For the video version, I basically took the same shape-detection pipeline I built for the static image and just wrapped it so it runs frame by frame. I started out trying a couple different preprocessing tricks (including grayscale at one point, but like before that didn't really help since the shapes are color-driven), and HSV again ended up being the way to go. I defined ranges for the main colors, built masks, cleaned them up with morphological operations, and then ran contour detection to outline the shapes.

One thing I noticed when I first ran it was that I had the program printing out the number of shapes detected for every single frame. That was fine at first for debugging, but it slowed everything down a lot once I started processing longer videos. I cut that reporting back so it only does lightweight tracking internally, and that made the whole thing much less computationally expensive.

Performance-wise, it works pretty well for picking up the shapes and overlaying outlines/labels as the video plays. The main challenge is similar to the static case: gradients and background colors can confuse the mask sometimes, so tuning those HSV ranges is key. But overall, once I simplified the reporting and kept the operations efficient, the algorithm was solid for video.