# Pedestrian Detection

Using HOG features and SVM in Matlab

Ariyan Zarei 3/31/2017

## Abstract

This is the report for the seventh project of the Image Processing Course (2nd Semester of the 95-96) by Dr. Alireza Tavakoli. We managed to implement a Pedestrian Detection system in Matlab. The system uses SVM and HOG features. The first step towards the detection is to train the system which in our case it was done by means of SVM classifier and positive/negative image set provided by Cornel University. Then the classifier was tested and assessed by some other positive and negative images and its accuracy in detection was satisfactory. After all mentioned works it was time to move forward to have a complex pedestrian detector system which detects and highlights the pedestrians with various sizes in a large and detailed image, containing other objects. The final results was fascinating considering the small image set used as training data. You can see the results at the end of the report.

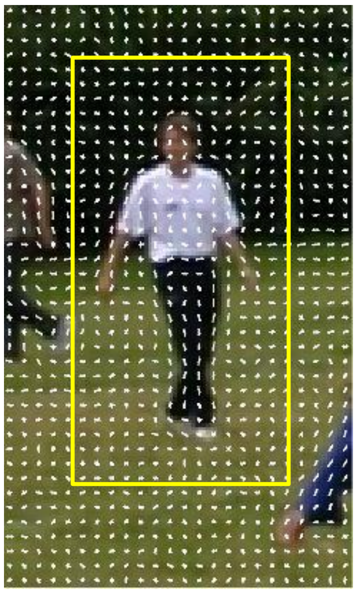

Image 1- applying HOG feature extractor on an image in order to extract the feature vector to feed the SVM classifier

# Pedestrian Detection

Using HOG features and SVM in Matlab

## Introduction

Detection of pedestrians in an image has been become important in the recent years by increase in the production of intelligent vehicles. As you can guess these vehicles has some kind of alarming system which in case of encountering a pedestrian, it will inform the driver in order to change the direction or decrees the speed of the car. As you know any detection problem take advantage of 2 important things, a classifier and a feature extractor. In case of our Pedestrian Detection system, we used SVM as our binary classifier and HOG as our feature extractor. In this project we trained a SVM classifier with sample positive and negative pedestrian images in a fixed and unique size. The images have been downloaded from Cornell University website[1]. Then by implementing an effective and simple method in Matlab, we tried to detect pedestrians in general, highly detailed images which contain other objects as well. In the following sections we are going to expand the methodology of our project.

## SVM Classifier Generator

This method, 'GenrateClassifier.m', deals with the important task of generating a SVM classifier based on the input images and their labels. It takes a directory for the input images and a simple number which represents the HOG window size. It should be mentioned that, the input image set should be in correct folder formatting, which is like having 2 folders each for the positive and negative image sets and as many as images inside each folder. This method goes through the below steps in order to generate the classifier:

- Defining the Feature and Label Matrices
- For each input image
    - Extracting the HOG feature using 'extractHOGfeatures' method of Matlab.
    - Adding the acquired informations into the Feature and Label Matrices
- Generating the SVM classifier using the 'fitcecoc' method. We can also use 'fitcsvm'.

By using the SVM model generated with this method we would able to classify images with the same size as the training images into two categories 'pedestrian' and 'non-pedestrian'. But this is not enough because input images are not always clean and cropped in a same size. They usually contain other object as well. In the next section we are going to explain our method to overcome this problem and detect multiple pedestrians in a highly detailed and general image.

## Pedestrian Detection in a Detailed Image

In order to find all the objects (pedestrians in this project) in an image, we have to check into all the possible image patches, with the size used in training step, of the main image and classify them into positive/negative classes using SVM. In our case the size used in training step was 160*96. There are 2 main problem with this simple approach:

- A single pedestrian (object) might be possible be detected in multiple patches (Image 2).

[1] http://www.cs.cornell.edu/courses/cs4670/2013fa/projects/p5/

- Pedestrians with different sizes in the main image (too small or too large) might not be detected because the SVM is trained with images in size 160*96.

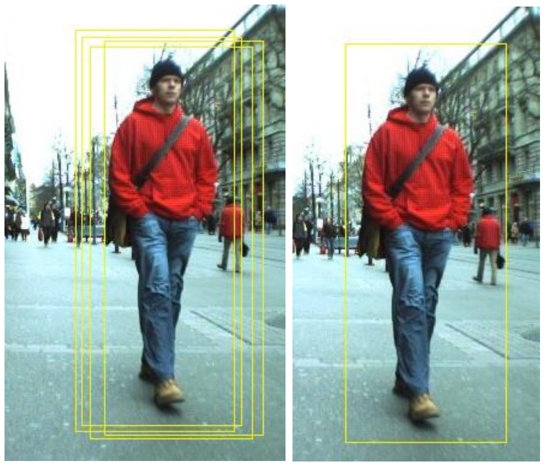

Image 2 - detection of a single object in multiple patches and solving the problem by combining overlapped ones.

In order to overcome these two problems, we added two more steps to this approach. For the first problem, the multiple detection of a single object, we combined the patches or boxes which had overlapped more than 50 percent of their area. The 'selectStrongestBbox' Matlab method used in order to merge these overlapped boxes. This procedure is called non-maxima suppression or NMS.

For the second problem, we generated an image pyramid and we detect pedestrian in each different sizes available in the pyramid. It is like running the same detection mechanism multiple times on the same image but in different sizes.

The result of having these two steps is almost clear and accurate detection. The functions 'FindPedestrian' and 'FindPedestrianNMS' were implemented to be used for the goal of pedestrian detection in large and detailed images. By using the first function, you ignored the two aforementioned problems, but the second one will run the non-maxima suppression and image pyramid procedures. The image pyramid used in this method is shown in the image 3.

Both functions mentioned earlier, return an array of n boxes (with size of n*4, which each row is like [x y width height]) and also an array of n scores. The scores represents the confidence of detection and are negative numbers in range of [-1, 0]. The higher the score, the more confidence of detection.

## Results and Conclusion

In order to test the system, you have follow the steps below:

- Run the method 'GenerateClassifier' on the downloaded image set from Cornell University (cropped_pedestrian folder) as:
  `classifier = GenerateClassifier('…\images\',8);`
- Generate the boxes of detected pedestrians by calling the 'FindPedestrianNMS' function as:
  `[boxes,scores] = FindPedestrianNMS(image,160,96,classifier,8);`
- Draw the boxes over the input image as:
  `Image2 = insertShape(image,'rectangle',boxes);`

- Show the final result by using 'imshow' function.

You can see some images and the detection results on them below.