CS 4476/6476 Project 2

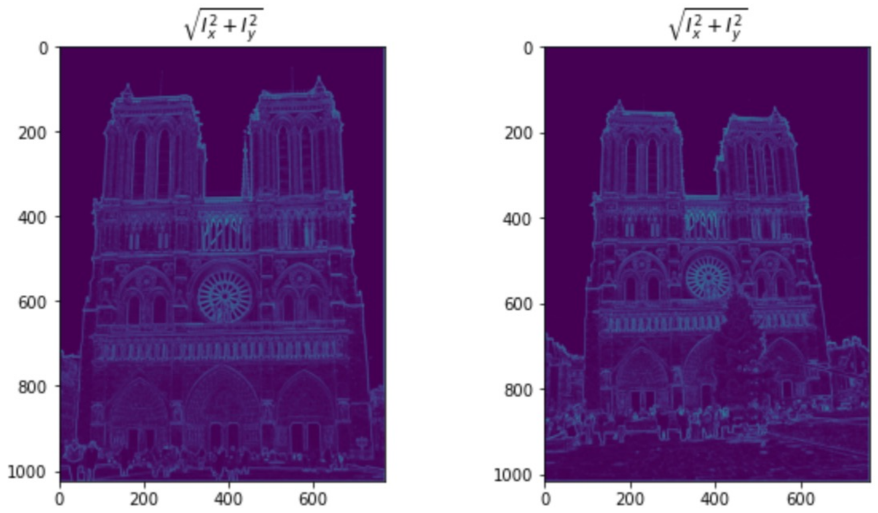
[Zixin Yin]
[zyin81@gatech.edu]
[zyin81]
[903718320]

Part 1: Harris corner detector

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

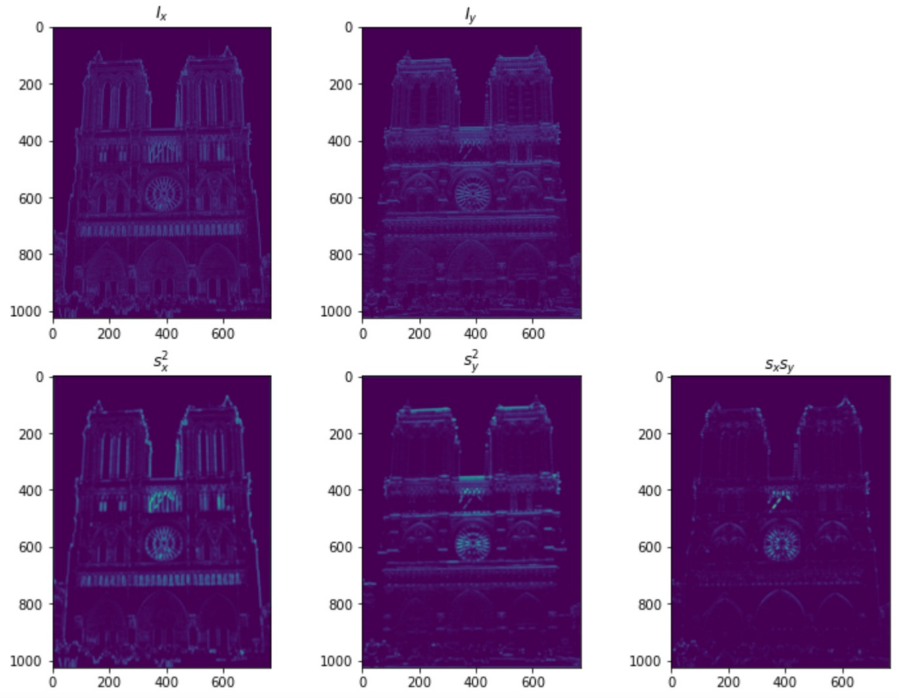
[Which areas have highest magnitude? Why?

The areas where frequency changes quickly, for example,
The circular wheel-like structure at the center. Because
at those places, frequency the partial derivatives can be
Large along both the x and y axis.



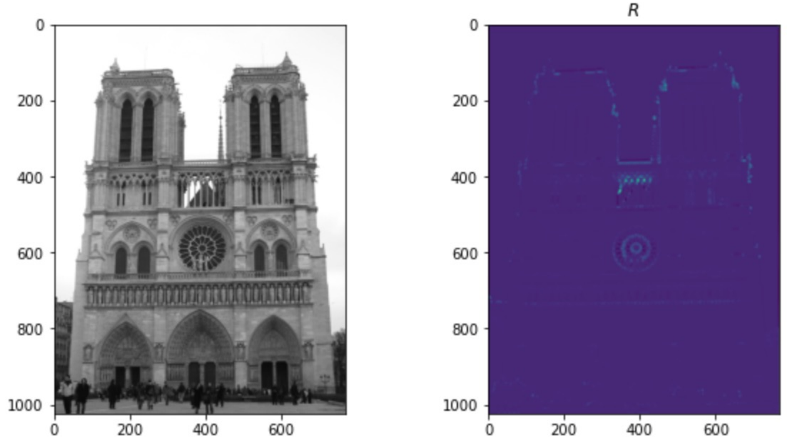
Part 1: Harris corner detector

[insert visualization of I_x, I_y, s_x^2, s_y^2, $s_x s_y$ 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]



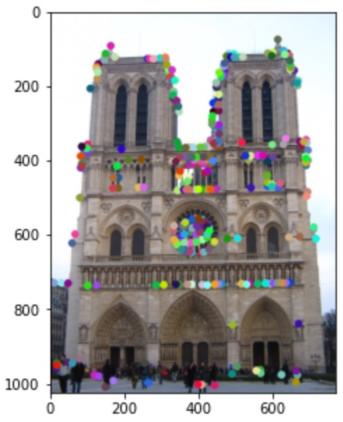
[Are gradient features invariant to both additive shifts (brightness) and multiplicative gain (contrast)? Why or why not? See Szeliski Figure 3.2]

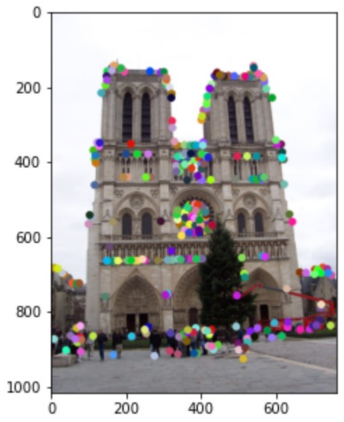
Gradient features will not change if a constant is added
To each pixel since it is calculated based on the change
Between each pixel.
Gradient features is invariant to contrast as well becaus
The multiplicative changes will be cancelled out by
Vector normalization.

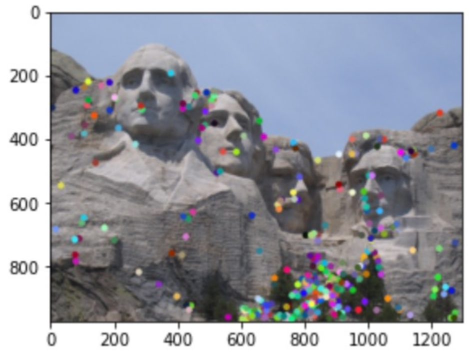
Part 1: Harris corner detector

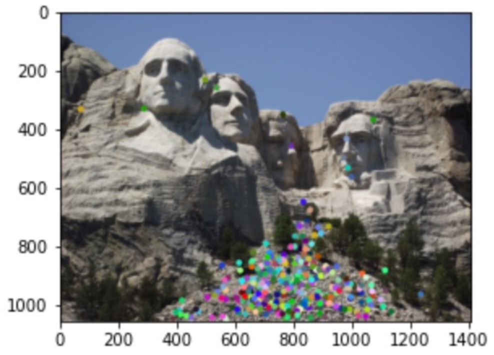
[insert visualization of Notre Dame interest points from proj2.ipynb here]

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



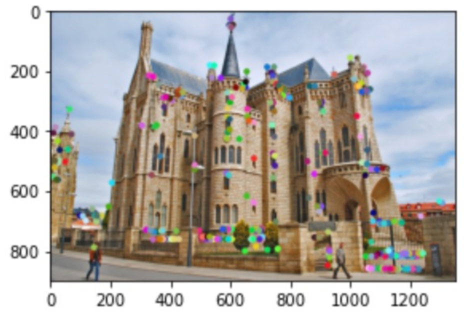


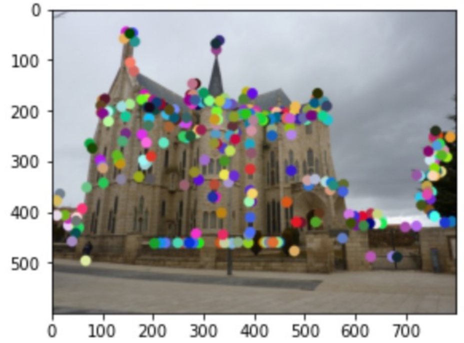




Part 1: Harris corner detector

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





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

Advantages: maxpooling finds the pixel with the highest
gradient to “represent” the original image with those
Characteristic pixel. This reduces computational cost.

Disadvantages: it ignores minimum values, so ”representation”
Of image loses some characters of the original image

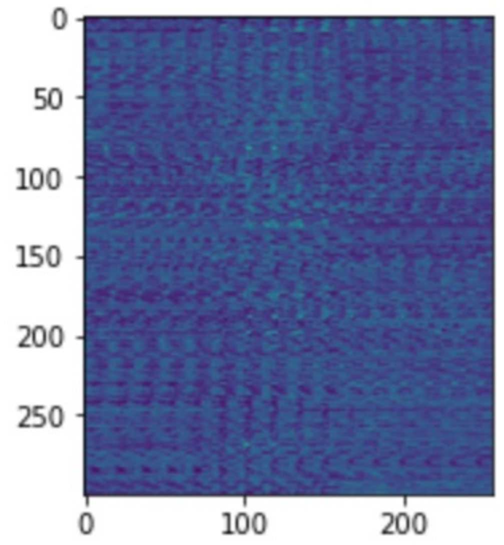
Part 1: Harris corner detector

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

The intuition is to slide a small window over the image which causes gradient changes
In different directions. It finds the areas that are the most eye-catching and ignores some
non-distinctive pixels.

Part 2: Normalized patch feature descriptor

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

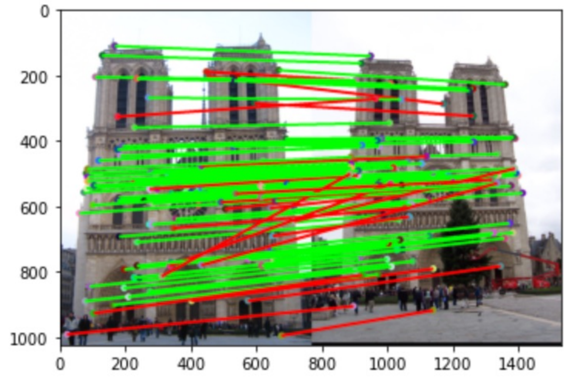


[Why aren't normalized patches a very good descriptor?]

Very sensitive to small changes in rotation, scale, view-point, and illumination; Computationally very expensive.

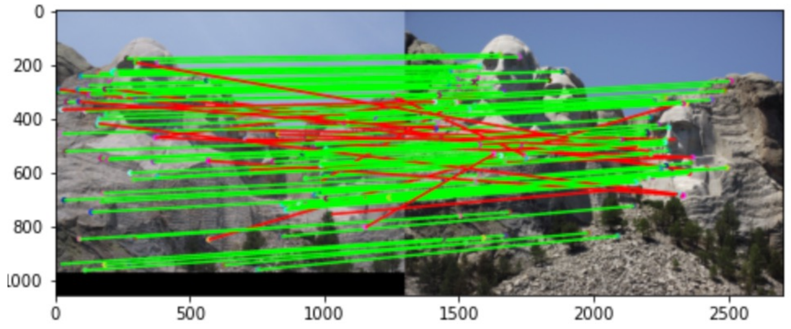
Part 3: Feature matching

[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): [95]
Accuracy: [0.74]

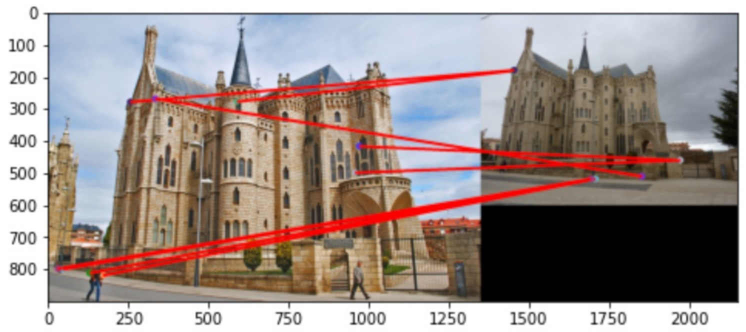
[insert visualization of matches for Mt. Rushmore image pair from proj2.ipynb here]



# matches: [101]
Accuracy: [0.792079]

Part 3: Feature matching

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



# matches: [9]

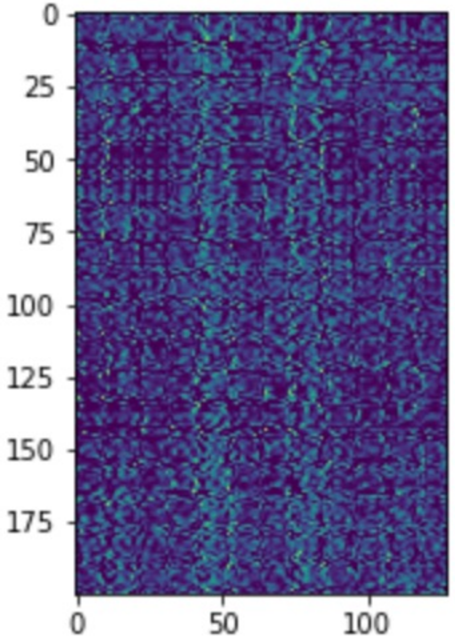
Accuracy: [0]

[Describe your implementation of feature matching here]

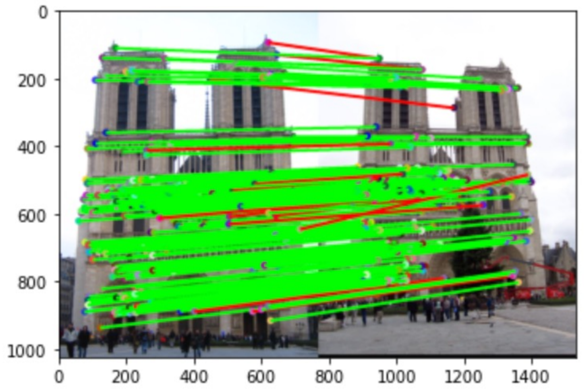
Calculate the distance between feature1 and feature2 by Normalizing the result of subtracting feature2 from feature1. Then calculate the nndr value for each row in distance matrix. If nndr is smaller than 0.79, we sort this row to find the smallest distance value in this row. Then divide the result using the maximum value to get confidences. Also keep this row number in the first column of matches, and keep the index of the smallest value in this row in the second column of matches.

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]

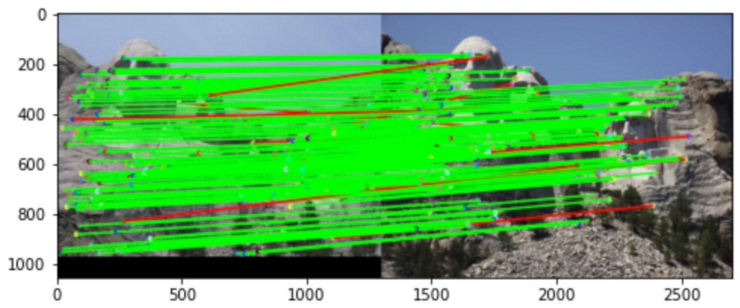


# matches (out of 100): [183]

Accuracy: [0.928962]

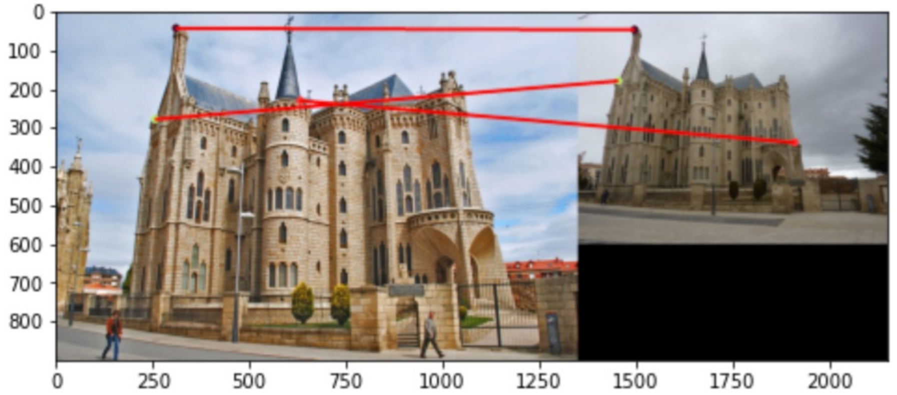
Part 4: SIFT feature descriptor

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



# matches: [169]
Accuracy: [0.934911]

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



# matches: [3]
Accuracy: [0]

Part 4: SIFT feature descriptor

[Describe your implementation of SIFT feature descriptors here]

Compute the gradient of image_bw along x and y Axises using compute_image_gradients() method. Then calculate matches and orientation using get_magnitudes_and_orientations. For each interest Point, we pass its coordinates, magnitudes, orientation, And feature_width to get_feature_vec. Then append And reshape the result matrix.

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

Since there are less data about the actual pixel Values in SIFT features than I normalized patches, SIFT descriptors are more insensitive to small changes in the original image.

Part 4: SIFT feature descriptor

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

Since Notre Dame has more distinctive points with large gradients, and the change from Feature1 to feature2 is not as much compared to Gaudi image pairs.

Part 5: SIFT Descriptor Exploration

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

Change of window size did not generate better performance. I changed window size to 32 and the Number of matches in Notre Dame image pairs is still 95.

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?

As I changed the number of local cells from 4 to 8, the number of matches stayed the same, 95/100.

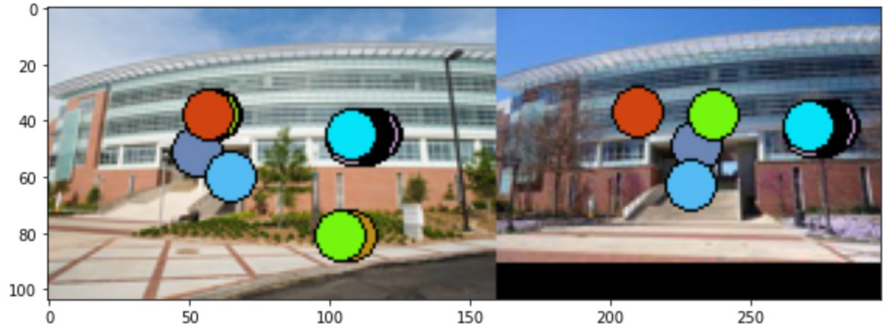
Part 5: SIFT Descriptor Exploration

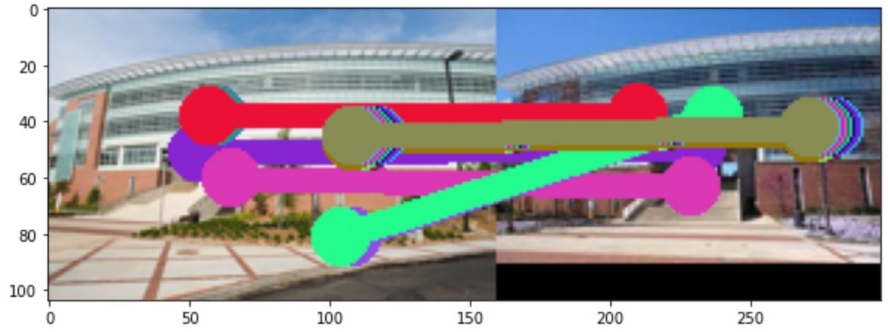
Describe the effects of changing number of orientations (bins) per histogram. Did different values have better performance?

I decreased the number of bins to 4, and I still got 95/100 matches with 0.74 accuracy. It was the same performance
As before.

Part 5: SIFT Descriptor Exploration

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





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]?

I think the performance was is not so good given that when nndr threshold is 0.9, and there are only
27 matches. I think the reason is that my images are taken from different viewpoints and scaled differently. Some
Parts of image1 are not shown in image2.

Conclusion

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

Because if there is rotation, the same angle of gradient from bin will actually represent
Different directions of the buildings. So the algorithm still considers the images using the same
Standard of angles.
I can perhaps correct the rotation first and then use current SIFT features.