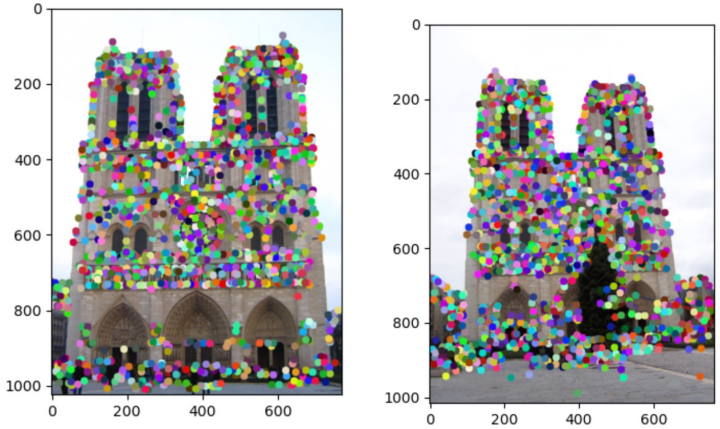
CS 4476 PS2

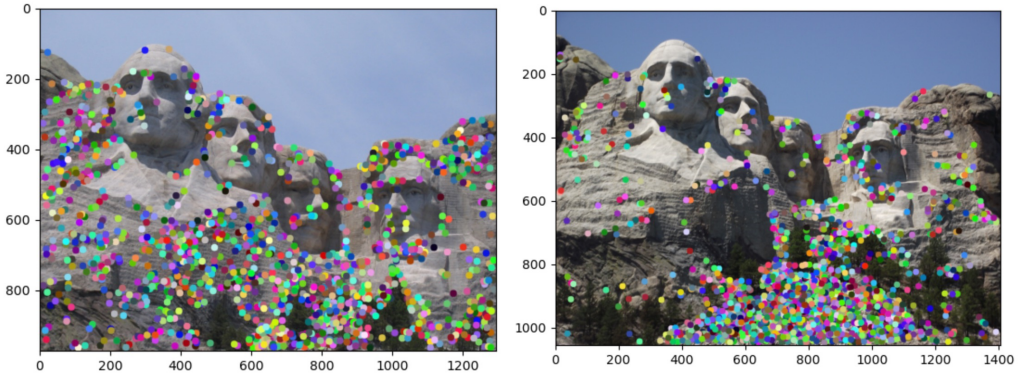
<Mohamed Ghanem>
<mghanem8@gatech.edu>
<mghanem8>
<903533880>

1.1: Harris Corner Detector

<insert visualization of Notre Dame interest points from ps2.ipynb here> [2.5 pts]

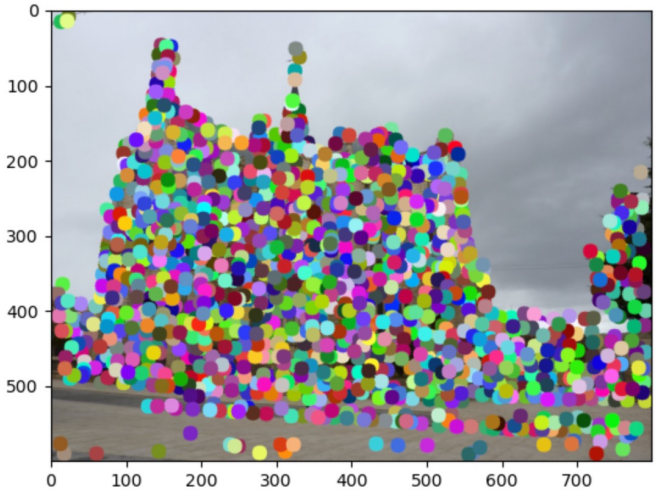


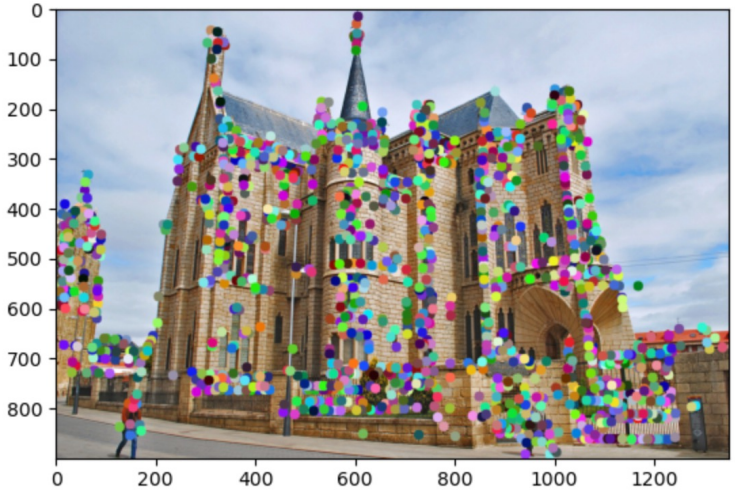
< insert visualization of Rushmore interest points from ps2.ipynb here > [2.5 pts]



1.1: Harris Corner Detector

< insert visualization of Gaudi interest points
from ps2.ipynb here > [2.5 pts]





1.1: Harris Corner Detector

- Briefly describe how the Harris corner detector works. [1 pt]

Harris corner detector begins by computing the x and y intensity gradient of an image using a Sobel filter. Then using those intensities, it computes the second moment which corresponds to the change in intensity. It then calculates the corner response using the second moment, the corner response is then used by non-maximum suppression to determine if a point is a corner.

- What does the second_moments() helper function do? [1 pt]

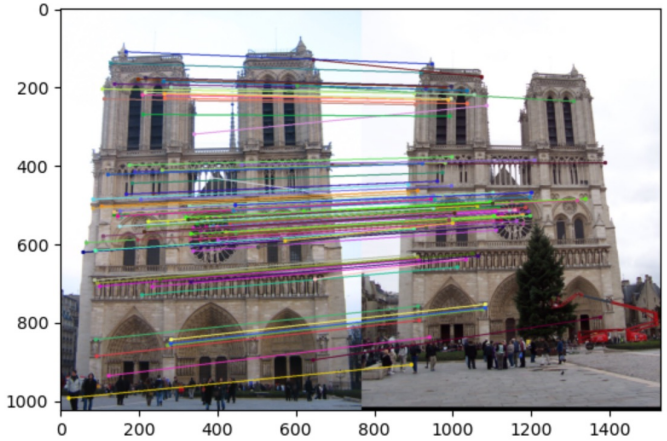
The second_moment() helper function uses the intensity gradients in the x and y direction (Ix and Iy) to compute the change in intensity or “second moment” in the x direction twice, y direction twice, and x then y direction.

- What does the corner_response() helper function do? [1 pt]

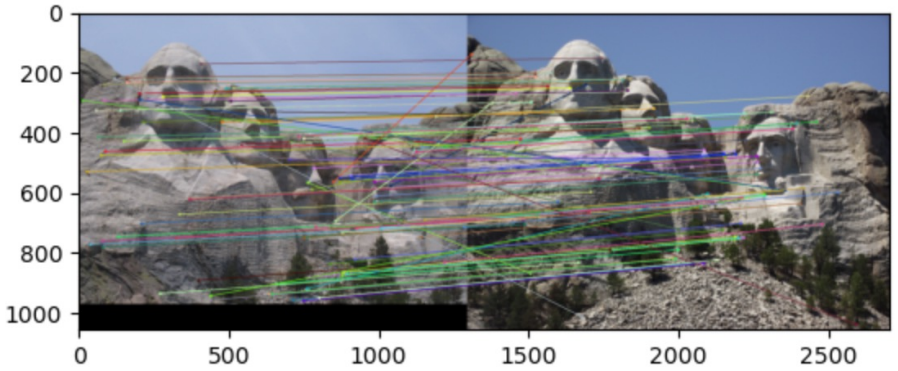
The corner_response() helper function uses the second moments matrix M (sxx, syy, and sxy) to compute the corner response function, R = det(M) - alpha(trace(M)^2) that is later used by non-max suppression to determine if a point is a corner.

1.3: Feature Matching

<insert feature matching visualization of Notre Dame from ps2.ipynb> [2.5 pts]

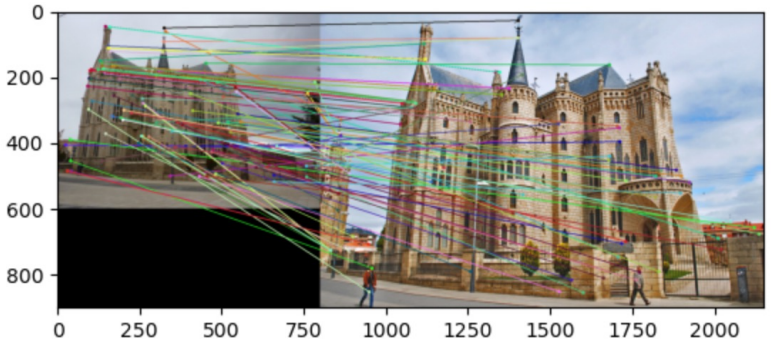


<insert feature matching visualization of Rushmore from ps2.ipynb > [2.5 pts]



1.3: Feature Matching

<insert feature matching visualization of Gaudi from ps2.ipynb > [2.5 pts]

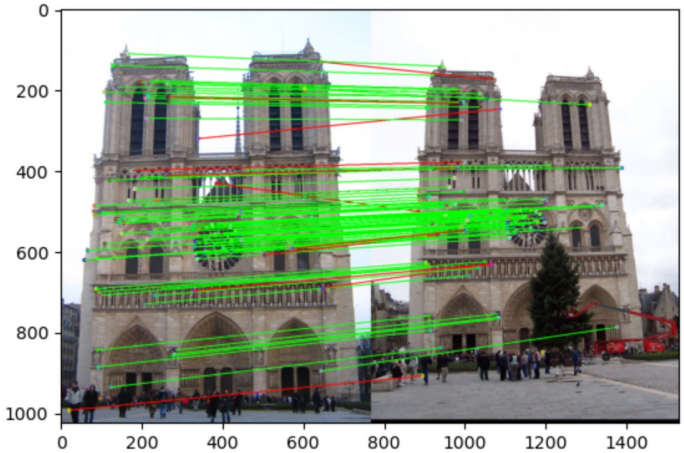


<Describe your implementation of feature matching.> [1.5 pts]

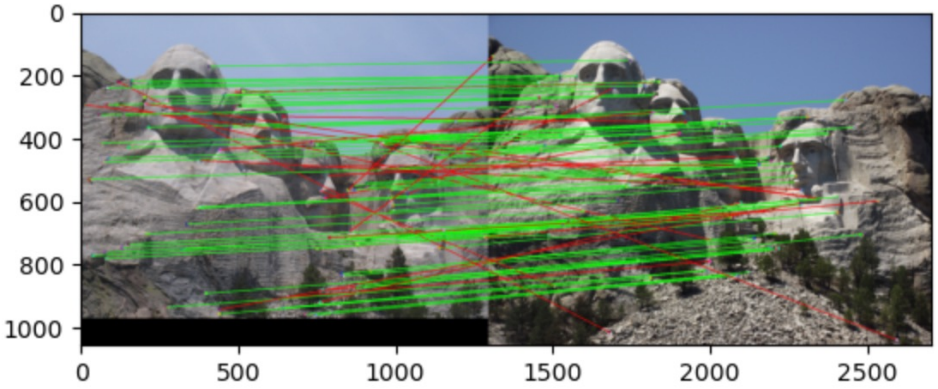
Compute the Euclidean distance between each pair of features in feature1 and feature 2. Iterate through feature1. For each feature in feature1 find the nearest neighbor feature and the second nearest neighbor feature in feature2, which correspond to the smallest and second smallest Euclidean distance features. Compute the NNDR by taking the ratio of nearest neighbor to the second nearest neighbor. Set an arbitrary threshold for evaluation (eg. 0.8). If the NNDR is less than the threshold, add the point = (current feature1 index, feature2 nearest neighbor) to the set of matches found.

Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Notre Dame from ps2.ipynb here> [1 pt]

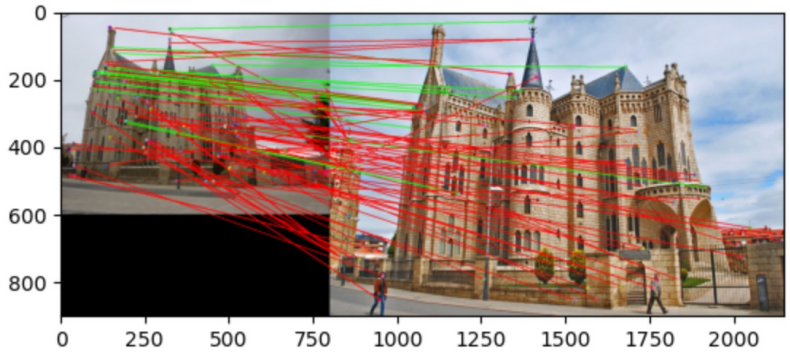


<Insert visualization of ground truth comparison with Rushmore from ps2.ipynb here> [1 pt]



Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Gaudi from ps2.ipynb here> [1 pt]



<Insert numerical performances on each image pair here. Also discuss what happens when you change the 4x4 subgrid to 2x2, 5x5, 7x7, 15x15 etc?> [2.5 pts]

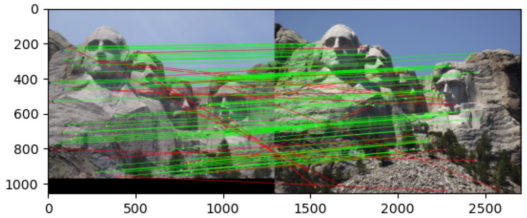
Accuracy initially increases with increase in subgrid size to 5x5 and 7x7, but at 15x15 I noticed a significant drop in accuracy (worst out of all tests). Changing the grid size means changing the number of histograms we make for a specific feature width. Therefore, an increase in subgrid size results in more features(larger feature vector).

Image	Accuracy
Notre Dame	85%
Mount Rushmore	94%
Episcopal Gaudi	19%

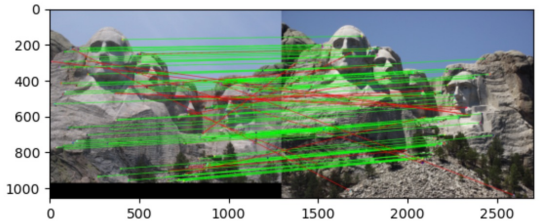
1.4(a): Hyperparameter Tuning part 1 [Extra credit]

<Insert images of the ground truth correspondence and their corresponding accuracies for varying sigma in the second moments [3, 6, 10, 30] > [0.5 pts]

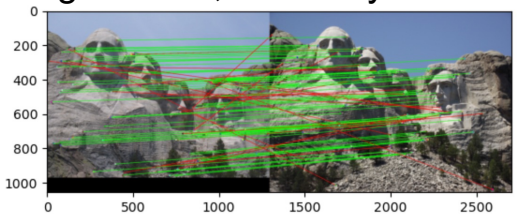
Sigma = 3, Accuracy = 77%



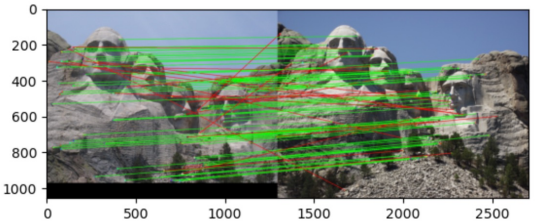
Sigma = 6, Accuracy = 81%



Sigma = 10, Accuracy = 77%



Sigma = 30, Accuracy = 78%



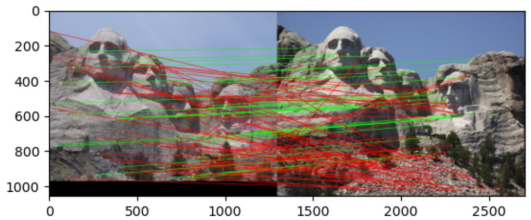
When changing the values for large sigma (>20), why are the accuracies generally the same? [0.5 pts]

Increasing sigma in the gaussian filter increases its smoothing, this causes the center pixel to become less significant. At large enough sigma the variations in second moment intensity becomes small making it less sensitive to noise causing the accuracy to remain the same.

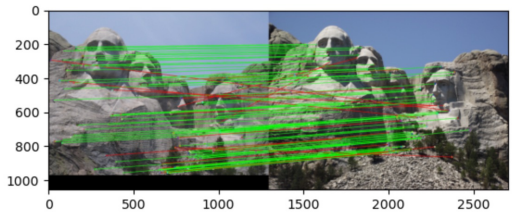
1.4(a): Hyperparameter Tuning part 2 [Extra credit]

<Insert images of the ground truth correspondence and their corresponding accuracies for varying feature width in the SIFT [8, 16, 24, 32] > [0.5 pts]

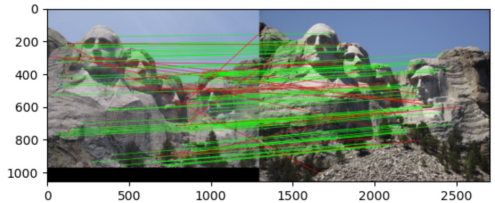
Feature Width = 8, Accuracy = 34%



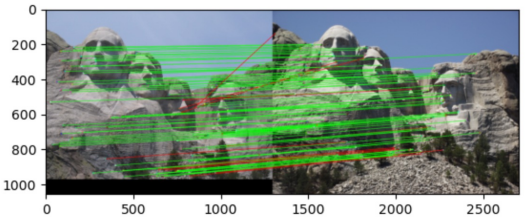
Feature Width = 24, Accuracy = 84%



Feature Width = 16, Accuracy = 78%



Feature Width = 32, Accuracy = 89%



What is the significance of changing the feature width in SIFT? [0.5 pts]

The feature width determines the size of the features we're searching for. A Larger feature width will be able to find larger scale features. In Mount Rushmore, the more significant feature are larger scale, so the accuracy increases with increase in feature width.

1.4(c): Accelerated Matching [Extra credit]

<Insert Runtime/Accuracy of your faster matching implementation. What did you try and why is it faster?> [1 pts]

Used KD Tree data structure to increase the speed of looking for nearest neighbor and second nearest neighbor distances. A KD Tree is a space-partitioning data structure for organizing points in a K-Dimensional space, so it can determine the nearest neighbor and second nearest neighbor distance a lot faster than linear search in normal arrays O(logn) vs O(n).

```
107 matches from 1341 corners
Time Elapsed (Accelerated): 0.23965907096862793
Time Elapsed (PCA): 4.012429237365723
Time Elapsed (no optimization): 4.065882205963135
```

```
You found 100/100 required matches
Accuracy = 0.940000
```