CS512 Project Report
HOG Methodology for Pedestrian Detection

Aditya Yaji
A20426486

Pallavi Chandrashekar
A20427289

Abstract:

In this report, we shall discuss about the Pedestrian Detection using HOG (Histogram of Oriented Gradients). We shall also discuss detailed implementation of HOG Methodology for Pedestrian Detection and the problems faced during implementation. Finally, we discuss the analysis of the results obtained and various dataset used.

Keywords: HOG, Pedestrian Detection, SVM.

1. Problem Statement:

Figure 1. Pedestrian Detection.

Pedestrian Detection (see Fig. 1) is widely used for intelligent control, intelligent transportation driving auxiliary, advanced human computer interaction and so on. Nevertheless, Pedestrian Detection is a challenging problem. Challenges faced are

i. Various styles in appearance i.e. clothing and trousers.
ii. Non-rigid objects i.e. various bodies and actions.
iii. Background environment.
iv. Frequent occlusion between pedestrians.

Despite challenges, Pedestrian detection remains as active research area in Computer Vision. Numerous approaches have been proposed. Holistic detection, part-based detection, patch-based detection, motion-based detection, detection using multiple cameras. Of all this, one of the typical and effective framework applies Histogram of Oriented Gradient (HOG) as descriptor and linear SVM to train the pedestrian detector.

2. Proposed Solution:

Here, we are using HOG descriptors to represent training samples, we can train a linear detector by SVM. In detection phase, we convolve the linear detector with HOG descriptors extracted from the dense detection windows within an image, assigning detection (prediction) score to each window. A threshold is enforced to decide whether a window contains a human.

- ### Histogram of Gradient:

HOG descriptor captures edge or gradient structure that is very characteristic of local shape. In addition, it is a local representation with controllably invariance to local geometric and photometric transformations.

Before illustrating the construction of HOG descriptor, we firstly describe the Laplacian Gradient.

a) Laplacian Gradient:

Laplacian Gradient (LG) is a fundamental concept in computer vision. This phase shows how to compute the gradient of a single pixel. Take a patch from an image and mark a certain pixel within this patch to be red. Assume, this pixel's value is 58. We can also access its four neighbors (left, right, top and bottom pixels) and their pixel values are 42, 31, 36 and 17. LG of the selected pixel (value of 58) consists of the gradients of horizontal (31-42) and vertical (36-17) directions, which is LG = [-11,19]. Its magnitude and angle can be figured out as 21.95 and 2.1 (rads) base on the LG.

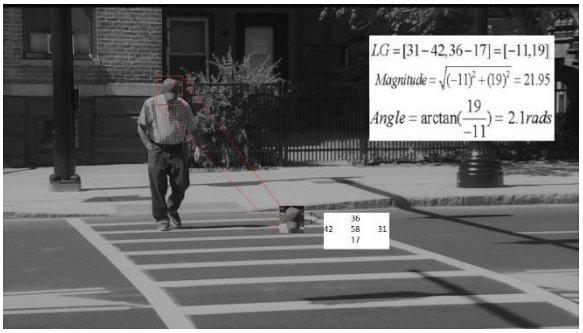

Figure 2: Gradient and angle

b) Constructing HOG Descriptor:

Given an image patch, which we call it a "cell" in the rest of this specification, we can obtain angles of all the pixels within the cell by following the above-mentioned computation. For example, we set a cell cover 8 x 8 pixels, then we can obtain 64 angles and magnitudes. We accumulate the 64 magnitudes into a 9-bin histogram according to the corresponding angles which ranges from $-\pi$ to π, i.e. the interval is 40 degrees. The histogram is the HOG of a cell.

In practice, we often group several cells into a block, e.g. the block 1 and block 2 which contain 2 x 2 cells shown in Fig. 3(a). The neighborhood blocks always overlap with each other by 50%.

Take block 1 for example, we respectively obtain 4 HOGs of the cells within this block. Then the 4 HOGs are concatenated into a vector. Denote the concatenated vector as v, we normalize v by L2-norm, i.e. $v = \frac{v}{\sqrt{||v||_2^2 + \epsilon^2}}$, where ϵ a small residual for avoiding "division by zero" in practice. The normalized vector v is the HOG descriptor of a block. Assume we have a 256x128 gray scale image and set the cell to cover 8x8 pixels. Then this image can be divided into 256/8=32 cells vertically and 128/8=16 cells horizontally. We also enforce each block contain 2x2 cells. Following the strategy of overlapping the blocks by 50% as shown in Fig. 3(b), we can group the cells into 32-1=31 blocks vertically and 16-1=15 blocks horizontally. Totally there are 31x15=465 blocks in this image. Since the dimensionality of the HOG descriptor for a block is 9 bins x (2x2) cells = 36, then dimensionality of final descriptor of this 256x128 image is 36 x 465 blocks = 16,740. In the rest of this specification and the implementation, we fix the number of bin to 9. We also fix the size of a cell to cover 8 x 8 pixels and block to contain 2 x 2 cells.

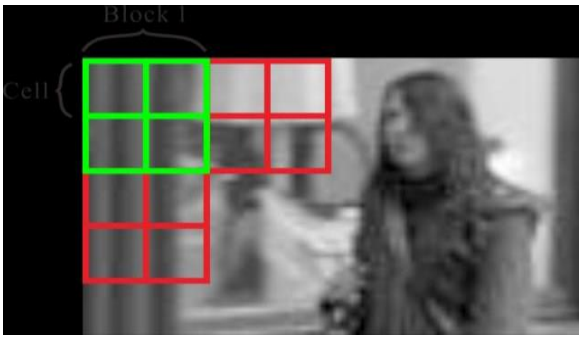

Figure 3(a): Cell and Block

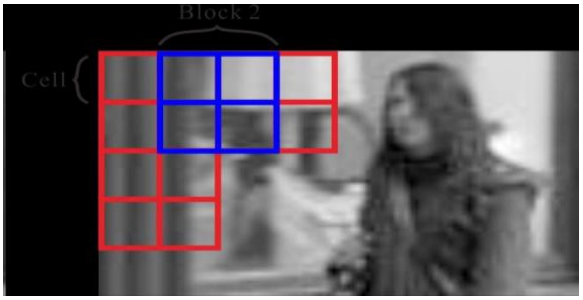

Figure 3(b): Cell and Block (Overlapping)

2.1 Pedestrian Detection Using HOG:

In this section, we describe how to train a linear detector and use the detector to detect pedestrian within a given image.

a) Train Linear Detector:

We have a set of training images, which is divided into positive samples and negative samples. The positive samples are the images containing only one pedestrian, while the negative ones contain no human. Fig. 4(a) and 4(b) shows the positive and negative samples.

Figure 4(a): Positive samples

Figure 4(b): Negative samples

We resize the sample to fixed size of i.e. 128 x 64. This size is applied to the training samples in the implementation later. Then we will extract HOG descriptor for each training sample. All the HOG descriptors are fed into two-class linear SVM to train human detector. In practice, we reshape the detector to a 3D matrix, for the convenience of detection in the following steps.

b) Sliding Window Strategy:

Given an image, we must detect the pedestrian(s) from it. In this step, firstly we will obtain the cells and blocks with the image. Image is converted into grayscale and then divided blocks and cells. We can extract the HOG descriptor for each block and then form a 3D matrix. In the above step, we train a linear pedestrian detector. Then we will convolve the detector on the 3D. We enforce a threshold to select the position where the score of convolution is above the threshold as hypothesis. In this project we are using OpenCV function “filter2D” to implement the sliding window strategy.

c) Multiple Scale Manner:

Practically, the scales of human vary. However, the scale of the detector is fixed, which makes it difficult to detect the human on other scale. For better performance, we perform detection on multiple scales of a given image as shown in Fig. 5. To achieve detection on multiple scales of images, we simple resize the image by several fixed

factors, e.g. 0.21,0.27, 0.3. The threshold for these scales are set to 3.0,3.5 and 4.5 respectively.

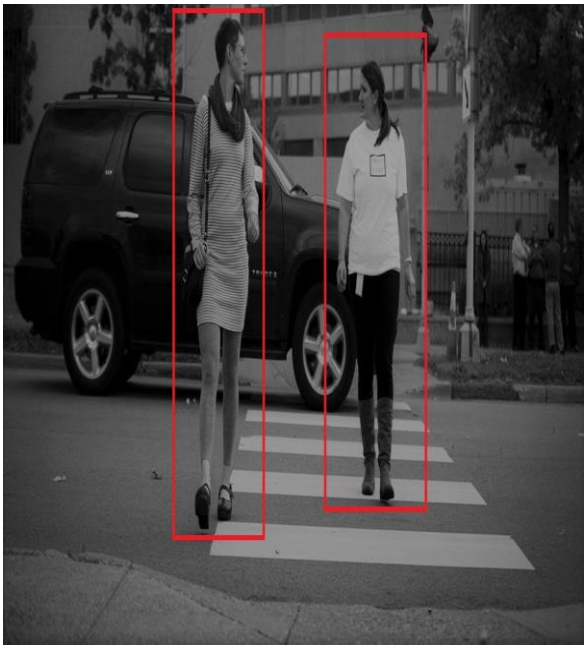

Figure 5: The sample result of detection

2.2 Support Vector Machine (SVM)

Support Vector Machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

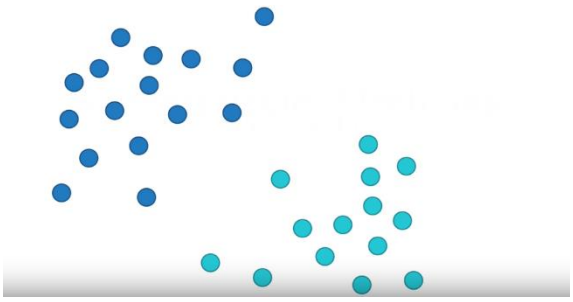

Figure 6: Data set before applying SVM

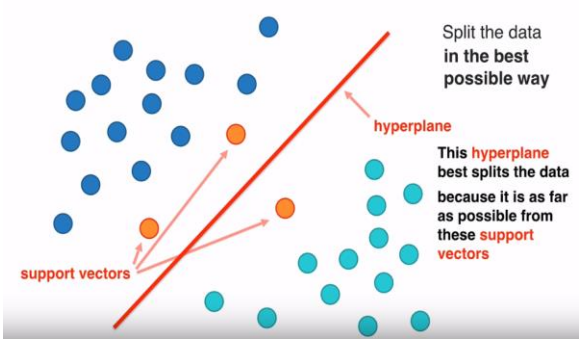

Figure 7: Data set after applying SVM

Suppose some given data points, each belong to one of two classes, and the goal is to decide which class a new data point will be in. In the case of support vector machines, a data point is viewed as a p - dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p - 1)$ dimensional hyperplane (a **hyperplane** is a subspace whose dimension is one less than that of its ambient space. If a space is 3-dimensional then its hyperplanes are the 2-dimensional planes). This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So, we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the ***maximum-margin hyperplane*** and the linear classifier it defines is known as a ***maximum margin classifier.***

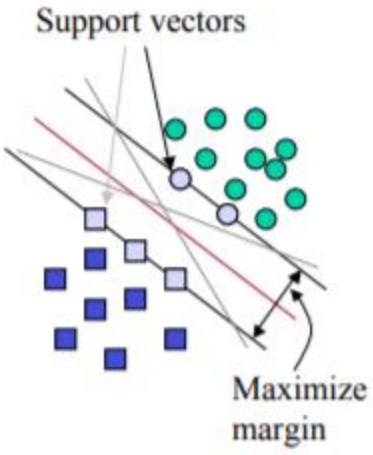

Figure 8: SVM with maximize margin

Linear SVM is the newest extremely fast machine learning algorithm for solving multiclass classification problems from ultra large data sets.

Features of linear SVM:

1. Efficiency in dealing with extra-large data sets.

2. Solution of multiclass classification problems with any number of classes.

3. No need for expensive computing resources.

4. Ideal for contemporary applications like e-commerce, text classification, banking service and many more.

3. Implementation Details

Code is implemented in Python 3.6.5

The implementation is as follows:

3.1. Compute angle and magnitude of gradient for each pixel (function: ExtractHOG):

For each pixel in the image, you should firstly compute the gradient vector, angle and magnitude as described in Section 2. Note that the range of angle should be from $-\pi$ to π.

The vertical and horizontal gradient can be computed by convoluting the kernel [1,0, -1] and [-1,0,1] on the input image, using the OpenCV function "filter2D". The angle can be computed by the math function "atan2", which returns the angle exactly ranges from $-\pi$ to π.

3.2. Construct HOG for each cell (function: ExtractHOG):

We fix the size of cell to make it cover 8x8 pixels and the bin of HOG to be 9. Before we segment the cells, we have padded the image to make its height and width to be the multiplication of 8. Our task is to construct the HOG for each cell within the input image. To achieve this goal, you should first quantize the range of angle ($-\pi$ to π) to 9-bins. And then, accumulate the pixel's magnitude to the 9-bin histogram according to its angle. For each cell, you can construct a 9-bin histogram.

3.3. Construct the normalized HOG for each block (function: ExtractHOG):

We fix the size of block to make it cover 2 x 2 cells. Note that the neighboring blocks overlap with each other by 50%. To construct the HOG of a block, we are to concatenate the 4 HOGs of the 4 cells within the given block, and then normalized the concatenated vector by its L2-norm. The OpenCV function "normalize" could be used to normalize a vector.

3.4. Convolute the learned detector on the HOG descriptor of the novel image (function: TrainMultipleComponent):

After all the implementations required in TODO#1-3, we can extract the

HOG descriptor from the novel image. Then our task is to train a detector. We firstly set the number of components e.g. n. Then apply k-means to cluster the HOG descriptors of positive samples into n-clusters.

3.5. Select the convolution above the threshold and compute its bounding box (function: TrainMultipleComponent):

After step - 4, we have n clusters. In this TODO, we are going to train n detectors.

3.6. Build image pyramid(function: MultiscaleDetection):

In this step, you are required to build an image pyramid. The pyramid contains the same image in different scales. The OpenCV function "resize" is well-designed for this purpose.

3.7. Perform detection on multiple scales of image (function: MultiscaleDetection):

After obtaining the image pyramid, our task then is to perform the detection on the images in the pyramid, by calling the implemented function in Step - 6. Please remember to recover the coordinate of hypothesis on different scales to the original one, by multiplying the inverse of corresponding scales. Fig 5 shows examples of the detection.

4. Dataset:

We provide two set of samples for training and validating respectively. The training set consists of 2,416 pedestrian images (positive samples) and 6,090 non-pedestrian images (negative samples).

5. Problems Faced During Implementation:

To successfully detect all the pedestrians in every image in valid set, I must set a low threshold which means I suffered a high false positive rate, consequently, there were many mistaken bounding boxes in my result. To modify it, I try to set a higher threshold to have lower false positive rate, while my project fails to detect any human in some validation images, the number is 4. This time, the results are listed below. It seems it performs better.

6. Discussion of the obtained results:

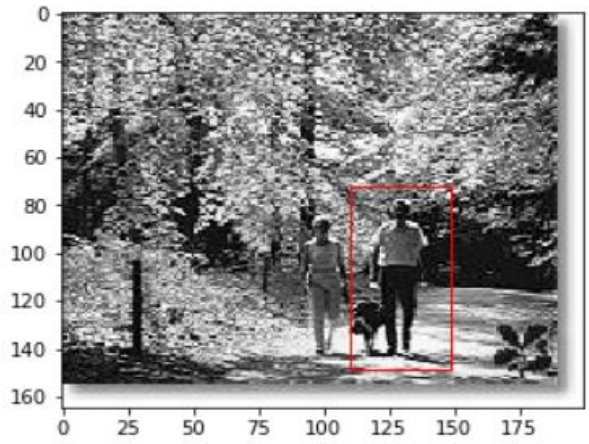

Figure 9(a)

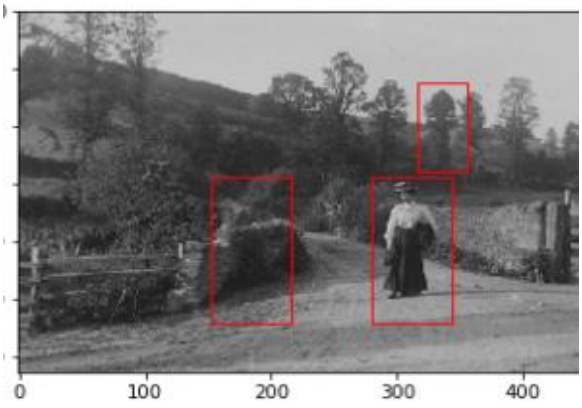

Figure 9(b)

- In order to successfully detect all the pedestrians in every image in valid sets, we have to set a low threshold, which means, we suffered a high false positive rate as shown in the figures 9(a) and 9(b).

- Consequently, there were many mistaken bounding boxes in the result.

- In order to modify it, I try to set a higher threshold in order to have lower false positive rate, while my project failed to detect any human in some validation images.

- Later, we set it to a moderate threshold where it helped by performing better.

- We are analysing the performance of pedestrian detection by plotting a graph using “average precision” and result is saved as “averagePrecision.png”.

- We are using averagePrecision sum =1:x of (precision at i * change in recall at i) ” formula to find the average precision, where x is number of samples on which pedestrian detection is tested.

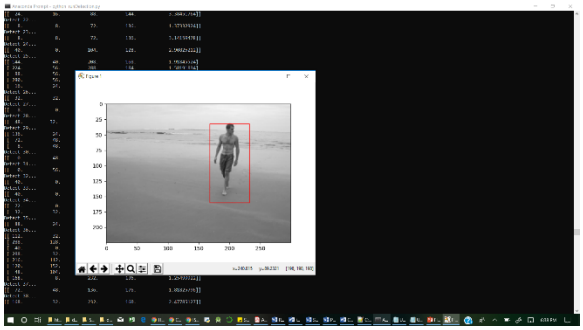

Figure 10(a)

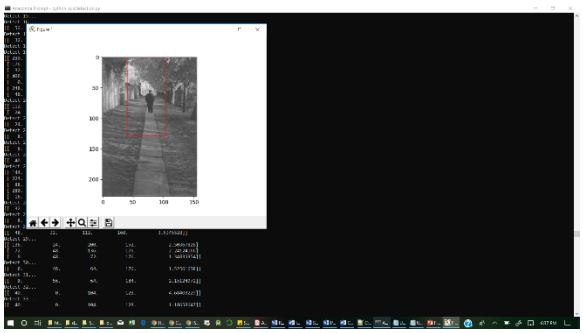

Figure 10(b)

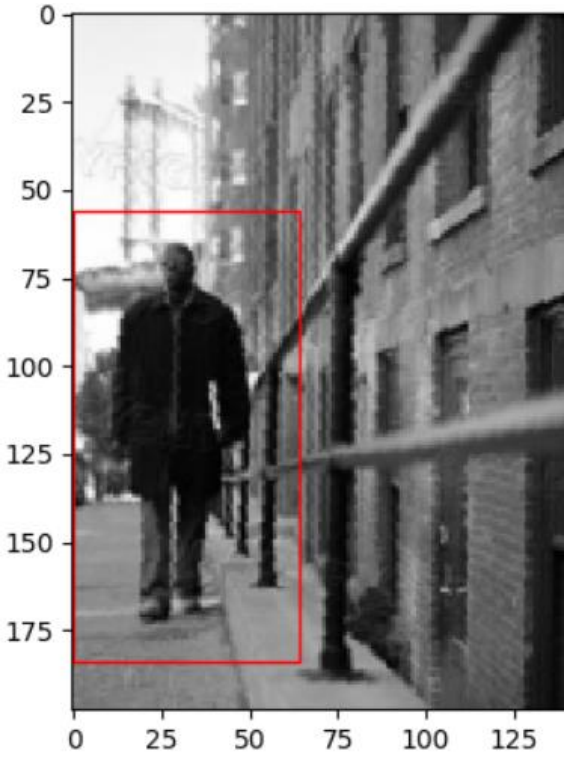

Figure 10(c)

7. Final Result:

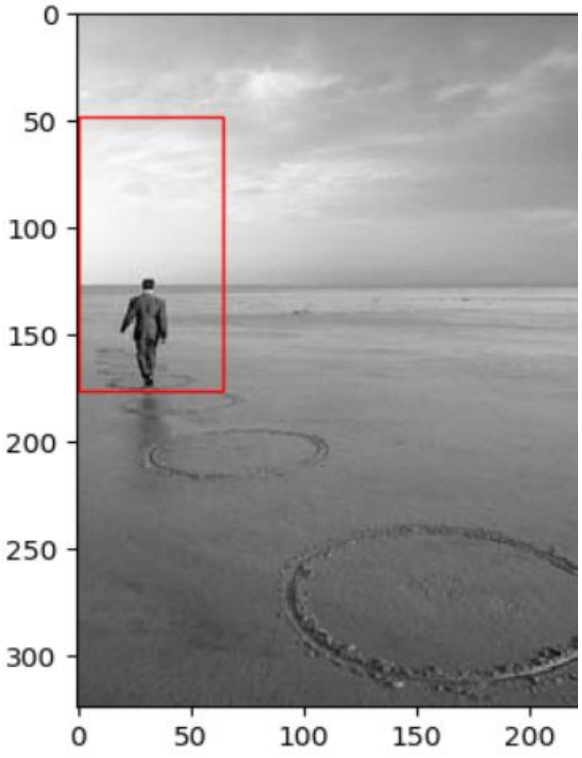

Figure 10(d)

Figure 10(f)

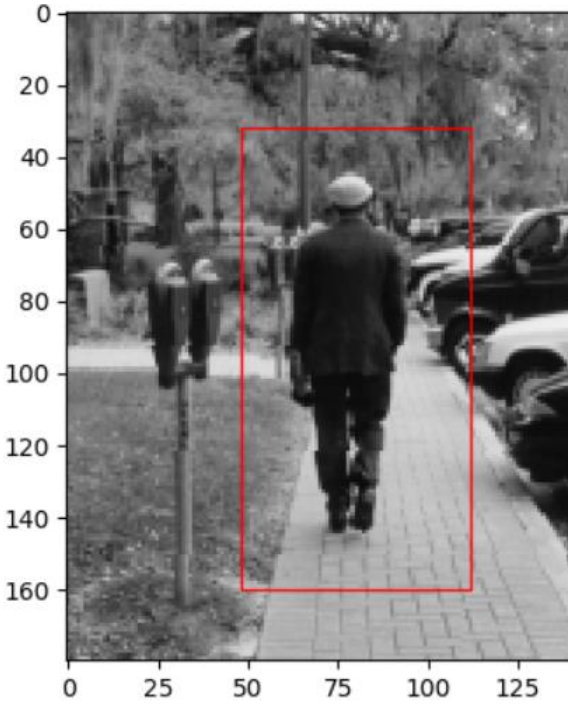

Figure 10(e)

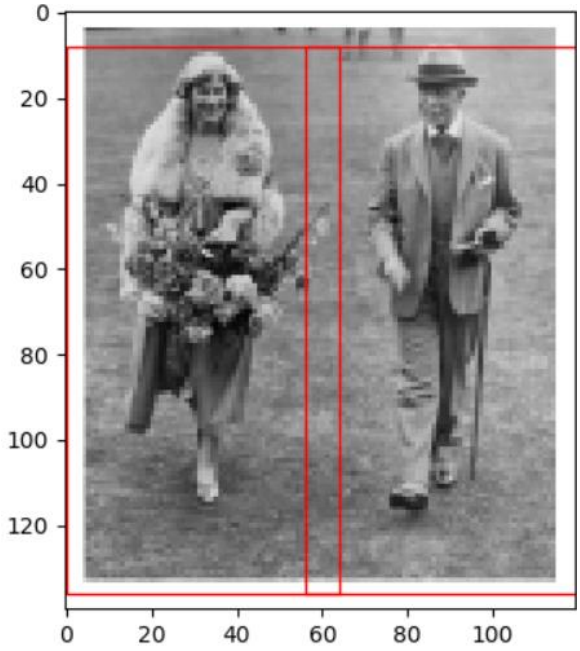

Figure 10(g)

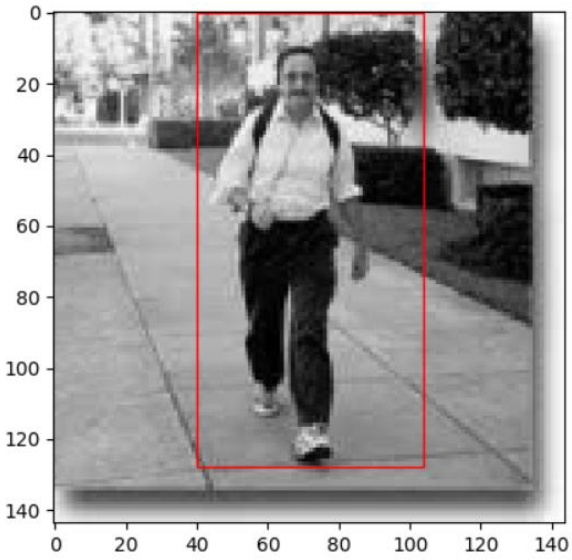

Figure 10(h)

https://ieeexplore-ieee-org.ezproxy.gl.iit.edu/document/6642755

3.) N.Dalal and B.Triggs:

Histograms of Oriented Gradients for Human Detection, CVPR, 2013

8. References:

1. Pedestrian detection: https://en.wikipedia.org/wiki/Pedestrian_detection
2. HOG introduction: http://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
3. Support vector machine:

   https://en.wikipedia.org/wiki/Support_vector_machine

   https://www.youtube.com/watch?v=N1vOgolbjSc

Papers used as a reference:

1.) Paper -1: https://ieeexplore-ieee-org.ezproxy.gl.iit.edu/document/7044743

2.) Paper-2: