Extending Navigation to Multiple Floors

One of the goals of the Building-Wide Intelligence (BWI) is to achieve cohabitation of the Gates Dell Complex robots and humans. In order to achieve this cohabitation it is necessary for the Segbots, a robotic platform built with in-house equipment and segway bases, to be able to navigate throughout the entire building without limitation to merely the third floor. Currently, the robot is not capable of traveling between floors without a large amount of human assistance. In order to change this, the robots need the ability to use the elevator, which will be the main focus of our project.

One of the major obstacles the robot currently faces when traveling between floors is the inability to call the elevator effectively without the assistance of humans. Its current limitations only allow it to enter the elevator given that the door is held open by a human. If the elevator is crowded, the robot will not enter into it properly. The robot is also unable to press the buttons inside the elevator and recognize when it's on the correct floor. Through the course of our project we aim to get rid of, or at least reduce these limitations by improving the robot’s ability to call the elevator, enter, and interact with humans.

`Background and Related Work.

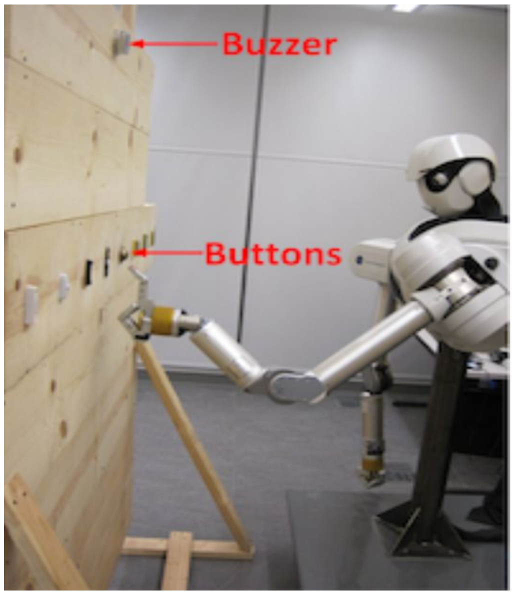
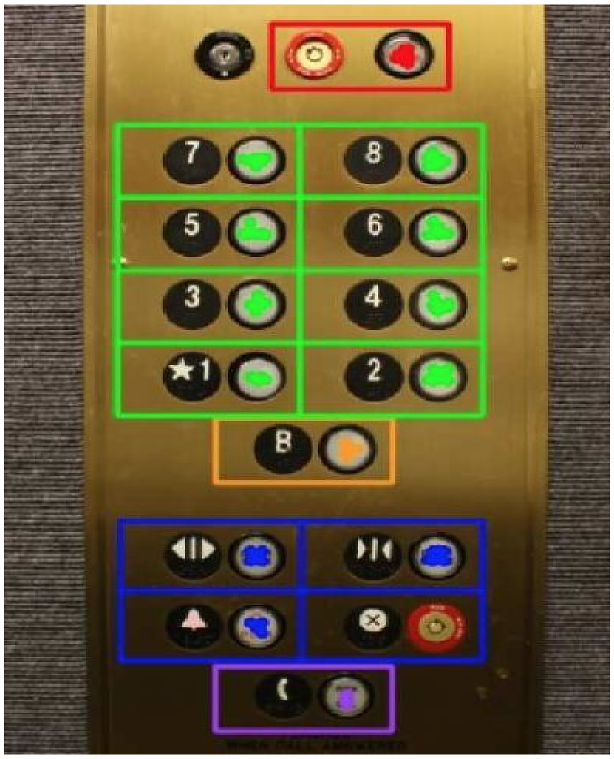


Fig (a). *The humanoid robot is pictured trying to press a button after detecting and localizing the button in its fron*t. Fig (b). *The red area is the functional part of the button which is based on the data from previous successful tests*.

For our work on improving the autonomy of the robots in the GDC, we considered a few projects that carried out research that is similar to ours. Both papers included locating different buttons and pressing the buttons in different circumstances. The first paper was published by researchers at Iowa State university. The paper featured a robot with a humanoid upper-torso that implemented a learning algorithm that helped the robot locate the buttons on different types of doorbells and successfully press them most times. The buttons were fixed into a sliding panel to vary the button in front of the robot. The robot arm was set in
its resting position before any trial to account for concave buttons. The robot either created a random plan to reach the button after locating it on the panel or it chose the path that was the most likely to succeed from previous trials or it chose the least certain path. Of all the three approaches, the most successful after about 1000 tries for each button was the uncertainty-driven method. After the button is pressed, some sort of feedback, either auditory, visual or other proprioceptive feedback, is expected by the robot for confirmation of a successful task.

The second paper which was published by researchers at the Stanford University focused on making a robot operate any elevator it comes in contact with. The algorithm used to solve this problem was broken up into 3 phases – detecting the buttons which was achieved using sliding-window object detection and optical character recognition, localization and labelling them. To do this, they made use of Expectation-Maximization to split the buttons into one or more grids which include the buttons and the labels to the left of the buttons. After the buttons are assigned to a grid, they are labelled using Hidden Markov Model and then it is stored. Now, the robot can make a plan to press the button it needs since it has located it on the panel. Although this paper didn’t make any provisions for feedback to confirm success, we plan to implement it by looking out for visual changes to the environment like changes in color.

From these two papers, a few outstanding problems were mentioned like the problem of lighting and size of the elevators. Since the camera on the robot will be publishing the image in 2D, the lighting in the elevator will affect the way the robot sees the button and the panel hence it may be hard for the detection and localization process to be completed. Also, if a robot is full, the robot will not go into the elevator since it is programmed to avoid obstacles which is what the robot sees a crowded space as. These papers give us solutions to some problems like finding the buttons and labelling them and they also draw our attention to the unresolved problems like the crowded elevators so we can find a solution to them.



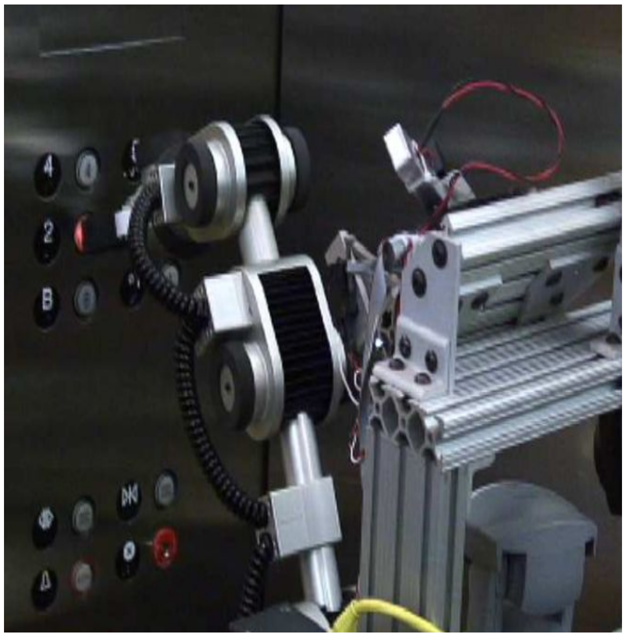
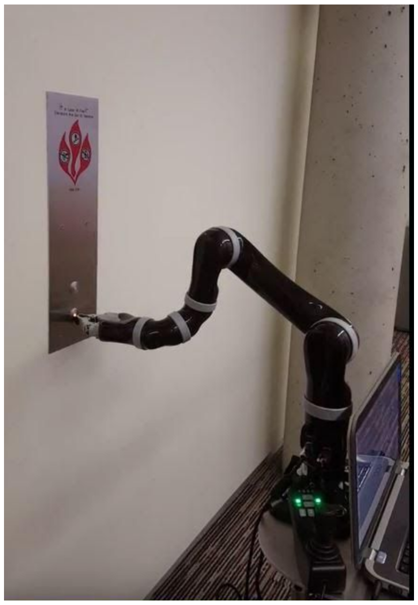


Fig (c) *A picture of a robot using a red pointer to mark its desired point of contact with the button*. Fig (d) *The buttons are divided into grids and the center of each button is marked*.

Technical approach

In order to allow the robot to push the elevator button we need to consider its current capabilities. The code currently uses a node which detects the elevator panel using a custom color filter and the PCL passthrough filter. This filters out everything in a cone of the robot's vision. Then, the same node uses a heuristic model to get the centroid of the extracted cluster (elevator panel). The model uses the minimum y point of the point cloud, which represents the bottom of the panel, to find the vertical position of the button. We then iterate over the cloud to obtain the average/middle of the button, which represents the horizontal coordinate. After this, an adjustment is made to deal with the xtion error. The xtion is the camera used to visualize the buttons and elevator. The error is purely hard coded and subject to change if the xtion is moved in any manner. After all of this, another node would generate an approach pose. After checking the inverse kinematics, the robot arm would would find a position directly in front of the button. From here the arm is published direct linear velocities in order to perform the button press.

Currently the arm works with a reasonably high accuracy, but only when placed in specific orientations without autonomous navigation. The robot will not function correctly if it is too close or too far away from the wall. The arm can only extend so far, but it also needs room to extend in order to press the button with the tips of its “fingers.” We aim to not only improve the functionality of this action, but to also allow the robot the capability to perform such an action after navigating autonomously to the region. This will be done by adding an object on the robot’s map for each elevator panel. This will help with placement of the robot in its correct position and orientation (facing the button). The robot will detect the closest point on the plane and then turns till it's directly facing that point. Afterwards it will move forwards (or backwards) very slowly to get to the desired position in which the robot can press the button.



To improve the current method of pressing the button, we first aim to add a verification step. Our current plan for execution is to save an image of the point cloud before the action is done and right after. Using the heuristic to find the general location of the button, we will search around this point for approximately 100 pixels and detect if there is a significant difference in color after the supposed button press. Since the button turns on a small orange light when it is pressed, the change in color would signify a successful attempt. If this change is not found, the robot would need to make another attempt. The next improvement will be will be made in a hardware sense. Physically mounting the xtion with screws instead of zip wires will reduce calibration error between the camera and arm. This will not only help to reduce error in both our own project and in all other projects involving the arm. Lastly we will attempt to improve the process by implementing a learning process each time there is an unsuccessful attempt at pressing the button. The plan of action is to place a marker on the arm to track the position of the hand from the camera’s view. We will the use the marker’s position to compare its distance from the button without the respective xtion error. The robot would then readjust its own error and create a new goal (position, orientation) to send to the arm. We would potentially use gradient descent with the distance from the button as a weight in order for the robot to learn and automatically adjust itself to reach the button over time.

Entering the Elevator

The robot heavily relies on human interaction to determine when the elevator has arrived and when it is safe to get in. Each elevator in the GDC has a vertical line at the center of its door extending from top to bottom. We plan to use computer vision to detect whether this line is present. After the button is pressed, the lack of a line in the robot's vision would mean the elevator is open. We are not sure how to tackle the issue of recognizing whether the elevator is moving in the correct direction (up vs. down), but plan to solve this issue later on. If the elevator is too full for the robot to get in, it should start the process all over again, just as a human would.

Choosing a Floor

Currently, the robot does not press any of the buttons in the elevator by itself. It must ask a human to choose a floor for it. This means that if there is no one else in the elevator, the robot cannot travel to another floor. Given time, we would like to implement some of the same approaches used to press the button outside of the elevator to choose a floor once inside. The process would involve a verification and self calibration similar to that used in pressing the first button. This task will be more complicated because an unsuccessful attempt at pressing one button could mean pressing a different button instead. This could result in false positives in the verification process. It will also be much more difficult to position the robot correctly in front of the panel once inside the elevator. If there is no room in the elevator for it to reach this panel, the robot might play an audio message asking the human in the way to press the floor button instead. Otherwise, it would attempt to position itself in front of the panel and press the button itself. Some possible approaches to recognizing which button corresponds to which number is to use text detection to find the numbers. Then we could put the location of the text into a grid and attempt to offset this grid in a way that the circles match up with the numbers.

Getting Out of the Elevator

The final step in traveling from floor to floor is exiting the elevator. This problem is complicated because as humans, we simply look up at the screen or lights in an elevator to determine which floor we are currently on. The robot, however, has a mounted camera that does not have the ability to tilt up and look overhead. Because of this, we must find ways other than vision to determine the current location of the elevator. Some possible approaches include using an accelerometer to calculate the distance traveled, or running experiments to determine how much time the elevator takes to get between floors. The latter approach would mean keeping track of when the elevator has stopped and when it is moving. Neither approach takes into account that the elevator might be moving in the wrong direction. Hopefully the direction will have been determined by this stage. The robot will also have to deal with getting around people that are blocking the door. This will also be done by playing an audio message asking people to move out of the way.

Conclusion

Autonomous use of the elevator would be a big step for the Segway Bots. It would expand the realm of possibility for both current and future projects in the research lab. We plan to compare the current method of navigating between floors to our own methods to determine success. We are also fully aware that this project will not be completed in the given two months. Because of the time constraint, we plan to focus mainly on pressing the initial elevator button, getting in the elevator, and choosing a floor. We plan to complete the first two steps and hopefully make a great deal of progress on the others.