HUMAN DETECTION IN IMAGES

TANVI RAUT

Introduction:

This project involves automatically locating humans in image or video. Human Detection comes under Object Detection. This project could also be considered as Pedestrian detection. Detecting humans becomes one of the challenging task in Computer Vision as it involves detecting human in different poses, at different scale space, with different style of clothing, different complex backgrounds and with partial occlusions. Also the illumination has a huge impact. Best detector would be the one giving best results irrespective of the variance. There are several algorithm being used since a decade to obtain accurate results.

This project will involve experimenting using Histograms of Oriented Gradients (HOG) descriptors [1]. A linear Support Vector Machines to classify each possible image region as a human. SVM is a type of machine learning algorithm used for classification. This project is an implementation of “Histograms of Oriented Gradients for Human Detection” by Navneet Dalal and Bill Triggs [2]. Their work has given impressive results. This concept has several applications such as automation of car systems, security, surveillance and civilian applications. Significant research is going on in the field of detecting human in images and video with much higher accuracy.

Previous work:

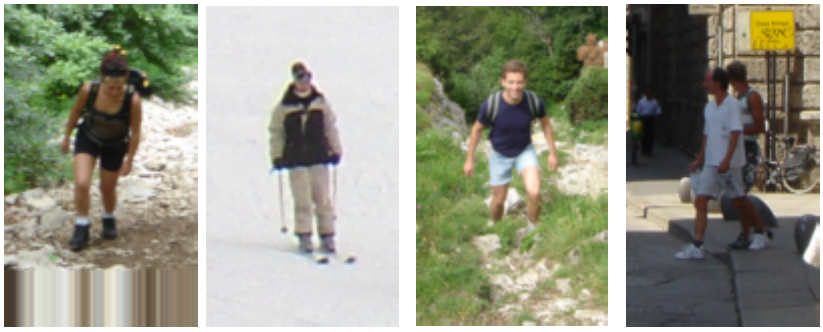
Human detection comes under the category of object detection and object recognition. One of the previous work done with object recognition was using Scale-Invariant Feature Transform algorithm done by David Lowe in 1999 [3]. In that algorithm, the given image is converted into a large collection of local feature vectors which are invariant of the image transformations. The second method was using HAAR wavelet transform along with SVM by Papageorgiou. In this algorithm, train a support vector machine classifier using a large set of positive and negative examples [4]. However, HAAR lacks in orientation information about the edges. Viola- Jones [5] built an algorithm that takes into account two consecutive frames of a video and trains the detector using Adaboost(which is a machine learning algorithm) to take advantage of both motion and appearance information. It uses HAAR-wavelet features. Training is slow but the detection was fast. Another method was using the edge templates and match them to the learned examples by Gavrila and Philomen[19].This was mainly modeled for car detection. Zhao et al [6] detects real-time pedestrians using a neural network that is trained on human silhouettes and stereo-based segmentation. Pedestrian detection using Kalman filtering along with algorithms of edge density and symmetry maps was done by M. Bertozzi, A. Broggi, A. Fascioli, A. Tibaldi [7].

Another method is using HOG descriptors Navneet Dalal and Bill Triggs [2]. They use these features to train a soft linear SVM classifier. HOG based features were capable

of capturing the pedestrian outline/shape better than HAAR wavelet [8]. SIFT and HAAR would be better for other object detections, but for human detection HOG features appeared more suitable. HOG is kind of successor of SIFT features. But HOG with SVM gives better performance when compared with SIFT [5]. HOG uses global features whereas SIFT uses local features. SIFT computes the gradient histogram only for patches(works as patch descriptor) whereas HOG computes gradient for the entire image. Both have their different applications. HOG was initially used for human detection, but now it could be used for many object detection research and it was found it works well from them as well. I will be using the HOG Descriptor algorithm along with linear SVM to train the model and then classify.

Dataset:

We need positive images which are labeled data and the bounding boxes. We use a lot of training data. The dataset used contains partial data images from the INRIA data set containing 1805 64 by128 images of humans cropped from a varied set of personal photos. People are upright and fully visible in these images by with different poses. I have taken 1484 images for training, out of which 1392 are positive images containing humans and 392 are negative images which does not contain any upright and fully visible human. The positive images contain the left-right images of 696 images, so total of all adding up to 1329 positive images. I am using some of the test images from the original images to test on the implementation. The dataset also contains the annotations as the ground truth. The annotations were written in Pascal , so I have used as CSV version of those annotation found on web [9].
Below are some images used to train the detector:



Overview of HOG descriptor algorithm:

This is a data-driven algorithm. When we say that HOG uses global features, it means that the entire human is represented using a single feature instead of using multiple local features to represent each part [9]. The feature descriptor produces a general feature vector of the person in such a way that the person produces the same features under variations. Features are used instead of pixels as they convey more

information about the texture, segmentation, etc.

This method uses the technique of sliding window. For the Validation images, at each of the location where we place the sliding window, HOG descriptor is found for that window. These descriptors are then send to the trained SVM to learn on them, which returns the classification as 1 or 0, since I am using a binary classification. The validation image is sub sampled to multiple sizes in-order to detect a human at different scales. On detection, a rectangular window is drawn around it. The figure shown below summarizes the entire process, Figure 1.

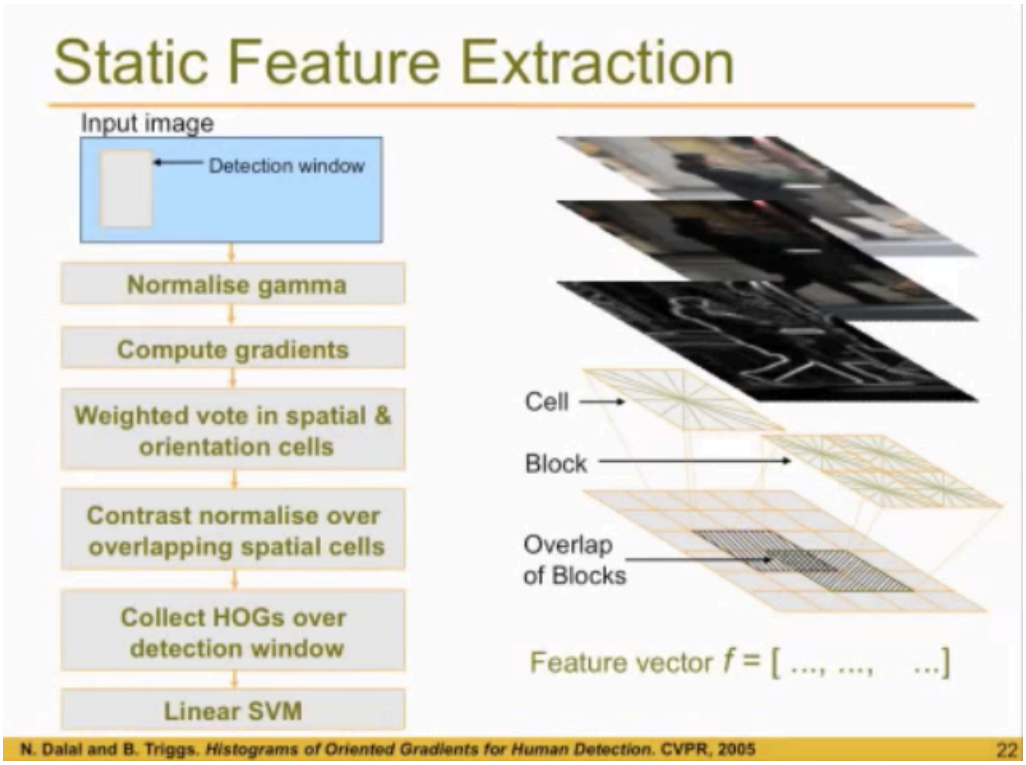


Figure 1. [20]

Implementation of the Algorithm [1,9]:

To start with, we take all the training images to train out detector first. The training image is of the size 130 by 66 pixels that is 128 x 64 with a 1-pixel border for computing the gradients at the edges. Using a window of 128 x 64 gives better result as shown below in figure 2. We convert the image into grayscale image.

1. **Gradient Computation:**

The image is divided into small spatial regions called cells. Each cell is of 8 x 8 pixels. These cells are organized into overlapping blocks. There will be 50% overlapping between the blocks. Each block will be of 16 x 16 pixels that is 2 x 2 cells. Thus, the window will be made of 8 horizontal cells and 16 verticals cells or we could say 7

horizontal blocks and 15 vertical blocks. Then for each of these cells we compute gradient vector. Gradient is the measure of change in the pixels along x and y direction around each pixel considered. Let's consider one pixel. Below is the illustration for the pixel in Figure 3.

The blue box represents the pixel currently considered. So now the rate of change in x direction from left to right will be 94 – 56 = 38, we moved from dark to light. Similarly, the rate of change in y direction from top to bottom will be 95-65 = 30. So now the gradient

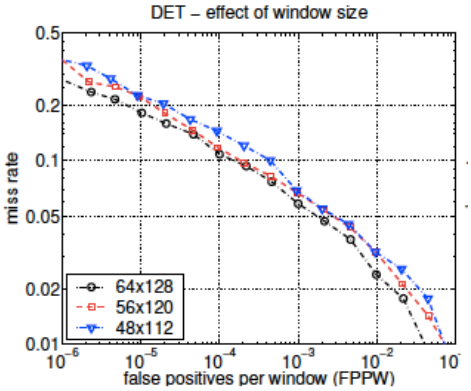


Figure 2. [1]

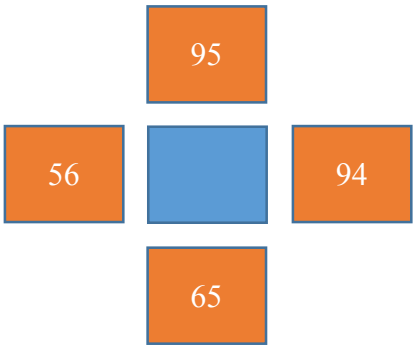


Figure 3

vector will be $\begin{bmatrix} 38 \\ 30 \end{bmatrix}$. This is nothing but simply using a 1-D mask [-1, 0,1] and its transpose.

Using larger mask decreases the performance according to the paper. Also no smoothing scale was applied. We then find the gradient magnitude and the angle/orientation at each cell. After finding the angle, we can then draw the gradient vector as an arrow with the angle found at that pixel. The larger the gradient the stronger are the edges.

2. Spatial Orientation/Binding:

In this step, we compute the histograms for each cell in the image. For each cell, we get 64 gradient vectors (8 by 8). We put these into histograms of 9 bins per cell. Using 9 bins significantly improves the performance as shown by Figure 4. The orientation bins are evenly spaced over 0 to 180 degrees of unsigned gradient. Unsigned, because we make all the negative angles positive by adding 180 degrees to it. Including the signed gradient was seen to decrease the performance. So now we will have 9 bins each of 20 degrees(180 divided by 9) .

We find the weighted vote for each histogram. The weighted vote will be nothing but the gradient magnitude at the pixel, so stronger the gradient magnitude, bigger will be the impact on the histogram. Using magnitude itself gives better results that taking the square or square root of it. The gradient angle for each pixel will fall between two bin centers. Each gradient vector's contribution is divided between the two neighboring bins, in proportion to the distance between the two neighboring bin centers that is, the magnitude will be split between the bin to the left and the bin to the right based on how far the angle is from the bin centers.

Converting the gradient value of each cell into histogram is a type of quantization where we quantize 64 values into 9 bins. In this way we get, 4 cells x 9 bins = 36 values

for each block. Thus, for total of 105 blocks (15 x 7 blocks) we get 3,780 values in the final vector. After this step, we perform block normalization on each blocks.

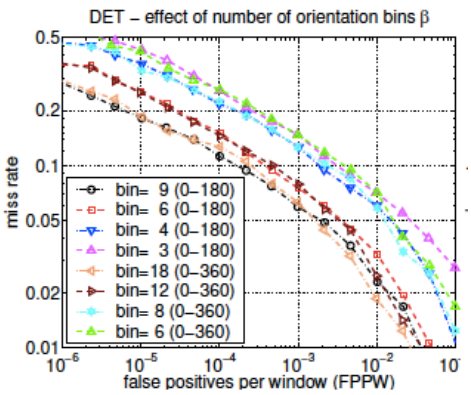


Figure 4.

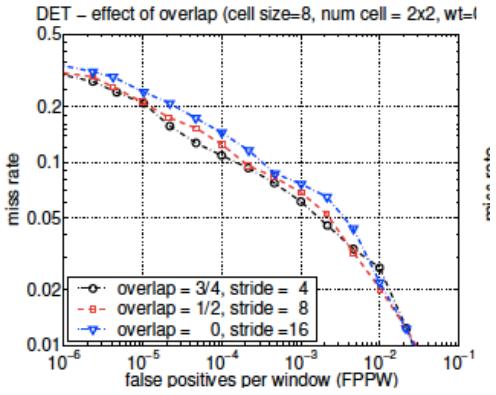


Figure 5.

3. Normalization and Descriptor blocks:

Normalization provides some invariance to changes in contrast, which can be thought of as multiplying every pixel in the block by some coefficient. We normalize the blocks to make it multiplication invariant that is, even if we multiply each pixel by certain amount, we should still get the same gradient vectors at every pixel.

When we multiply all the pixel value in the image by certain amount (say e.g. 2.5), we basically make the bright pixels became brighter while dark pixels only became a little brighter, thereby increasing the contrast between the light and dark regions of the image. Now, the problem here is, after multiplying by a factor, the gradient vector magnitude also increases by that factor. This means that the value of each bin of the histogram for that cell will also be increase by that factor (in our case by 2.5x). In order to make it multiplication invariant, we divide the divide the gradient vectors by their magnitude. This method of dividing a vector by its magnitude it referred as normalization of that vector.

So we take a block of 2 x 2 cells and normalize their histograms. Then get the histograms for the cells in that block. Place all the histogram values into a single vector and then compute its magnitude. Divide all the histogram values by this computed magnitude to normalize them. Thereby, normalizing the histograms, we now make it invariant to the change in illumination.

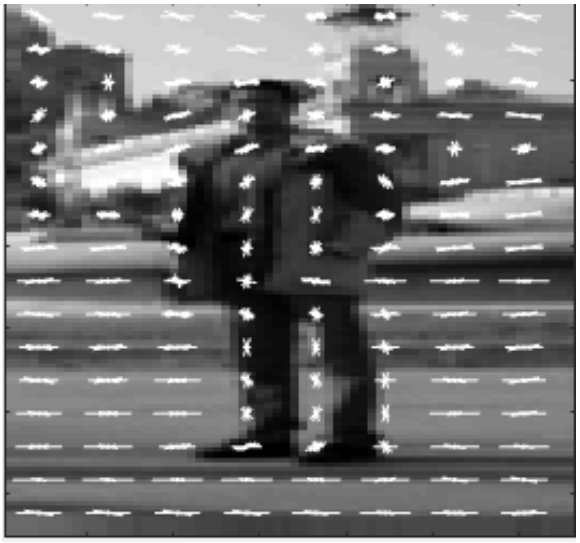
As we know, there will be 50% overlap between two consecutive blocks. This overlap is done so that each cell contributes several components to the final descriptor vector, each normalized with different blocks [1]. This may seem that each cell will appear multiple times in the final descriptor vector but good normalization is important and including overlap significantly improves the performance as seen above in Figure 5. Various block normalization schemes are presented in the paper by Dalal and Triggs namely L2 norm, L2-hys, L1 norm and L1 sqrt [12].

4. Detector window and Context:

So now, our final descriptor will have 3780 values for the detection window of 128 x 64 pixels. In Matlab, we have an inbuilt function to obtain this HOG final descriptor, '`extractHOGFeatures(I)`'. It returns the extracted HOG features for the given grayscale or colored image I. The features are returned in 1 by N vector [13]. This function also returns 'visualization', that could be used in plotting the descriptors. You can specify the size of cell in that function. In our case I have used an 8 by 8 cell.

The input and output image after extracting HOG descriptors are presented below: The input image shown below is a 130 by 66 pixels' window from the test data set of positive images

Input image



Output image

5. Classifier:

We will be using a soft linear Support Vector Machine [14]. SVM are supervised learning models with learning algorithms used for classification. Since, our data has only two classifications i.e. 'person' or 'not a person', we will be using binary classification. "An SVM classifies data by finding the best hyperplane that separates all data points of one class from those of the other class." [15]. With the help of the training data set, we train the SVM model. There is a function to train SVM in matlab namely, '`fitcsvm(X,Y)`', where X is Predictor data and Y are the class labels [16]. In our case, Y will contain '0' for the negative data and '1' for the positive data. Length of Y has to be equal to the length X. In our case, X will contain all the feature vectors. This function will return the SVM classifier model. This model is saved for the purpose of testing.

While testing the data, we use the function named '`predict(SVMmodel,X)`'[17]. This method predicts the classification labels 'Y', given the trained model and the dataset.[18] In our case, SVMmodel will be the one we saved after training and X will be the feature vector for the test data obtained after performing HOG on the test image. On the basis of the labels obtained, we could then classify the test image whether it has or does not have a person in the given test image. For testing purpose also, we are using

an image of size 130 by 66. So, out test program will only find detect the presence of human in the given test image. The test image is given as input, then we extract the HOG descriptor vector and then this descriptor vector is sent to the classifier that predicts the classification given a pre-build trained model.

6. **Testing and detecting humans in typical image**:

For a typical image (like 640 by 480 pixel), we can get 10k to 200k windows. There might be multiple people in the given image. In this step, we will not just detect the presence of human in the image but also detect the location and plot a bounding box around the human detected. We send a test image of any size to a pre-trained HOG detector model for the purpose of validation. Then convert it into gray-scale image. Now out task is to find or location the human. But the human could be present at any scale. This is the scale space issue. To resolve this problem, we perform computations at different scales. However, the HOG descriptor window is never resized, instead the image is resized for every computation. We rescale the image many times. Each image scale is 5% smaller than the previous one. We repeat this until the resulting image is too small to fit even a single descriptor. For each scale the image is down sampled and then HOG descriptor vectors are computed for every detection window in the resized image.

We basically apply the sliding window technique. As a detection window, we again take a 128 by 64 window. We start with placing the window on the top left corner of the image. Now for this window we extract the HOG descriptors feature. The HOG descriptors are fed to the linear classifier to determine whether their detection window contains a person. If the classifiers prediction is greater than the threshold, then the window contains a person. The threshold value is computer based on multiple observations. Once the window is detected with presence of human, then we find the co-ordinates of the top left corner of this detection window. Then compute the detection window co-ordinate and size relative to the original image that is before scaling down. We save the co-ordinates and the size (width and the height) of the detection window.

After this, slide the detection window horizontally by 1 cell and repeat the process again. At the end of this process, we will have all the detection windows containing the presence of human. Remember, even the presence of any body part of the human would be considered. So we will have multiple detection windows per person. As we scan the image at multiple scales, we would find multiple overlapping detections We then plot the boxes around them. To limit the number of boxes around a single person, I have used a threshold. This is done by taking the centroid of every detected window and then keep only those which are in same cluster (closely grouped) and discard the outliers and those more than the threshold. This technique is similar to mean shift.

The following figure shown below shows the output:

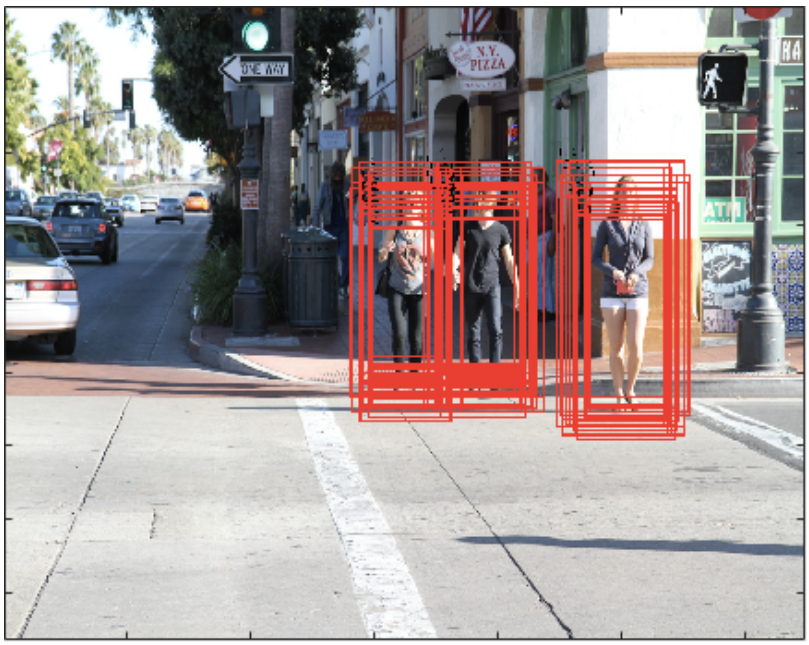


Figure 6. Output of my implementation

As seen from the output, there are multiple overlapping detections. There was one of the approach of using heuristic method for combining the multiple detections [23] to solve this problem. However, while implementing this technique, it still gave me two-three detection window per human detected depending on the image used. Another technique mentioned by Navneet Dalal in his thesis was non-maximal suppression which would give better results. In non-maximal suppression, the detections are represented in scale pyramids [11]. In his thesis, he mentioned non-maximal suppression technique as "*Each detection provides a weighted point in this 3-D space and the weights were the detection's confidence score. A non-parametric density estimator is run to estimate the corresponding density function and the resulting modes (peaks) of the density function constitute the final detections, with positions, scales and detection scores given by value of the peaks. We will call this process as non-maximum suppression."* [11]. However, this technique was difficult to understand and implement.

The complete algorithm is as follows [11]:

Input:
(a) Test image
(b) Trained window classifier with normalised window of width W_n and height H_n
(c) Scale step size S_r, stride for spatial scan N_s, and sigma values σ_x, σ_y, and σ_s
Output: Bounding boxes of object detections

Initialise
(a) Let start scale $S_s = 1$; compute end scale $S_e = \min\left(W_i/W_n, H_i/H_n\right)$, where W_i and H_i are image width and height, respectively
(b) Compute the number of scale steps S_n to process

$$S_n = \text{floor}\left(\frac{\log(S_e/S_s)}{\log(S_r)} + 1\right)$$

For each scale $S_i = [S_s, S_s S_r, \ldots, S_n]$
(a) Rescale the input image using bilinear interpolation
(b) Extract features (Fig. 4.12) and densely scan the scaled image with stride N_s for object/non-object detections
(c) Push all detections with $t(w_i) > c$ to a list

Non-maximum suppression
(a) Represent each detection in 3-D position and scale space $\mathbf{y}_i$
(b) Using (5.9), compute the uncertainty matrices $\mathbf{H}_i$ for each point
(c) Compute the mean shift vector (5.7) iteratively for each point in the list until it converges to a mode
(d) The list of all of the modes gives the final fused detections
(e) For each mode compute the bounding box from the final centre point and scale

Discussion:

The important parameters to be considered while implementation are: Gradient scale, Number of orientation bins, Percentage of block overlaps, the color space to work on and then the normalization technique. All these parameters play an important role in the performance of the algorithm. The current implementation is slow due the in-built SVM classifier used, which is very slow. According to me, using Adaboost classification would have boosted the speed of computation as well. The training accuracy was found to be 100%, the testing accuracy was approximately 84%. The accuracy could be improved by combining HOG with other algorithms like HAAR cascade or Local Binary Pattern [21]. The vision toolbox named vision.PeopleDetector implements the entire algorithm correctly including the non-maximal suppression to obtain exact window blocks for detected humans [24].

Not using blocks with multiple scales is one of the factor missing in the approach used in this paper [22]. I have also observed that if we perform any kind of smoothing on the image before calculating the HOG descriptors, then it gives incorrect results. So

basically, edge plays the key role in this method.

Conclusion:

This approach worked great for upright and non-occluding images, especially if the two humans are significantly distant from each other. With further improvements in the accuracy and the time taken, this method is very useful for real-time object detection. With this algorithm, I have learned the importance of features in any image. After implementing this method, it encourages me to do more research and implementation work towards object detection. This method could also be extended to detect humans in Video, considering each frame as one image and then applying the same approach. Navneet Dalal has described about human detection in video in his thesis.

References:

[1]. Dalal, N. & Triggs, B (2005) Histograms of oriented gradients for human detection Computer Vision and Pattern Recognition, 2005. Computer Vision and Pattern Recognition 2005. IEEE Computer Society Conference.
[2]. Dalal, N. & Triggs, B. & Schmid, C. (2006) Human detection using oriented histograms of flow and appearance. European Conference on Computer Vision.
[3]. Lowe, D.G. (1999) Object recognition from local scale-invariant features.
[4]. C. Papageorgiou and T. Poggio. A trainable system for object detection. International Journal of Computer Vision., 38(1):15.33, 2000.
[5]. P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. The 9th ICCV, Nice, France, volume 1, pages 734??741, 2003
[6] L. Zhao and C. Thorpe. Stereo-and neural network-based pedestrian detection. ITS, IEEE Transactions on, 1(3):148{154, 2000.
[7]. M. Bertozzi, A. Broggi, A. Fascioli, A. Tibaldi, R. Chapuis and F. Chausse, “Pedestrian localization and tracking system with Kalman filtering”, in Proc. IEEE Intelligent Vehicles Symposium, pp. 584-589, June 2004.
[8]. Ish Rishabh and Arjun Satish, “Human Detection in RGB Images,” University of California, Irvine
[9]. http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/ . Date: November, 26th 2016.
[10]. Dalal, N. (2005) INRIA Person Dataset http://pascal.inrialpes.fr/data/human/
[11]. Dalal, N. (2006) Finding People in Images and Videos. Ph.D. Thesis.
[12]. D. G. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 60(2):91.110, 2004.
[13]. https://www.mathworks.com/help/vision/ref/extracthogfeatures.html Date: November 27th, 2016.
[14]. Christianini, N., and J. C. Shawe-Taylor. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge, UK: Cambridge University Press, 2000.

[15]. https://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html Date: November 27th, 2016.
[16]. https://www.mathworks.com/help/stats/fitcsvm.html#inputarg_X Date: November 27th, 2016.
[17]. https://www.mathworks.com/help/stats/compactclassificationsvm.predict.html Date: November 27th, 2016.
[18]. Platt, J. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods." In Advances in Large Margin Classifiers. MIT Press, 1999, pages 61–74.
[19]. D. M. Gavrila and V. Philomin. Real-time object detection for smart vehicles. CVPR, Fort Collins, Colorado, USA, pages 87.93, 1999.
[20]. http://pascal.inrialpes.fr/soft/olt/
[21]. X. Wang, T. X. Han, and S. Yan. An HOG-LBP human detector with partial occlusion handling. In Proc. of ICCV, 2009. 5, 7
[22]. Q. Zhu, M.C. Yeh, K.T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In CVPR, 2006.
[23]. H.A. Rowley, S. Baluja, T. Kanade, Neural network-based face detection ,In IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 20, Issue: 1, Jan 1998)
[24]. https://www.mathworks.com/help/vision/ref/vision.peopledetector-class.html Date: November 27th, 2016.