Project 2 Writeup

Instructions

- This write-up is intended to be 'light'; its function is to help us grade your work.
- Please describe any interesting or non-standard decisions you made in writing your algorithm.
- Show your results and discuss any interesting findings.
- List any extra credit implementation and its results.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

This project implements image filtering using SIFT. Equation 1.

$$a = b + c \tag{1}$$

Implementation Detail

I first implemented match_features and followed the guide to find matrix D, the distances between every feature in image 1 and image 2. Then I sorted D and found the indices for the nearest neighbors for each row using `numpy.argmin`. I used this to create the matches array and to find confidences I divided the second nearest neighbor distances by the first nearest neighbor distances.

I then implemented get_features. I used `numpy.gradient` to calculate the gradient (which increased my accuracy compared to using the sobel filter) and adapted the pseudocode from project 2's questions to loop through all the x and y values by using a for loop to call a helper function that took in one x and one y value and outputted the descriptor at that point.
I had trouble trying to debug these two functions before seeing a piazza post and realizing I had mixed up the indexing ordering- I just had to switch some x and y's and my

accuracy with cheat_interest_points jumped 80% on the notre dame image.

Lastly I implemented `get_interest_points`. I followed the steps on slide 48 of the Interest Points and Corners presentation. I calculated the gradients using `numpy.gradient`, applied a Gaussian filter with sigma = 0.54, and computed the cornerness using:

$$C = det(M) - \alpha trace(M)^2 = g(I_x^2) \circ g(I_y^2) - g(I_x \circ I_y)^2 - \alpha[g(I_x^2) + g(I_y^2)]^2$$

Then I applied `peak_local_max` to eliminate clusters and thresholded the result.

After this I played around with alpha, threshold, and the parameters for `peak_local_max` to increase the accuracy. For my results I used alpha=0.01253, threshold=5.75E-6, a minimum distance of 2 and 400 peaks.

Result

For the Notre Dame photo I was able to get:

1. 86% accuracy on the 50 most confident matches
2. 79% on the 100 most confident matches
3. 42% on all matches.

For the Mt Rushmore photo I was able to get:

1. 76% accuracy on the 50 most confident matches
2. 80% on the 100 most confident matches
3. 35% on all matches.

For the Episcopal Gaudi photo I got:

1. 18% accuracy on the 50 most confident matches
2. 12% on the 100 most confident matches
3. 5% on all matches.

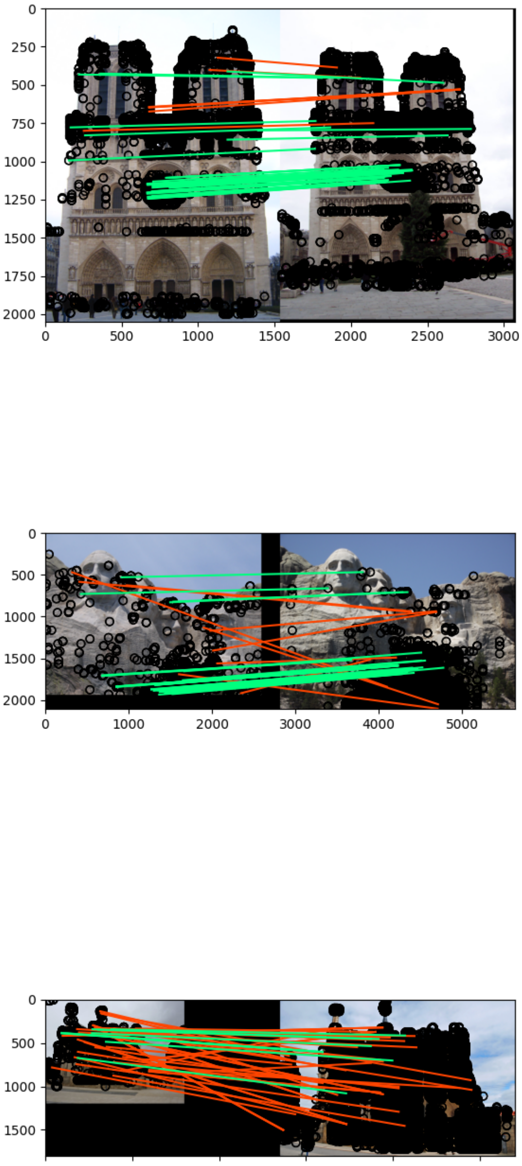

Figure 1: *Top:* Notre Dame result. *Middle:* Mt Rushmore result. *Bottom:* Episcopal Gaudi.