CS 477 HW #9

Questions completed: All undergrad level questions in MATLAB

A1. Visualizing SIFT keypoints.

Below in Figure 1, the collage of slide 1 and frame 1 is shown.

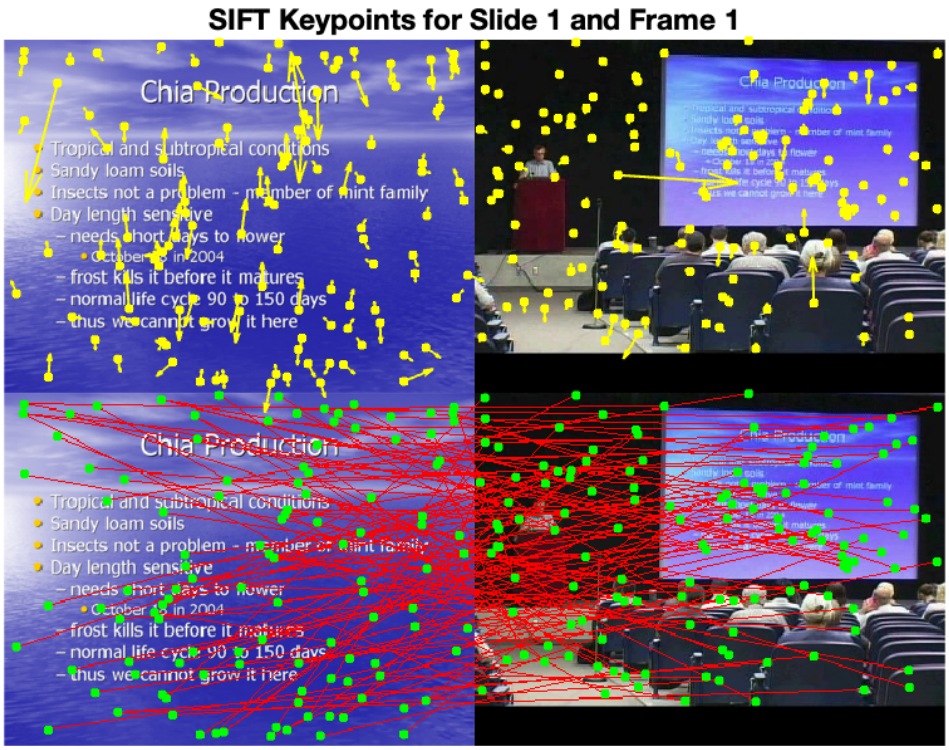


Figure 1: The left two images are a picture of slide 1, and the right two images are a picture of frame 1. Every 10th keypoint along with their gradient vectors from the frame 1 sift file were selected to be plotted in the top right image, and the corresponding keypoints and their gradient vectors from slide 1 were plotted in the top left image. The corresponding keypoints in slide 1 are the Euclidean distance nearest neighbors of the selected keypoints of frame 1, computed from the 128-length descriptor vector of every keypoint. The bottom two images show the same keypoints for the respective images connected by a red line to indicate their nearest neighbor match.

Every 10th keypoint was selected to be plotted to reduce visual clutter. The func-

tion `knnsearch()` was used to compute the Euclidean distance nearest neighbor of every selected keypoint from frame 1 to slide 1. Other notable functions used were `draw_segment()` which was provided in the assignment and `quiver()` which was used to draw the keypoint vectors. Other than that, the main challenge of creating the collage was using careful logic and programming to ensure correct generation of the keypoints.

Below in Figure 2, the collage of slide 2 and frame 2 is shown.

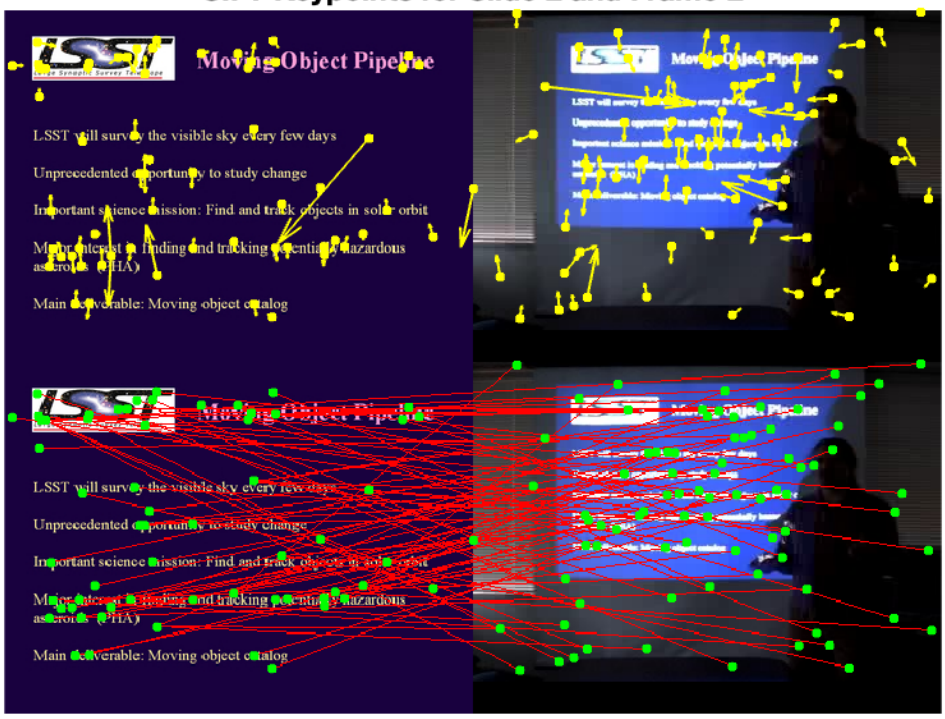


Figure 2: The collage of slide 2 and frame 2 was generated using the same code that was used to produce Figure 1.

Below in Figure 3, the collage of slide 3 and frame 3 is shown.

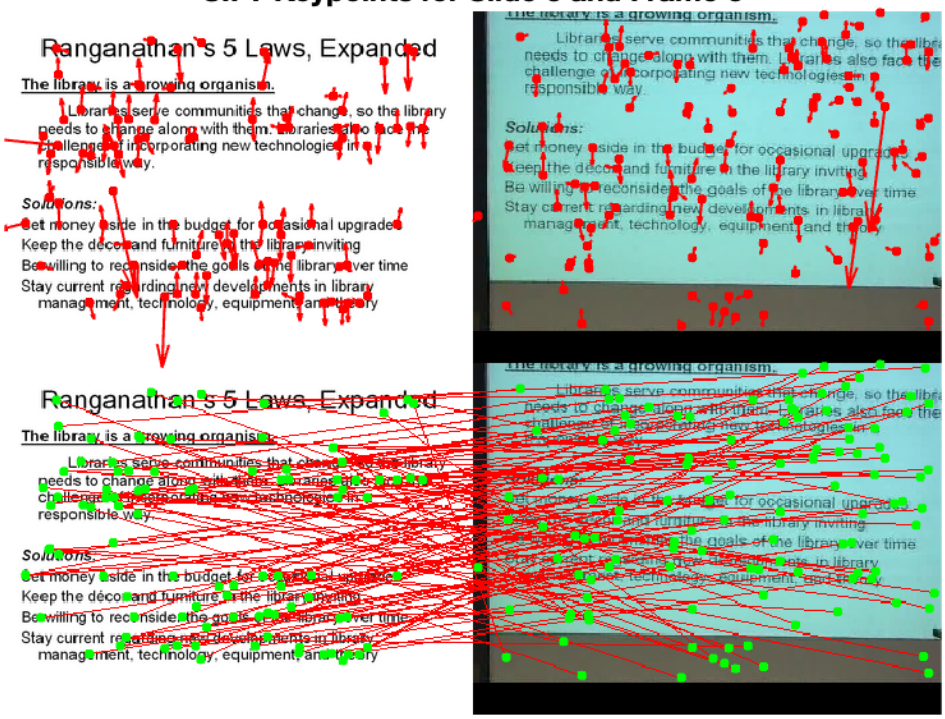
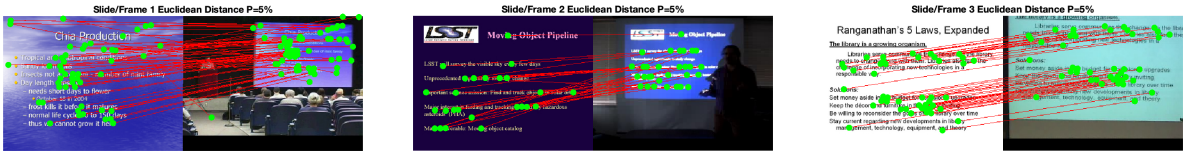


Figure 3: The collage of slide 3 and frame 3 was generated using the same code that was used to produce Figure 1.

A2. Measuring error between features.

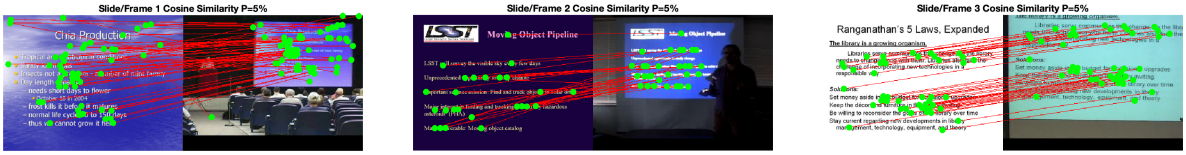
The top 5% best-matching keypoints based on Euclidean distance, cosine similarity, and Chi-square test were plotted. To do this, P was set to 5%, and no keypoints were skipped unlike in A1 where only every 10th point was selected to be plotted. Here, we do not want to skip keypoints because we need to know the actual best-matching keypoints that the different error measures give us. Figure 4 below shows the results of all three error measures.



(a) Top 5% Euclidean distance for slide/frame 1.

(b) Top 5% Euclidean distance for slide/frame 2.

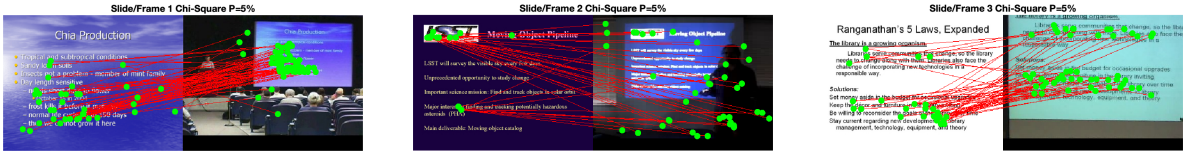
(c) Top 5% Euclidean distance for slide/frame 3.



(d) Top 5% cosine similarity for slide/frame 1.

(e) Top 5% cosine similarity for slide/frame 2.

(f) Top 5% cosine similarity for slide/frame 3.



(g) Top 5% Chi-square for slide/frame 1.

(h) Top 5% Chi-square for slide/frame 2.

(i) Top 5% Chi-square for slide/frame 3.

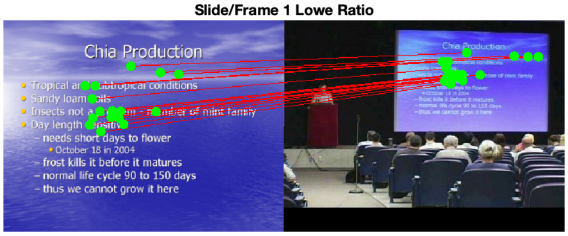
Figure 4: The top 5% of keypoints for the three different error measures (Euclidean distance, cosine similarity, and Chi-square test) for the 3 pairs of slide/frame images.

My findings were that Euclidean distance and cosine similarity produced the exact same pairings of keypoints. This is expected because angular distance is basically the same as Euclidean distance when you think in terms of the components of a vector in the vector space. The Chi-square error measure however did not produce acceptable

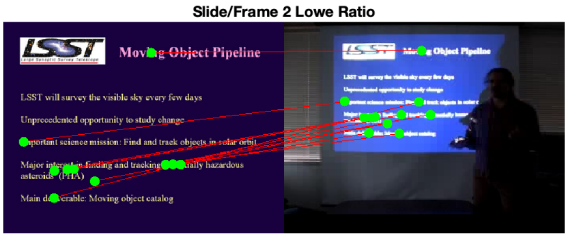
results as seen by Figure 4(h). I am not sure why Chi-square performed worse than the other two error measures as Chi-square is a valid error measurement for a feature vector that is a histogram count. It could be that my Chi-square function was incorrect, or that the nature of squaring values makes it more sensitive to noise and therefore increases the likelihood of producing unwanted results as seen in Figure 4(h). Based on my findings, I would select Euclidean distance at $P = 5\%$ although cosine similarity produces the same results. Higher and lower P values that were above and below 20% were also tested, but nothing interesting came out so I settled on $P = 5\%$ as a nice number that produced matches mostly within the slide portion of every frame (at least for Euclidean distance and cosine similarity).

A3. Pruning keypoint matches.

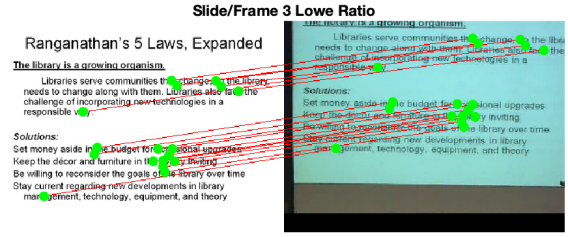
Below in Figure 5, a Lowe ratio of 92% was used to prune the keypoint matches found from Euclidean distance. The filtering was done by first computing the nearest and second nearest Euclidean distance neighbor of every keypoint of the frame and then filtering out keypoint matches if the second nearest neighbor distance was too close in value (over 92% ratio) compared to the nearest neighbor distance.



(a) Lowe ratio of 92% for slide/frame 1.



(b) Lowe ratio of 92% for slide/frame 2.



(c) Lowe ratio of 92% for slide/frame 3.

Figure 5: The keypoint matches for Euclidean distance were filtered out if the nearest neighbor distance was greater than 92% of the second nearest neighbor distance. The original Lowe ratio of 80% filtered out too many matches to be of use.

Using the a Lowe ratio of 80% produced too few matches, so a final value of 92%

was decided after testing various thresholds. The Lowe ratio approach helped filter out incorrect and/or undesirable keypoint matches and is therefore a good pruning method to use alongside Euclidean distance error measure. Comparing Figure 5 to Figure 4(a-c), one can see that Lowe ratio pruning filtered out most of the incorrect keypoint mappings from the Euclidean distance matching, specifically keypoints that mapped more than once to another keypoint. I realize now that the effect of the Lowe ratio pruning would have been easier to see if I used a higher P value in A2 to allow for more filtering from Lowe ratio. Lowe ratio and the P value can be seen as two knobs for fine-tuning results. It would have been interesting to see the results of a higher P value and a lower Lowe ratio. For at least these 3 sets of images, matching the SIFT feature vectors as close as possible (Euclidean distance) and than filtering out weaker candidate feature vectors that look close to other feature vectors (Lowe ratio) seems to work well, but maybe we could get away with using just one or the other at an extreme threshold.

A4. Testing preferred error measure and threshold.

All combinations of P values and Lowe ratio values were tested in a nested for-loop. A diagonal confusion matrix was produced with $P = 73\%$ and a Lowe ratio of 0.65. The confusion matrix for this combination is shown below.

```
 15    0    0
  0   13    0
  0    0  161
```

The maximum scoring combination if we are subtracting the summation of the non-diagonal from the summation of the diagonal was $P = 44\%$ and a Lowe ratio of 1 (so no Lowe ratio filtering). The confusion matrix for this combination is shown below.

```
302  215   36
 52  233   22
 55   13  525
```

There were many more interesting combinations. For example, we could optimize for the skew caused by slide/frame 3 since the entire frame is basically the slide in this case.

As long as the diagonal of confusion matrix contains the highest counts, we can say that the combination of Euclidean filtering and Lowe ratio accurately detects the right slide in the right frame. This is because the value of the ith row and jth column is the number keypoints matched to the ith slide and the jth frame. Therefore, only matches along the diagonal are correct. We can scale the confusion matrix as the number of slides and frames grow, and it has the benefit of providing information on the mismatches between slides and frames so that we can fine tune our parameters.