# CPE558 2014Spring Project 4

Junjie Wang, jwang44@stevens.edu

## Introduction

The project is to implement an object recognition algorithm on the given Graz02 dataset starting from scratch. The main procedure includes the feature extraction and description, building the dictionary, image representation and classification. I write my own codes for the most of the functions, including the corner detector, the SIFT descriptor, image representation, k-NN classifier and multiclass support for SVM. I try use two algorithms to classify the dataset with various parameters. Then I make a table to compare different results between them.

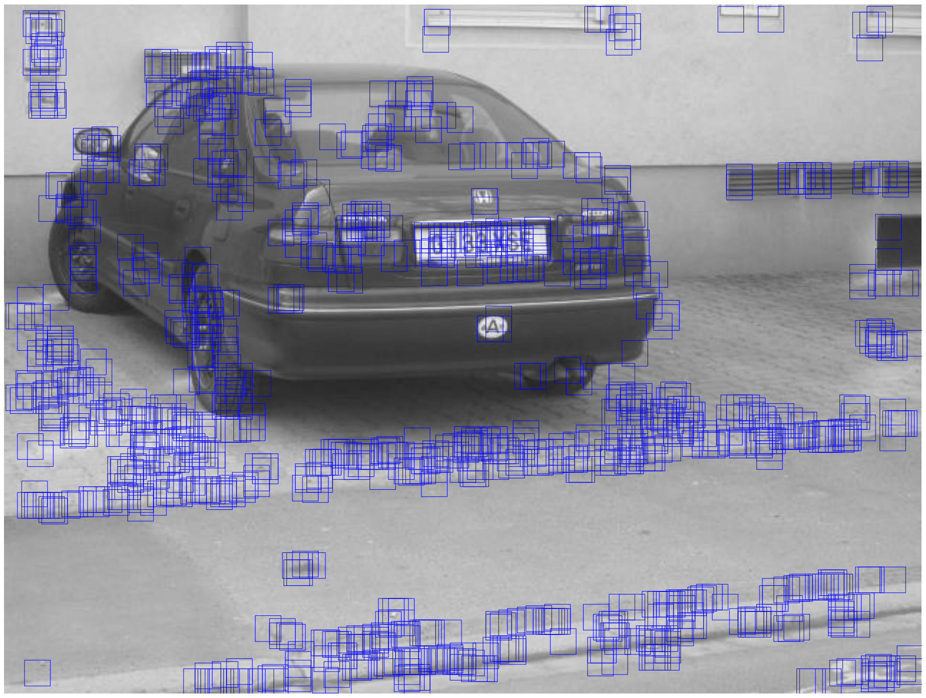
## Detailed Steps

### Feature Extraction

The method used for feature extraction is my Harris corner detector as the second project of this course. The steps for the detector are shown below.

1. Compute derivatives at each pixel
2. Compute second moment matrix M from the derivatives
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$
3. Compute corner response function
$$R = \det(M) - \alpha\, trace(M)^2$$
4. Threshold R
5. Find local maxima of response function

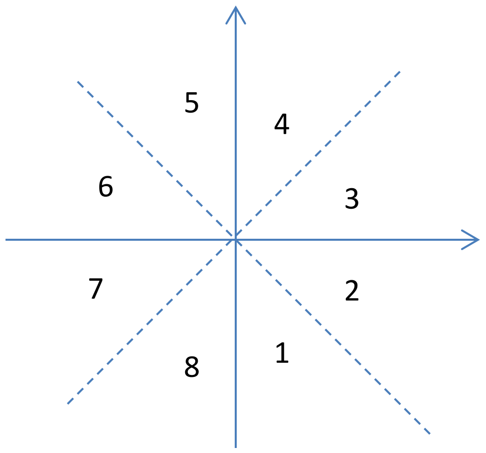
The window size is 16 pixels since in later calculation the descriptor requires the input window size to be 16 pixels. The threshold of R is set to be 1 and α is set to be 0.05. The window function $w(x,y)$ is 1 inside the window and 0 outside.
For images in the training set, usually thousands of feature points can be detected in bike, car and person categories for one image. However for the image in the 'none' category, sometimes only several hundred points can be detected.
In addition, for later calculation every point is recorded according to the value of response function R in a descending order.

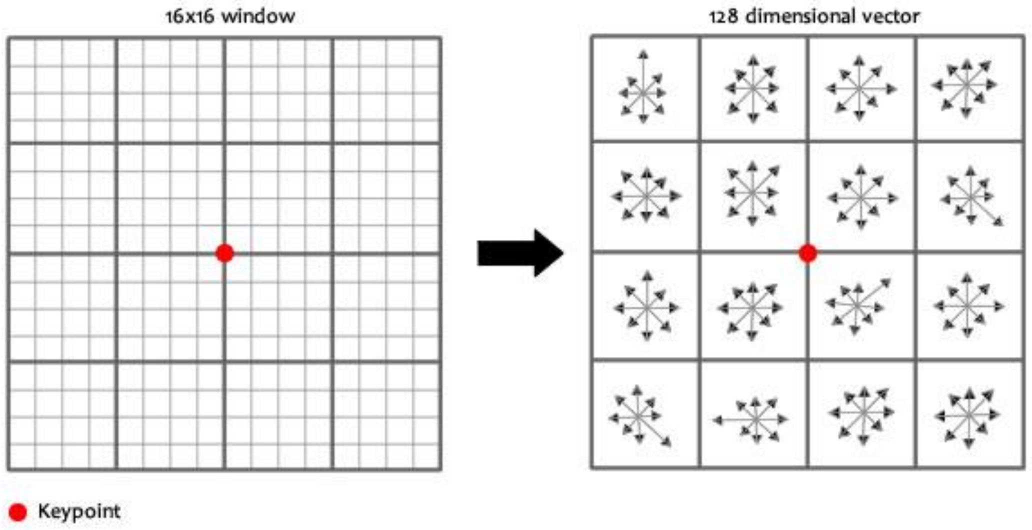


The figure shows one example of the detector. Many detected points, which are sufficient to describe most part of a car like the license plate, wheels, and mirrors and so forth, are around the car, while still many other points are in the background.

## Description

A simple version of SIFT [2] descriptor is written by myself and used here. Only the first half of the points, which have higher values of response function detected in the previous step are used in computation for descriptors. In this way the computation resource such as time and storage can be reduced. Since I have sorted the detected points, this method is equivalent to setting a different threshold in the previous step.
The image patch from the previous step has a size of 16 pixels * 16 pixels. Each patch is divided into 4*4 sub regions with size of 4*4. For each sub region 8 bins that represent 8 regions for directions are used. Gradients of the 16 pixels in each sub region vote for the 8 bins according to their directions and their magnitudes are accumulated.
The figure below shows the labels of 8 bins that correspond to regions of directions.





The image above intuitively shows the principles for the calculation of SIFT. [3]
The result matrix is then vectorized. Therefore the descriptor of one image patch is a vector with 4*4*8=128 dimensions. In the end the vector is normalized to achieve invariance to some certain level. However the version I implement is not rotation invariant.

## Dictionary

One image has hundreds to thousands of detected points. For the whole training set, more than 2 million points are detected and I use the first half of them, which are more vital to images and count more than 1 million to build the dictionary. K-means clustering on the detected feature points is used to learn the dictionary.
K-means uses an iterative method to find clusters in the data. The first is to initialize the centroids of each cluster, which are randomly selected among all the points. Then in each step, every point in the data is assigned to the nearest cluster centroid and the new centroid of each cluster is updated to the mean of points that belong to one cluster. The procedure is repeated until the centroids of clusters do not change any more. Due

to the limit of time, the value of k meaning the number of clusters is set to be 700 and only this value is tested.

## Image Representation

After the dictionary is built, the all the images in the training, validation and test set are computed for representation. The dictionary has a dimension of 700 and each code is a vector with a size of 128.

Every feature of an image is assigned to the closest code in the dictionary and every image is represented by a histogram by counting the number of features quantized to the codes. Hence each image is represented by a vector of 700 integers in this way.

## Classifier

After we have all the image representations and groups, we can begin to classify them. I tried two classifiers here, k-Nearest Neighbors algorithm and Support Vector Machine.

### k-NN

I implement my own k-NN classifier with one parameter k, which specifies the range of neighbor for comparison. The k-NN inputs all the training data including image representation points and related labels of 4 categories at one time. Then for each test case, k-NN computes the distance between the test case and all the training cases and finds k nearest cases. Next the categories of the k points vote for the final category of the test data. To compute the nearest cases, the concept of distance function plays an important role. In the project, I use histogram intersection as the distance function defined as below.

$$I(h1, h2) = \sum_{i=1}^{n} \min(h_1(i), h_2(i))$$

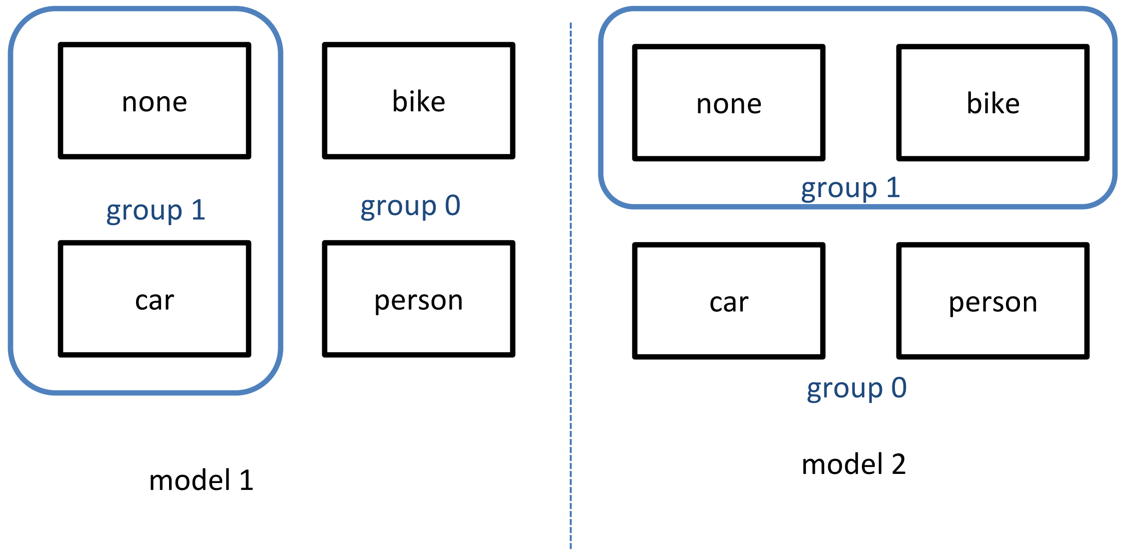
where n is the dimension of the histogram, which is 700 as in the image representation. With histogram intersection, one case is near to another in the sense that the two cases have a high value of histogram intersection.

Various choices of k are tested to find better parameters.

### SVM

Support Vector Machine is the other classifier I use here. Matlab has a built-in pair of functions, ‘svmtrain’ and ‘svmclassify’ [4], to perform the classification. However ordinary SVM only solves the classification problem with two categories. I develop my own way to deal with problems in this project which has four categories.

The training set has 4 categories, none standing for the background images, car, bike and person. I decide to group them up and train two models. I first try to make none and car in one group and bike and person in the other group. One model is trained in this way. Then I make none and bike in one group and car and person in the other group. The second model is trained to fit this way. The figure below explains the method intuitively.



After the two models are trained we can classify the data in validation and test set. In the classification part we can calculate the results of two models for each test case and recover the categories according to the following table.

| Model 1 | Model 2 | Category | |
|---|---|---|---|
| 1 | 1 | 1 | none |
| 1 | 0 | 2 | car |
| 0 | 1 | 3 | bike |
| 0 | 0 | 4 | person |

This is a modified version similar to the one-versus-one approach to solve multiclass SVM problems. This method is simple and effective.

## Recognition

Many experiments have been conducted to test various methods and parameters that can impact on the accuracy of recognition.

### k-NN

The k-NN classification has an upper limit of 40%. I have tried different values for k. In addition I try to use half of the detected feature points and full of the points for image representations. As the result shows, increasing feature points for image representation and different k values cannot help improve the accuracy.
The following table shows different results of classification from k-NN and SVM with various parameters. The numbers indicate the number cases that are correctly classified.

| kNN | 1 | 2 | 3 | 4 | Accuracy |
|---|---|---|---|---|---|
| 100%, k = 5 | none | cars | bike | person | |
| validation | 27 | 50 | 79 | 12 | 42% |
| test | 18 | 34 | 71 | 18 | 35.25% |
| | | | | | |
| 50%, k = 10 | | | | | |
| validation | 23 | 42 | 81 | 10 | 39% |
| test | 27 | 40 | 73 | 11 | 37.75% |
| 50%, k=3 | | | | | |
| validation | 48 | 26 | 71 | 23 | 42% |
| test | 35 | 20 | 64 | 23 | 35.5% |
| | | | | | |
| SVM,100% | Gaussian Radial Basis Function kernel, sigma = 10 | | | | |
| validation | 69 | 70 | 65 | 56 | 65% |
| test | 38 | 65 | 66 | 42 | 52.75% |
| | sigma = 20 | | | | |
| validation | 36 | 79 | 83 | 69 | 66.75% |
| test | 20 | 74 | 85 | 64 | 60.75% |

50% indicates that half of the detected points are used for image representation and 100% indicates that all detected points are used.

## SVM

The SVM classifier has overall a much better performance. The best result I can get is to use Gaussian Radial Basis Function as the kernel with sigma equal 20. The accuracy is shown in the table above. It is 66.75% for the validation set and 60.75% for the test set. Tables below show the confusion matrix. The column shows the input test data for training, validation and test set. The rows show the classification results. The diagonal elements are number of correctly classified cases and the rest are numbers of misclassified ones.

| training | results | none | car | bike | person | total |
|---|---|---|---|---|---|---|
| sources | | | | | | |
| none | | 130 | 36 | 12 | 2 | 180 |
| car | | 6 | 205 | 0 | 9 | 220 |
| bike | | 7 | 3 | 145 | 10 | 165 |
| person | | 0 | 13 | 2 | 96 | 111 |

Accuracy for training set: 576/676=85.2%

The following table shows the confusion matrix for the validation set. The classification accuracy is 66.75%.

| validation | results | none | car | bike | person | total |
|---|---|---|---|---|---|---|
| sources | | | | | | |
| none | | 36 | 50 | 7 | 7 | 100 |
| car | | 11 | 79 | 2 | 8 | 100 |
| bike | | 6 | 5 | 83 | 6 | 100 |
| person | | 2 | 15 | 14 | 69 | 100 |

The following table shows the confusion matrix for the test set. The accuracy is 60.75%.

| test | results | none | car | bike | person | total |
|---|---|---|---|---|---|---|
| sources | | | | | | |
| none | | 20 | 38 | 20 | 22 | 100 |
| car | | 16 | 74 | 5 | 5 | 100 |
| bike | | 3 | 2 | 85 | 10 | 100 |
| person | | 4 | 15 | 17 | 64 | 100 |

## Discussions

From the results of k-NN algorithm, we can see that the performance is relatively poor. Different values of k when choosing neighbors of a test case do not make a difference. Also the bike category has always a better accuracy and the person category has a very low rate of recognition. This becomes extremely obvious when k becomes larger. The recognition rate for none and person categories can drop to very low and become even worse than a random guess. I think the main reason is that the number of training cases of the bike category is much larger than that of the person category. Therefore in the space separated by the given training set, the bike category always occupies more area than other categories and every random test case has a higher probability to be classified into the bike category. Another reason might be the chosen distance function which is the histogram intersection. The function might not be able to well separate different categories so that the result is sensitive to noise.

In the case of SVM algorithm, generally the 'none' category has a rather low rate of recognition. The 'person' category has a lower rate of recognition but it is not that obvious. The reason might be that normal images in other categories may have features similar to those in the 'none' category since a large portion of any image can be the background and they all share the same pattern. Therefore it can be difficult to tell background images from the others.

## Conclusion and Future Work

The overall performance shows that SVM is a better classifier than k-NN in this application. Various parameters for k-NN do not result in much different consequences. The k-NN also seems to be sensitive to the number of training cases. The classification tends to be biased if different categories count differently in the training set.

SVM has a stronger ability to correctly classify the cases. Compared to the upper limit of 40% of the k-NN classifier, SVM achieves accuracy of 60% with only a few tests.

Both classifiers have a poor accuracy recognizing the background images that belong to the 'none' category. I think background images should be specifically treated to enhance the recognition rate.

SVM uses the Gaussian radial basis function with different values of sigma as the kernel function. In addition the histogram intersection used in k-NN could also be implemented as the kernel function for more tests.

Good features should also help improve the accuracy. My corner detector only operates with fixed window size in one scale and my SIFT descriptor is a simplified version without rotation invariance. Hence multi-scale feature could be added to the feature extraction step and the SIFT descriptor can be improved for more complicated functions so that we can get better features and descriptors. Then we may have better image representations that can be more easily differentiated.

## References

[1] C.Harris and M.Stephens. "A Combined Corner and Edge Detector." Proceedings of the 4th Alvey Vision Conference: pages 147—151, 1988.

[2] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.

[3] http://www.aishack.in/wp-content/uploads/2010/06/sift-fingerprint.jpg

[4] http://www.mathworks.com/help/stats/svmtrain.html