Journal Report 11
12/2/19-12/9/19
Austin Huang
Computer Systems Research Lab
Period 1, White

Daily Log

Monday December 2

Brainstormed how to synthesize the dxdy and background subtraction. Because background subtraction uses blackbox CSRT tracker provided by OpenCV, integrating the two is harder than it seems. May be hard to combine the two to find a more accurate location, but instead better to use dxdy to check CSRT tracker-background subtraction. Perhaps could be used for other things as well?

Tuesday December 3

Really hated how things take so long. So focused on making algorithm more efficient. Realized from previous week that process that took longest was thresholding each pixel to determine if it belonged to background. Made a plan to use dxdy to limit the amount of pixels that are thresholded each frame.

Thursday December 5

Lots of problems with code. Spent class debugging. Commented out previous synthesis preliminary algorithm to just default to background subtraction to get rid of potential confounding variables. Most of errors come from limiting box for thresholding. Needed some way to guarantee that the space I threshold over contains the ball. For now I can gradually increase limiting box until it is big enough. After debugging, it was not the sizing, but it was my programmed dimensions for the box. Forgot that in openCV, rectangles are made from left corner instead of in center. This made me adjust rectangle. Also dependent on what sign dxdy was, I had to flip which parameter went first to draw an openCV agreeable rectangle.

Timeline

Date	Goal	Met
Dec 2	Finish implementing dxdy and combine it with background subtractor	one class this week, worked on repository sorting instead.
Dec 9	Finish implementing dxdy and combine it with background subtractor	was able to make code run faster by limiting background subtraction thresholded region with dxdy.
Dec 16	Make code more accurate by either adding another optical flow method or combining dxdy and background subtraction	
Dec 23	Rest	
Dec 30	Rest	
WINTER GOAL	I will have an application that will reliably track any pass in reasonable time. (¡10 seconds per frame).	

Reflection

I initially had hoped to make my algorithm more accurate, so for instance it could even track the ball when it is obscured. However, I managed to improve the other aspect this week, the speed, which, in retrospect considering how it took an unholy amount of time (20 seconds per frame) to debug and test code between every edit, this may actually help me save more time in the future. Now frames take less than a second to process.

I did this by doing what I had set out to do: I combined dxdy with Background Subtraction (BS). I realized early in the week that trying to combine dxdy and BS was harder than I had anticipated because finding the ball with my BS algorithm was done through a CSRTTracker in openCV, which is a blackbox. While it works quite well, I cannot test if it is the fault of the tracker if something unpredicted happens with my code. In this way, for now I determined it was better to use dxdy as a test to see whether BS was tracking in the right area. But then, I realized instead of testing BS afterwards, what if I used dxdy before?

Logically, dxdy, because it uses the previous ball's change in position to predict the next frame's motion, it should not get too thrown off frame to frame. Therefore, I just had to make a reasonable area around my predicted ball's location using dxdy to apply BS. Because I am confident that this area will contain the ball, I safely only need to apply BS to this area, rather than the whole frame. The good thing about this approach is that it bypasses the blackbox tracker problem.

Essentially, if the ball is still within the region that I BS, then nothing from my old algorithm following changes. The tracker compares the white ball against a black background and find it in the region that I applied BS to. However, now the area outside the region where the ball is is completely filtered out, instead of potentially having confounding non-background elements that stayed through the previous BS algorithm. This I believe is helpful for time most obviously, but may also be helpful for the tracker (now it only needs to search for a rare white spec inside the region specified by dxdy, instead of maybe confusing another white area for the ball)

For now, I just made the region dxdy limited equal essentially a 3x3 grid, where the bbox, tracking the ball, is in the center. There are better ways to determine the region for sure, but this seems to be big enough to always contain my ball and easy to calculate.