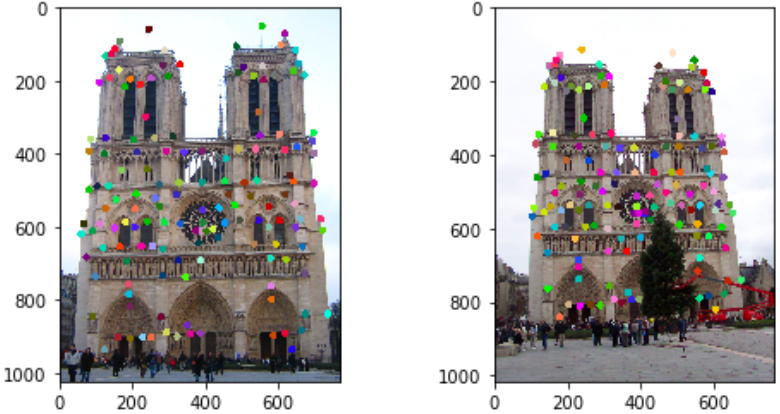
CS 4476 PS3

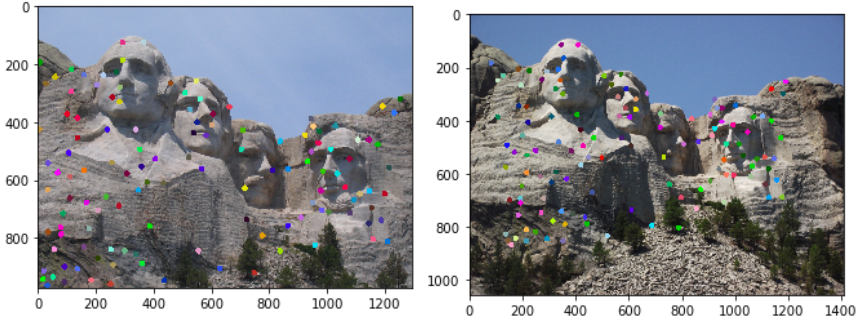
Advaith Sekharan
asekharan7@gatech.edu
asekharan7
903282224

1.1: Harris Corner Detector

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

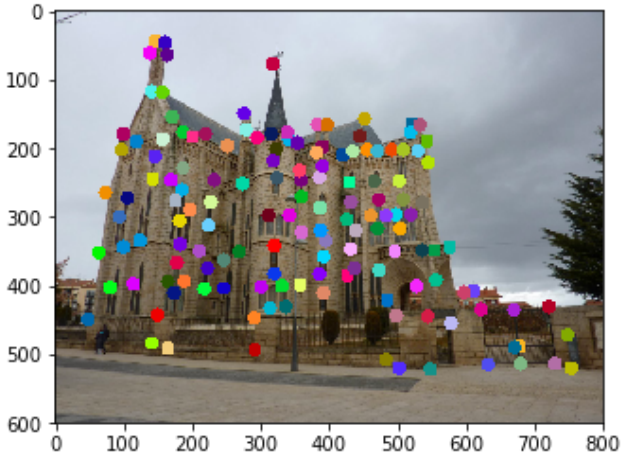


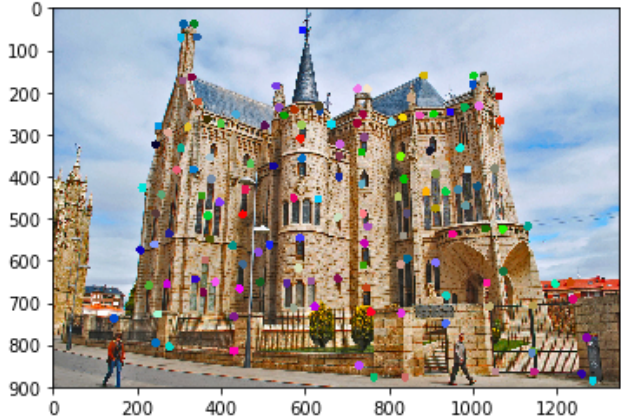
< insert visualization of Rushmore interest points from proj3.ipynb here >



1.1: Harris Corner Detector

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





1.1: Harris Corner Detector

- Briefly describe how the Harris corner detector works.

  The Harris corner detector uses gaussian filters and gradients to create a corner response score for each pixel and then applying non maximal suppression to narrow down on fewer interest points and then sorting them based on confidence and returning the top n points.

- What does the second_moments() helper function do?

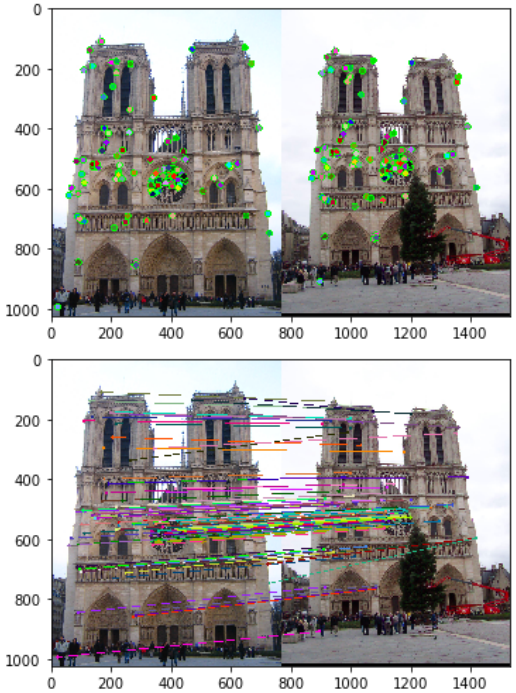
  - The second moments function computes the second moments (ix^2, iy^2, ixiy) given the first moments (gradients)

- What does the corner_response() helper function do?

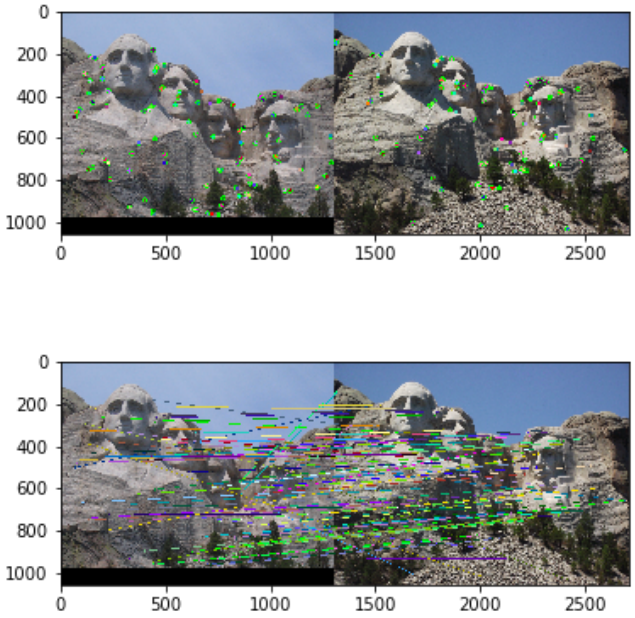
  - The corner response function computes the corner response score as per the formula we learned in class for each pixel

1.3: Feature Matching

<insert feature matching visualization of Notre Dame from proj3.ipynb>

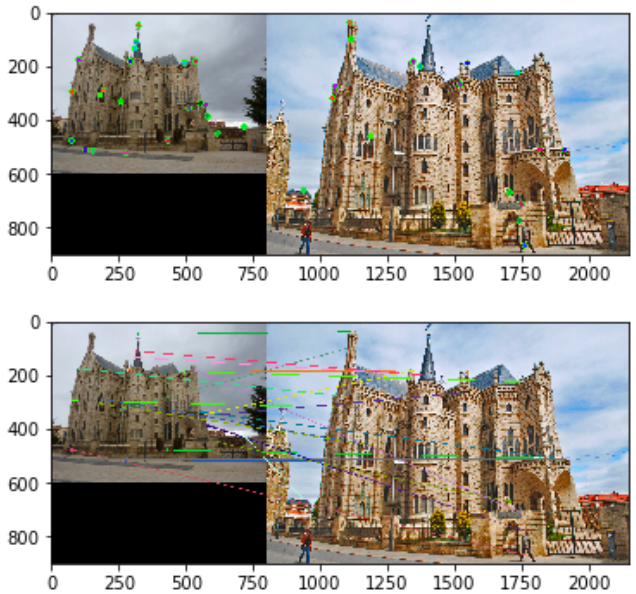


<insert feature matching visualization of Rushmore from proj3.ipynb >



1.3: Feature Matching

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

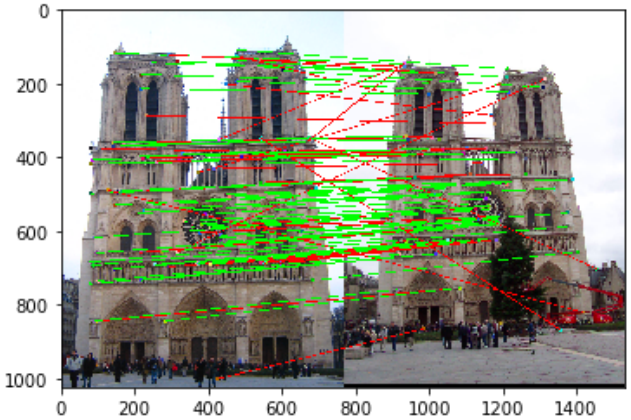


<Describe your implementation of feature matching.>

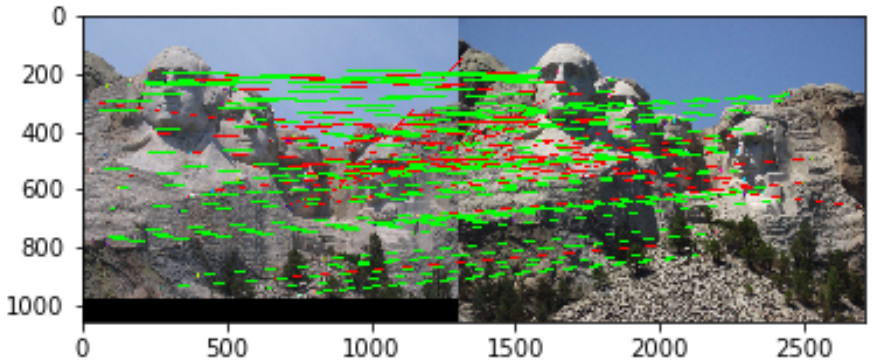
I first compute the distances between all pairs of features then iterate over the features in the first feature array, sorting the distances between that feature and all other features in the second feature array, and then appending the one with the smallest distance, applying ratio test as an additional filter, before finally adding it to the matches and confidences I return. I also normalize confidences and sort based on confidence before returning them.

Results: Ground Truth Comparison

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

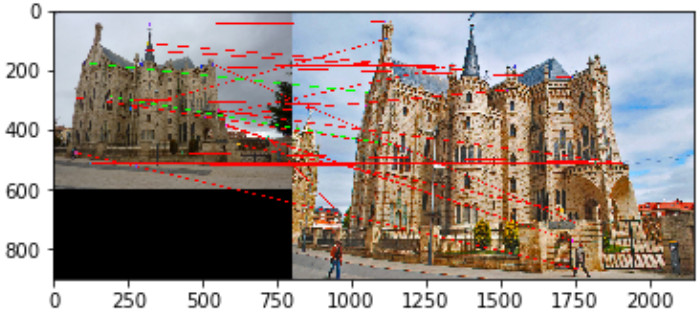


<Insert visualization of ground truth comparison with Rushmore from proj3.ipynb here>



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?>

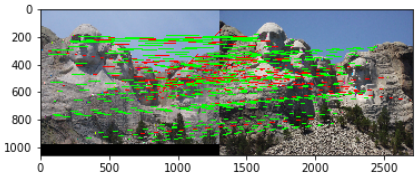
Results are in a range because I tweaked alpha values to maximize accuracy.
Notre Dame: 81-89%
Rushmore: 69-71%
Gaudi: 2-4%

When you change the 4x4 sub grid to a smaller subgrid, you get more feature vectors that are less descriptive (each fv has a smaller length) and you get less feature vectors that are more descriptive when you increase the gridsize.

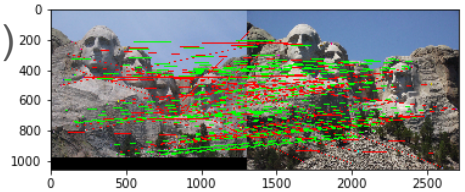
1.4(a): Hyperparameter Tuning part 1 [Extra credit]

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

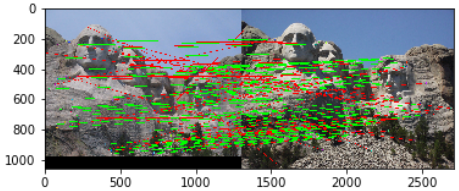
3: accuracy (62%)



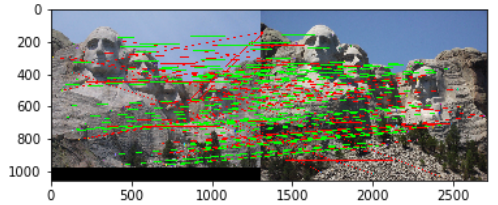
10: accuracy (53%)



6: accuracy (54%).



30: accuracy (57%)

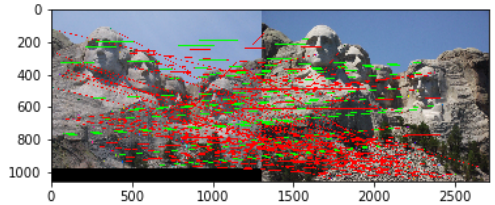


When changing the values for large sigma (>20), why are the accuracies generally the same?

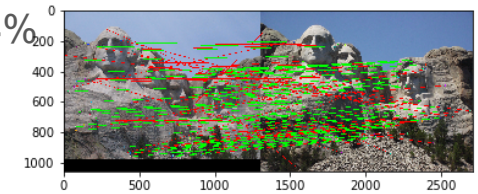
1.4(a): Hyperparameter Tuning part 2 [Extra credit]

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

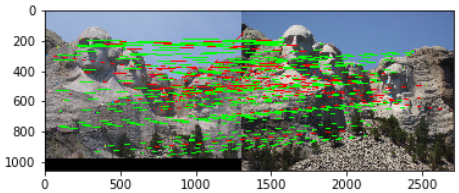
8: accuracy 24%



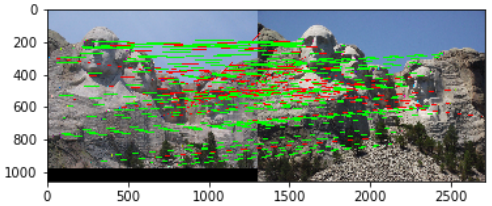
16. Accuracy 54%



24: accuracy 62%



32. Accuracy 69%



What is the significance of changing the feature width in SIFT?
Increasing feature width increases the size of the window around each interest point we generate a feature vector for. Thus, the feature vectors will be more descriptive and fewer (since more will be too close to the boundary).

1.4(c): Accelerated Matching [Extra credit]

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