Jkilver3

Assignment 7

FEATURE DETECTION AND MATCHING

JACOB KILVER

Input images

I took the recommendation to get the subject by itself literally. I cropped the image down to just the subject itself, an Isaac Newton fridge magnet/finger puppet.

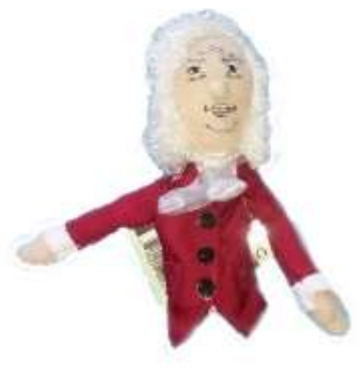

Figure 1: Original image

For the sample image, I used the same setup as the original image, but left the background in

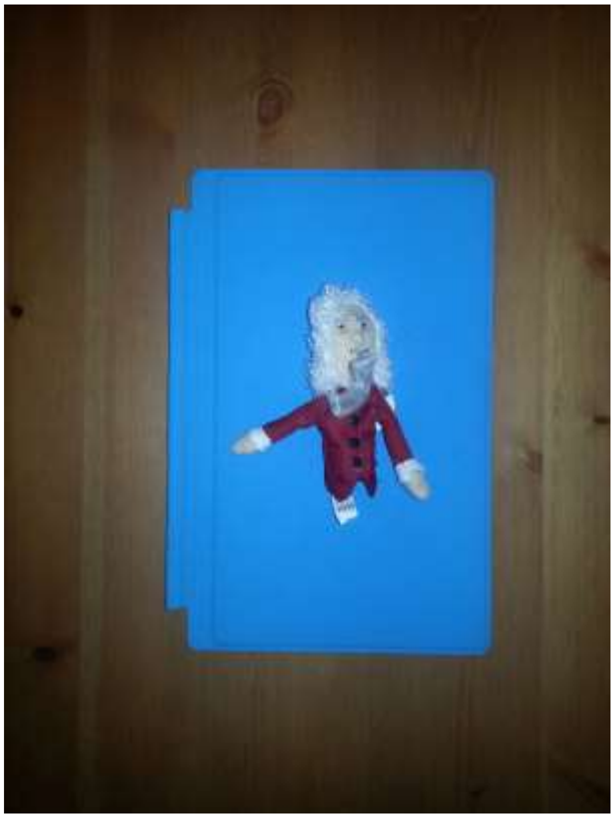

Figure 2: Sample image

For the lighting image, I put Isaac on a darkened bookshelf.

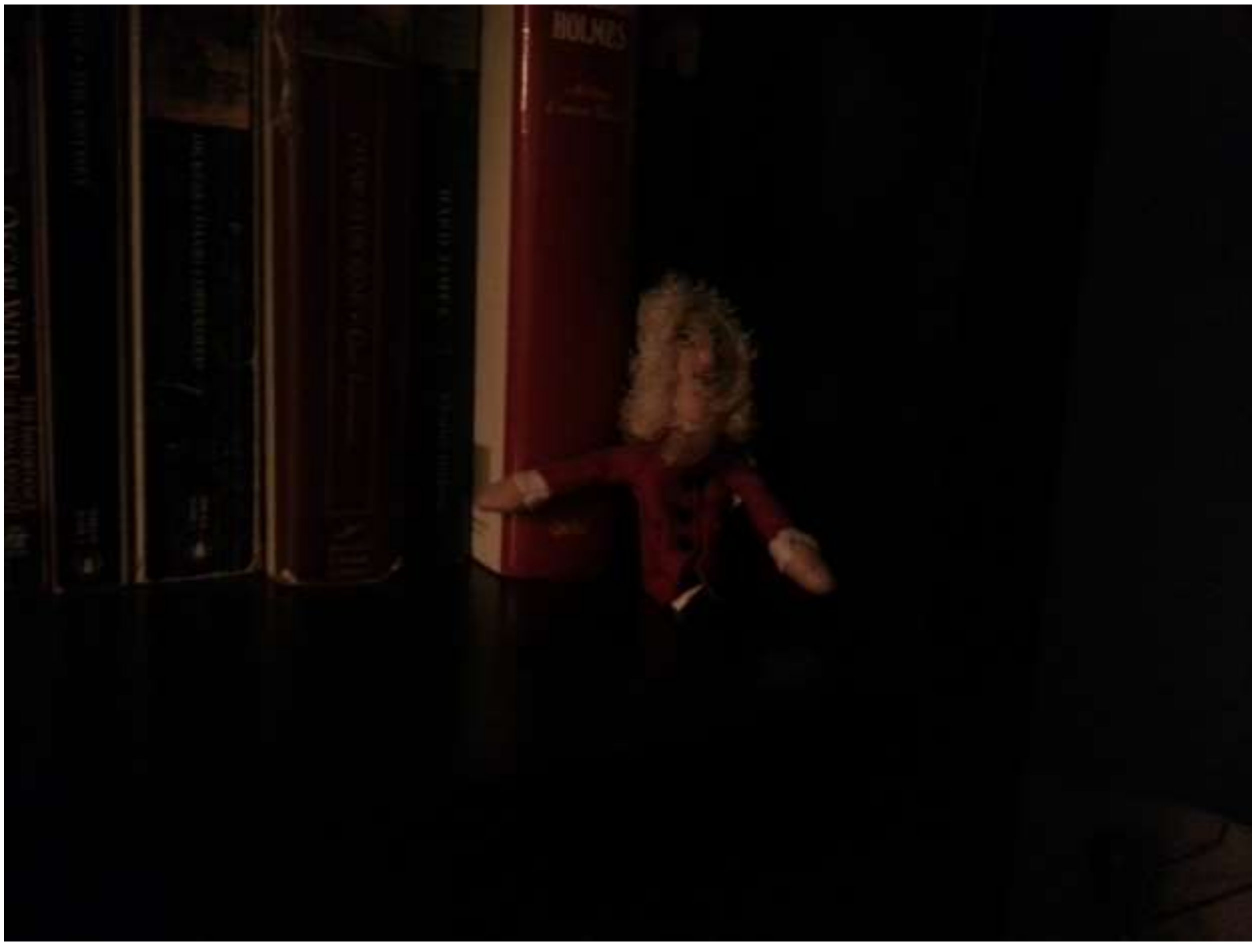

Figure 3: Lighting image

For the rotation image, I placed Isaac sideways on a seat cushion.

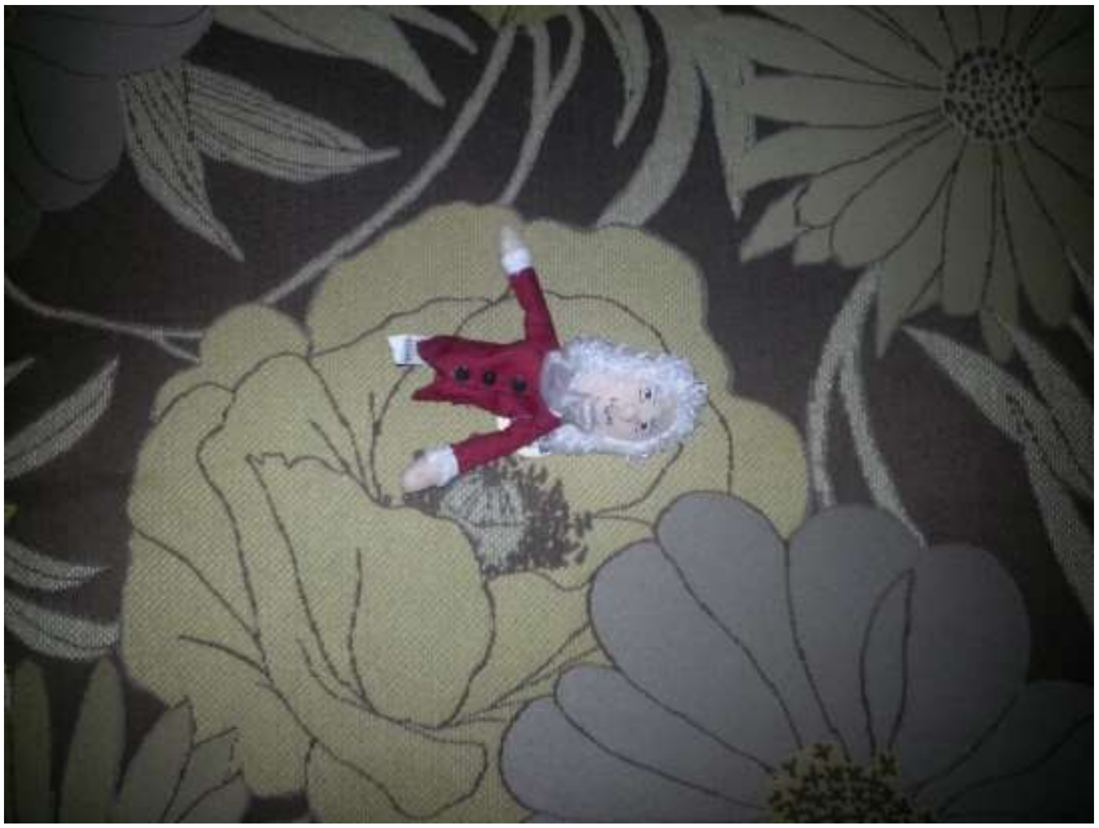

Figure 4: Rotation image

Finally, for the scale image, I placed Isaac on a fridge and took the picture from a few feet back.

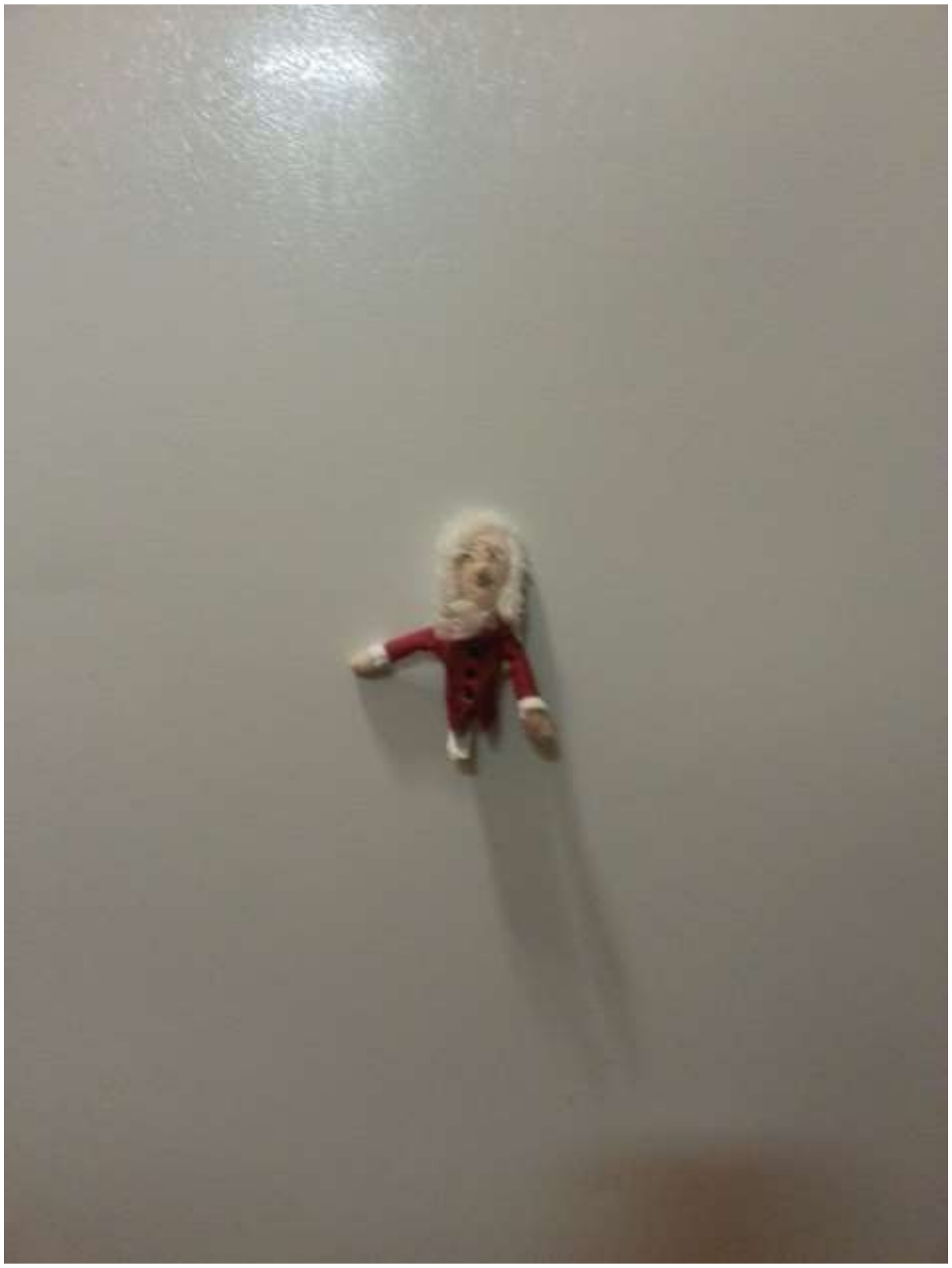
Figure 5: Scale image

Feature Matching Code

To implement feature matching in Python using OpenCV, I first created an instance of ORB. (Note: I tried using SIFT, but had errors with memory. I had to use ORB to avoid this.) I used the detectAndCompute() method to detect keypoints and compute descriptors for the keypoints.

Next, I created an instance of a Brute Force Matcher (BFMatcher) with settings to use Hamming distance and enable cross checking. I passed in the two sets of keypoint descriptors to the BFMatcher's match()

function. This returned a list of DMatch objects. To get the top ten matches, I used python's sorted() function with the arguments taken from the example at the following address: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html. I added the top ten matches to a separate list and returned these matches. The instructor provided code performed the rest of the actions.

Results

Sample

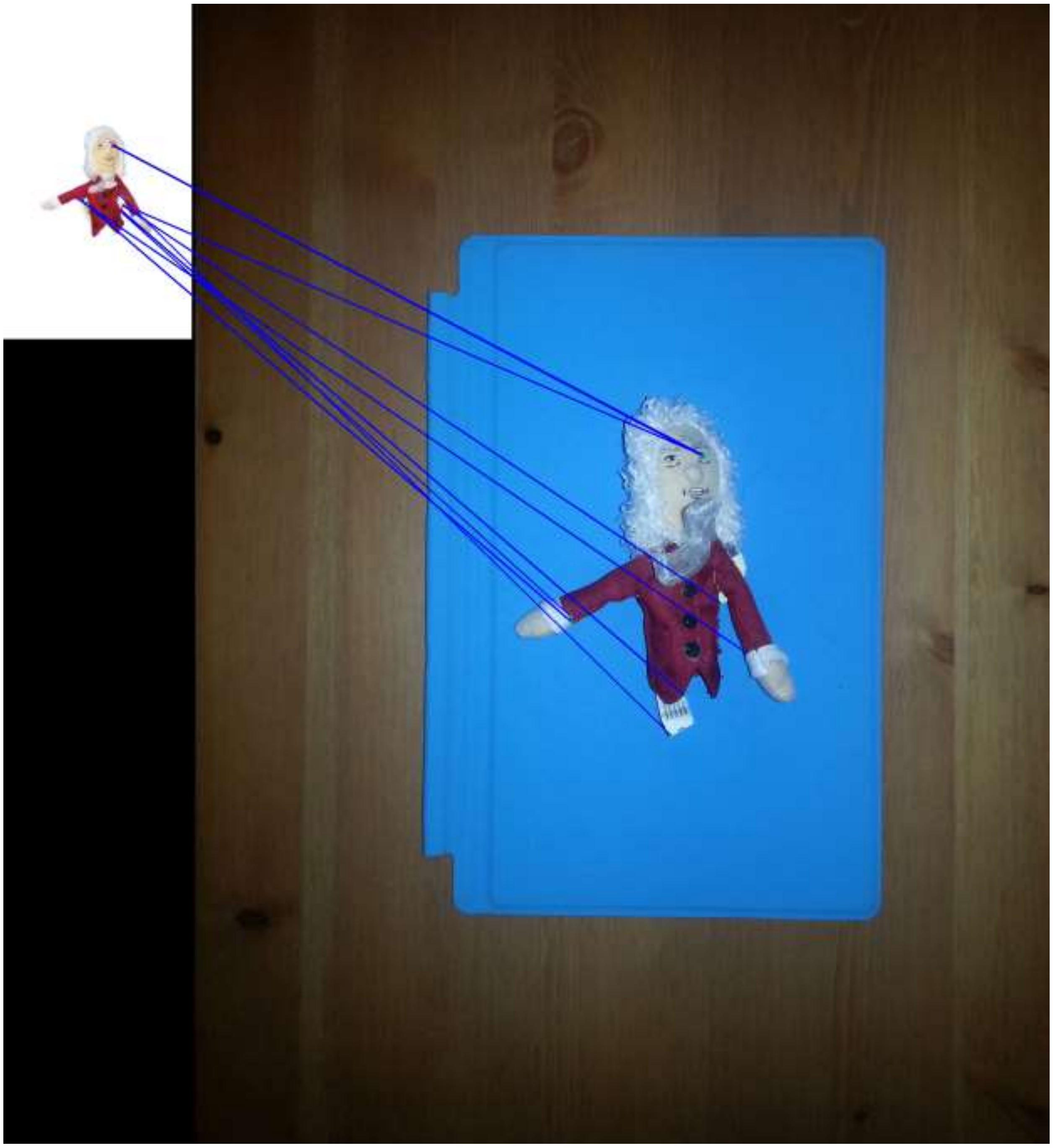

Figure 6: Sample matches

It seems as though only 2 of the feature points were matched correctly here: the right eye and under the right shoulder. Everything was done to make the keypoints as easy to detect as possible – adequate lighting, similar (but not the same) scale and orientation, and a distinct background. I believe part of the problem is with using ORB instead of SIFT. SIFT would throw “insufficient memory” errors, forcing the use of ORB.

Lighting

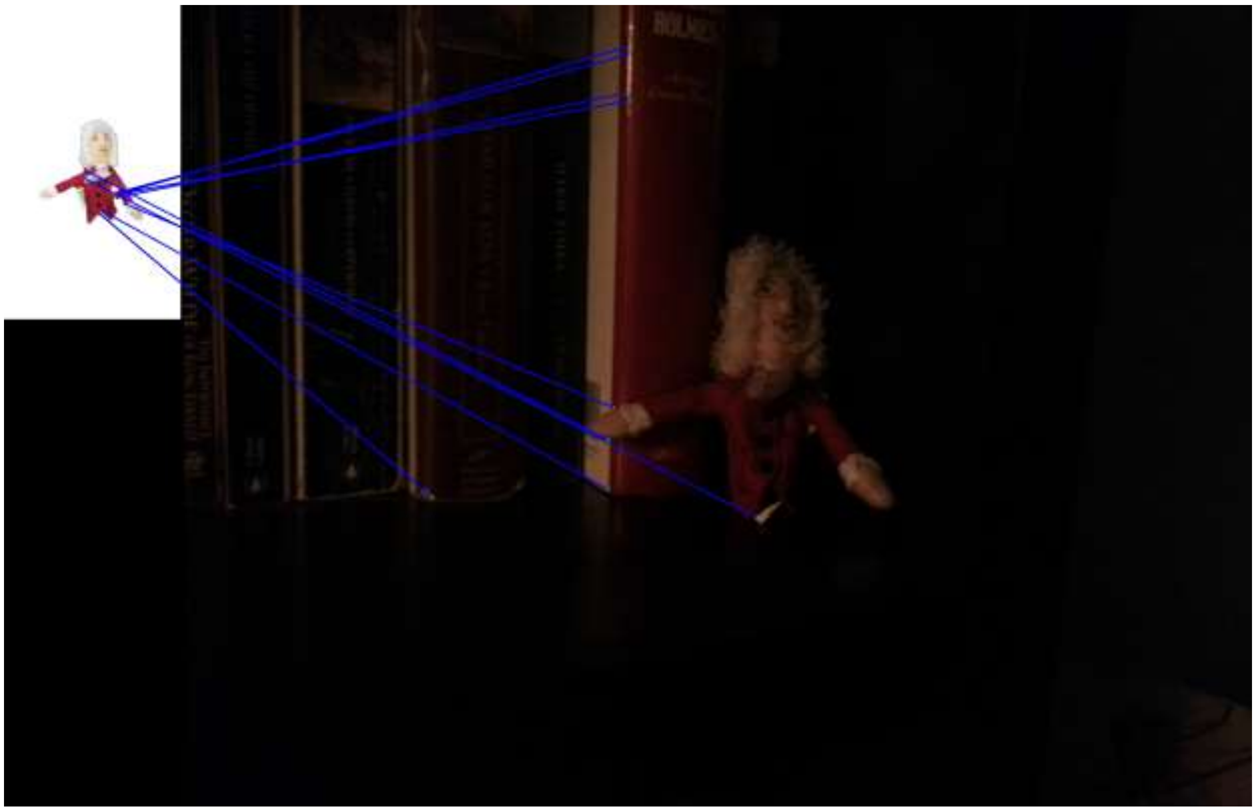

Figure 7: Lighting matches

I wanted to test the limits of ORB with the lighting image. As you can see, there are no successful matches, mostly due to the fact that many of the feature points are not clearly visible in the reduced lighting image. ORB seeks to find the best match, which happens to be incorrect because the true match is too shaded to find. I didn’t try taking the image multiple times here because I wanted to push the limits of matching here.

Rotation

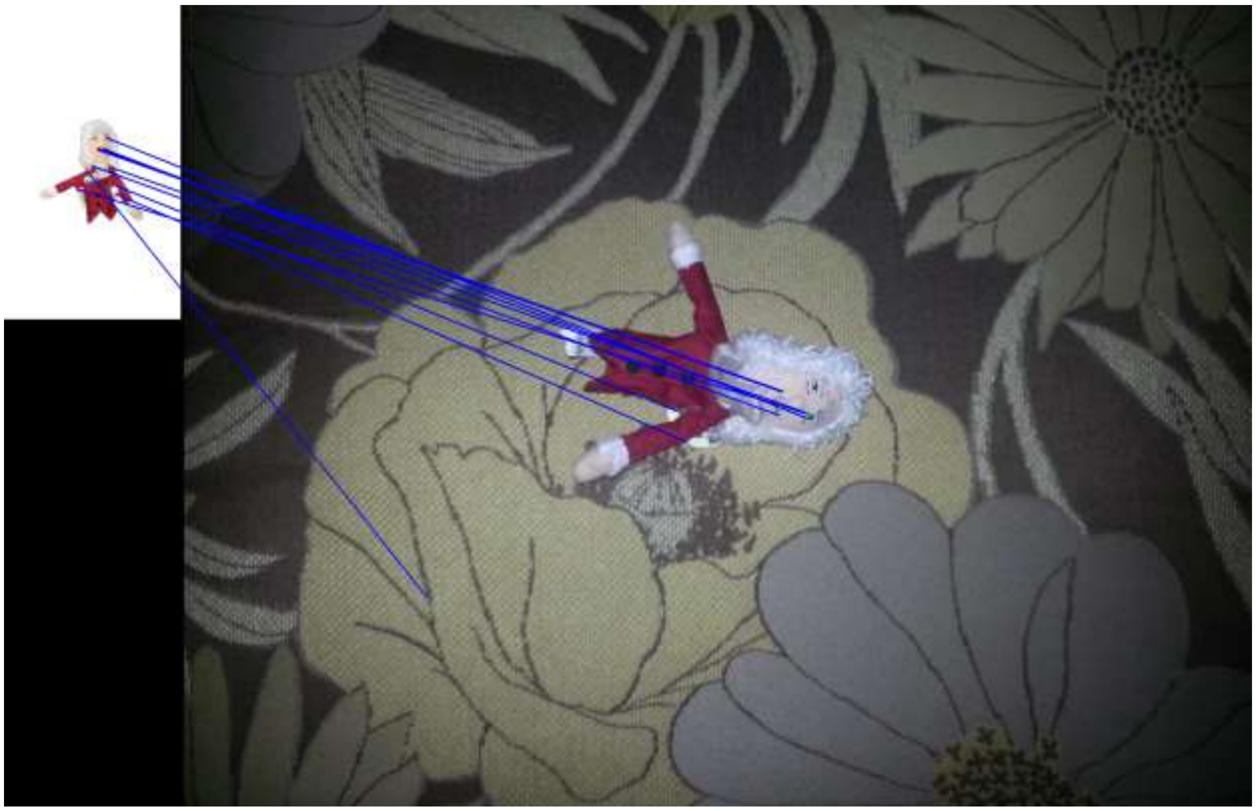

Figure 8: Rotation matching

It was a little hard for me to tell, but I think there are 4 correct matches here. Several keypoints, like the right eye and the edges of the mouth, are successfully matched despite the rotation. I attribute the failed matches in part to using the inferior ORB instead of SIFT. Additionally, the textured background I think caused at least 2 failed keypoint matches.

Scale

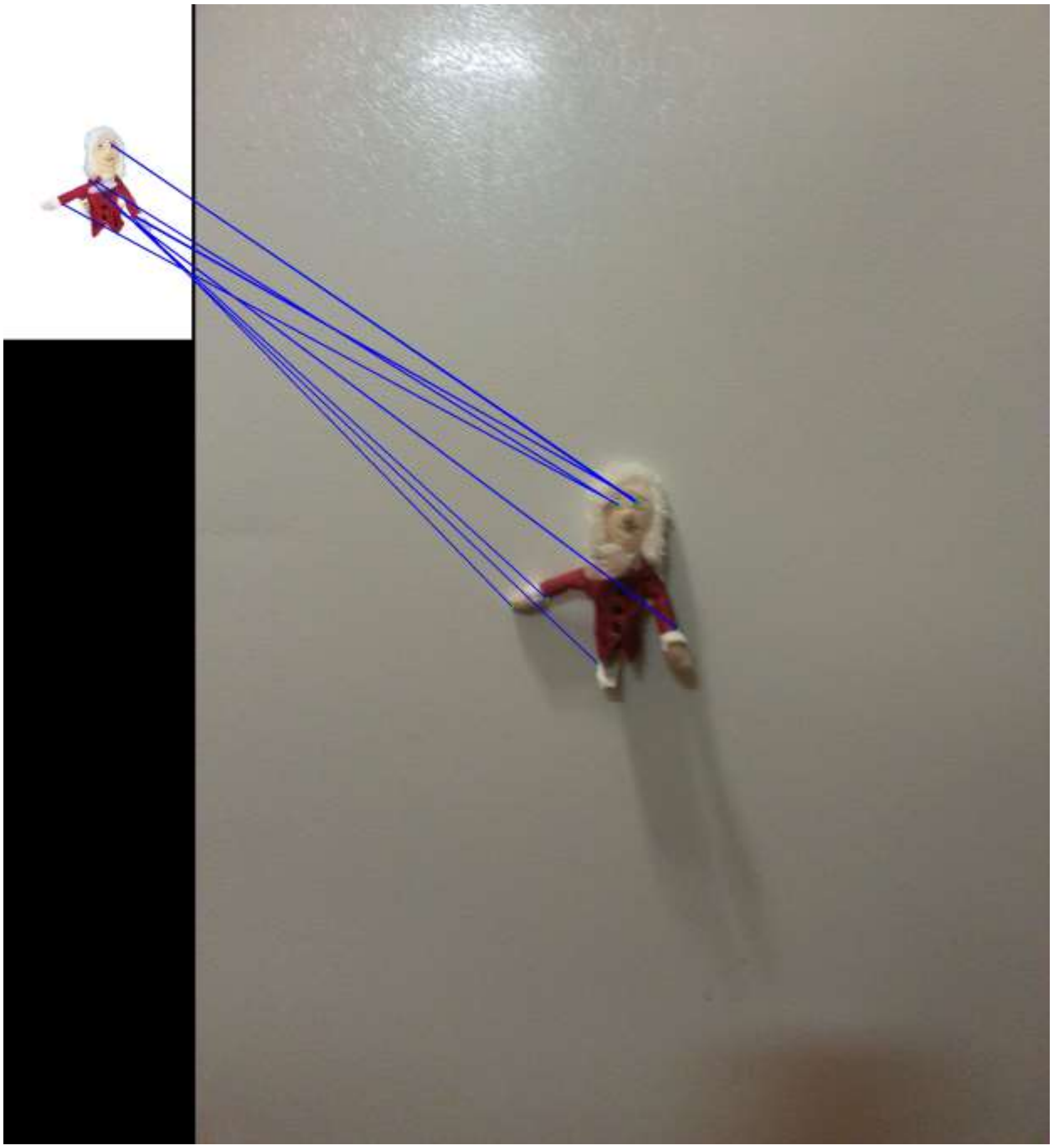

Figure 9: Scale matches

For the scale matches, I see only 1 definitive match – the right eye. Once again, ORB is an inferior feature detection and matching algorithm to SIFT, so some of the error I attribute to ORB. Second, the similar backgrounds I think are throwing off some of the matching. Some of the keypoints on the original are on the edge between the finger puppet and the white background, for instance the cuffs of the coat. Using the scale and rotation invariant matcher, it could get confused with the descriptors since the backgrounds are similar.

Final Notes

I know I attribute a lot of the error to ORB, but having used ORB and SIFT in the Computer Vision course, I can say with certainty that the descriptors produced by ORB are not nearly as good as those produced by SIFT.