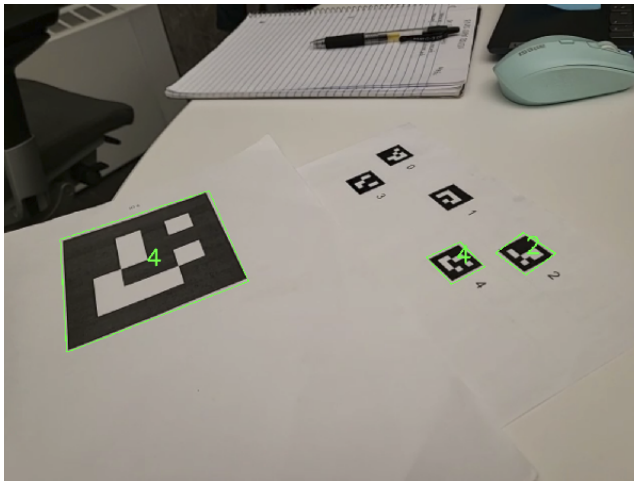
For the final testing phase of my ArUco marker detection system, I used a live video feed rather than a saved sample of images. This better represents the conditions in which a system like this will be used and the challenges attached with it. Compared to the training and validation data sets, this data will have more consistent lighting, motion blur in frames from the live stream, and changes in distance and perspective as I move the camera around. For these reasons, I actually expect better performance as the images will be less difficult to process and detect. Keep in mind that I have been comparing accuracy by measuring the number of images in which the system detects *at least one* marker, rather than all of them present. In reality, the accuracy is much lower as markers at the edge of the frame, or not in the main focus will lose tracking.

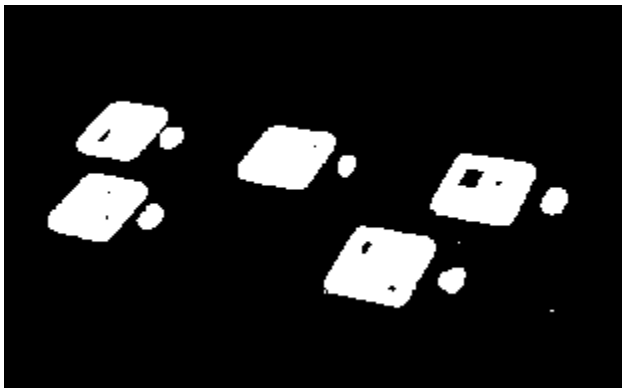
For this phase of testing, I conducted three different tests in slightly different conditions. The first test of five equally sized markers on a white background recorded for approximately 30 seconds achieved a one marker detection accuracy of 93.5%. The next test was the same as the first, but over a less uniform carpet background. It achieved a one marker accuracy of 97.9%. The last test used five small markers and one large marker over a uniform background, and achieved a one marker detection accuracy of 93.7%. However, it often lost tracking of the smaller markers and rarely detected them all simultaneously.



Example of partial detection.

Failed detection, especially of smaller markers, happens for a variety of reasons. A major cause is imprecise polygon detection, which happens early in the pipeline to pass on potential markers to the detection system. If valid markers are approximately poorly and don’t match up with their actual corners, the normalized marker after perspective transformation will not properly line up with the grid used for pixel detection. Adding an offset to

only sample the middle of pixels improves the accuracy by ignoring skewed edges, but this can only go so far. Extremely deformed markers will not be detected.



Rounded edges after processing decrease the accuracy of polygon approximation.

Overall my ArUco marker detection system is fairly robust in good lighting conditions for well space out, decently sized markers. I've included a video below demonstrating the system in action, detecting markers from the camera on my phone. In the first half, the system is functioning in "consumer mode" where only detected markers are shown, and the marker id is displayed in the middle. In the second half, debug mode is enabled where we can see the contours of invalid markers. Blue outlines are rejected for extreme narrow angles, an invalid number of sides, or a small contour area. Red outlines are potential markers that after being normalized did not match any ArUco codes. In addition, it also displays perspective corrected readouts for each successfully detected marker. This view shows the pixel grid, subpixel detection region, average value, and the final decision on the value of each pixel (indicated by font color). See the video below:

https://www.youtube.com/watch?v=d7ALW8XDoB0