Journal Report 10
11/18/19-11/25/19
Austin Huang
Computer Systems Research Lab
Period 1, White

Daily Log

Monday November 18

Went over the algorithms from before and after optimization. Realized fastest is to combine aspects of both. Perhaps there is a way to even more optimize it. Actually, would certainly be the case. Dad is getting me a new computer, so hopefully things will speed up regardless with less broken hardware.

Tuesday November 19

Started researching about Kalman filtering again. Realized that Kalman is essentially just a comparison of other data. Tried to research other methods. Ran across sparse vs. dense optical flow. Also looked into Lucas Kanade. But this requires observation set to be really small. Probably best to brainstorm how to determine a smaller area of where ball could be.

Thursday November 21

Was inspired by Kalman. Started setting pixel location of the center of the bounding box as position, and change between the centers of the bounding box as the dxdy/dt. I believe that if I can combine using the previous dxdy to predict the next location of the ball with my already installed background subtractor with hueristic and the necessary weighting, this will be the way to go.

Timeline

Date	Goal	Met
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	realized probably not the way to go (yet), but what I am doing now is inspired by it.
Dec 2	Finish implementing dxdy and combine it with background subtractor	
Dec 9	Search and implement other optical flow methods	
Dec 16	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

From my understanding optical flow is the term used to analyze motion of objects in video with computer analysis. With optical flow, there exists dense and sparse optical flow, and the difference between these two are the amount of objects and thus pixels they analyze. For a sparse optical flow analysis, there is only a handful or fewer items that need to be tracked, and so there only needs to be a careful analysis of the area through which the object may exist. However with dense optical flow, each pixel's movement is calculated from frame to frame, and as one would imagine this would take much more computational power and frankly, probably not needed for my project.

Taking into account sparse optical flow, I needed to find a way to limit the area I was searching and applying my algorithm. This may be a way to not only increase accuracy, but also make my background subtractor more efficient. I thought that the bounding box that the openCV tracker comes up with is usually pretty accurate and when it's not it starts to either gradually snowball out of control very quickly or just report a enormous error on the first incorrect tracked frame. I connected this problem with my previous research with Kalman filtering, and I realized that if I were able to give the tracker a certain ball park to search, it would make my time much better spent.

This can easily be done by calculating the difference between positions of the ball between each frame and using that to predict the next frame's location. I figure that because it's a pass, although it may be fast, the pass will not unexpectedly take a turn or suddenly accelerate. Therefore, while using the delta-position from the previous frame to predict the location in the next frame may not be entirely accurate for a parabolic pass with real world error, it will land myself in a reasonable area. This is the whole idea of the Kalman filter essentially; trying to present multiple reasonable predictions to combine them into something more reasonable.

I am still in the process of figuring out how to combine this predicted delta-position position with the color background subtraction algorithm. Some ideas I had were to use a weighted heuristic where I would count the delta-position prediction greater than the color subtraction. But also another idea would just be to do delta-position prediction, limit the area, then apply color background subtraction to find the ball. Although then the part I would be stuck on is how to limit the area accurately.