Final Project

Angel Macias

Initial Process and Plan

My initial project plan was to use a depth sensing camera (PMDtec) to scan the ground and gather arrays of distance data from the camera. Next I would use the camera’s MATLAB library functions to access the data and create software to manipulate the data. The manipulation of the data would parse the distances into many blocks of space, then patch them together into a map of the ground in variable resolution, and finally calculate the blocks normal vectors and colorize the blocks based on their steepness. The robot I intended to implement this for would then analyze the map to find a good place to step on based on proximity to the foot and steepness of the patch.

Learning MATLAB and Camera Usage

At the beginning of this project I had terribly slow progress due to having just learned how to code in MATLAB, which I learned through the first Humanoids assignment and some practice. I also needed to learn how to interact with the camera which took a couple of weeks due to not having too many resources to look at, such as forums, which were rather sparse. When I finally got the correct library for windows and MATLAB for the camera to operate correctly on my machine I began experimenting with the different functions, such as adjusting modulation frequency and integration time to visualize the effect they had on the data I received. The modulation frequency didn’t seem to affect the quality too much, while higher integration times for the images made the depth points a little less noisy and sharper. Then I used the depth measuring functions, one that gets 3D coordinates tuples and one that simply gets the distance data from points on a grid. After a while, gathering data from the camera and looking at it on MATLAB became rather easy to do, so I started coding on the actual topography and patching algorithm.

Patching Algorithm Development Process

Initially I checked for how integration time and modulation frequency changes affect the quality of the data. Modulation frequency had no apparent affect, however the integration time made the data a bit less noisy and sharper. I need the program to run relatively quickly since the robot moves rather fast, as in the video (LINK) that I showed in the presentation, so I set the integration time to the default speed. As for choosing between 3D coordinate tuples and single distance arrays, I went with gathering 120x165 arrays of distance points since it’s less information to work with and made understanding the problem easier for me.

I first worked on cleaning up the data but adding a few filters:

1. Depth filter that removed points that were too far away, about 1m away, since they wouldn’t be necessary for the Snake Monster robot considering how short it is and how small the range for the camera is.
2. Area filter that removed blocks of distances that had areas that were too large.
3. Block filter that removed points that weren’t part of block.

I concatenated the blocks into squares of points,

1---2

| |

3---4

In that fashion. If the area of the square made from the points was too large or the points were too far from each other, then the data was too skewed and then filtered. As for the block filter, points that didn’t conform into squares like shown above (as in there was a group of only 2 or 3 points left over somewhere) then those points were removed. The blocks themselves can be changed in size between 2x2 distance point blocks and 10x10 blocks.

Once filtered and concatenated, I get a vertical and horizontal vector from each plane and cross them to get the normal vector. Then, if I want to show the normal vectors I collect the centroid and call the quiver3 function to display normal vectors, though it is purely for visual purposes and usually turned off. If not displaying the vectors, I dot the normal vector of the block with a reference vector that points straight up (<0,0,1>) to get the angle of the bock relative to the ground. The blocks are then assigned the angle and color based on steepness.

This loops for each frame and at the end of each I’m able to give a delta-x and delta-y that tells the plot how much the Snake Monster could’ve moved between the frames, then plots based on that changed origin.

Overall, this program is a topographical mapper with a depth sensing camera

Future Work

I’m going to continue working on this project during the summer for some while. I wasn’t able to get the last part to finish, however I did get the bulk done. The Snake Monster robot is rather popular and out of my reach to work with. The grad student that works with the robot made a simulator for me to start working with in Simulink for MATLAB. However, I didn’t have enough time to learn how to use that as well considering how many other libraries I worked with learning on for this project.

The first thing I’d want to incorporate is a dynamic reference vector. Instead of a <0,0,1> vector that is static, I would like the vector to changed based on the angle of the camera to the ground, which I would do by either attaching an IMU to the back of it, or gathering [pitch,roll,yaw] data from the Snake Monster(SM) robot.

Then I would finish the project by creating a cost function for the robot to place its feet in the correct orientation based on the proximity and steepness of a good patch of ground.

Conclusion

I may not have finished the entire project that I set out to do, but I did learn a lot in the process in terms of computer vision data manipulation, depth data manipulation, and how to use new technologies (MATLAB, Time of Flight depth cameras, and Simulink in some regard). I created a decent piece of software, at this point, that maps out the topography of a surface rather well and patches the pieces together in some manner.