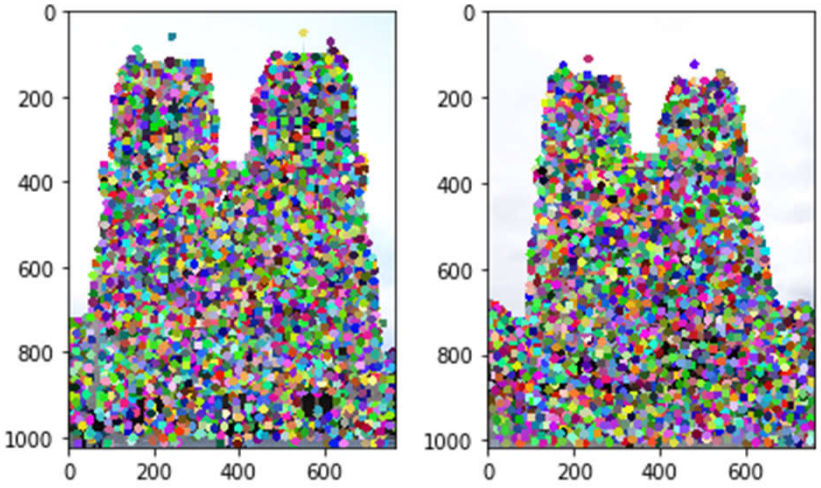
CS 4476 Project 2

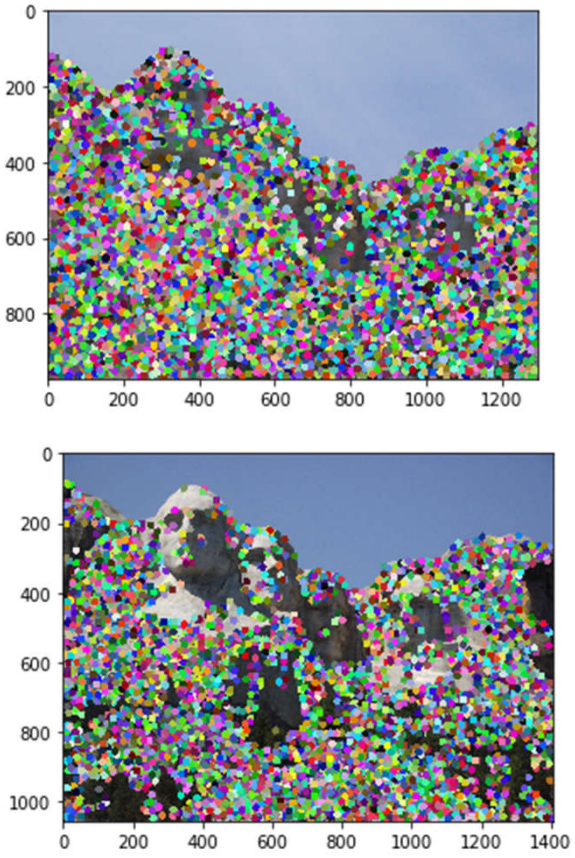
Jian Yu, Kok
jkok7@gatech.edu
jkok7
903550380

Part 1: HarrisNet

<insert visualization of Notre Dame interest points from proj2.ipynb here>

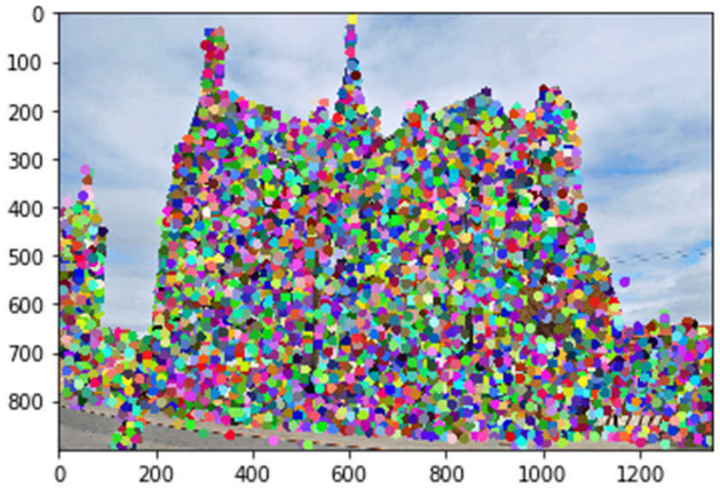


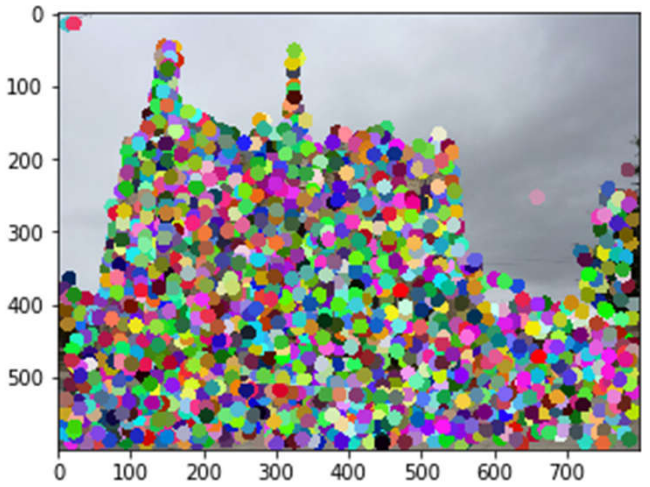
< insert visualization of Rushmore interest points from proj2.ipynb here >



Part 1: HarrisNet

< insert visualization of Gaudi interest points
from proj2.ipynb here >





Part 1: HarrisNet

<Describe how the HarrisNet you implemented mirrors the original harris corner detector process. (First describe Harris) What does each layer do? How are the operations we perform equivalent?)>

The original Harris Corner Detector process consist of 3 main steps.

1. Compute cornerness score of each image window.
2. Find points with a larger corner response (f>threshold)
3. Perform non-maximum suppression.

Harris net has 5 Layers. Image Gradients, Channel Product, Second Moment Matrix, Corner Response, and NMS layer. Image Gradients convert the pixels into gradients, Channel Product computes Ixx, Iyy and Ixy, which will be used by Second Moment Matrix layer to calculate the Second Moment Matrix, which will then be used to estimate the change in appearance in a neighbourhood.
The NMS layer will then set all points with value < threshold(median) to 0, and perform non-maximum suppression.
From this, it can be said that Image Gradients, Channel Product, Second Moment Matrix and Corner Response layers corresponds to Step 1 as they are used to calculate the cornerness score, and the NMS Layer corresponds to step 2 and 3.

Part 2: SiftNet

<Describe how the SiftNet you implemented mirrors the Sift Process. (First describe Sift) What does each layer do? How are the operations we perform equivalent?)>

Sift process starts by computing gradients around detected keypoint. This is done in the Image Gradient layer, which calculates gradient at every pixel.

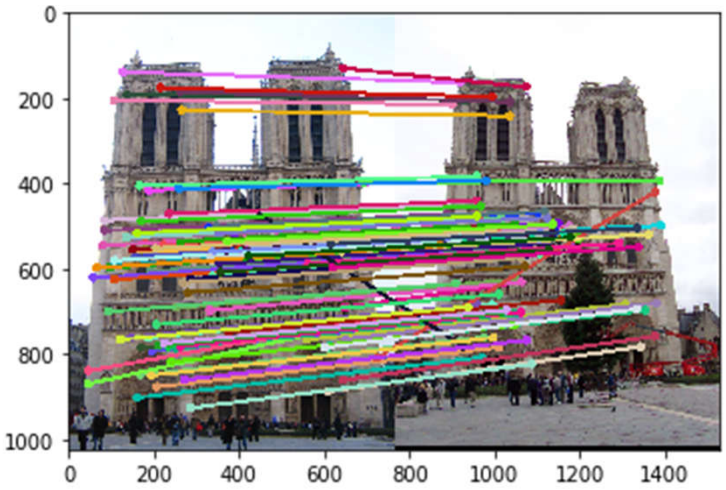
We then form a histogram by adding the gradients into one of eight histogram bin. This is done with SiftOrientation Layer and Histogram Layer. Where SiftOrientation calculates the gradient's magnitude in each of the 8 bins. Histogram Layer then creates a weighted histogram at every pixel with values obtained from SiftOrientation.

In Sift, the histogram in formed in 4x4 subgrids. This is replicated in this siftnet by the subgrid accumulation layer, which basically sums up the vectors in a 4x4 subgrid around the pixel.

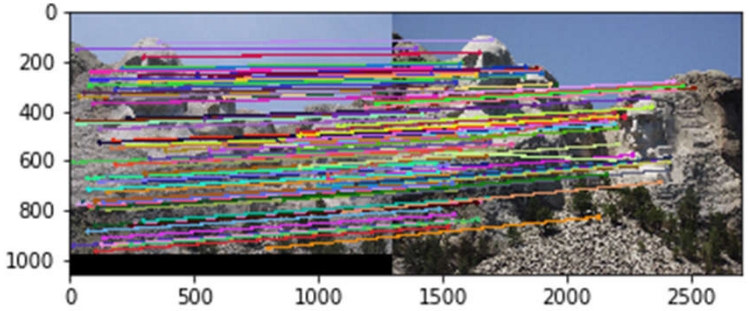
To retrieve feature with Siftnet, we get 16 coordinates around the provided coordinate pair (x, y). Each coordinate representing a 4x4 subgrid. We get the values of the weighted histogram for the 4x4 subgrid, and concatenate with all 16, 4x4 subgrid to get a 128 dimension vector. This is then normalized, and raise to the power of 0.9

Part 3: Feature Matching

<insert feature matching visualization of Notre Dame from proj2.ipynb>

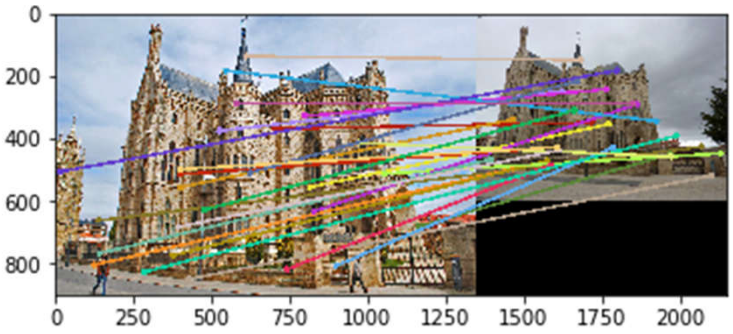


<insert feature matching visualization of Rushmore from proj2.ipynb >



Part 3: Feature Matching

<insert feature matching visualization of Gaudi from proj2.ipynb >

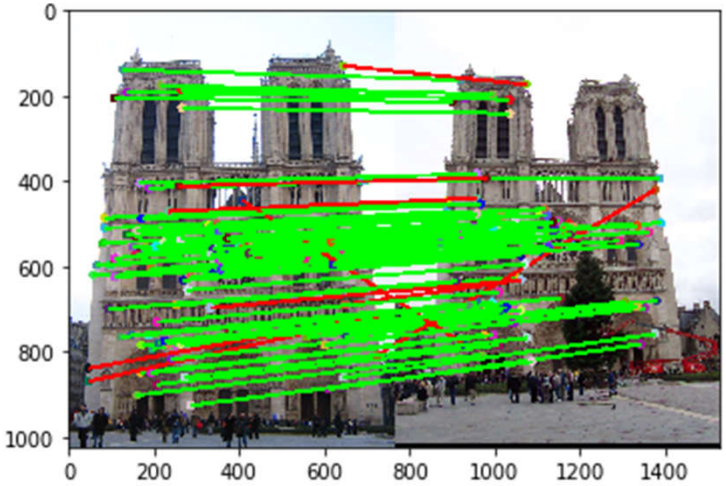


<Describe your implementation of feature matching.>

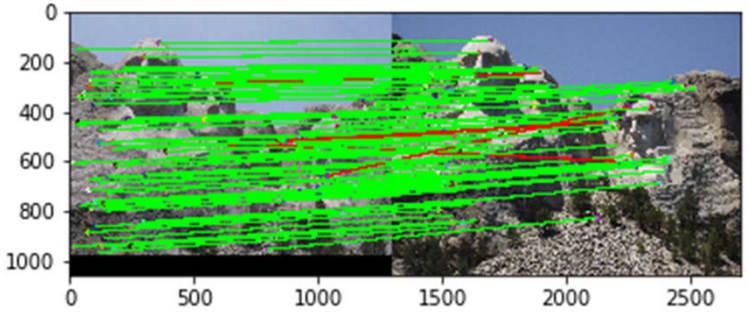
1. Firstly, I computed the distances between all features in features1 and features2.
2. For every features in features1, retrieve the 2 features in features2 which it has the smallest and 2nd smallest distance with.
3. Calculate nearest neighbor distance ratio(NNDR) for features in feature1 by dividing the smallest distance with 2nd smallest distance.
4. Sort features1 with the nearest neighbor distance ratio.
5. Filter out features if it has a NNDR of <= 0.8
6. Create matches array, where first column is features in feature1 sorted by NNDR, and second column is the corresponding feature in features2 with the smallest distance to the feature in column1.
7. Set confidences as NNDR sorted, with all values <= 0.8 filtered.
8. Return matches and confidences

Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Notre Dame from proj2.ipynb here>

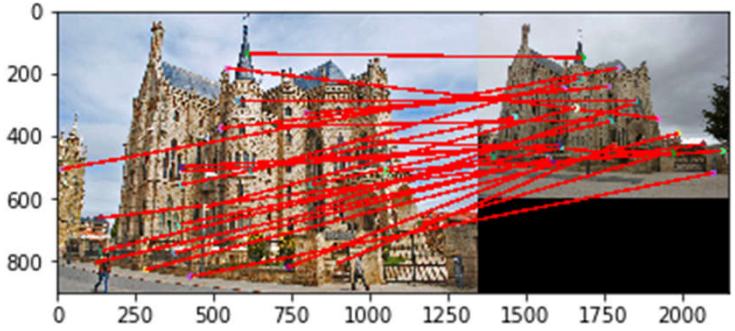


<Insert visualization of ground truth comparison with Rushmore from proj2.ipynb here>



Results: Ground Truth Comparison

<Insert visualization of ground truth comparison with Gaudi from proj2.ipynb here>

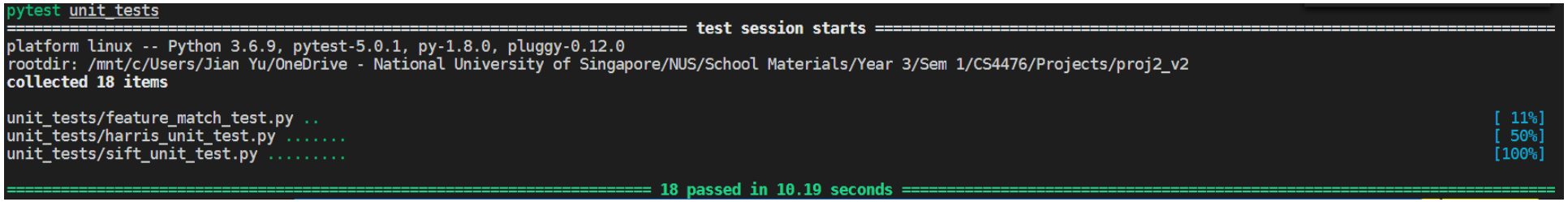


<Insert numerical performances on each image pair here. Also discuss what happens when you change the 4x4 subgrid to 2x2, 5x5, 7x7, 15x15 etc?>

Notre Dame: 100/100 matches, 0.90 accuracy
Rushmore: 100/100 matches, 0.95 accuracy
Gaudi: 26/100 matches, 0.00 accuracy

Tests

<Provide a screenshot of the results when you run `pytest unit_tests` on your final code implementation (note: we will re-run these tests).>



Conclusions

<Describe what you have learned in this project. Feel free to include any challenges you ran into.>

I have learnt a lot on how corner detection and feature matching is done with Harrisnet and Siftnet. I don’t claim to fully understand it, but I do believe that I understand the basics and at the very least how it is being done.

This project was really tough, but mainly because I started the project barely knowing anything about Harrisnet and Siftnet. I believe that if I had started the project with a better understanding, it would have proceeded a lot smoother.