Journal Report 7
10/15/19-10/28/19
Austin Huang
Computer Systems Research Lab
Period 1, White

Daily Log

Tuesday October 15

Decided to pursue gnumpy. Researched what it is all about (https://www.cs.toronto.edu/ tijmen/gnumpy.html), seems to be like what was discussed. Does not look easy to set up.

Thursday October 17

Easy to install actual gnumpy library. Error about cudamat. Researched cudamat. Started process to install cudamat. While waiting, started research on Kalman Filtering. Algorithm requires prediction points.

Monday October 21

Continued to research Kalman Filtering. Went back to installing cudamat. For reasons unknown, my computer bluescreened. The process to download cudamat successfully is equally irritating. Figured out needs SDK.

Tuesday October 22

Computer search bar does not work. Now needs to locate files manually. Figured out that I need to run simulation GPU because my computer does not have a GPU. Installed NVIDIA program, which should be able to simulate.

Thursday October 24

Looked on github and stack overflow for solutions to downloading the cudamat library. Run into TypeError and then Microsoft visual c++ error. Able to download Build tools from Microsoft Visual Studios. Looked for an exe file to run. Not quite sure how stuff happened, but the library showed up on PyCharm after I ran some obscure files hidden in my repository. Figured it out at home. More in reflection.

Timeline

Date	Goal	Met
Oct 14	Create a background subtractor to find balls faster	Achieved but it is slow.
Oct 21	Install gnumpy	struggled to install on computer
Oct 28	Install gnumpy	downloaded onto PyCharm
Nov 4	Optimize matrix operations with gnumpy	
Oct 11	Implement Kalman filter	

Reflection

Based on the discussion we had, I figured the best way to proceed first was to quicken the algorithm and then add on. This makes the most sense because debugging can become tedious with a slow program. As a result, I decided to pursue gnumpy first, then implement the Kalman filter. Ultimately, I would like to add both color-based tracking as well as trajectory based tracking to improve my overall accuracy.

What I did not anticipate was the difficulty of installing gnumpy onto my computer. This was even harder than installing OpenCV initially at the beginning of the year. Unlike OpenCV, there were no video tutorials or any well documented guides online (in my opinion), and the whole process is quite confusing.

For example, gnumpy cannot be implemented without installing this other package, "cudamat," first. From what I researched, gnumpy operates off cudamat, as cudamat is the basic package that optimizes matrix operations in a GPU. Why they didn't merge these two libraries, I do not know.

Furthermore, the whole process to download cudamat is just as if not harder than gnumpy. From what was on the official github documentation of the cudamat library, it seemed like I needed to download the SDK for cudamat. This proved to be a whole monster in it of itself. For example, there was a TypeError, and an online post said to use a pip command. This worked, but then another error about Microsoft Visual C++ missing on my computer.

Finally, at home I realized that somehow the python on my computer (2.7) was different than the one on PyCharm (3.0). This gave me the idea to localize everything on the terminal in PyCharm. I redid all my previous steps in the terminal and miraculously the cudamat seemed to load. It seems like that most if not all the installation I did outside on my computer was not necessary.

The other research I did these weeks were about Kalman filtering, most of which I did while I waited for downloads and installations. It seems that an important part of the algorithm is to include a past state and predicted future states. I imagine this can be done easily by storing a past state and looking ahead using either Hough Circle method in OpenCV or even using my color based tracking to predict a future state.

Now that I managed to install gnumpy on PyCharm, my next step is to transfer everything to matrix operation, and test for time optimization. Then, I will proceed with Kalman filtering.