Journal Report 9
11/11/19-11/18/19
Austin Huang
Computer Systems Research Lab
Period 1, White

Daily Log

Monday November 11

Debugged to test for speed. Noticed the times I was spending the most time for both matrix operation version and original version was during background subtraction thresholding process.

Tuesday November 12

Looked over comprehensions to see if error is there. Mr. White proposes slicing. Edited code accordingly. Noticed a new problem: can comprehension output a 2D matrix? Right now I have a comprehension generate a 1D array from a 2D matrix to put back into a 2D matrix. Is there a way to compress this all into one step? This was done beforehand with two nested for loops, but I think it can be possible with a single comprehension, or at least be optimized with python.

Thursday November 14

Spent some time cleaning up my code because it looks really ugly and disorganized right now. Need to upload to github. I forgot to upload at school. When I remembered I had enough time to create an account at least. Found some good web pages on Kalman filter. Couldn't solve efficiency problem with n^2 problem. Logically may be faster to go with original nested for loop.

Timeline

Date	Goal	Met
Nov 11	Optimize code with just matrix operations	Matrix operations achieved, optimized–not really.
Nov 18	Diagnose why it is not going fast and work on Kalman filter	diagnosed and started working; found a website and a section on a textbook.
Nov 25	Finish implementing Kalman filter	
Dec 2	Optimize Kalman filter	
Dec 2	Optimize everything overall and organize code and diagnose any remaining problems	
WINTER GOAL	I will have an application that will reliably track any pass in reasonable time. (¡10 seconds per frame).	

Reflection

This week I decided to go back to examine my code to determine what was wrong with my algorithm, and I saw that it was mainly the thresholding that took up most of the time in both the comprehension-updated program and the original program.

In the original program, I programmed the algorithm to go through each element in the frame and compare the distance away from the average and made it binary based off the distance was above or below a certain threshold. In order to this, I traversed a nested for loop in order to access all elements. Within the same nested for loop, once I determined whether a pixel should be 0 or 255, I matched the element location in another matrix and assigned the value there. Overall efficiency: n2.

However, in what I had initially thought was a faster comprehension list algorithm was actually slower. What I had done was program in a comprehension that extracted a 1-dimension list of ordered elements from the frame with the distance-threshold algorithm as the filter. Then, I went through each of the elements in the 1D array and put them into a new matrix. I had thought that because I had separated each operation, instead of having a nested n2 effect, it would have an n+n=2n effect. However, now I realize this was not the case because just by traversing the matrix to put back each of the elements I had in the 1D array, it was a n2 operation, even if I was doing it starting from the 1D perspective. So this method could actually have been slower because it had a n2 operation plus the additional comprehension.

I am left confused with this, but decided to move on to research kalman filtering, which is very technical, I quickly figured out. I found a good introductory website online: $https : //www.bzarg.com/p/how - a - kalman - filter - works - in - pictures/$ which actually goes pretty in depth after the initial paragraphs

I also browsed the computer vision books in the other computer systems classroom and found a textbook with a section on kalman filtering.

Next week, the goal will be to program in this long awaited filter finally.