Assignment 10 (Group)

Pratik Bhandari and Simon Swenson
pratikbhandari@email.arizona.edu
simonswenson@email.arizona.edu
Graduate Students

April 12, 2019

Group Contribution

For the assignment, the two of us worked together for each question. Every question was brainstormed and discussed with both members being involved. Similarly, the coding was done by both members in parts, one adding on to the work of the other and vice versa. The report was also worked on together as a team by using *Overleaf* as a platform for both team members to work concurrently on the report.

Part A

Problem 1

First, the SIFT features of the frame and slide images were computed using the VLFEAT executable provided in the assignment. The following command was used to get the '.sift' files:

```
./sift <frame|slide.pgm>
```

Then, the euclidean distance between all pair of SIFT keypoints between the corresponding frame and slide keypoints were computed. From this distance matrix, slide keypoints that were the nearest neighbors to frame keypoints were determined.

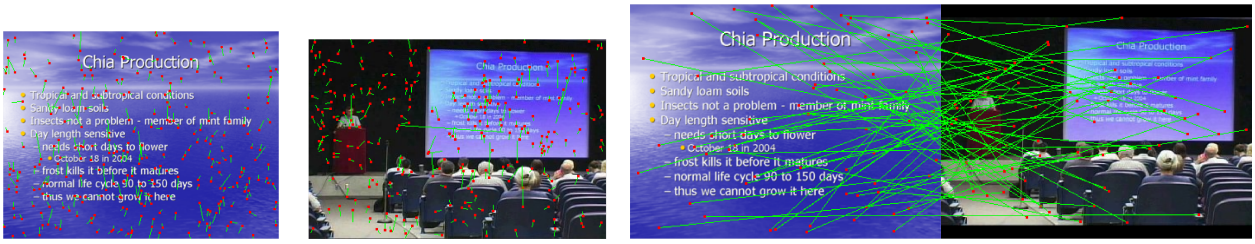

Figure 1: Four image collage of the first frame-slide image pair. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined.

In the first and second image, the keypoints are visualized using the scale and angle measures for each keypoint. The scale is shown using the length of the line (green in color) and the direction is shown using the orientation of the line. Each keypoint coordinate is represented using a red dot on one end of the green line.
Similarly, for the last two images, matched keypoints are joined using green lines and keypoints themselves are represented by red dots. In the first two images, every 5 keypoints are shown whereas for the last two images, every 20th keypoints was shown to reduce clutter.
We can see that by using just the euclidean distance to match keypoints, we get very bad results with

minimum number of actual matches and a lot of false positives. The results for the other two frame-slide pairs are shown below:

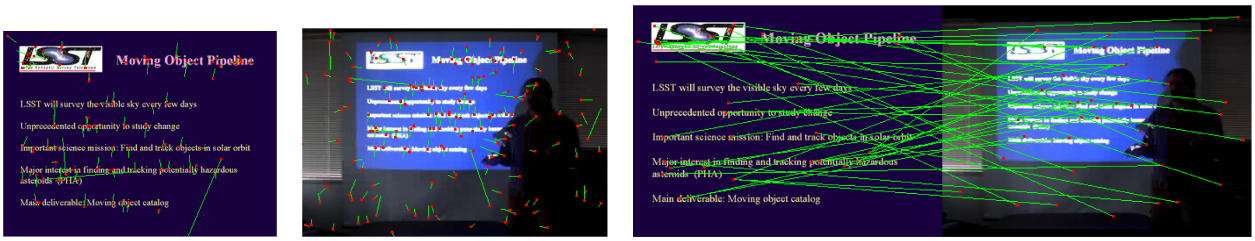

Figure 2: Four image collage of the second frame-slide image pair. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined.

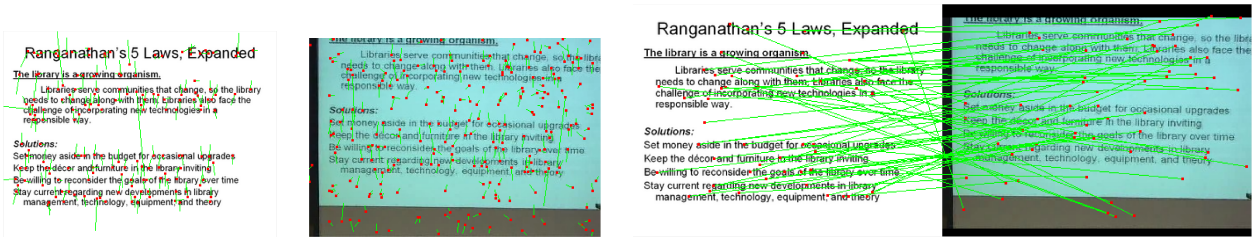

Figure 3: Four image collage of the third frame-slide image pair. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined.

Problem 2

Now, in order to clean the clutter and produce only the best P percentage of the matches, we experimented with different values of P. Starting at P=10%, we found a lot more bad matches then good ones. Going down to P=2.5% produced fewer results than we wanted to show. Finally, we settled at P=5% where there was a good balance between the good and the bad matches.
Here, we wrote out all the pixels and did not step over them to decrease the output lines. We observed that the output was better in terms of matches with closer matches being better. The output for the three frame-slide pairs in the form of collages are given below:

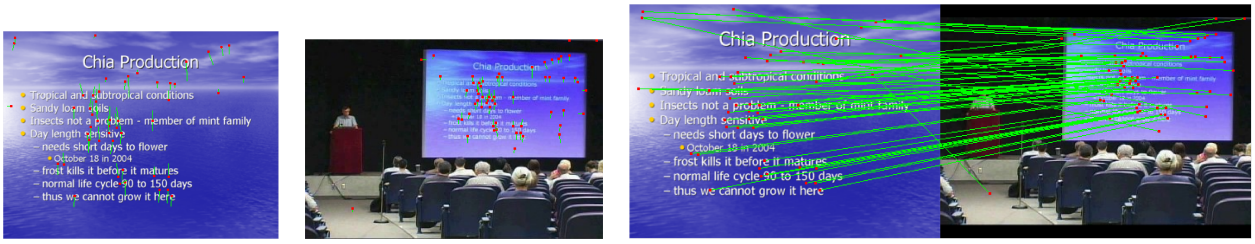

Figure 4: Four image collage of the first frame-slide image pair with the best 5% matches. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased from last time.

Figure 5: Four image collage of the second frame-slide image pair with the best 5% matches. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased from last time.

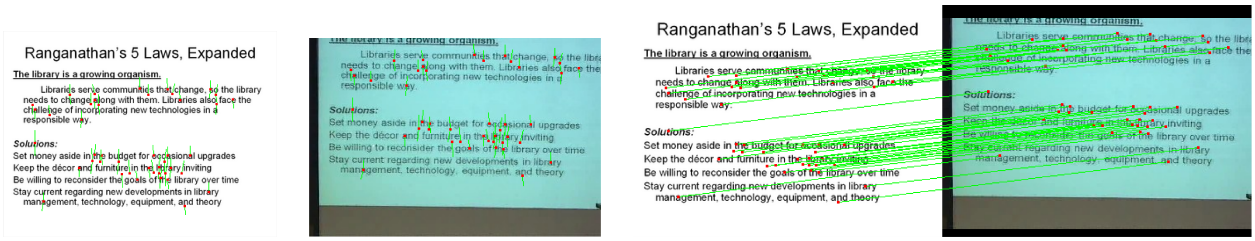

Figure 6: Four image collage of the third frame-slide image pair with the best 5% matches. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased from last time.

Problem 3

Now, instead of using the Euclidean distance as a measure of neighborhood, we calculated the angle between the two 128 element feature vectors of for each pair of keypoints for the frame and slide images and used that value as the measure of distance. The angle (θ) was calculated using the following relation:

$a.b = |a|.|b|cos(\theta)$

Here, a and b are the 128 element feature vectors.

The results were definitely better than the ones obtained from the basic Euclidean distance. We obtained more accurate matching. There were a few outliers which were pointing to sharp opposite directions but their count was much lower compared to the accurate matches. Just like in the previous question, we plotted only the top 5% of the matches to reduce the clutter. The output collages are shown below:

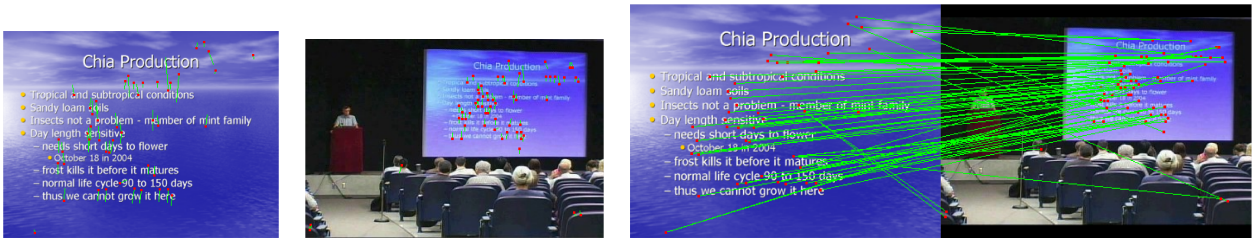

Figure 7: Four image collage of the first frame-slide image pair using angle between feature vector as a measure of distance. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach.

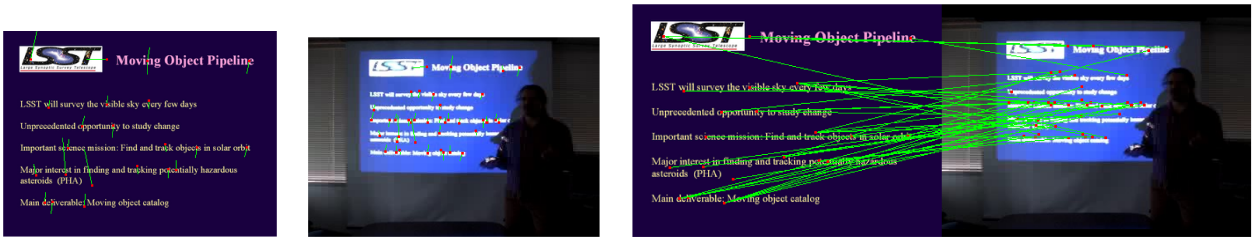

Figure 8: Four image collage of the second frame-slide image pair using angle between feature vector as a measure of distance. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach.

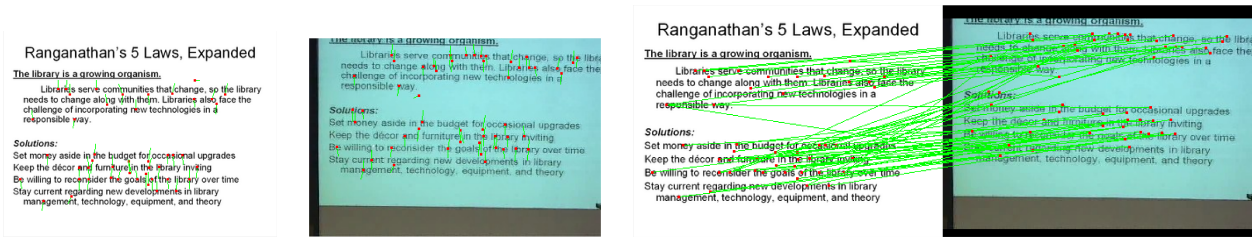

Figure 9: Four image collage of the third frame-slide image pair using angle between feature vector as a measure of distance. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach.

Problem 4

For this question, we used the χ^2 based measure to find matching keypoints in the frame-slide image pair. Since the 128 features of a keypoint as analogous to histogram counts, it makes sense to use the χ^2 based measure. This is given by:

$$\chi^2 = \frac{1}{2}\sum_{k=1}^{K}\frac{(h_1(k) - h_2(k))^2}{h_1(k) + h_2(k)}$$

Like before, the keypoints pair with the minimum χ^2 value were chosen as matching keypoints between the frame-slide pair. The results from using the χ^2 based measure was found to be better than using the angle between the keypoint vectors as distance measures. For cases when the feature vector had 0 value on both vectors, a divide-by-zero bug appeared (since we are dividing by the sum of the elements). In order to prevent this from happening, we smoothed the valued by adding 0.0001 to all feature vector values. This did not affect the overall result by a considerable amount. In order to reduce the clutter, we plotted only the top 2.5% of the total matches. This also helped in visualizing only the relevant matches and removed most of the false-positives.

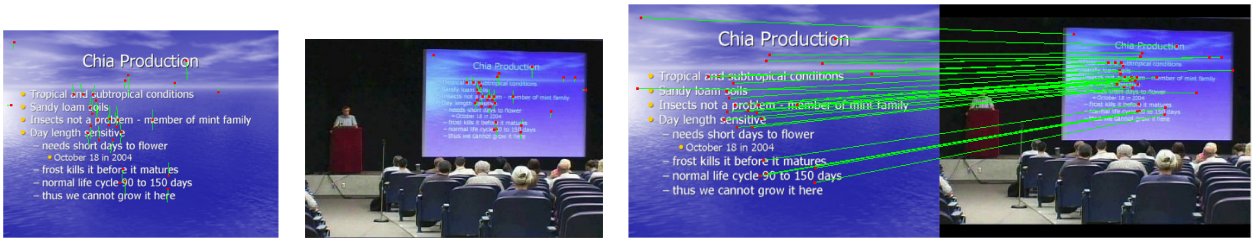

Figure 10: Four image collage of the first frame-slide image pair using the χ^2 based measure. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the angle between vectors as distance approach.

Figure 11: Four image collage of the second frame-slide image pair using the χ^2 based measure. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the angle between vectors as distance approach.

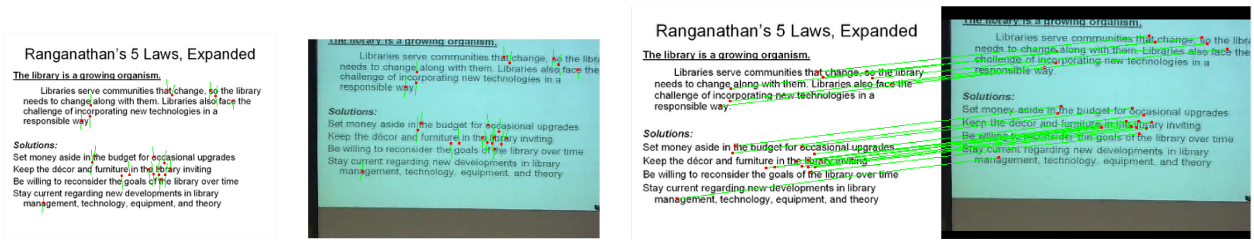

Figure 12: Four image collage of the third frame-slide image pair using the χ^2 based measure. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the angle between vectors as distance approach.

Problem 5

For this question, we used Lowe's method of thresholding. Having already computed the distance to the nearest neighbor using Euclidean distance, we calculated the distance to the second nearest neighbor and divided that to the distance to the nearest neighbor to get our desired ratio. This ratio was used to make the matching more robust.
As described by Lowe, instead of taking the top P% of the raw distance values, we pruned the matches based on a threshold of the ratio value. We found that for the images, a threshold ratio of 0.7 seemed a good fit. So, we rejected any matched with ratio greater than 0.7.
This approach did help with getting better results compared to the basic Euclidean distance measure. However, compared to the top 5% approach, the results did not improve by a drastic amount and seemed to be of the same quality. There were some false positives but were mostly because of noise and could be neglected. We observed that both methods (Top 5% or ratios) have a parameter to tune (how many to keep vs the ratio), but the top 5% method assumes that 5% of the features will have valid matches. We assume that if the pictures are of two very different things, then the ratio approach will be much better compared to the Top 5% approach.

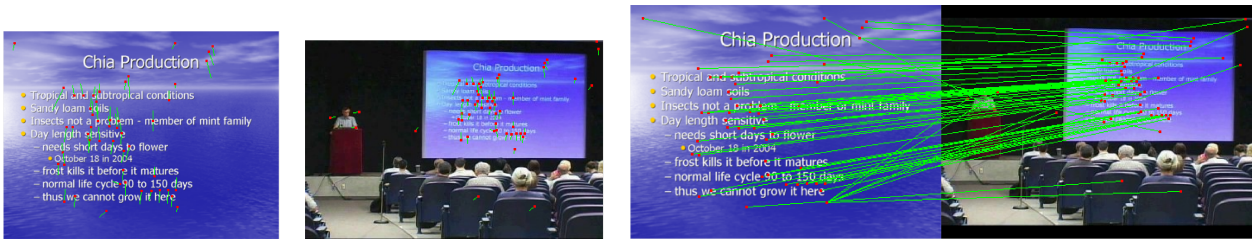

Figure 13: Four image collage of the first frame-slide image pair using using Lowe's method for thresholding at 0.7. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach but does not change much from the Top 5% approach.

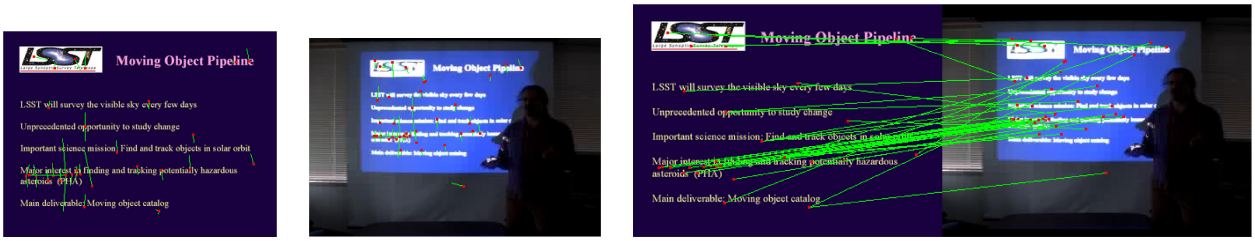

Figure 14: Four image collage of the second frame-slide image pair using using Lowe's method for thresholding at 0.7. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach but does not change much from the Top 5% approach.

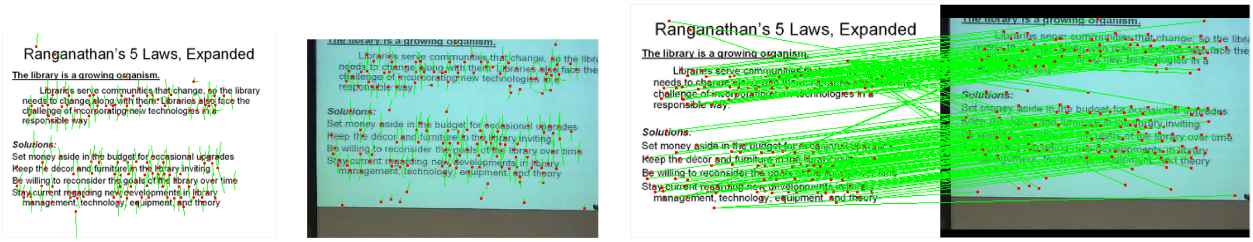

Figure 15: Four image collage of the third frame-slide image pair using using Lowe's method for thresholding at 0.7. The first image is the image of *slide* with its keypoints, the second image is the image of $frame1$ with its keypoints. The last two images have lines drawn between them such that matches keypoints are joined. We see that the match quality is increased compared to the basic Euclidean distance approach but does not change much from the Top 5% approach.

Problem 6

For this part of the assignment, we explored the situation of matching the feature points of multiple pairs of images with each other, without knowing which image belongs to which. This was an interesting experiment to see just how well the techniques used above could perform with a bunch of unlabeled images, which is often the case when applying feature detection techniques (as in SfM). We decided to use the distance metric, as above, since we found it to give the best results when compared with angular difference and χ^2. We also opted not to use the ratio metric, since, earlier, we did not notice that much of an improvement over just taking the best matches for the pairwise problems, and we expected about the same results for the combined 3-way image matching.

With the same 5% top results that we used in problems 2-3, we got very good results. Only one entry appeared outside of the diagonal in the confusion matrix.

	Frame 1	Frame 2	Frame 3
Slide 1	4	0	0
Slide 2	1	24	0
Slide 3	0	0	136

One interesting result is that the top 5% matches is actually quite biased toward the third slide/frame pair. This is likely due to the fact that the slide is plain black-and-white, and it fits in the frame much better than the previous two. This means that the image is much simpler, and the features can be matched to a much higher degree of certainty. In this case, using the ratio technique might have actually helped un-bias the data, since ratios are normalized in some sense, whereas distances can be arbitrarily small or large.

Note that the code we used for Problem 6 actually caused a 16 GB machine to run out of memory. (A 32 GB machine worked fine, however.) This is due to the large amount of entries in the concatenated matrices. We used matrix multiplication techniques, so, at this point, we had three matrices that were about 1 GB. This might also have been exacerbated by the previous homework problems, as we did not use the "clear" keyword to free any previous variables.

Problem B (Grad Portion)

This portion of the assignment focused on a specific feature detection algorithm: the Harris Corner Detector. Whereas for the previous problems, we were given the sift feature points and descriptors, and worked from there, our goal for this part was to actually come up with the feature points.

The Harris corner detector first appears complicated, but can be easily broken up into several steps. First, the image gradient is computed similar to Homework 8, using two blurred convolutions (one for x-direction edges and one for y-direction edges). The blurring allows us to consider corners at different image

scales. For example, if we wanted to omit many of the higher-frequency corners in the "climber" image below, we would increase the standard deviation of that Gaussian. From here, things start to differ. Each image coordinate has a 2x2 matrix associated with it, called the outer product, which reminded me a bit of a covariance matrix.

From there, we picked a window size to sum over. You can think of this as representing the total "cornerness" over a certain range. Note that, because a dark-to-light edge transition will have the opposite magnitude of a light-to-dark transition, windows with opposing edges will tend to obliterate eachother, so this sum gets larger as the corner types are similar. This also helps to combat noise, similar to the Gaussian convolution above.

To find a corner, specifically, we want both eigenvalues to be large and similar. This makes sense, as, if only one were large, this would be an edge rather than a corner. The Harris edge criterion has two terms, and together, they accomplish the task:

$$det(H) - k(\frac{trace(H)}{2})^2$$

Here, I think the $det(H)$ term, which is equivalent to $\lambda_1\lambda_2$, ensures that both are large. The $trace(H)$ term, which is equivalent to $\lambda_1 + \lambda_2$, ensures that both have a similar magnitude. Finally, the local maxima are found, and a threshold is used to filter out corners that are not strong enough for our purposes.

We applied this technique to five images, tweaking the parameters for the algorithm each time, since each image is unique and requires special care to find the corners. The parameters we used are as follows: (We did not modify k from the suggested 0.5.)

	Gaussian Std-dev	Window Size	Cutoff
Building	1.0	4	0.00001
Chandelier	1.0	4	0.001
Climber	3.0	2	0.00000001
Leaf	1.0	4	0.001
Tent	2.0	4	0.0001

The results are encouraging, and the corners found seem to correspond with our intuitive notion of corners well. In summary, the parameters seemed to affect the corners detected in the following ways:

- Gaussian Standard Deviation - This changed the image scale, causing the detector to omit smaller-scale corners, such as those in high frequency areas. We also found that corners detected with higher Gaussian standard deviations were offset from their true positions. We hypothesize that this is due to the shape of Gaussian distributions: Gaussians can steeply increase before leveling off before steeply decreasing again, causing those "steep" portions of the function to be detected as corners rather than the actual corners. We generally avoided this situation by decreasing the standard deviation if we noticed this in the images.

- Window Size - This ensured that corners were essentially consistent over a larger or smaller size of the image. This is similar to ignoring noisier areas of the images.

- K - Given the explanation of the terms, above, increasing k will give more weight to stronger edges and corners, while decreasing the weight will ensure that we find *corners*, no matter how strong or weak they are.

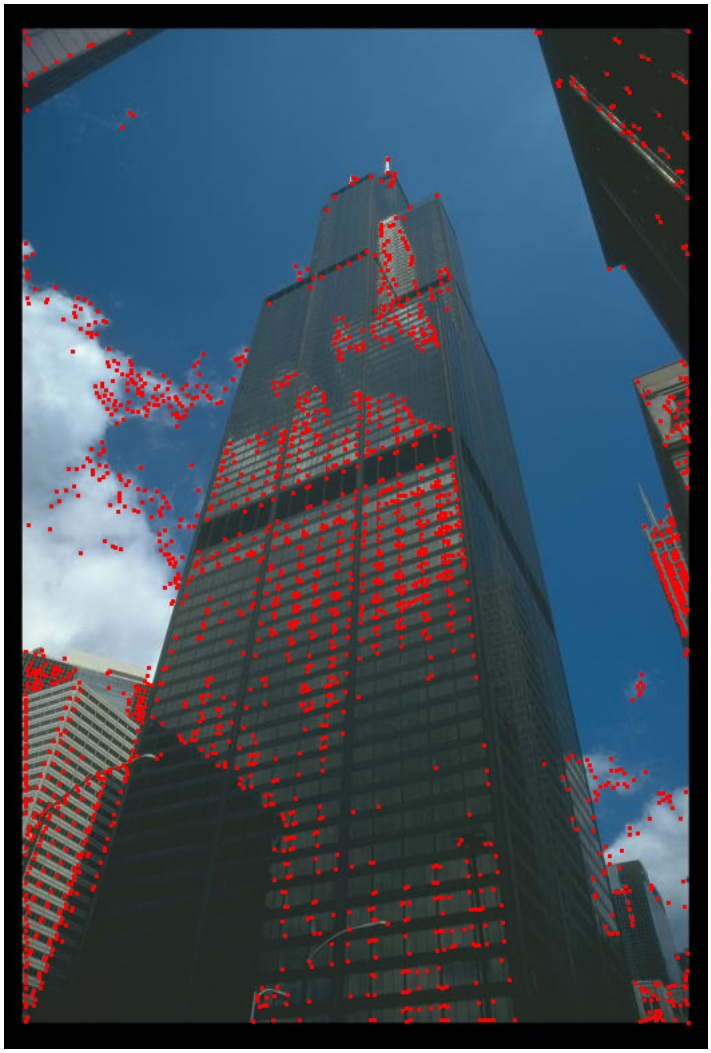

Figure 16: The building image, seen in a previous assignment. Since buildings have many sharp angles, this should be a good image to test the corner-finder on.

Figure 17: The chandelier image, as seen previously on an assignment. Here, many corners are found, but pay attention to the bottom-right. Since the background here is so close in color to the chandelier, the threshold removes such matches. However, decreasing the threshold too much results in more corners in strange spots, so it's a trade-off, in this case.

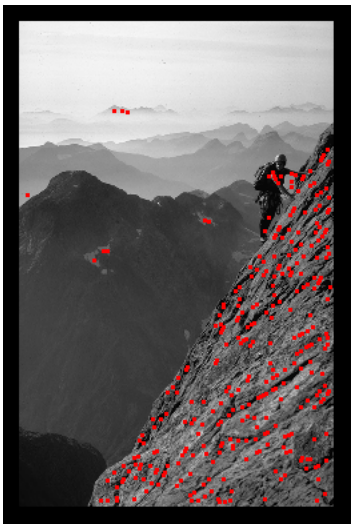

Figure 18: The climber from the convolution assignment. Here, we were hoping to highlight the mountain more, but we realized that those are simply edges, not corners. Instead, though, the detector finds the corners between the lighter rocks (snow?) and darker rocks very well.

Figure 19: This image represents an ideal use-case for the Harris Corner Detector. There is clear delineation between the background and the leaf, and the leaf has many points, which should be detected. Indeed, using the values above, we get very good results, where only the points (corners) of the leaf are detected.

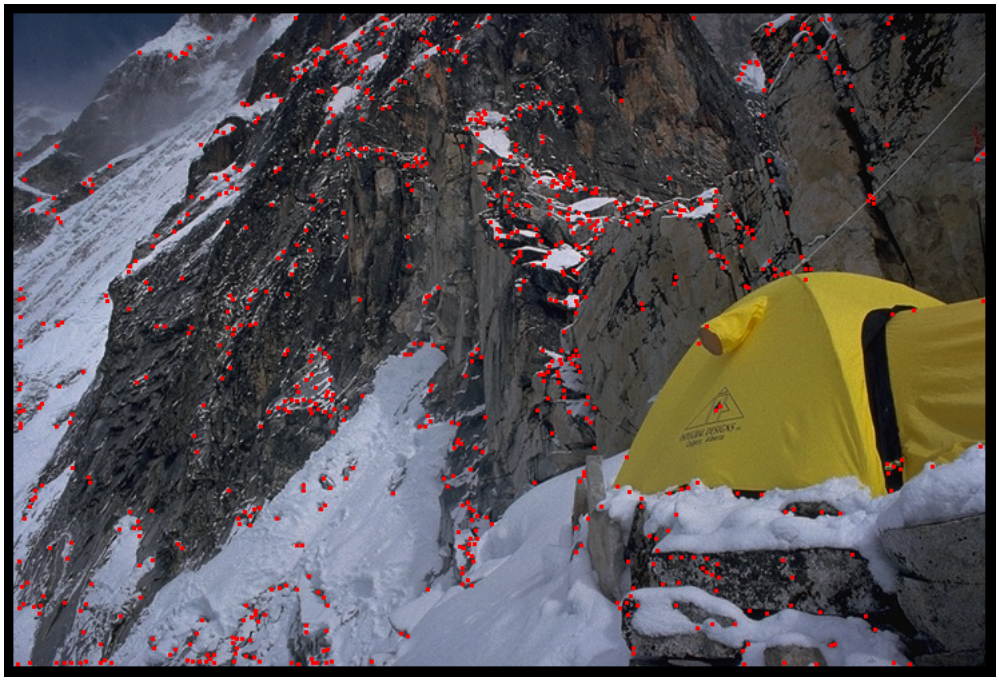

Figure 20: The corners between the dark rocks and white snow are found with ease in this image. However, the yellow tent, being close in color to the white snow, did not have many meaningful matches. In addition, the tent is fairly rounded, so it does not have many corners to begin with.