Graham Skeats
Computer Vision Project 2 Report
3/1/19

1. Edges and Corners in Video

I implemented the first part of this project in EdgesandCorners.py using my built-in laptop camera as the source for my images and passing either a user defined function that performed a canny edge detection or a harris corner detector to a video capture function. For the actual detectors I used cv2.harriscorner and cv2.canny instead of implementing my own.

For canny edge detection I experimented primarily with a variety of hysteresis thresholds as well as scale.

Figure 7 Original image with no edge detection.

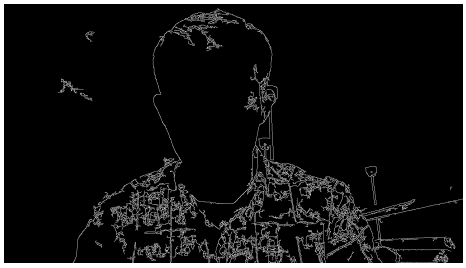

Figure 6 Canny edge detector with threshold1=250 and threshold2=10.

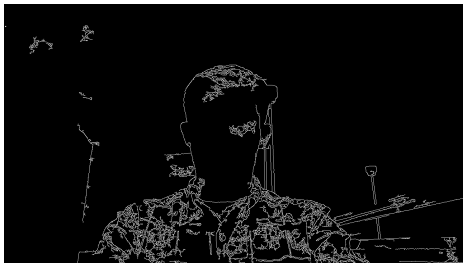

Figure 5 Canny edge detector with threshold1=10 and threshold2=250.

Figure 4 Canny edge detector with threshold1=150 and threshold2=150.

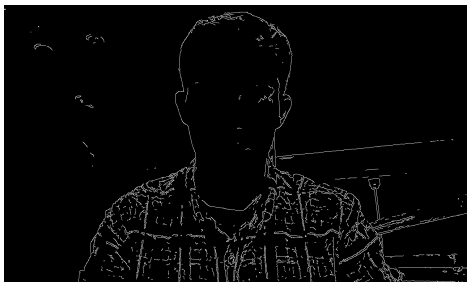

Figure 3 Canny edge detector with threshold1=150 and threshold2=100.

Figure 2 Canny edge detector with threshold1=150 and threshold2=50.

Figure 1 Canny edge detector with threshold1=500 and threshold2=200.

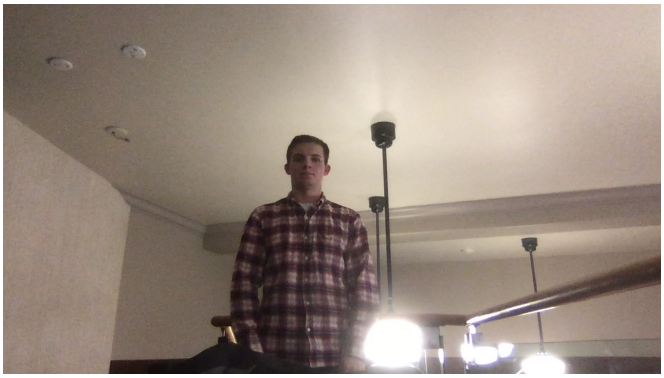

Figure 13 Canny edge detector at scale.

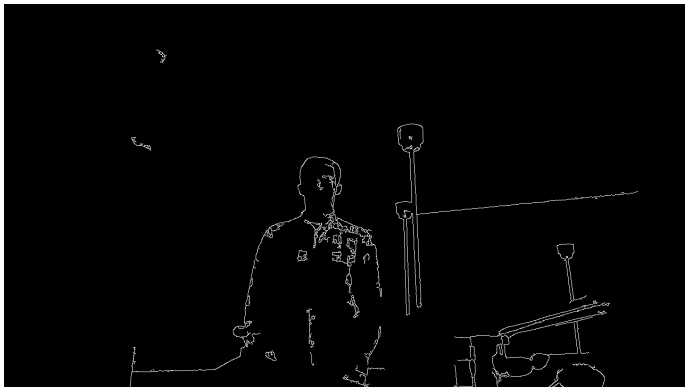

Figure 12 Canny edge detector with threshold1=300 and threshold2=50.

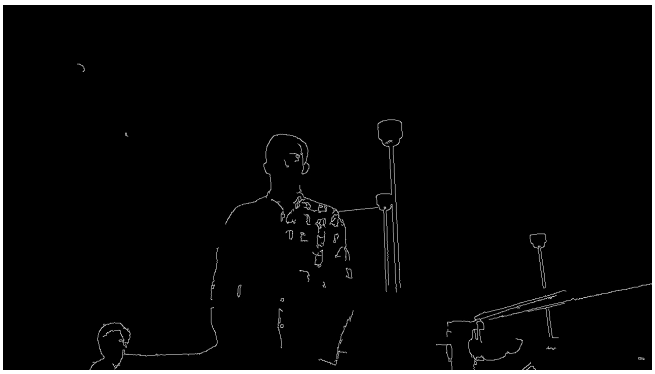

Figure 11 Canny edge detector with threshold1=500 and threshold2=200.

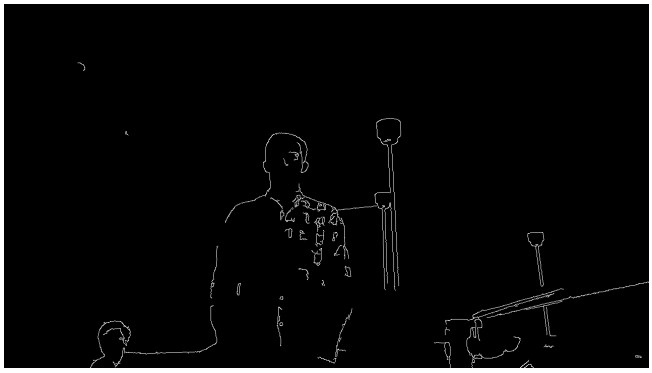

Figure 10 Canny edge detector with threshold1=300 and threshold2=100.

Figure 9 (Upper parameter range that's useful) Canny edge detector with threshold1=600 and threshold2=300.

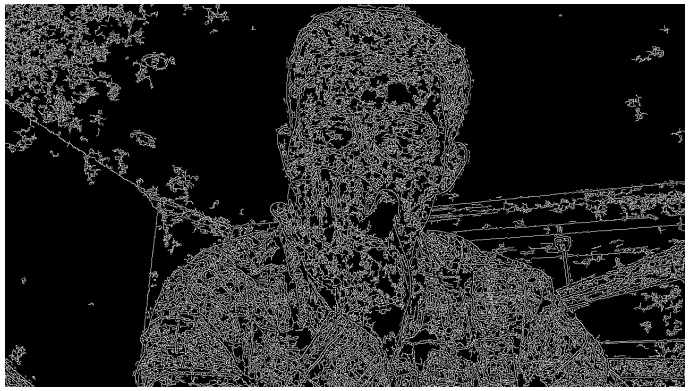

Figure 8 (Lower parameter range) Canny edge detector with threshold1=50 and threshold2=5.

As the above images demonstrate, canny edge detection is effective at a variety of scale and threshold values for the hysteresis thresholding. If you look closely at figures 4,3, and 2 on the previous page you can see that as the second threshold drops the edge map includes more and more discontinuities. This is consistent with our understanding of cany since the lower threshold allows smaller intensity gradients to be identified as edges. This is similar to figure 8 above where the lower threshold is set to only 5 allowing many discontinuities into the resulting edge map and preventing us from getting very useful information. Similarly the high

threshold in figure 9 demonstrates that only extremely well defined edges (those edges with very high intensity changes as represented by their gradient) are included in the image. This suggests that cany is a useful candidate for dynamic adjustment of these threshold values while searching for certain patterns in the image since these intensity changes are affected by blurring caused by light or distance. Examining figures 13 and 12-10 shows that canny can effectively identify edges at different scales however, with the object in question farther away changes to the thresholds do not produce as great a change in the resulting edge map. This suggests that dynamic adjustment of the thresholding values based on the distance to the object is necessary and must be different at all ranges. Similarly, if you look at the bottom right of those same images you'll notice that the light source in the bottom right washes out edges since the intensity gradient won't show a large enough change as a result of the added brightness. This suggests that canny would be a poor choice in very light or dark environments as objects blur together and the total range that a intensity gradient could change decreases.

For edge detection I printed out a paper with several shapes of different sizes and orientations to test the Harris corner detector.

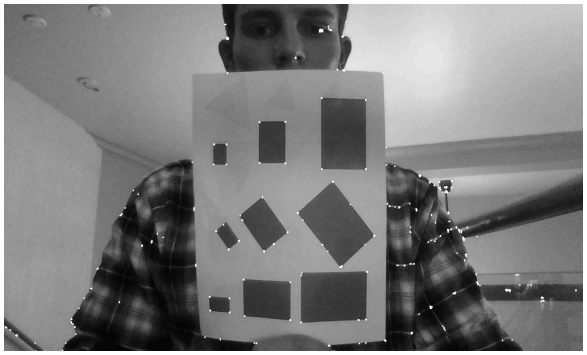

Figure 15 Harris corner detection block size=2, kernel size=3, k=.04.

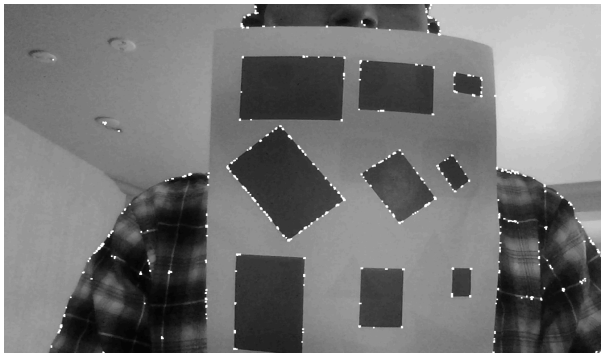

Figure 14 Harris corner detection block size=2, kernel size=3, k=.01

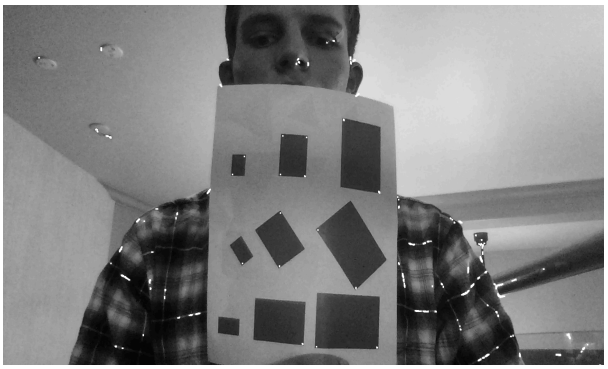

Figure 16 Harris corner detection block size=2, kernel size=5, k=.04.

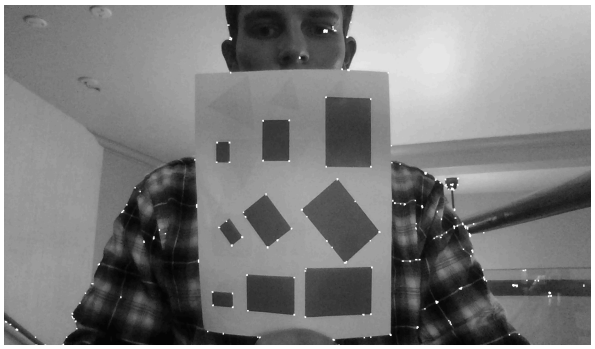

Figure 15 Harris corner detection block size=2, kernel size=3, k=.04.

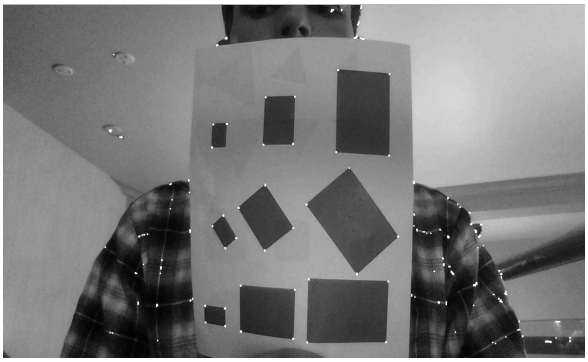

Figure 20 Figure 14 Harris corner detection block size=2, kernel size=5, k=.04.

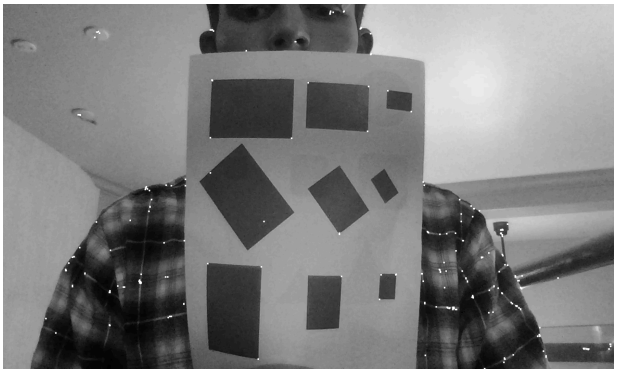

Figure 19 Harris corner detection block size=2, kernel size=4, k=.14

Figure 18 Harris corner detector at an angle block size=2, kernel size=3, k=.04.

Figure 17 Harris corner detection at a distance, block size = 2, kernel size =3, k=.04.

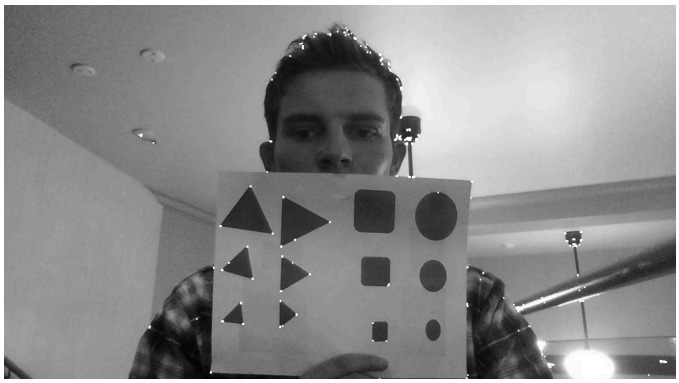

Figure 22 Harris corner detection with non rectangular shapes, block size=2, kernel size=3, k=.04.

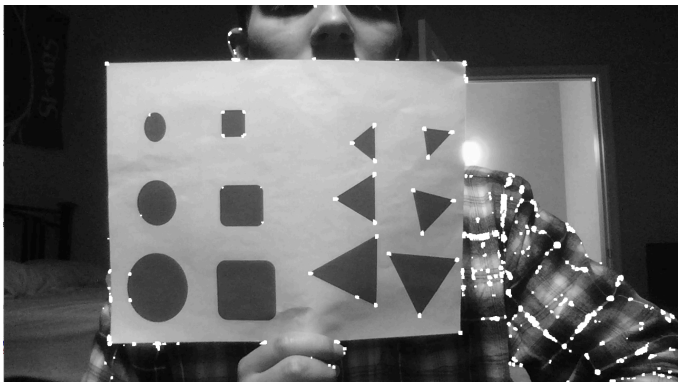

Figure 21 Harris corner detection with block size=6, kernel size=3, and k=.04 under bright light.

As you can see in the set of images above the harris corner detector was able to correctly identify the corners in a variety of situations. Each shape in our grid included three versions of different sizes to test the detectors response to scale. Similarly, the shapes were included in several orientations so that the response to different orientations could be checked. As you can see in figure 22, the corners of the triangles of all sizes and orientations were correctly identified while the circles and rounded corners next to them had very few false

positives. We can also see that in figure 21, the small rounded rectangle has corners while the larger ones don't. This is because by increasing the block size the "threshold" for what is considered a corner increases since the more pixels around each point are considered. This suggests that greater block size could be used to increase the likelihood a particular feature with corner like attributes is classified as a corner.

2. Sift Descriptors and Scaling

Figure 23 Clear photo of sift capturing the keypoints on a wall in starbucks.

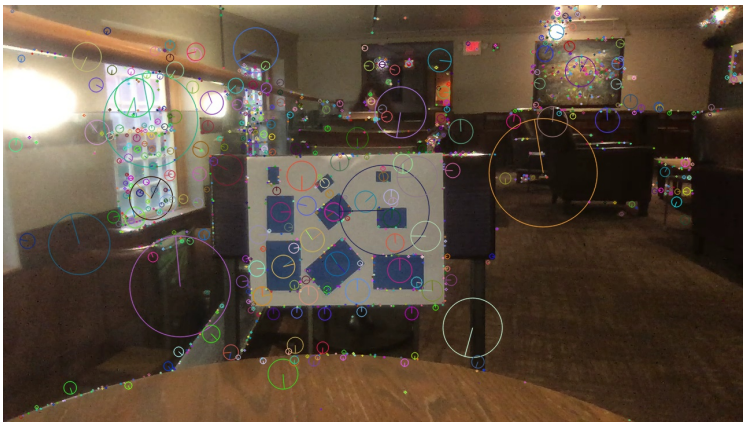

Figure 25 Sift close up.

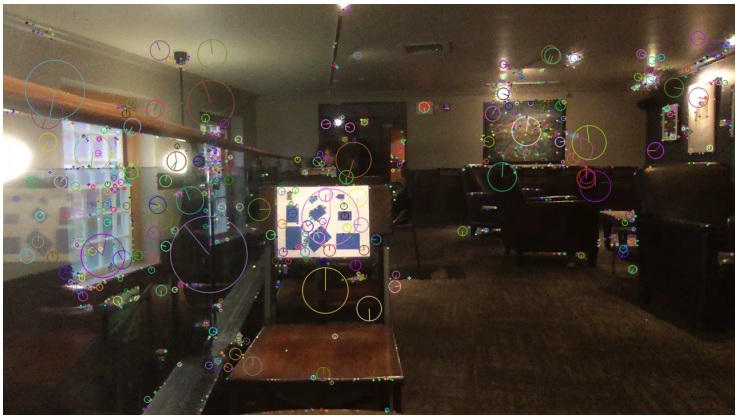

Figure 24 Sift farther away.

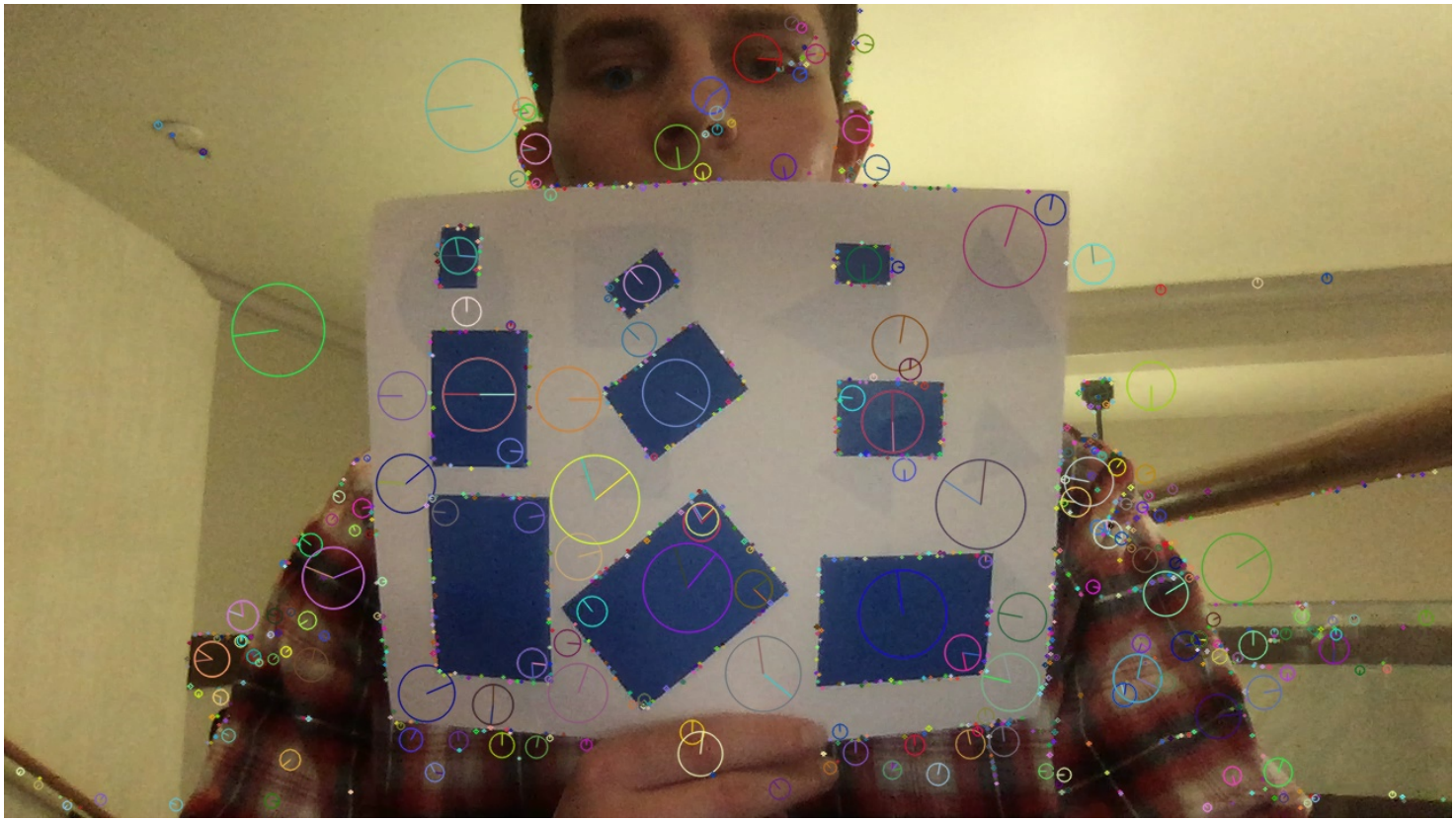

Figure 27 Sift for various shapes at various sizes and orientations.

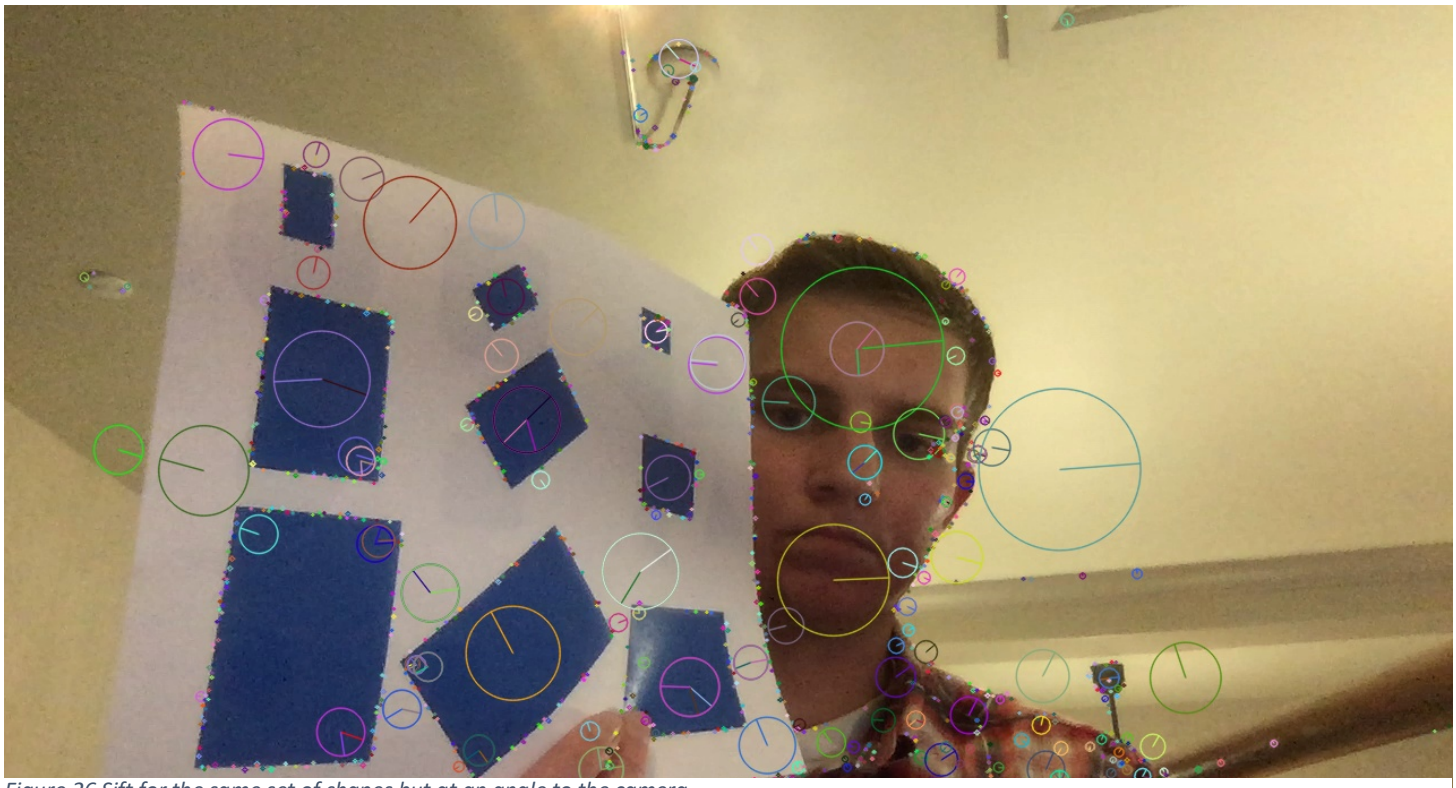

Figure 26 Sift for the same set of shapes but at an angle to the camera.

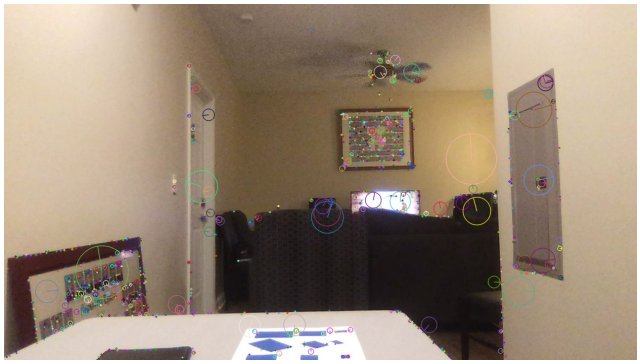

Figure 29 Key points in my living room.

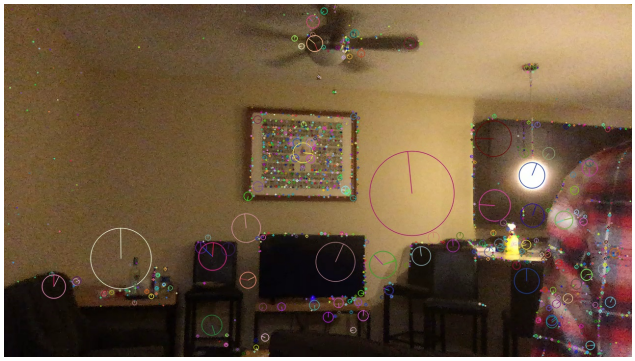

Figure 28 Key points in my living room but about five feet farther down the hall.

SIFT identified keypoints are not as sensitive to changes in scale as we saw in the other feature detection algorithms. This is mostly like because the detection considers the texture surrounding a particular feature so that at a different scale it will still pick out and identify the same features. Looking closely at figures 25 and 24 you can notice that many of the same features on the grid of shapes are picked out even though the grid is at a different scale (and slightly different height). You can see something similar in figures 29 and 28 which identified several key points in common even though the same scene was at a different scale such as the wall to the right of the tv, the left side of the ceiling fan, and the back of the chair to the left of the tv. A worthwhile investigation of this approach would include thresholding these keypoint values so that there are fewer keypoints in an image but since this would mostly make it more viewable to the human eye to a machine it might not make a significant difference.

3. Keypoints and Matching

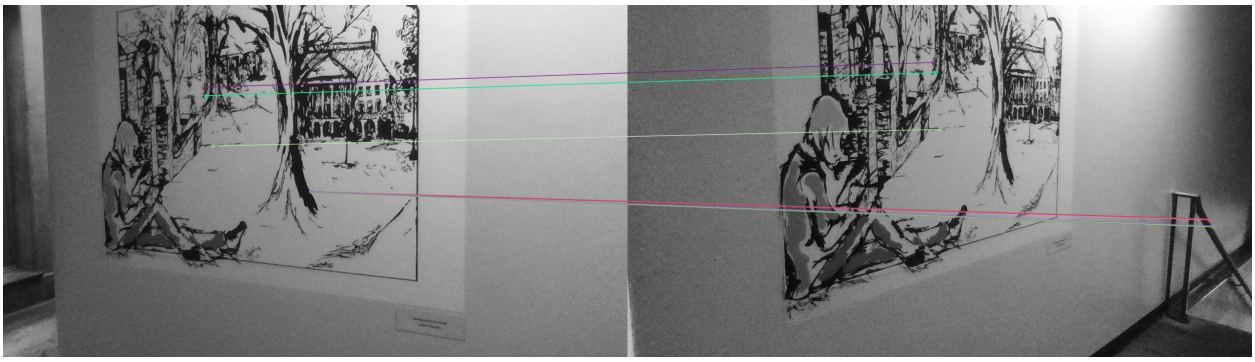

Figure 30 Matching keypoints on two different images of the same painting using harris corners.

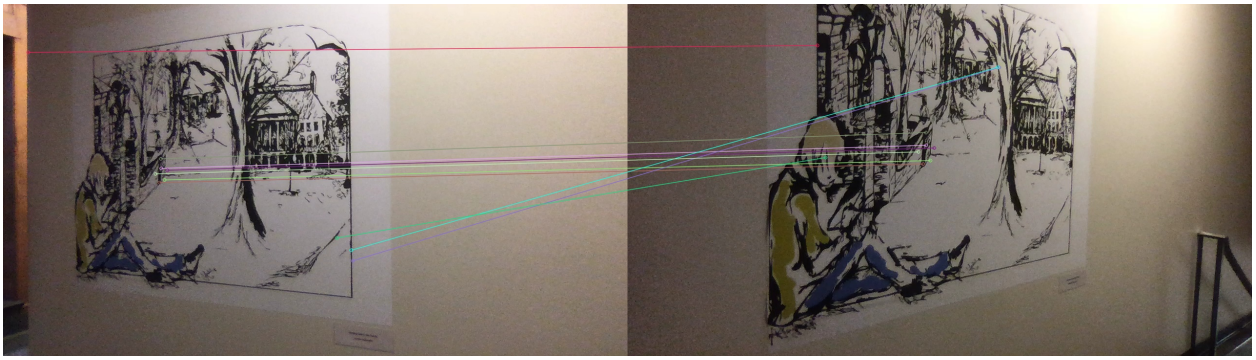

Figure 33 Matching two different images of the same painting using SIFT descriptors/keypoints.

Figure 32 Matching two different angles of the upstairs area of ZSR starbucks using harris corners.

Figure 31 Matching two different angles of the ZSR starbucks using SIFT descriptors/keypoints.

What I found when experimenting with keypoint matching was very interesting. Firstly, there was a noticeable improvement when the two images were as similar as possible, this intuitively makes sense since otherwise the matcher will be forced to work with two very different sets of keypoints. Secondly, keypoint matching using SIFT descriptors was much faster than using harris corners which also makes sense intuitively since SIFT will perform similar calculations but in one step rather than two. The performance of SIFT matching was also much better than harris corners and I think that is because the harris corners are searching only for a particular feature set, corners, whereas keypoints can be many different shapes with a greater variety of attributes. That means that by only searching for keypoints that are corners, better

keypoints are actually ignored that could otherwise have been used to more accurately match the two images. Similarly, since SIFT and its difference of gaussians is much more robust to changes in scale than harris corner and its second moment matrix situations where scale is variable would be a better suited to using SIFT than harris corner. However, after examining some of the math that underpins the two approaches I think that with a bespoke harris matching algorithm there are fewer computational steps instead of the SIFT feature detection. While our implementation happened to be slower I think that harris corner could actually be faster than SIFT making the use of harris is low latency situations more appropriate.