Fei Xia

Ikanna Uzoije

Linze Zhao

Louae Tyoan

Zhongtao Yang

Week 3 Report Team TurtleSeek

Before this week, we had a TurtleBot capable of moving around the room with keyboard control and would scan the world into a 2D map. We also had the openCV examples for facial recognition and body detection/tracking running. The goal for this week was to modify the facial recognition and body detection algorithms to work with still images as opposed to a video input as it would be redundant to constantly scan as the TurtleBot moves and it would also save on processing power. We also wanted the facial recognition to match to a single target photo as opposed to a database comparison. The body detection would also need to output a location which we could then place into the 2D map. However, due to the complexity of the openCV examples, we ran into a few complications. Details of those complications and what we did to work around them are elaborated later in this report.

Facial Recognition

Upon further investigation of the openCV example "facerec_video.cpp" which uses the fisherfaces algorithm, I noticed that the facial recognition is highly sensitive to illumination and exposure. I saved a picture of my face with the exact size and resolution as the other images in the database to see if it recognized me, which it didn't. Here is the output of this test. The predicted value is subject 18 when my face is in subject 35.

Because of this unreliability, we decided to change the approach to revolve around "Sparse Respresentation". This approach should be a much better starting point for the facial recognition which we can also make more robust later on. Here is the link to that paper:

http://perception.csl.illinois.edu/recognition/Files/PAMI-Face.pdf

We will be using the open source code from this github collection:

https://github.com/409544320/face_recog.src

Here are examples of the code in action.

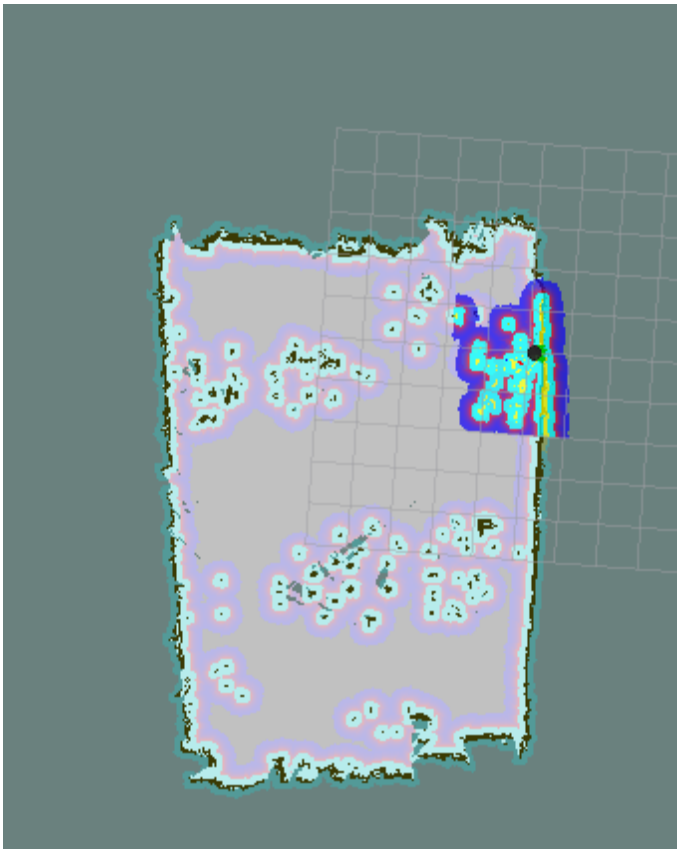
https://www.youtube.com/watch?v=hm-cBYsH9ZE

https://www.youtube.com/watch?v=hplmAxK95yw

Currently we're working on the code to make it compatible with the TurtleBot. We already have a basic package, but still need to do some work on it so that it will run. This will be the basis for next week's task.

Mapping

For this week I further tuned the TurtleBot navigation part. First I built a map for the room by driving the TurtleBot around. The map looks like below:

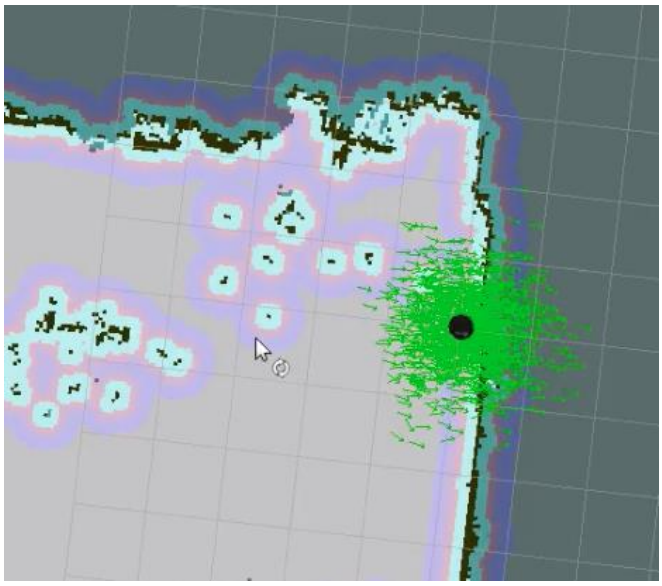


The mapping used gmapping program and icp algorithm. It works nicely on TurtleBot

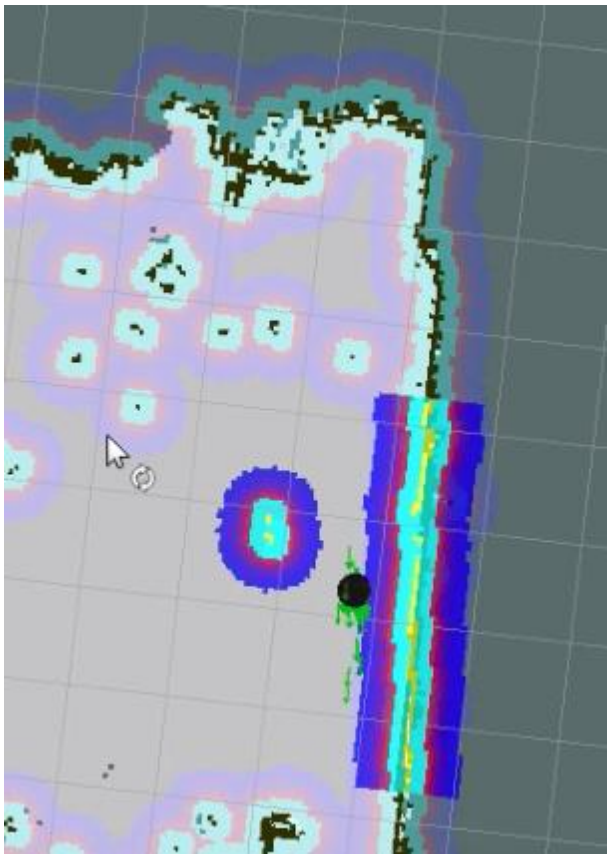
Localization

Also, I tested localization of TurtleBot by using the package amcl. The amcl uses particle filter to do the estimation of the pose.

The program starts by setting an initial guess of the pose.

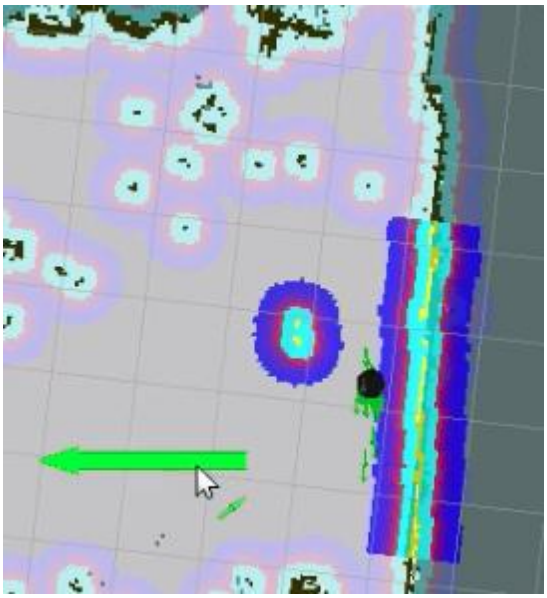


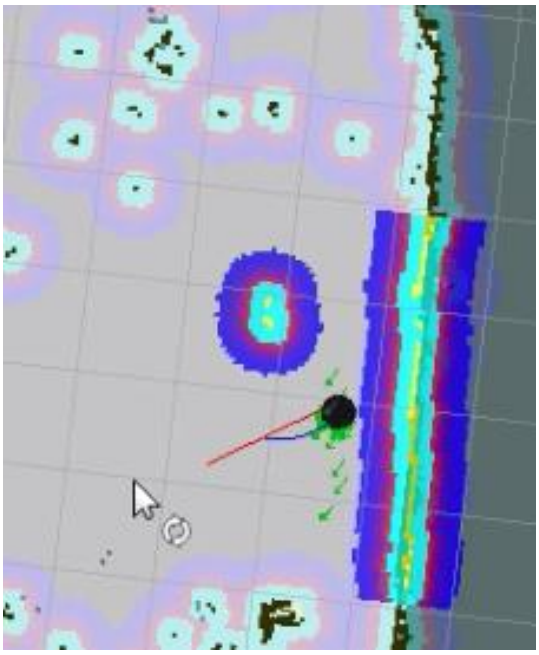
After driving the robot a little bit. The algorithm will combine the odometry data and depth sensor data to make the localization more accurate.



Path Planning

I tested the path planning of TurtleBot. After setting a destination, TurtleBot will automatically design a path and drive to that point.

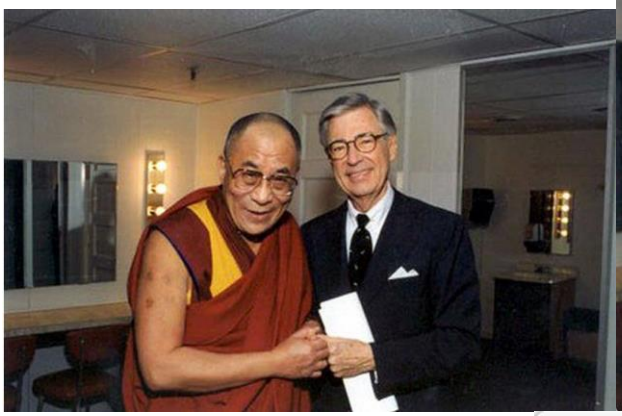


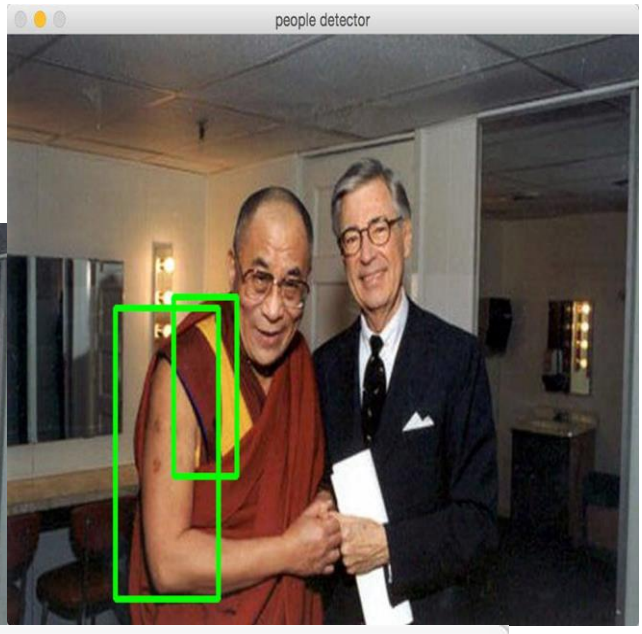


There are two video clips to show:
https://www.dropbox.com/s/7hbnflpidyr1yic/turtlebot_navigation2.mov?dl=0
https://www.dropbox.com/s/lj0rzxqir3okfn1/turtlebot_navigation.mp4?dl=0

Body Detection:

This week, Zhongtao and I (Ikenna) decided to use a different algorithm for the body detection from still images. Due to the midterms we had this week, we couldn't get the algorithm to work all the time. It works when the background is "pure" i.e. when there isn't a lot of a variation going on in the background. It also only detects one person when there is a group of people. Again, we will attach a copy of the .cpp file of our code and the result of the algorithm on the image is shown below. We plan to fix the code as our next week's work and also get the relative position of the body printed on the screen.




people detector


people detector


people detector


people detector
Tag Photo
Options
Share
Send