Report of CS 558 Project 4

Name: Yunfeng Wei
CWID: 10394963
E-mail: ywei14@stevens.edu

Step:

1. To extract features of a picture, I use Harris Corner Detector to detect corners from the image and then get a 5x5 image patches around the feature location as Figure 1.1 and Figure 1.2. Then I get a 1x25 feature descriptor from each corner and set as the feature descriptor.

```
% Project 4
truthSrc = dir('/Users/wyf920621/Documents/Computer_Vision/Project4/groundtruth_GRAZ_02_900_images/*.jp
truthDir = '/Users/wyf920621/Documents/Computer_Vision/Project4/groundtruth_GRAZ_02_900_images/';
descriptor = getCategoryDescriptor(truthSrc, truthDir);
```

Figure 1.1

```
function descriptor = getCategoryDescriptor(src, dir)
    count = 1;
    for a = 1:length(src)
        filename = strcat(dir, src(a).name);
        I = imread(filename);
        [rows, columns, numberOfColorChannels] = size(I);
        if numberOfColorChannels == 3
            I = rgb2gray(I);
        end
        I = im2double(I);
        % Feature Extraction
        % Get the corner position
        %position = corner(I, 'Harris', 'SensitivityFactor', 0.02);
        position = corner(I, 'Harris');
        positionCount = size(position,1);
        % end
        % Feature Description
        for i = 1:positionCount
            descriptor(:,count) = SetDescriptor(I, position(i,2), position(i,1));
            count = count + 1;
        end
    end
    descriptor = descriptor';
end
```

Figure 1.2

2. Run K-means clustering on all training features to learn the dictionary. Set the K as 500 to get 500 center descriptors as Figure 2.1 shows.

C

500x25 double

	1	2	3	4	5	6	7	8	9	10	11	12
3	0.0063	0.3023	0.5143	0.0410	0.0021	0.0661	0.6293	0.8054	0.1677	0.0042	0.3623	0.8
4	0.7059	0.9793	0.9961	0.8664	0.2684	0.4837	0.9508	0.9880	0.7584	0.1142	0.2294	0.8
5	0.0028	0.0035	0.0015	0.0029	0.0036	0.0462	0.0408	0.0337	0.0057	0.0022	0.6997	0.7
6	0.9985	0.9965	0.9969	0.9881	0.8060	0.9987	0.9962	0.9954	0.9422	0.5036	0.9797	0.9
7	0.0022	0.0312	0.4491	0.8881	0.9866	0.0036	0.2026	0.7967	0.9851	0.9971	0.0103	0.3
8	0.0261	0.4369	0.4010	0.0275	0.0023	0.0147	0.4359	0.7438	0.2624	0.0075	0.0098	0.4
9	0.0173	0.3070	0.7820	0.9585	0.7532	0.2048	0.6983	0.9486	0.9714	0.6256	0.6537	0.9
10	0.4857	0.1982	0.0474	0.0059	0.0027	0.9414	0.8255	0.5622	0.1335	0.0053	0.9952	0.9
11	0.0020	0.0299	0.6285	0.9726	0.9961	0.0023	0.0458	0.6804	0.9796	0.9964	0.0021	0.0
12	0.0031	0.0019	0.0048	0.0107	0.0368	0.0019	0.0832	0.3248	0.4912	0.6119	0.0085	0.4
13	0.8512	0.5261	0.1295	0.0080	0.0037	0.9892	0.9240	0.6640	0.2367	0.0213	0.9970	0.9
14	0.6316	0.9785	0.9836	0.7755	0.1564	0.6704	0.9809	0.9947	0.8918	0.3160	0.7345	0.9
15	0.9804	0.9859	0.9922	0.9984	0.9898	0.7576	0.9953	0.9718	1	0.9875	0.0180	0.0
16	0.1707	0.8090	0.9932	0.9969	0.9984	0.0787	0.7325	0.9893	0.9959	0.9978	0.0201	0.5
17	0.9976	0.9973	0.9751	0.6414	0.0679	0.9988	0.9967	0.9791	0.6397	0.0399	0.9964	0.9
18	0.0021	0.0124	0.2477	0.3819	0.0658	0.0069	0.0652	0.4938	0.3360	0.0142	0.0066	0.1
19	0.9638	0.9953	0.9960	0.9962	0.9685	0.7801	0.9510	0.9861	0.9310	0.7389	0.2601	0.6
20	0.9977	0.9647	0.5794	0.0155	0.0023	0.9969	0.9806	0.6709	0.0347	0.0017	0.9971	0.9
21	0.0027	0.0042	0.4409	0.9545	0.9984	0.0023	0.0052	0.2321	0.9115	0.9964	0.0030	0.0

Figure 2.1

3. For the features extracted from any image, quantize each feature to its closest codevectors in the learned dictionary and accumulate a histogram by counting how many features are quantized to each codevector in the dictionary as Figure 3.1 shows. The bag-of-features are represented as the percentage of all 500 centers.

features

120x500 double

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0.0417	
3	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0.0127	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	0	
9	0	0.0127	0	0	0	0	0.0127	0	0	0	0	
10	0	0	0	0	0	0	0.0127	0	0	0	0	
11	0.0127	0	0	0	0.0127	0	0.0253	0	0	0	0	
12	0	0	0	0	0	0	0.0253	0	0	0	0.0127	
13	0	0	0	0	0	0	0.0127	0	0	0	0	
14	0	0	0	0	0	0	0.0127	0	0	0	0	
15	0	0	0	0	0	0	0.0127	0	0	0	0	
16	0	0	0	0	0	0	0.0127	0	0	0	0	
17	0	0	0	0	0	0	0.0127	0	0	0.0127	0	
18	0	0	0	0	0	0	0	0	0	0	0.0127	
19	0	0	0	0	0	0	0	0	0	0	0	

Figure 3.1

4. Use the LIBSVM to train the Support Vector Machine with the histogram representation of all images as Figure 4.1. The label for bike is set as 1, car is 2 and person is 3. The total training dataset number is 900.

```
*
optimization finished, #iter = 300
nu = 1.000000
obj = -598.717923, rho = -0.001796
nSV = 600, nBSV = 600
*
optimization finished, #iter = 300
nu = 1.000000
obj = -598.896008, rho = -0.002778
nSV = 600, nBSV = 600
*
optimization finished, #iter = 300
nu = 1.000000
obj = -598.613859, rho = -0.000349
nSV = 600, nBSV = 600
Total nSV = 900
```

Figure 4.1

5. Finally, use the SVM to predict the test dataset, and the accuracy is shown as Figure 5.1. We can see that the accuracy for bike is 99.17%, the accuracy for car is 90% and the accuracy for person is 56.67%.

```
Accuracy = 99.1667% (119/120) (classification)
Accuracy = 90% (108/120) (classification)
Accuracy = 56.6667% (68/120) (classification)
```

Figure 5.1