Journal Report 9

11/11/19-11/14/19
Aneesh Boreda
Computer Systems Research Lab
Period 4, White

Daily Log

Monday November 11

I did research on why OpenCV wasn't working properly, even though I installed it through apt-get earlier. I found a few people with similar issues, but the advice was just to reinstall and build from source, which takes a lot of time. I tried looking for any other options, but couldn't find anything.

Tuesday November 12

I tried to just reinstall OpenCV from scratch, since I don't think I installed it properly the first time. It took most of class to build OpenCV from the source, and I had to figure out how to properly deal with linking the library with g++. I added it to path and finished all the basic setup again. While it was building OpenCV, I found an article that explained the stored transformation matrices in more detail.

Thursday November 14

Even when the library was properly linked with g++, the OpenCV defined classes were still unrecognized, even though the import statements were working properly. I tried installing different IDE's and seeing if they would better handle dependencies. I tried installing Netbeans and the C++ extension for VSCode, but neither of those resolved the issue, even after I added OpenCV to the list of included directories and they recognized the import statements.

Timeline

Date	Goal	Met
Week of 10/31	Start setting up configuration of stereo vision prototype in C++, make sure all necessary libraries are installed	Combined with next week
Week of 11/7	Continued from previous week	I made good progress on the calibration step, as I found a sample on github that I can adapt.
Week of 11/14	Figure out necessary information to store for camera calibration, work on generating calibration from chessboard training images	Yes, I found what exactly I need to save, and made good progress on fixing opencv
Week of 11/21	Fix OpenCV library linking and get chessboard detection working	
Week of 11/28	Calculate transformation matrices from chessboard corner detection for rectification	

Reflection

I had some issues with OpenCV this week, but I think I finally am on the right track with the installation, and once I fix all the issues I'm having right now, it should be good for the rest of the year. I think I should have all the information to finish the point cloud generation now, as I found specifications for almost all parts of the algorithm. The two main parts, calibration and actual point cloud generation aren't overly complicated by themselves, but most of the difficulty comes from ensuring that calibration data can properly be used to rectify images before calculating the differences, since without proper rectification, the generated point cloud will end up being very noisy and inaccurate.