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.

In the beginning...

match feature

Starting with the match feature function. The implementation of the Nearest Neighbour distance ration was simple. Never the less i did some smaller mistakes on the confidence evaluation, that keeps me up searching for bugs later on.
I calculated the confidence by dividing the nearest distacne of two different points. This leads to negative confidences and i realized that there must be a error in the function, so i fixed it to:

```
confidences[i] = 1 - (nearest_distance[i,0] / nearest_distance[i,1])
```

After adjusting this line, my algorithm scores the desired percentage and i could focus on the parameter adjustment.

get descriptors

After working on the feature matching function, i start implementing on the descriptor function. At the beginning i struggled with understanding the idea of histogram bins and need to fixed the missing knowledge with paper research. After getting a understanding for the basic idea, i start coding and iterative solve the function. Again there was a problem with my code, that i found later on.

```
bins = np.abs(grad_angles_small_square // 45)
for b in range(8):
   sum = np.sum(grad_mag_small_square[bins == b])
```

In the way i calculated the angles, they were all negative. The problem was to not taking the absolute value when preparing for the binning. Having negative values, there where no values labeled with positive bin-numbers and it results in bad matching later on.

get interest points

Implementing the harris corner detector was not hard compared to the descriptor. Following the instructions of the lecture slides was enough to proper implement the algorithm.

Interesting Implementation Detail

```
for row in range(0, descriptor_window_image_width, 4):
    for col in range(0, descriptor_window_image_width, 4):
        grad_mag_small_square = grad_magnitudes[row:row+4, col:col+4]
        grad_angles_small_square = grad_angles[row:row+4, col:col+4]
        bins = np.abs(grad_angles_small_square // 45)
        for b in range(8):
            sum = np.sum(grad_mag_small_square[bins == b])

            feature[8*row + 2*col+b] = sum
```

The code above shows the way i build the feature for every detected corner point. Looping through the 16×16 square and calculate for the smaller 4×4 squares, the magnitude and angle. To speed up the implementation, i used vectorization. The final assignment of the summed values to the feature, is maby not that good looking, but it works :).

A Result

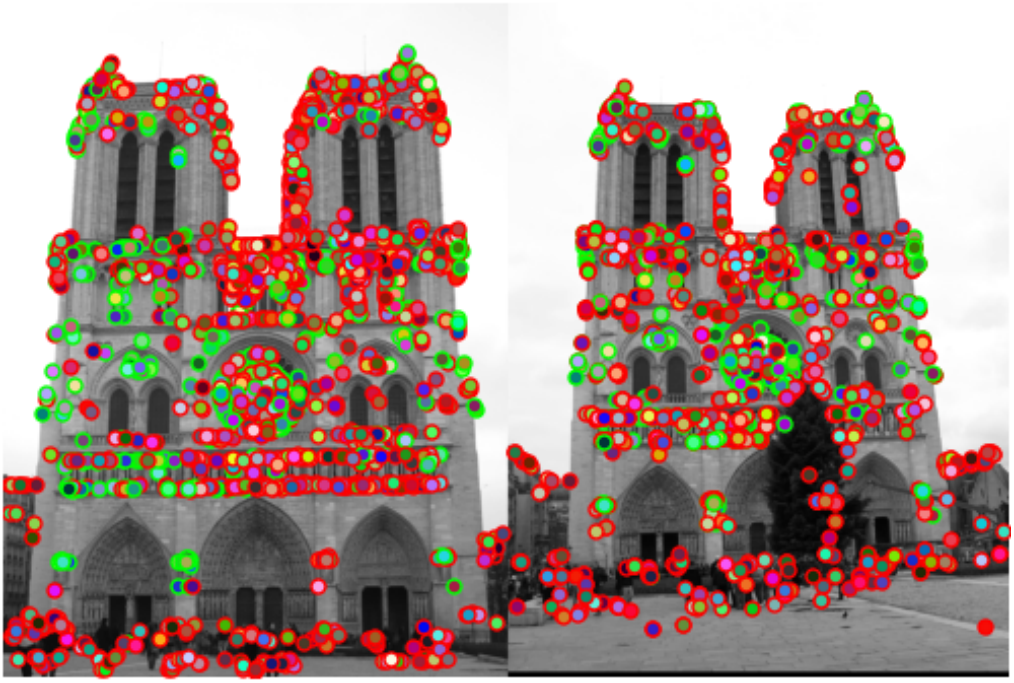

Figure 1: All results of ND

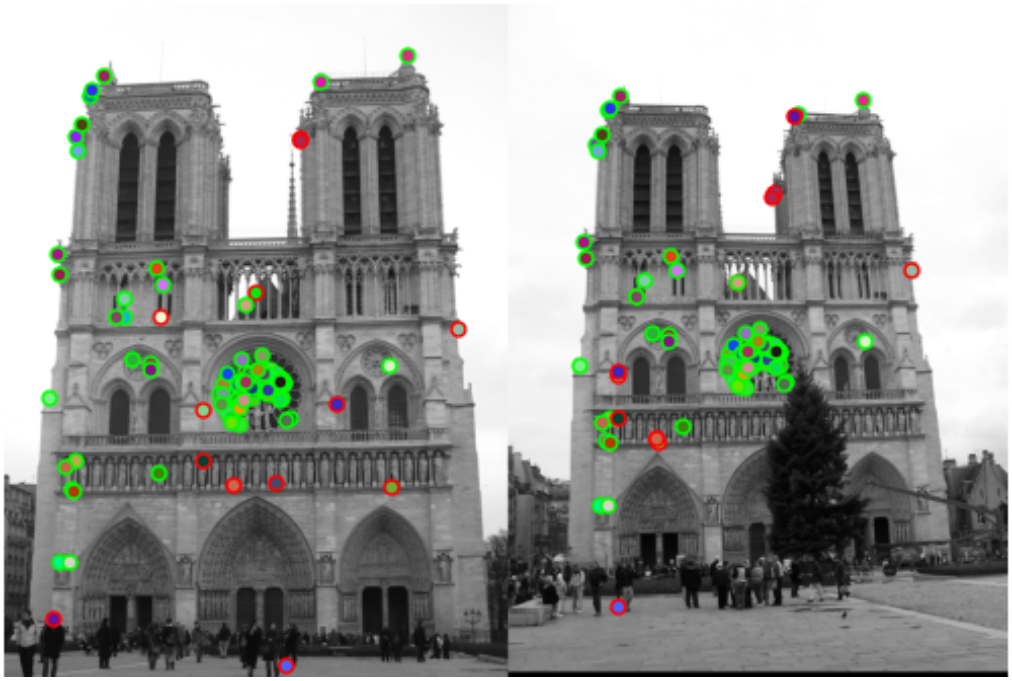

Figure 2: Top 100 results of ND

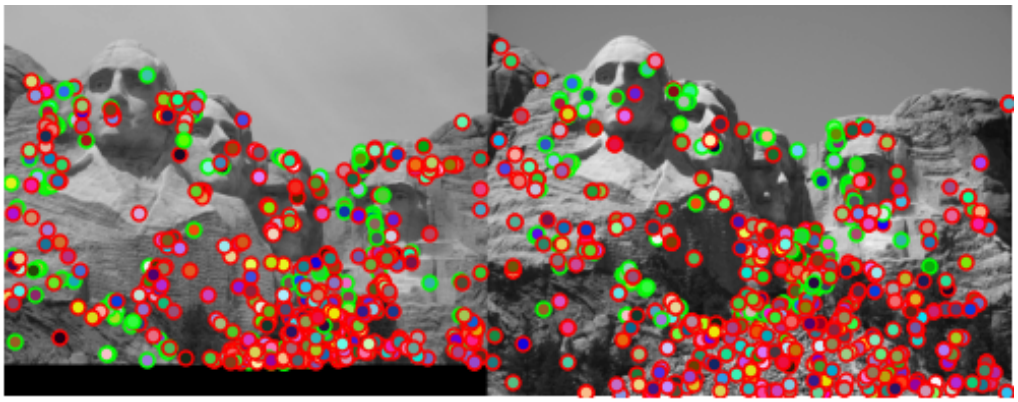

Figure 3: All results of MR

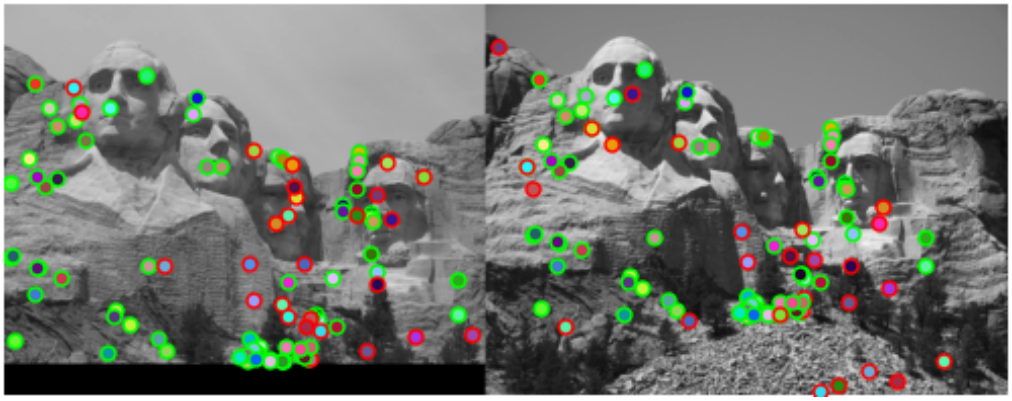

Figure 4: Top 100 results of MR

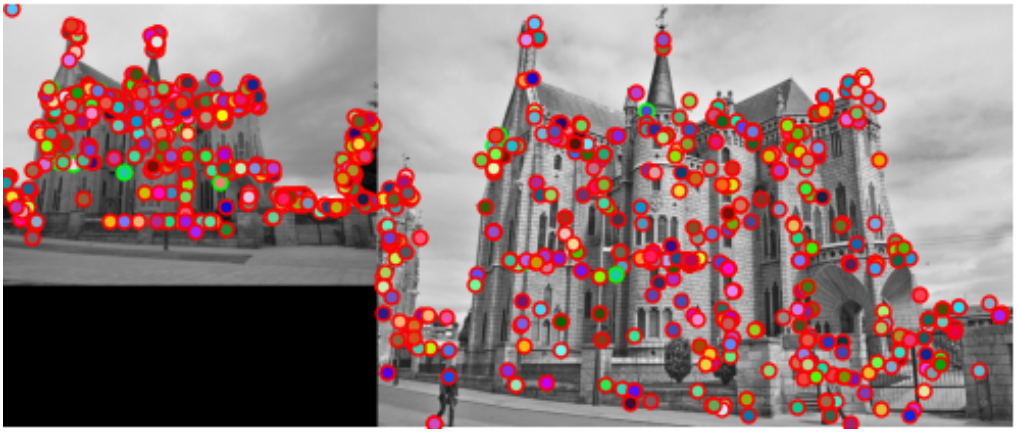

Figure 5: All results of EG

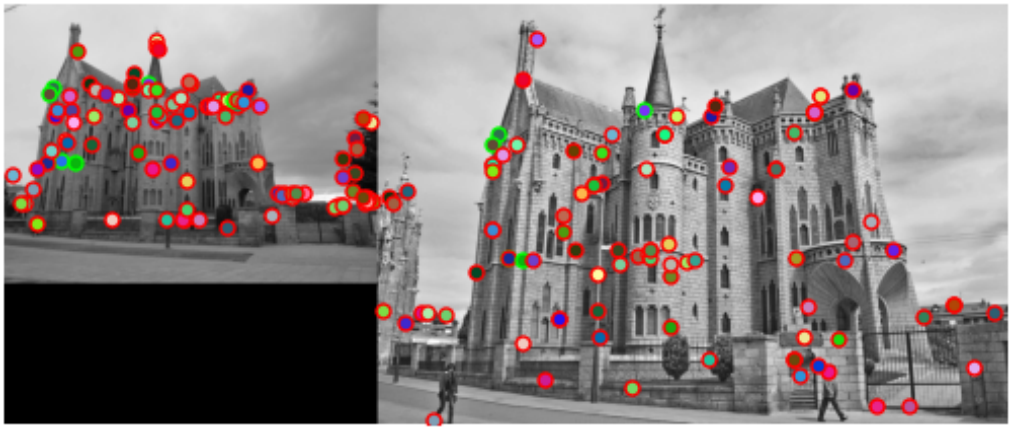

Figure 6: Top 100 results of EG

Image	Feature-style	Total(%)	Top 100 (%)
ND	Patch	23,94	74
	Bin	26,85	87
MR	Patch	10,14	35
	Bin	23,19	73
EG	Patch	1,1	3
	Bin	2,2	6

Table 1: Performance: Patch vs Bin on the 3 images

As you can see in the table above, using the patch as a feature performs worse compared to the Bin method. It has a score rate that is at some of the images only have of the score of the bin method. This massive differences in classification is due to the fact that the images are gray scale and so have not that high contrast. A image having a RGB-channel probably performs better, because there are three layers in which differences could appear.