Homework 3 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.**

In the beginning...

Harris corner detection and SIFT method is famous method for finding and matching the corners in two different images, has some same content in it. In this project, I have implemented Harris corner detector to find interesting points. Then implemented some SIFT-like to get the orientation for each windows and made histogram. Finally, I had to implement feature matching algorithm, by calculating euclidean distance and extract nearest neighbors' ratios.

Interesting Implementation Detail

```
localmaxima_C = zeros(size(C));
WINSIZE = 2;

for i = 1 + WINSIZE : size(C, 1) - WINSIZE
    for j = 1 + WINSIZE : size(C, 2) - WINSIZE
        tmp_window = C(i - WINSIZE : i + WINSIZE, j -
           WINSIZE : j + WINSIZE);
        local_maxima = max(tmp_window(:));
        if C(i, j) == local_maxima
            localmaxima_C(i, j) = C(i, j);
        end
    end
end
```

I have implemented my own non-maximum suppression algorithm. After calculated C, this algorithm take a look at neighbor values for every values. "Neighbor values" are defined by the constant WINSIZE. If the value is local maximum, we keep it. If it's not, it will be 0.

```
for fx = 1:4
    for fy = 1:4
        tmp_hist = zeros(1, 8);

        for wx = 1:4
            for wy = 1:4
                curr_x = (fx - 1) * 4 + wx;
                curr_y = (fy - 1) * 4 + wy;

                curr_dir = frame_dir(curr_x, curr_y);
                idx = floor(mod(curr_dir + 180, 360) /
                    45) + 1;
                tmp_hist(idx) = tmp_hist(idx) + frame_mag
                    (curr_x, curr_y);
            end
        end

        idx = (fx - 1) * 32 + (fy - 1) * 8 + 1;
        features(i, idx : idx + 7) = tmp_hist;
    end
end

features(i, :) = features(i, :) / norm(features(i, :));
```

I made a histogram for every frames, composed by 4x4 windows. In histogram, [-180, -135) degrees will be the index 1 in histogram, [-135, -90) will be 2, ... Every histogram calculated from each windows would be a part of the feature.

```
[n1, f] = size(features1);
[n2, f] = size(features2);

dists = zeros(n1, n2);
for i = 1 : n1
    for j = 1 : n2
        dists(i, j) = norm(features1(i, :) - features2(j,
            :));
    end
end
```

norm function in the MATLAB was useful to calculate the euclidean distance.

A Result

Sample	Accuracy(All)	Accuracy(100)
Notre Dame de Paris	57.92%	90%
Mountain Rushmore	63.18%	97%
Gaudi's Episcopal Palace	6.667%	7%

Figure 1: Result of Notre Dame de Paris

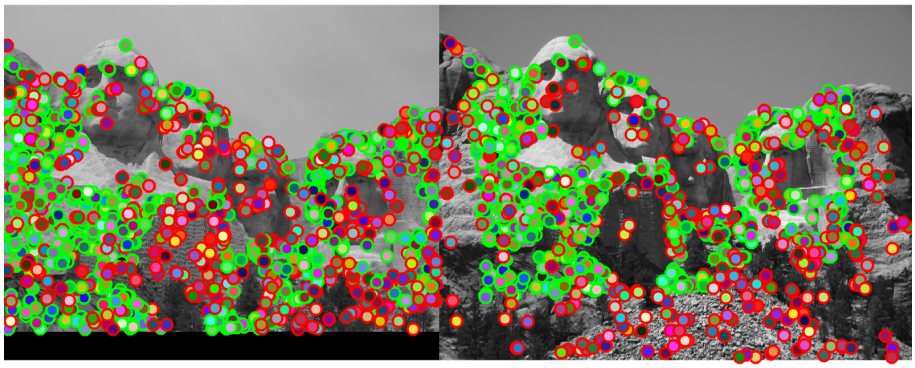

Figure 2: Result of Mountain Rushmore

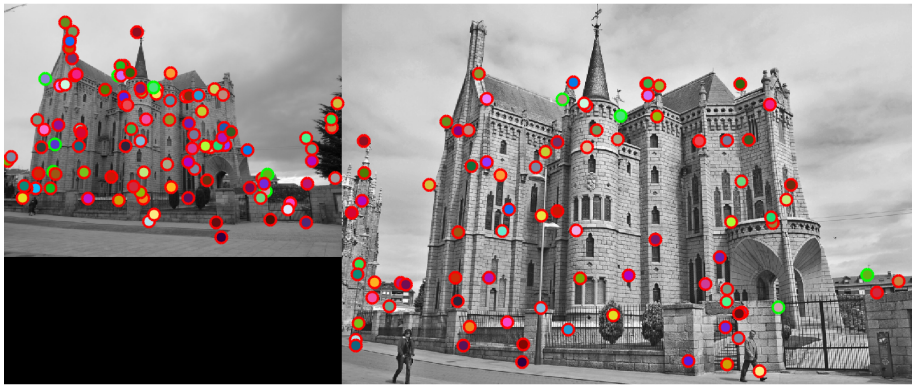

Figure 3: Result of Gaudi's Episcopal Palace