Homework 4 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Algorithm

get_interest_points.m

First, the image was blurred with a gaussian filter of width 5 and std 1 to remove noisy high frequencies that may cause extreme gradient values. Then, we used another gaussian filter of width 7 and std 1 to calculate the x and y gaussian derivatives. The x and y derivatives were approximated by correlating the gaussian filter with $\begin{bmatrix}0 & -1 & 1\end{bmatrix}$ and $\begin{bmatrix}0\\-1\\1\end{bmatrix}$ respectively.

Using these gaussian derivative filters, the x derivative (I_x) and y derivative (I_y) of the image was estimated. Then, the elements of the second moment matrices were calculated based on the following formula:

$$\begin{aligned} I_{xx} &= I_x^2 \\ I_{xy} &= I_x \cdot I_y \\ I_{yy} &= I_y^2 \end{aligned} \tag{1}$$

Then, the cornerness values can be found using the following formula proposed by Harris:

$$C = g(I_{xx}) \cdot g(I_{yy}) - g(I_{xy})^2 - \alpha \cdot (g(I_{xx}) + g(I_{yy}))^2 \tag{2}$$

where $g(\cdot)$ is the blurring gaussian used to initially blur the image, and $\alpha = 0.06$ as suggested by Harris.

The threshold value was set to 0.0000002 after a number of experiments on all three images. Adaptive non-maximum suppression was then applied by using a $(descriptor\_window\_image\_width \times descriptor\_window\_image\_width)$ window.

get_descriptors.m

A SIFT-like feature descriptor was implemented using Sobel filters to approximate directional gradients. For diagonal gradients, a Sobel filter was modified accordingly to capture appropriate gradients. For example, for the 45 degree gradient, the following 'Sobel' filter was used.

$$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{3}$$

Because the algorithm observes a 16×16 neighborhood of our feature point, appropriate padding was first added to the image to prevent out-of-range errors. Then, each Sobel filter was correlated with the image to give the approximate gradients in 8 directions. The resulting gradient matrix has dimensions of $(image\_height) \times (image\_width) \times 8$

Then, for each feature point, a $16 \times 16 \times 8$ patch was extracted from the matrix of gradients. This patch was element-wise multiplied by a 16×16 gaussian filter with std 8 in order to make a smooth falloff at the edges of the patch.

Then, for each $4 \times 4 \times 8$ crop of the $16 \times 16 \times 8$ patch, the gradients were added along the first and second dimension to create a feature descriptor of size $4 \times 4 \times 8$, which was resized to 1×128. This feature vector was then normalized to a unit vector and added to the list of feature descriptors.

match_features.m

A simple NNDR (nearest neighbor distance ratio) algorithm was implemented to match feature descriptor vectors. The Euclidean distances were calculated between the points of both images. Then, for each point in the first image, a point in the second image was matched.

The match was calculated by simply calculating the Euclidean distances from a point in the first image to all points in the second image, sorting the distances in increasing order, and finding the index of the uppermost point.

Confidence was calculated by $1 - \frac{NN_1}{NN_2}$ where NN_1 is the nearest neighbor and NN_2 is the second nearest neighbor.

Experiments and Results

Normalized Patches vs. SIFT-like Descriptors

First, we compared the performance of SIFT-like descriptors with normalized patches. The function for getting the normalized patches were implemented in a separate file,

get_patch_features.m. For the Notre Dame image, normalized patches recorded an accuracy of 23.14% for all submitted matches and an accuracy of 80% for the 100 most confident matches. On the other hand, the SIFT-liek descriptors recorded an accuracy of 24.36% for all submitted matches and an accuracy of 83% on the 100 most confident matches. Although not by a large margin, the SIFT-like descriptors perform better than normalized patches.

Visualization of Performance

We now visualize the performance of our feature matching algorithm on the Notre Dame, Mount Rushmore, and Gaudi's Palace in Figures 1, 2 and 3.

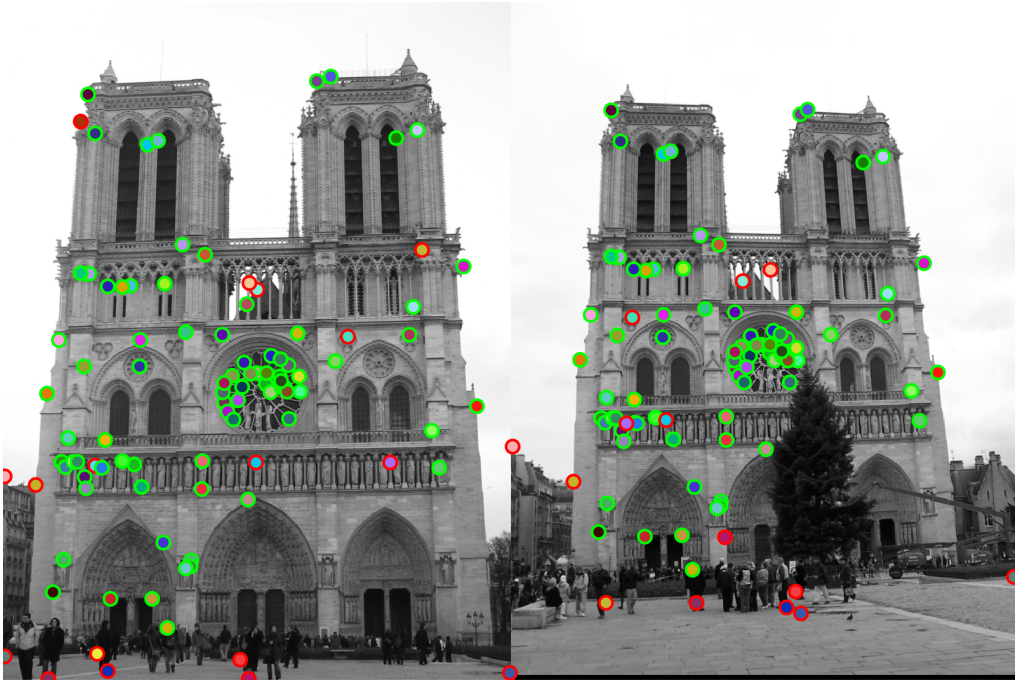

Figure 1: 100 Most Confident Matches on Notre Dame

For the 100 most confident matches, Notre Dame, Mount Rushmore, and Gaudi's Palace achieved 83%, 92% and 4%, respectively. The low performance for Gaudi's Palace is mainly due to the large difference in scale and rotation, which were not accounted for in the SIFT-like feature descriptors.

Extra Credit

Adaptive Non-maxima suppression was implemented. Details are included in section *get_interest_points.m*

Figure 2: 100 Most Confident Matches on Mount Rushmore

Figure 3: 100 Most Confident Matches on Gaudi's Palace