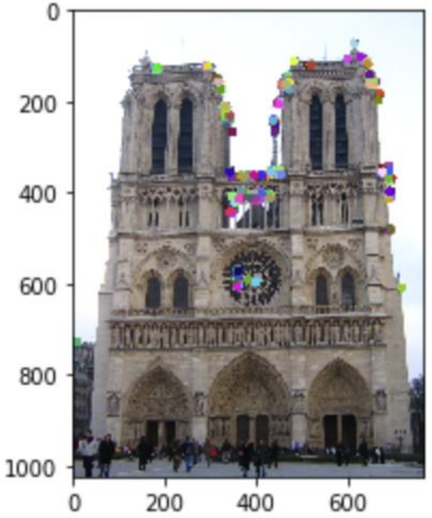
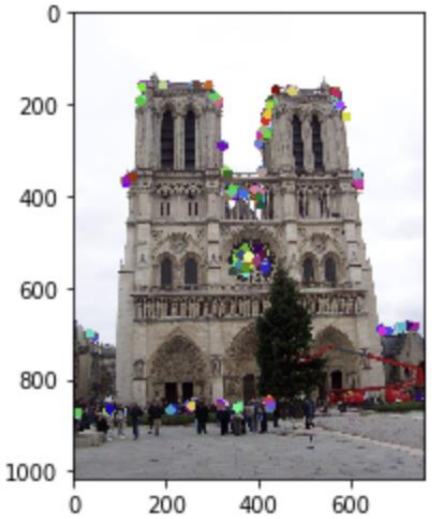
# CS 4476 Project 3

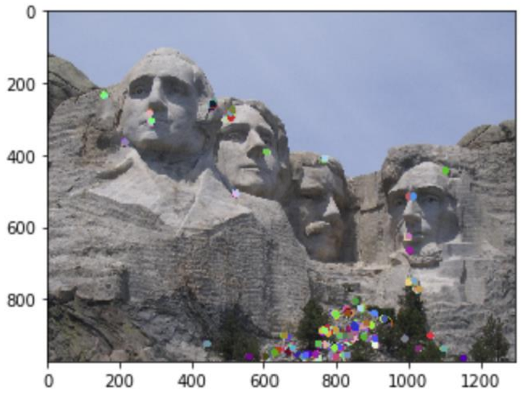
Wenyue Wang
Wenyue_wang@gatech.edu
wwang413
903204153

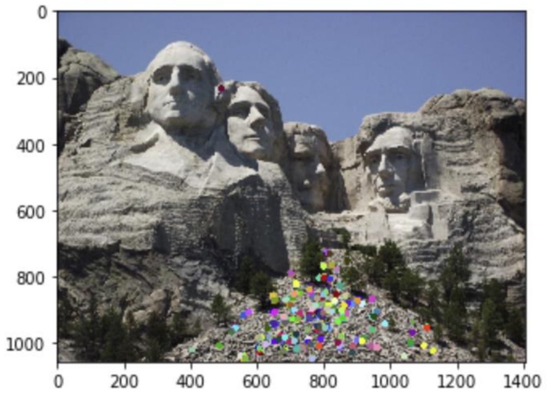
## Part 1: Harris Corner Detector

< insert v
points fro



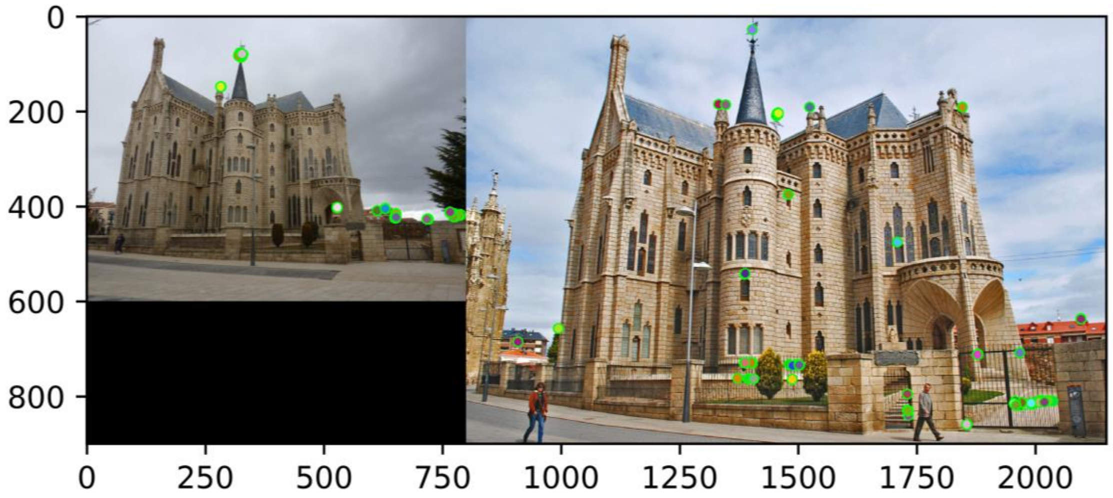






## Part 1: Harris Corner Detector

< insert visualization of Gaudi interest points
from proj3.ipynb here >



## Part 1: Harris Corner Detector

<Describe how your implementation mirrors the original harris corner detector process. (First describe Harris) What does each helper function do? How are the operations we perform equivalent and different?)>

Original Harris method has three steps, first is using cornerness as a representation of feature, so calculating the cornerness of each            each pixel in that image by getting gradients to form M matrix; secondly calculate f based on eigenvalues, and find points whose surrounding window would give higher corner response. And the last step is performing the non_maximum suppression and get the detected matrix.

Here, we have these helper functions: get_gaussian_kernel is to get 2D gaussian filter, my_filter2D is to perform the convolution, get_gradients is to get horizontal and vertical gradients of each pixel, remove_border_vals is to make all borders be zero, second_moments is to get the M matrix, and corner_response is to get response score of each pixel, and non_max_suppression is utilized to make great corner response be more obvious.

In my implementation, the major difference is that I did not set threshold and did not go through the eigenvalue analysis. Instead, I used the quantity proposed by Harris and Stephens as calculating $\det(A) - \alpha * trace(A)^2$. Then based on this result, I calculate the corner response. In addition, remove_border_vals is also implemented to make sure the moving window always takes reasonable input values.

## Part 2: Sift

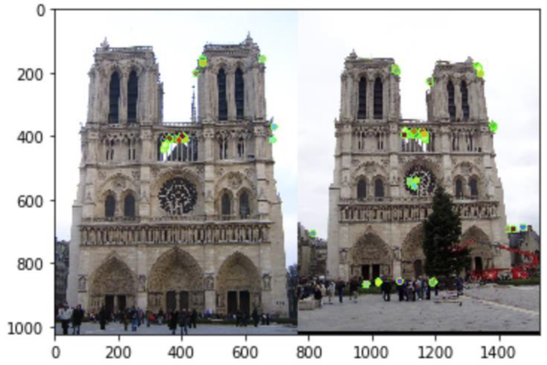
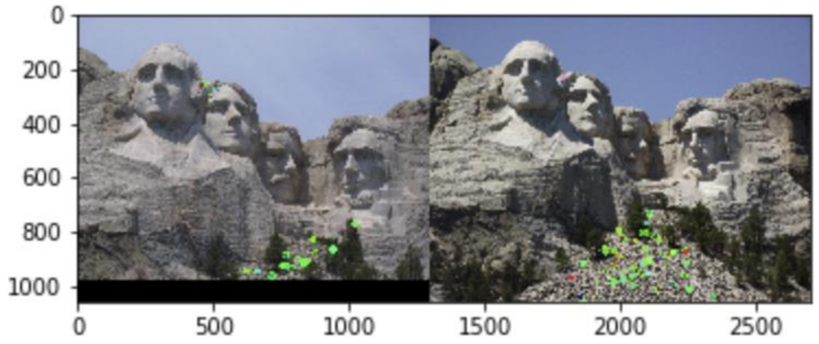
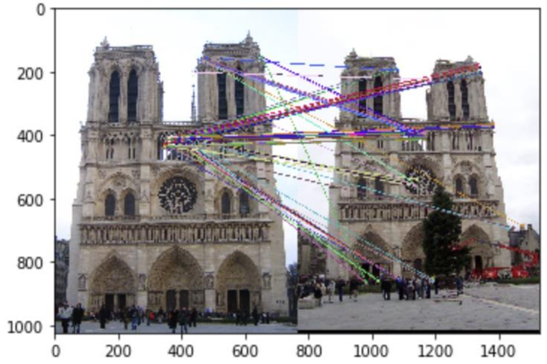
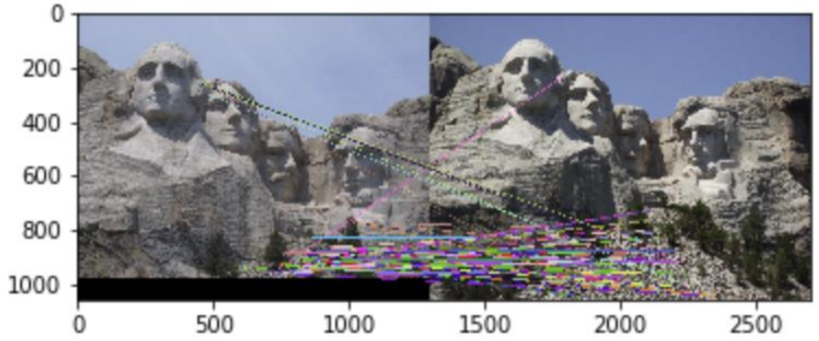
<Describe how your implementation mirrors the Sift Process. (First describe Sift) What does each helper function do? How are the operations we perform equivalent and different?)>

The essence of the Sfit algorithm is to find key points (feature points) on different scale spaces, calculate the size, direction, and scale information of key points, and use these information to form key points to describe feature points.

Here we have these helper functions: get_magnitude_and_orientation is used to get magnitude and orientation based on gradients, get_feat_vec is the main helper function, where for each 4*4 ceil, we put these orientations into a histogram with 8 bins, and each bin represents an angle. After that, we appeaned these histrograms to form the final feature vector.

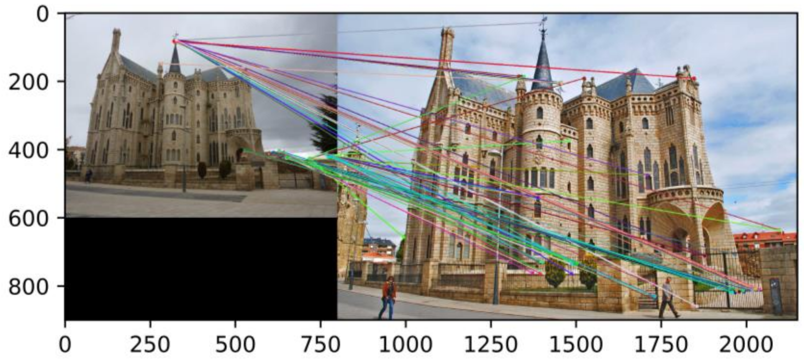
All these subfunctions are similar as we used in this program.

## Part 3: Feature Matching

## Part 3: Feature Matching

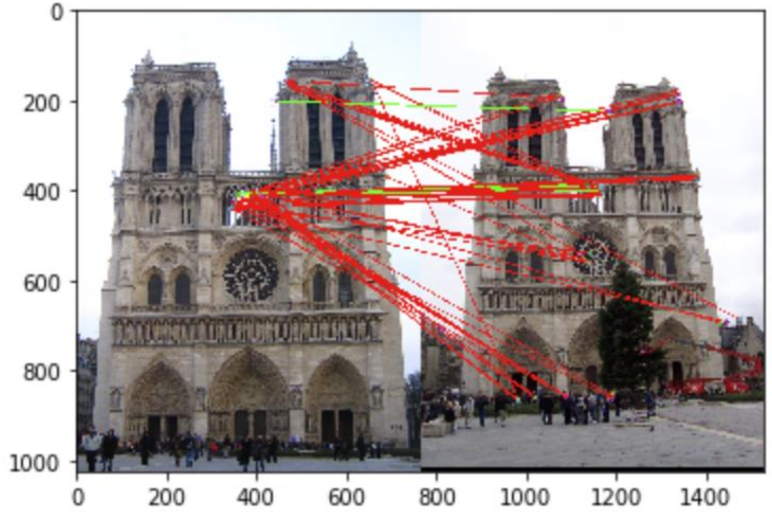
<insert feature matching visualization of Gaudi from proj3.ipynb >



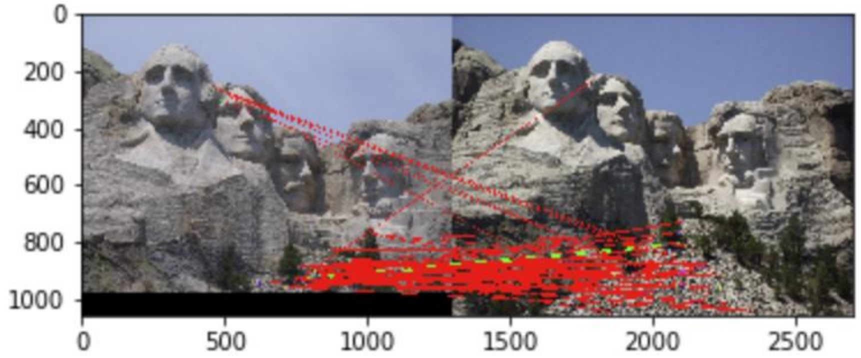
<Describe your implementation of feature matching.>
Firstly, I implement compute_feature_distance, so that calculate distance from one feature to every other features. Then we sort the distance to get the nearest neighbor and the second nearest neighbor, and confidences is calculated as nearest distance/second nearest distance. Then, we judge if the confidences is smaller than a threshold. Then get all features within that threshold be our matched features.

## Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Notre Dame from proj3.ipynb here>

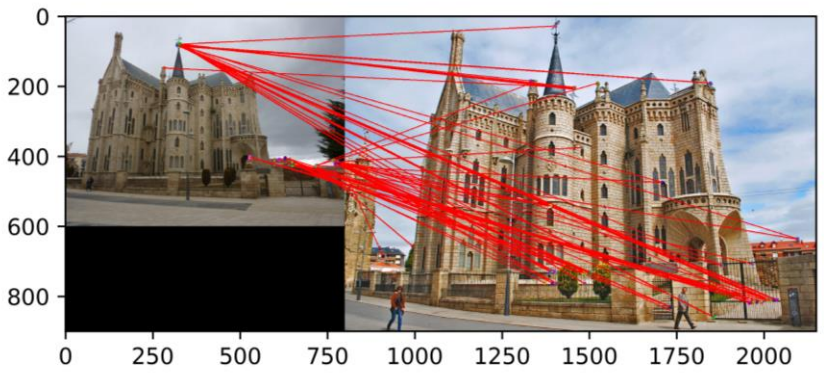


<Insert visualization of ground truth comparison with Rushmore from proj3.ipynb here>



# Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Gaudi from proj3.ipynb here>



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

## Extra Credit: Hyperparameter Tuning part 1

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

**When changing the values for large sigma (>20), why are the accuracies generally the same?**

## Extra Credit: Hyperparameter Tuning part 2

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

**What is the significance of changing the feature width in SIFT?**

## Extra Credit: Accelerated Matching

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