detecting and reporting human positions.

References

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

[2] http://answers.ros.org/question/201535/move_basedwa-parameters-for-a-large-robot-w

[3] https://github.com/xlz/rosbeam

[4] http://wiki.ros.org/navigation/Tutorials/RobotSetup/TF

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

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

[7] wiki.ros.org/differential_drive

[8] Viola, Paul, and Michael J. Jones. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.

[9] Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition. (2001). CVPR. Proceedings of the IEEE Computer Society Conference on (Vol. 1, pp. I-I). IEEE.