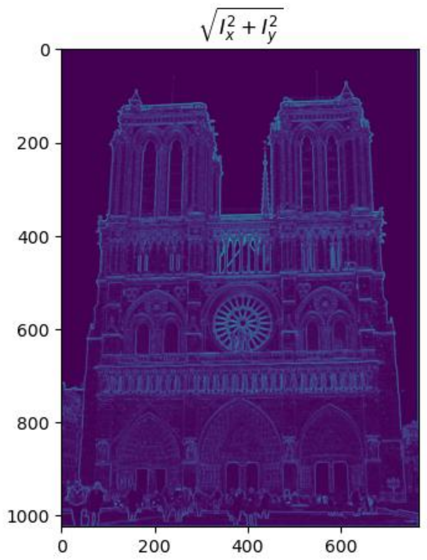
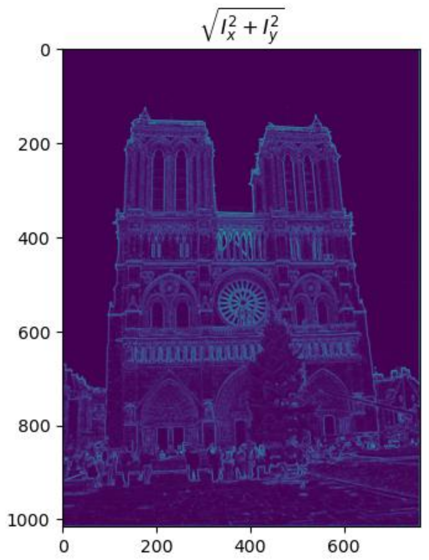
CS 4476/6476 Project 2

Shuvo Newaz
shuvo.newaz@gatech.edu
snewaz3
903614132

Part 1: Harris corner detector

[insert visualization of \sqrt(I_x^2 + I_y^2) for Notre Dame image pair from proj2.ipynb here]



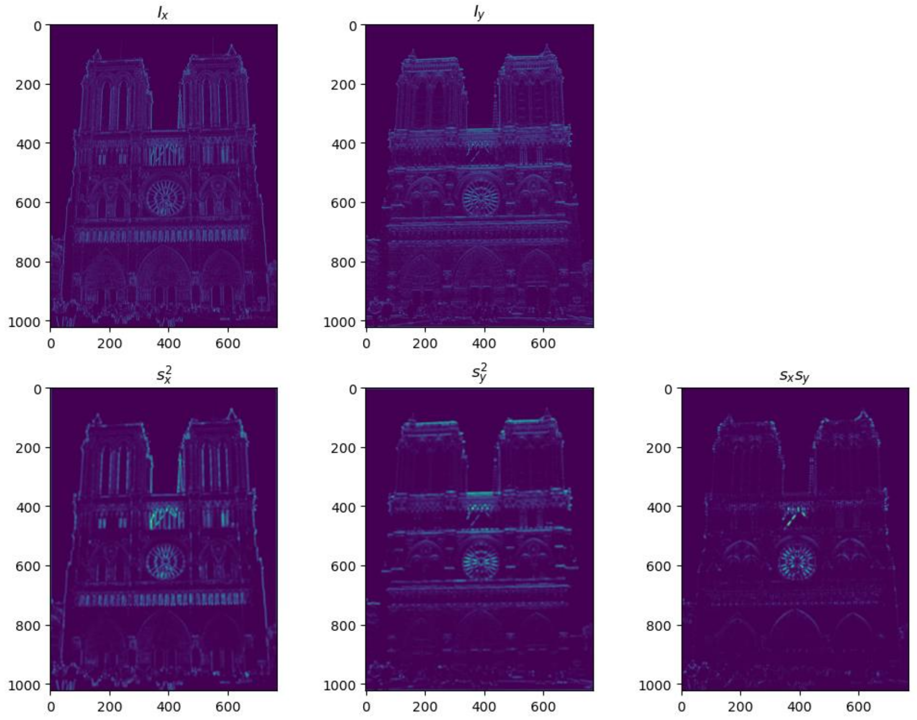


[Which areas have highest magnitude? Why?]

The outer boundary, the circular part and other parts with clear boundaries have very high magnitudes. This is because the gradient is high at the boundaries that separate distinct color regions.

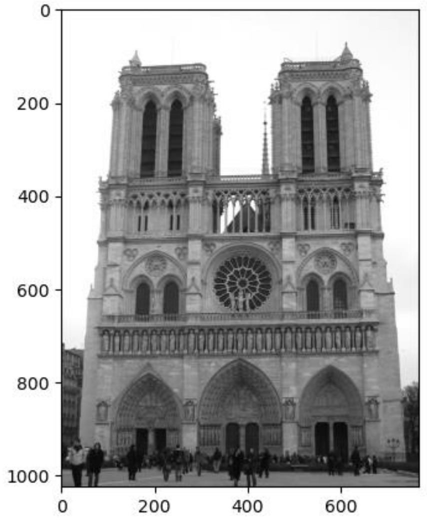
Part 1: Harris corner detector

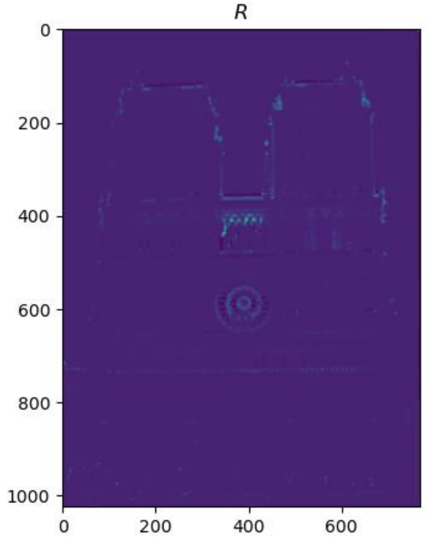
[insert visualization of I_x, I_y, s_x^2, s_y^2, s_xs_y for Notre Dame image pair from proj2.ipynb here]



Part 1: Harris corner detector

[insert visualization of corner response map of Notre Dame image from proj2.ipynb here]





[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

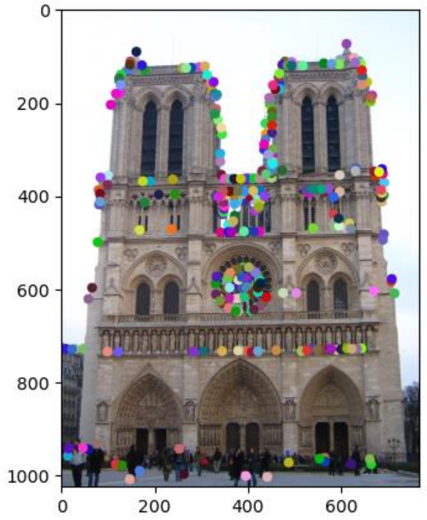
Gradient features are invariant to additive shifts. Since gradients are differences in pixel intensities, equal shift in intensities of all the pixels (not exceeding extreme values) do not change the gradient. $x_2 - x_1 = (x_2 + a) - (x_1 + a)$

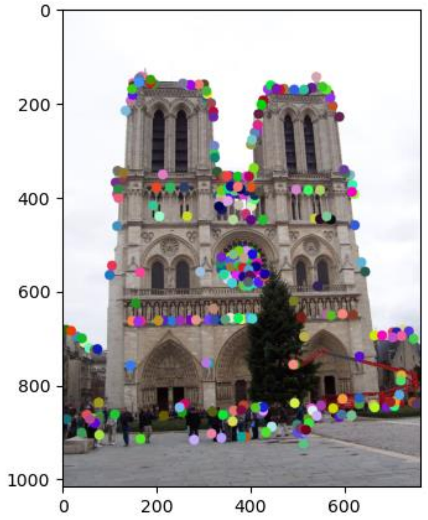
They are not invariant to multiplicative gains. Similar argument as above, but the difference changes when everything is multiplied by a constant. $x_2 - x_1 \neq ax_2 - ax_1$.

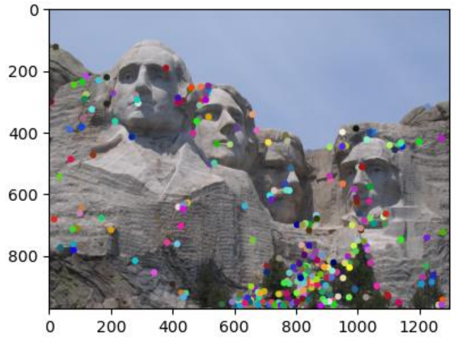
Part 1: Harris corner detector

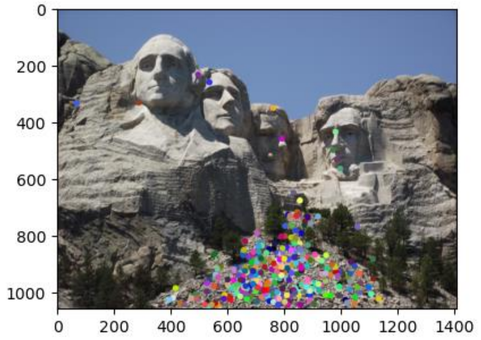
[insert visualization of Notre Dame interest points from proj2.ipynb here]

[insert visualization of Mt. Rushmore interest points from proj2.ipynb here]



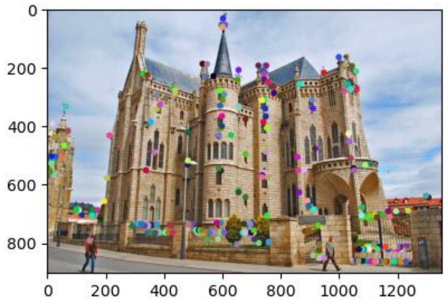


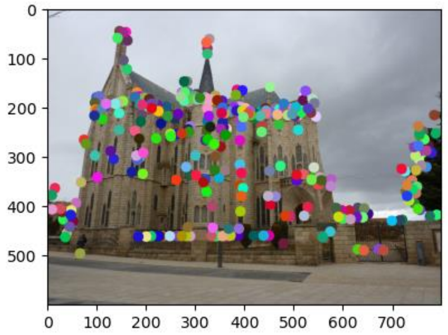




Part 1: Harris corner detector

[insert visualization of Gaudi interest points from proj2.ipynb here]





[What are the advantages and disadvantages of using maxpooling for non-maximum suppression (NMS)?]

Advantage: Maxpooling is fast. It can extract the local maximums very quickly.

Disadvantage: If zero-padded, the local maximums at the edges may be smaller than a neighboring non-maximum value. This can happen because any confidence value not zero is greater than zero, thus treated as local maximum, whereas there can be a group of large confidence values in the middle of the matrix, which leads to some large values being discarded. This can also happen in non-edge regions where neighboring values are all small, making the local maximum also small.

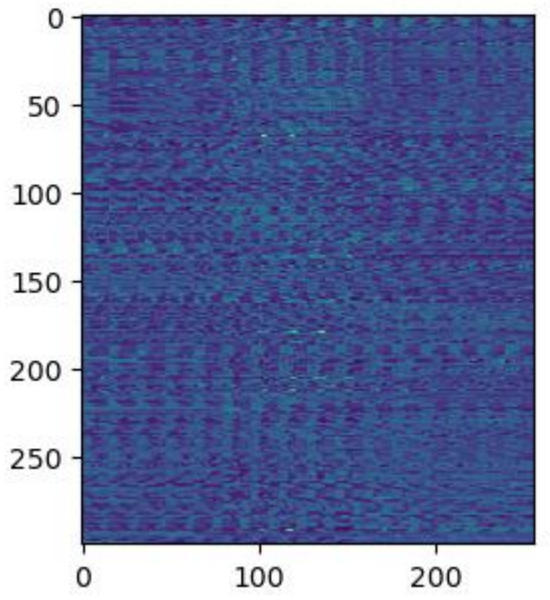
Part 1: Harris corner detector

[What is your intuition behind what makes the Harris corner detector effective?]

The intuition is that a point that has a high value for gradient in both directions is likely to be a corner.

Part 2: Normalized patch feature descriptor

[insert visualization of normalized patch descriptor from proj2.ipynb here]

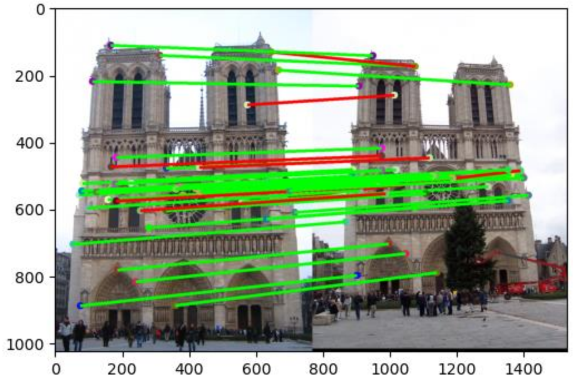


[Why aren't normalized patches a very good descriptor?]

These descriptors do not take into account the image gradient magnitudes and orientations at all. They work directly with normalized image pixels, which is a less reliable in terms of finding matches because it will match with anything that has a similar intensity.

Part 3: Feature matching

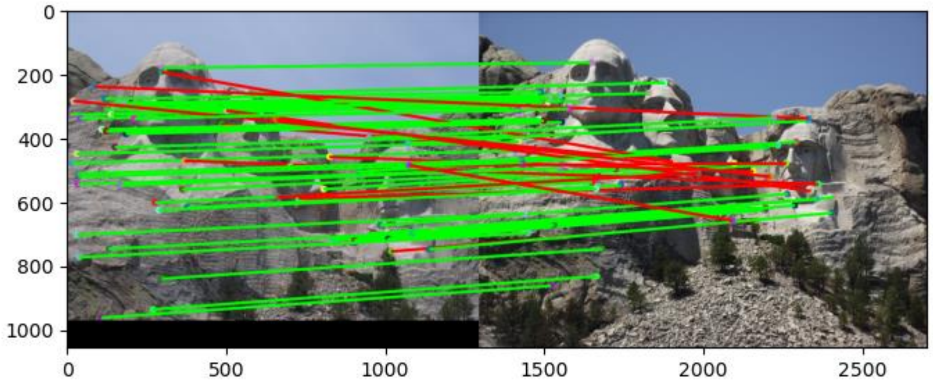
[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]



# matches (out of 100): 41

Accuracy: 34%

[insert visualization of matches for Mt. Rushmore image pair from proj2.ipynb here]

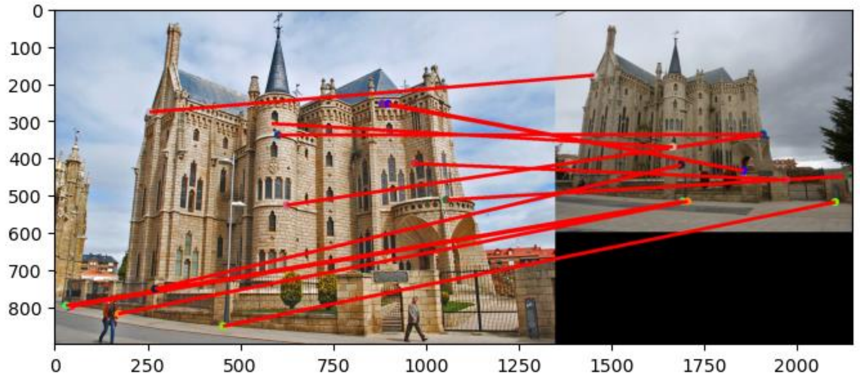


# matches: 65

Accuracy: 52.00%

Part 3: Feature matching

[insert visualization of matches for Gaudi image pair from proj2.ipynb here]



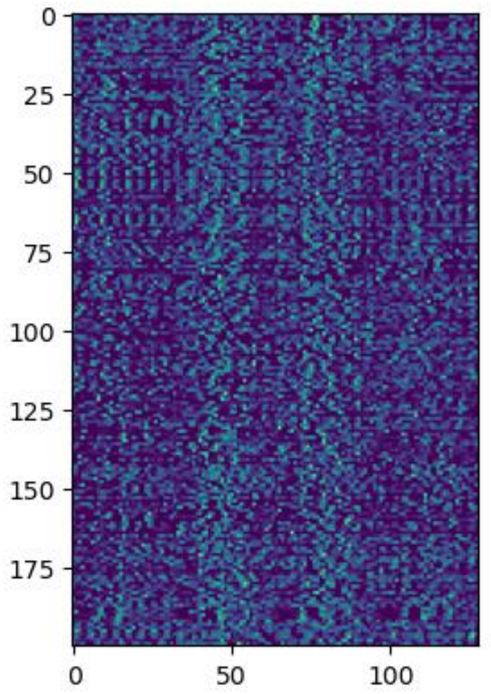
# matches: 13
Accuracy: 0.00%

[Describe your implementation of feature matching here]

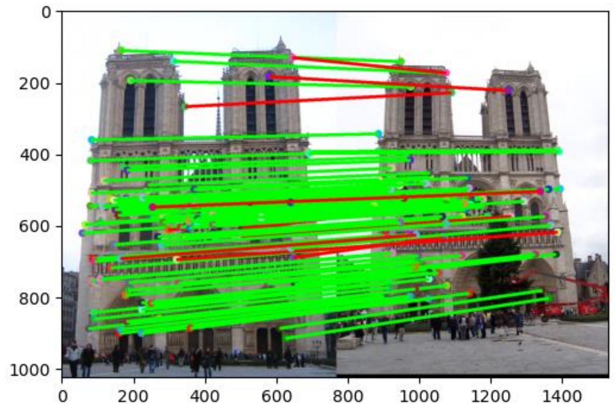
1. Compute distances between the features from set 1 to features from set 2.
2. Sort the distances and keep track of the sorted indices.
3. For every point in feature set 1, compute the distances to its nearest neighbor and second nearest neighbor in feature set 2. Take the ratio of these 2 distances (should be less than 1). This is the nearest neighbor distance ratio (NNDR)
4. Set a threshold. Select only the points from the two features sets for which the NNDR is less than or equal to the threshold.

Part 4: SIFT feature descriptor

[insert visualization of SIFT feature descriptor from proj2.ipynb here]



[insert visualization of matches (with green/red lines for correct/incorrect correspondences) for Notre Dame image pair from proj2.ipynb here]

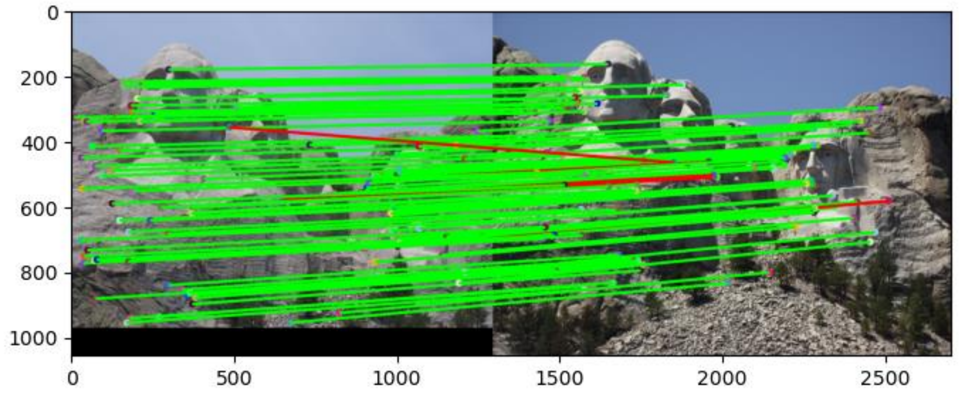


# matches (out of 100): 102

Accuracy: 88.23%

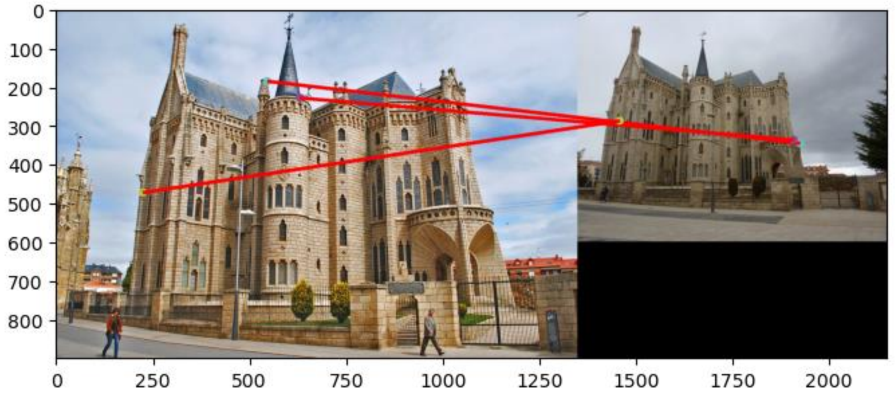
Part 4: SIFT feature descriptor

[insert visualization of matches for Mt. Rushmore image pair from proj2.ipynb here]



# matches: 98
Accuracy: 93.00%

[insert visualization of matches for Gaudi image pair from proj2.ipynb here]



# matches: 3
Accuracy: 0.00%

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

1. Compute the image gradients.
2. Get magnitudes and orientations from the gradients.
3. Isolate a 16×16 patch of the magnitude and orientation matrices.
4. Create an 8-bin histogram with given bin centers. The histogram holds orientations in a 4×4 region, weighted by the corresponding magnitudes. This will yield a 128-length feature vector.
5. Compute this feature vector for all available interest points.

[Why are SIFT features better descriptors than the normalized patches?]

As explained in Part 2, normalized patch descriptors do not take into account the image gradient magnitudes and orientations at all, whereas SIFT descriptors do. A descriptor that makes use of the gradient magnitudes and directions works well on top of a corner detector.

Part 4: SIFT feature descriptor

[Why does our SIFT implementation perform worse on the given Gaudi image pair than the Notre Dame image and Mt. Rushmore pairs?]

Our SIFT implementation perform worse on the Gaudi pair because the color contrast of the two image are too dissimilar. Also, the images are of different sizes. We have not implemented the “scale-invariant” part of SIFT. The other image pair did not have these issues, or the issues were less severe.

Part 5: SIFT Descriptor Exploration

Describe the effects of changing window size around features. Did different values have better performance?

Increasing window size means accommodating more local cells. The following table shows the metrics obtained from testing different window sizes on the Notre Dame (left) and Mount Rushmore (right) pairs. The threshold for NNDR is the same for all test sessions. Increasing the window size had varying effects in terms of number of matches and accuracy for the image pairs as seen from the table. In terms of precision, window size of 16 gives the best result.

Window Size	Cell Size	# Histogram bins	# Matches	Accuracy
8	4	8	107	55.14%
16	4	8	102	88.23%
32	4	8	40	38.00%

Window Size	Cell Size	# Histogram bins	# Matches	Accuracy
8	4	8	67	34.00%
16	4	8	98	93.00%
32	4	8	119	98.32%

Part 5: SIFT Descriptor Exploration

Describe the effects of changing the number of local cells in a window around a feature? Did different values have better performance?

Increasing number of local cells in a given patch means reducing the cell size. The following table shows the metrics obtained from testing different cell sizes on the Notre Dame (left) and Mount Rushmore (right) pairs. The threshold for NNDR is the same for all test sessions. Greater number of cells (smaller cell size) resulted in fewer matches, but the relative precision for the matches were better.

Window Size	**Cell Size**	# Histogram bins	# Matches	Accuracy
16	**2**	8	23	22.00%
16	**4**	8	102	88.23%
16	**8**	8	166	77.71%

Window Size	**Cell Size**	# Histogram bins	# Matches	Accuracy
16	**2**	8	39	39.00%
16	**4**	8	98	93.00%
16	**8**	8	108	85.19%

Part 5: SIFT Descriptor Exploration

Describe the effects of changing number of orientations (bins) per histogram. Did different values have better performance?

The following table shows the metrics obtained from testing different window sizes on the Notre Dame (left) and Mount Rushmore (right) pairs. The threshold for NNDR is the same for all test sessions. 8 bins seems to be the optimum number for both image pairs. When the number of bins is increased, the number of matches goes down drastically.

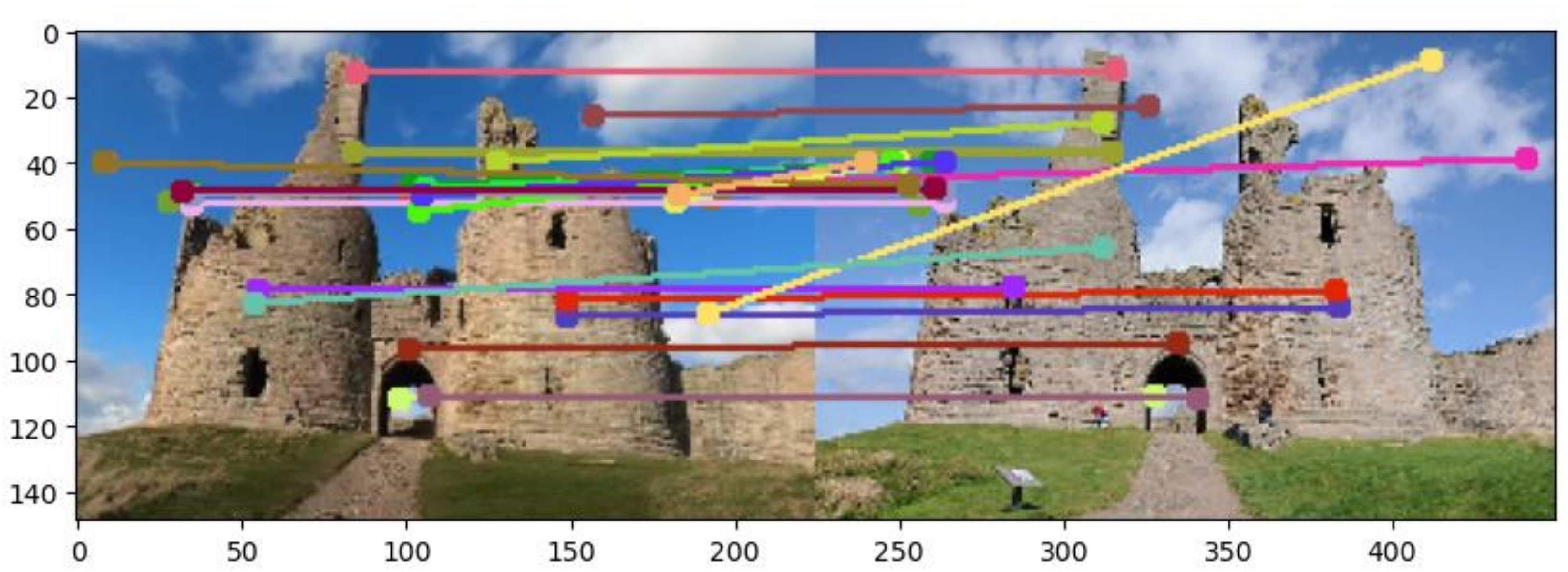
Window Size	Cell Size	**# Histogram bins**	# Matches	Accuracy
16	4	**4**	135	70.37%
16	4	**8**	102	88.23%
16	4	**16**	7	6.00%

Window Size	Cell Size	**# Histogram bins**	# Matches	Accuracy
16	4	**4**	94	75.00%
16	4	**8**	98	93.00%
16	4	**16**	4	4.00%

Part 5: SIFT Descriptor Exploration

[insert visualization of matches for your image pair from proj2.ipynb here]

Dunstanburgh Castle (Images collected online)



Part 5: SIFT Descriptor Exploration

[Discuss why you think your SIFT pipeline worked well or poorly for the given building. Are there any characteristics that make it difficult to correctly match features]?

The SIFT pipeline worked reasonably well for my selected building. It is probably because there are some clearly sharp/pointy corners and dark windows to match with. Also, the images I found are very close in color contrast and has similar scale, which helped with the matching. Some spurious matchings were due to different cloud positions in the background.

Conclusion

[Why aren't our version of SIFT features rotation- or scale-invariant? What would you have to do to make them so?]

Scale-invariance can be achieved by defining key locations as extrema of the result of difference of Gaussians applied in scale space to a series of smoothed and resampled images.

Rotation-invariance can be achieved by detecting the highest local peak (dominant orientation) in the histogram adjusting for it.

Our version is not rotation- or scale-invariant because we didn’t do any of the above.