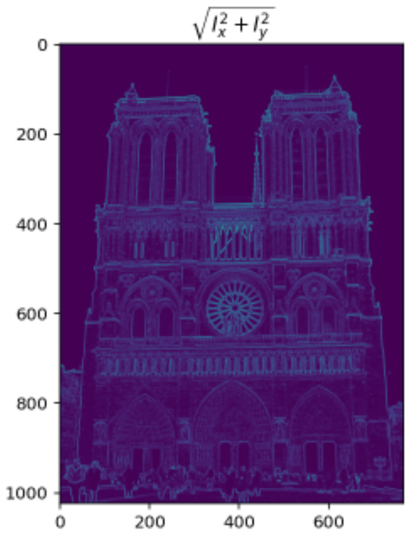
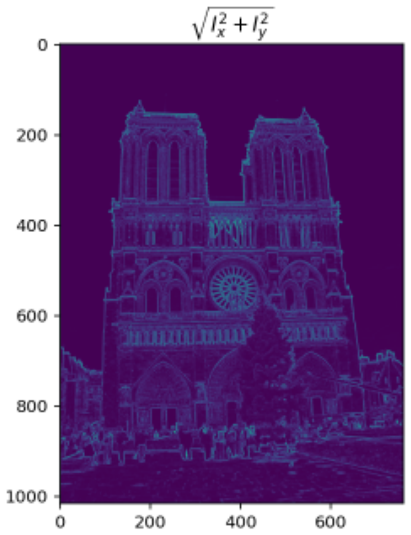
CS 6476 Project 2

Bojun Yang
byang301@gatech.edu
byang301
903254309

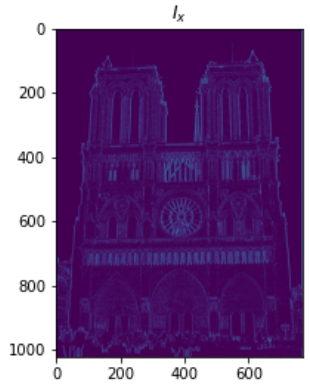
Part 1: Harris corner detector

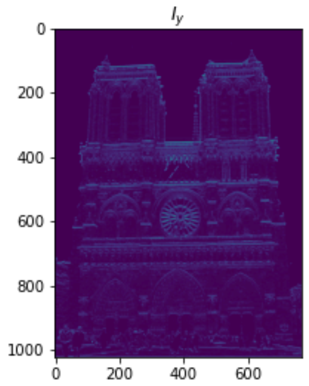


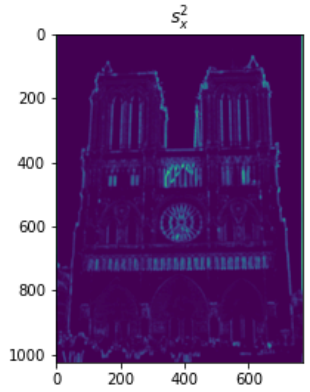


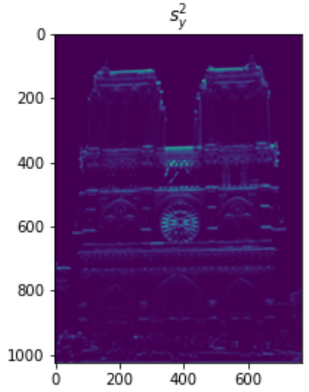
Corners will have the highest magnitude, followed by edges. Corners will have the highest magnitude due to the X and Y direction sobel filters that I applied. When the partial derivatives of the image w.r.t. X and Y direction are combined, corners will be most visible because they change both in the X and the Y direction. It is obvious that edges also have a high magnitude because we use X and Y direction sobel filters.

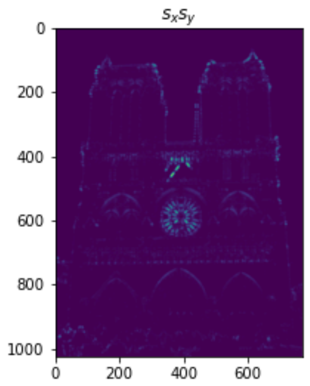
Part 1: Harris corner detector



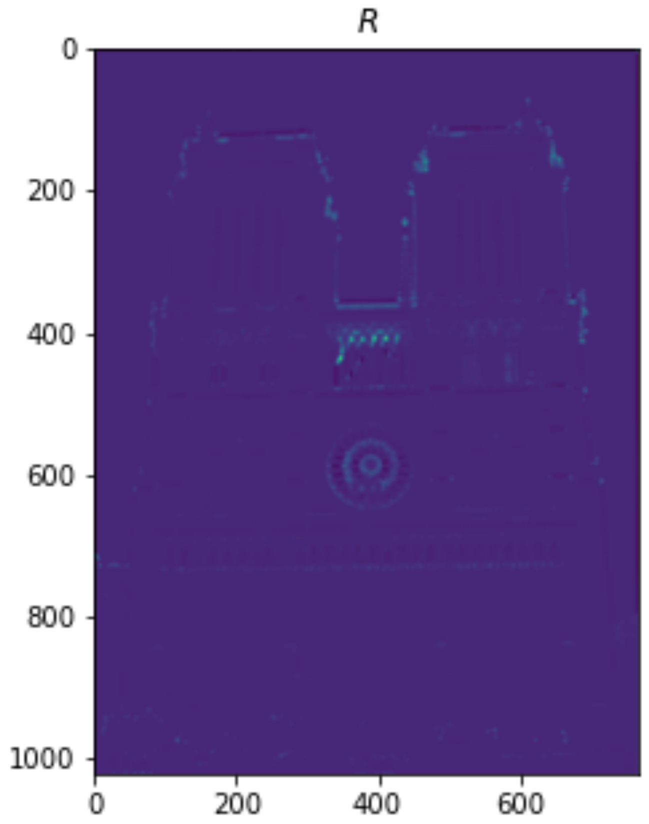






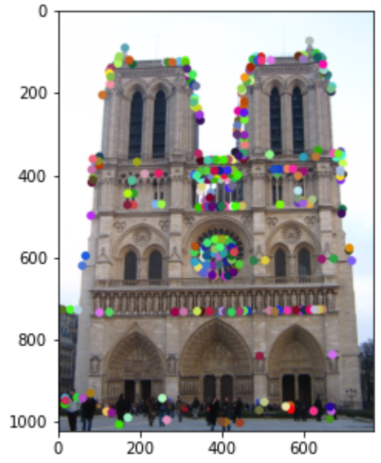


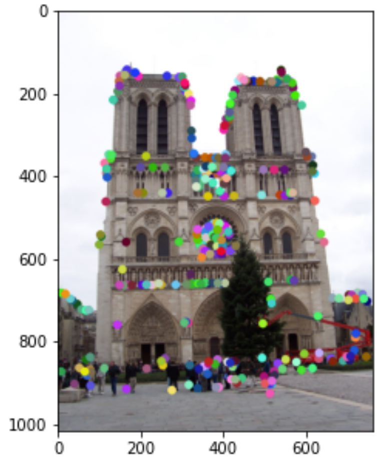
Part 1: Harris corner detector

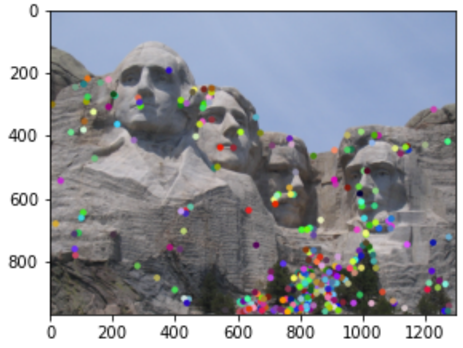


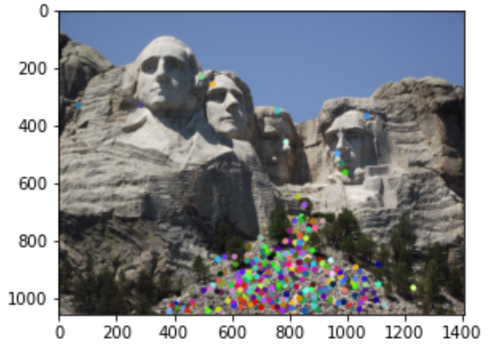
Gradient features are partially invariant to brightness/intensity change. The increase in intensity can bring peaks that were otherwise below our threshold above our threshold and be detected as a corner. The inverse could happen if intensity was lowered. Changes to contrast could also affect the gradient features. The change in contrast could also bring a patch that was not detected as a corner to be detected as a corner.

Part 1: Harris corner detector

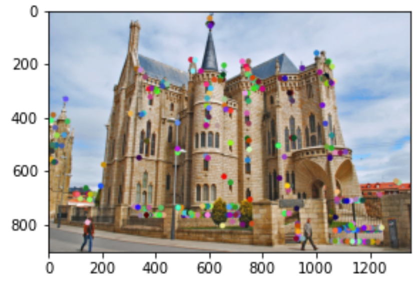


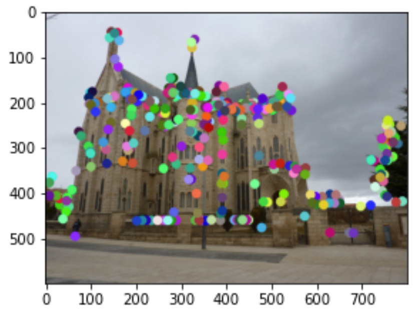






Part 1: Harris corner detector



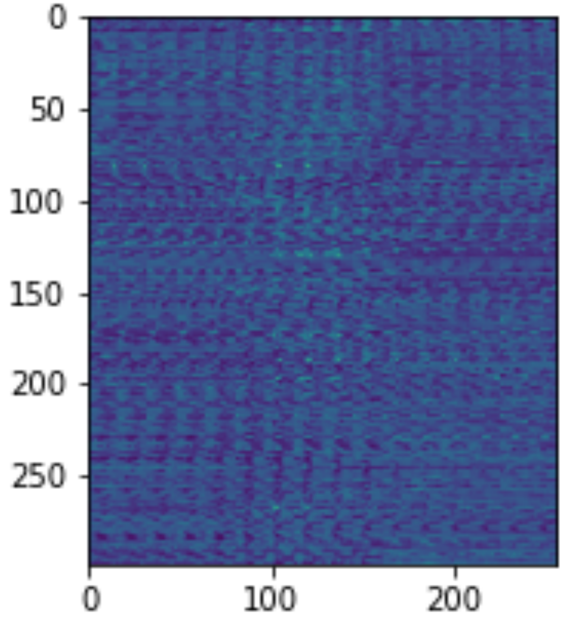


The advantages of using NMS is so we select one (x,y) coordinate out of many possible coordinates for one corner. We can use this to suppress the less likely corners. However, using nms can also suppress corners that should be included if there are more correct corners than the max number of interest points to take.

Part 1: Harris corner detector

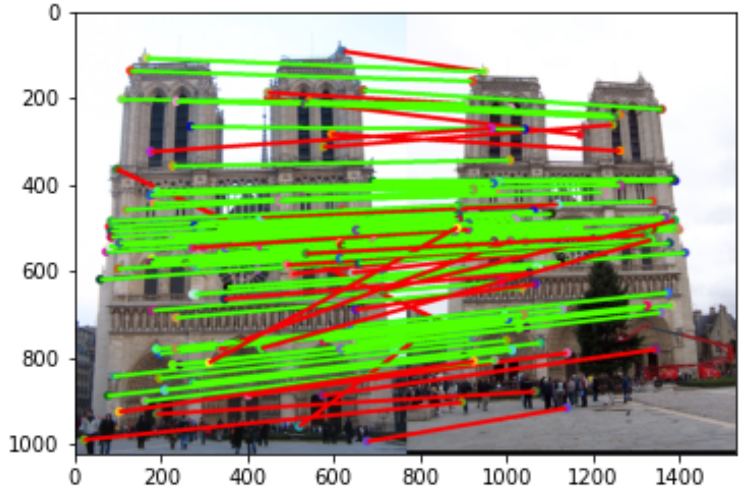
Using the image derivatives to take a Gaussian weighted sum, the Harris corner detector uses the values on the diagonals of the second moments matrices to effectively find corners. Parts where the image changes in brightness in any direction are detected by the image derivatives and this characteristic can be estimated through the determinant of the second moments.

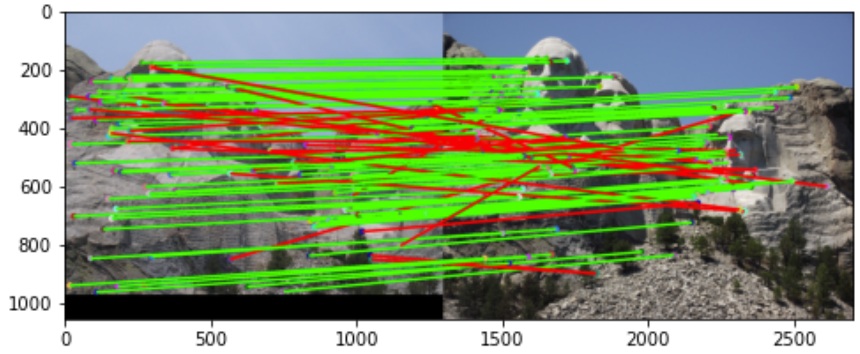
Part 2: Normalized patch feature descriptor



A normalized patch is just the original image but normalized. It doesn't contain any extra information about features that we're looking for. They are also not invariant to image scaling or rotations.

Part 3: Feature matching

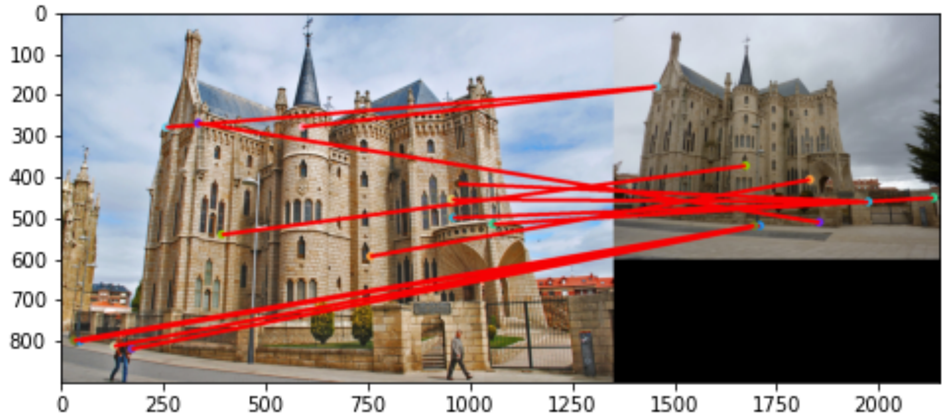




# matches (out of 100): 105
Accuracy: 0.81

# matches: 111
Accuracy: 0.73

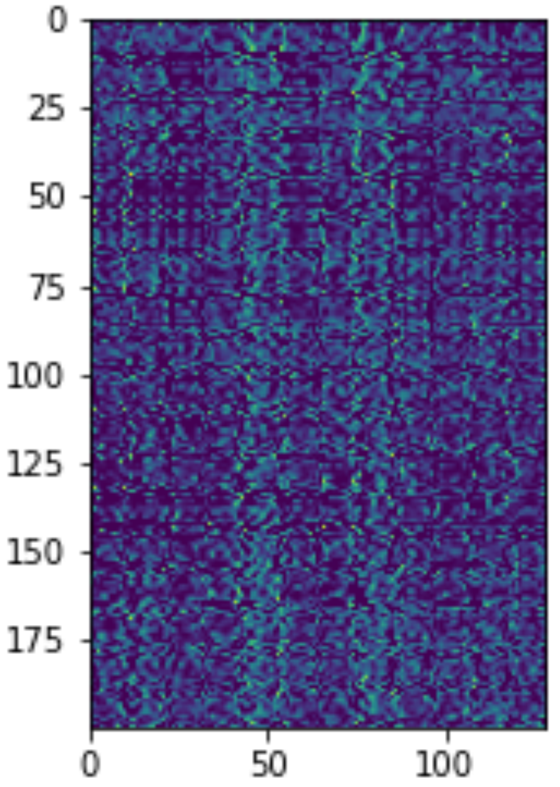
Part 3: Feature matching

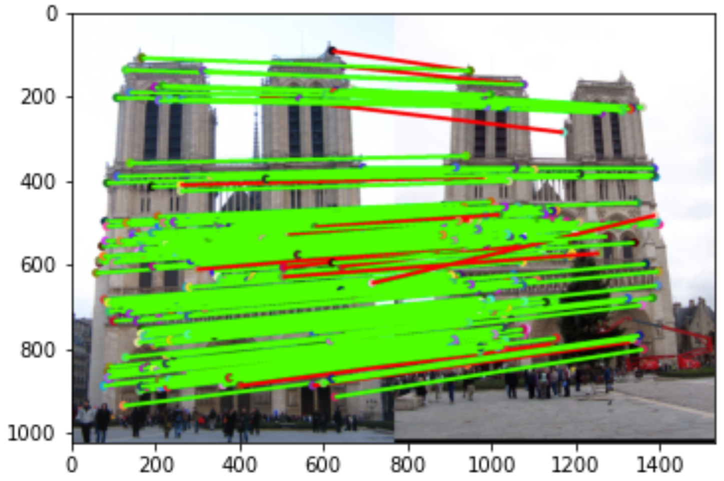


I compute the distances between the feature vectors of the two images. Then I sort them using argsort, keeping track of the sort order. I use the first sorted distance divided by the second sorted distance as my confidence and set my threshold as 0.8. The Nearest-Neighbor-Distance-Ratio is my confidence. Then I use an array mask of the confidences that is within my threshold and the previous sort order from sorting distances to obtain my matches.

# matches: 13
Accuracy: 0

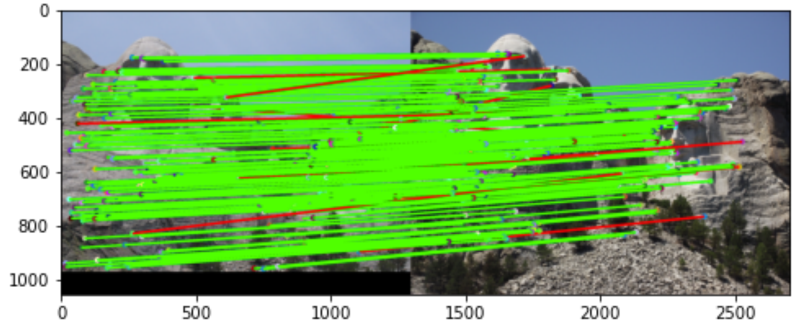
Part 4: SIFT feature descriptor

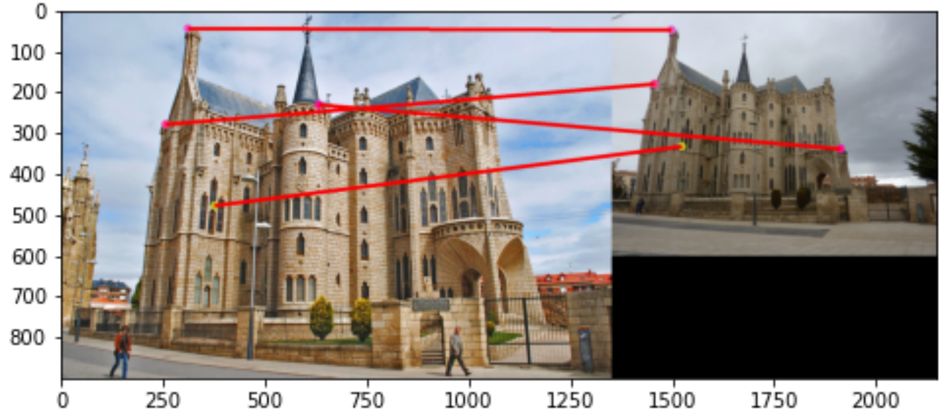




# matches (out of 100): 193
Accuracy: 0.93

Part 4: SIFT feature descriptor





# matches: 181
Accuracy: 0.93

# matches: 4
Accuracy: 0

Part 4: SIFT feature descriptor

I first compute the image gradients Ix and Iy. Using these gradients I calculate the magnitudes and orientations by splitting up each patch into cells that are (4x4). For each (4x4) cell, I return a vector that represents a gradient orientation histogram of the magnitude of orientations (8 bins). Then I L2 normalize this feature vector and use the root SIFT implementations to return my final feature vector.

SIFT features are better because of it’s use of the gradient orientation histogram. Because of this, our SIFT is resistant to changes in image illumination and color spaces. Blurring will affect the gradient magnitudes but the orientations will still be similar. A fully implemented SIFT is also scale and rotation invariant.

Conclusion

Our versions of SIFT only use 1 size patch for feature vectors. To be scale or rotation invariant, we need to use an appropriate level of the Gaussian pyramid at which a keypoint is determined. To be scale invariant, we need to have various Gaussian pyramid sizes to evaluate a patch with. We also were not required to softly distribute magnitudes to adjacent histogram feature vectors.

Vectorized SIFT

My SIFT runs in under 5 seconds with over 0.80 accuracy on Notre Dame. I vectorized all parts of my magnitude and orientation calculations by using numpy functions with numpy arrays. My implementation of getting the gradient histogram also uses the numpy histogram function to vectorize the calculations for each patch.