

# Object Detection On Android Smartphones

Shantanu Kamath

School of Computer Science and Engineering,  
Nanyang Technological University

Assoc Professor Felix Klanner  
Alexander Hanel

Supervisors  
Future Mobility Research Lab,  
BMW@NTU

**Abstract** – In this paper, the ability of robust pedestrian recognition on an android device is addressed, this is done by adopting Histogram of Oriented Gradients (HOG) Descriptors and LibLinear classification library. HOG descriptor is used to process images and extract a feature vector that contains information on the brightness of each pixel present in the image. This feature vector can then be used to determine the presence of a pedestrian once it is supplied to LibLinear. LibLinear provides the necessary support vector machine (SVM) required to perform such classification. 2000 labeled images were extracted from KITTI dataset to train a classifier model while another 2000 images were used for testing and improving the model. The image processing, description and classification were implemented in C++ while the Native Development Kit (NDK) of android was used to support the reuse of this code in the application. The application has the functionality to allow the use the rear camera of an Android smartphone to click an image in real time or upload an existing image from the storage of the device to perform pedestrian detection on the image. The result are one or more rectangles plotted into the image showing the position of detected pedestrians.

**Keywords** – Android Smartphones, Object Detections, Pedestrian Recognition, HOG Descriptors, LibLinear, SVM, HOG-SVM.

## 1 INTRODUCTION

Pedestrian detection is an arduous task owing to the difference and variety in postures and appearances humans can adopt. Existing technologies have been implemented on high power computing devices mostly. The ability of such a system on an android mobile operating system enhances its usability owing to the fact that it can now be supported by the operating systems present in vehicles.

This paper describes the use of HOG descriptors and the SVM classifier for the purpose of pedestrian detection. A software was developed consisting of a backend and frontend part. The backend is where the image is processed and classified while the frontend is handles the user interface and access to camera. The detection of pedestrians is to be done on a per image basis and the positive results would be displayed with a highlighted rectangle. The front end involves java programming to build a native android [16] application to access the smartphone camera as well as storage to find an image.

KITTI dataset and labels [3] [13] were used as the only source of images to train the classification model. Further images were taken using the rear camera of a Samsung Galaxy S6 for the main testing phase. An assumption made during the detection phase is that all pedestrians would assume a standing posture.

In the following topics, terms like descriptor mean algorithms or applications that produce descriptions in terms of shape, size, color or motion. Another important term would be classifiers, which would use descriptions provided to it and classify it under categories based on a machine learning algorithm using support vector machines (SVM).

Further sections briefly discuss previous work on human detection in §2, explain the HOG method for descriptors and SVM classification in §3, give a description of our dataset in §4 and discuss the results in §5. The summarized conclusion is included in §6.

## 2 PREVIOUS WORK

There is a broad range of literature research available on the topic of Pedestrian detection and its methods, but here only a few important relevant ones are listed.

Ronfard et al [4] fused support vector machine based limb classifiers over first and second order Gaussian filters in a dynamic programming framework to construct a detailed body detector like those of Felzenszwalb and Huttenlocher [5] and Ioffe and Forsyth [6]. Gavrilu and Philomen [7] took a more straightforward method involving mining for edge images and coordinating them to an arrangement of scholarly models utilizing chamfer distance. This has been utilized as a part of a viable real-time pedestrian detection frameworks [8].

Papageorgiou *et al* [9] used rectified haar wavelets to depict a detector in view of a polynomial support vector machine with a parts variation in [10]. A mix of orientation-position histograms with binary-thresholded gradient-magnitudes was used by Mikolajczyk et al [11] to construct a sections based methodology containing identifiers for heads, faces and side and front profiles of lower and upper body parts.

However, Dalal & Triggs [1] created a detector that used a simpler architecture with only a single detection window, but gave appreciably superior performance on pedestrian pictures due to its ability to detect pedestrians in even clutter and badly illuminated backgrounds.



Figure 1. Some of the images that were extracted which include the pedestrian is present in the foreground were used in the training of the classifier. The subjects are always upright, but with some partial occlusions and a wide range of variations in pose, appearance, clothing, illumination and background. [3]

### 3 METHODOLOGY

This section describes the approach to detect pedestrians in images taken by an Android smartphone camera.

First, the algorithm behind HOG is described, then the principle of SVM is explained. Afterwards, the integration into an Android app is described.

#### 3.1 HISTOGRAMS OF ORIENTED GRADIENTS DESCRIPTORS

##### 3.1.1 FEATURE EXTRACTION

Features are detected on gray scale images having a dimension of 64 x 128. A pedestrian must be of this minimum size. For pictures larger than the given size, a sliding-window and resizing algorithm was built that moves the 64x128 window across the image horizontally and vertically step by step by 8 pixels. The image would be then resized to 80% of its original size and the screening would continue until the image was ultimately resized to the required dimensions. This ensures that all different sizes of pedestrians are detected.

Each 64 x 128 image is divided into multiple non-overlapping cells. Each cell consisting of 8x8 pixels, therefore there are in 8 columns and 16 rows and a total of  $8 \times 16 = 128$  cells per image as seen in Figure 2. It is important to note here that the size of each cell can be changed, here typical values are used for such an algorithm. For each of the 64 pixels in a cell, compute the centered horizontal and vertical gradients with no smoothing. Using the two gradients, determine the gradient orientation and magnitude for each of those cells. For each cell, quantize the magnitudes and orientations of all pixels within the cell into a cell histogram of 9 gradient orientation bins represented by Equation 1 and shown in Figure 4.

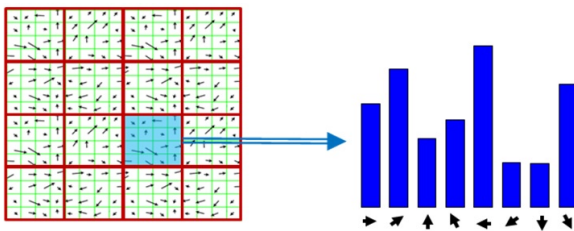


Figure 4: Formation of histograms from pixel magnitude and gradient. [17]

After calculating the above values, start forming blocks. Blocks are a group of 2x2 cells and are 50% overlapping. Hence, there are 7 blocks horizontally and 15 blocks vertically, making a total of  $7 \times 15 = 105$  blocks. The quantized histograms of each cells are added up to form another quantized histogram for the block and then normalized using l2-normalization shown in equation 1. By normalizing the histogram can make it invariant to illumination between different parts of the image.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad |\mathbf{x}| = \sqrt{\sum_{k=1}^n |x_k|^2},$$

Equation 1. Shows the formula for l2-normalisation with n values and |x| as the normalized value. [15] In our case n is 9 because 9 orientation bins are used.

The calculated normalized value is used to adjust each individual cell histogram and finally collated into a vector having  $8 \times 16 \times 9 = 1152$  values. This vector is known as the feature vector for the image patch.

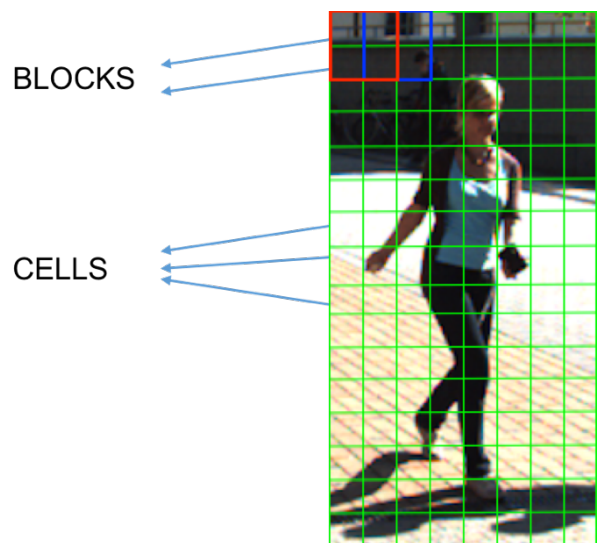


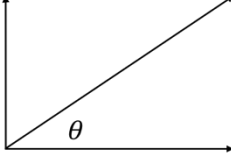
Figure 2. 64x128 image with a cell size 8x8 pixels and blocks of 2x2 cells. [13]

### 3.1.2 COMPUTING GRADIENTS

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) + f(x-h)}{2h}$$

Equation 2. Formula for centered gradients.

$$\text{Magnitude : } S = \sqrt{S_x^2 + S_y^2}$$

$$\text{Orientation : } \theta = \arctan\left(\frac{S_y}{S_x}\right)$$


Equation 3. Formula to calculate magnitude and orientations of each pixel using their centered vertical and horizontal gradients.

The formula in Equation 2 is applied to each pixel in a vertical and horizontal manner to calculate the centered vertical and horizontal gradients. The formula in Equation 3 is applied to get the magnitude and orientation for each pixel. Repeating the above process for each pixel in a cell, can concatenate the magnitudes and orientations to form a cell histogram of 9 bins. These are normalized as mentioned earlier to receive a final feature vector.

### 3.2 LIBLINEAR

LibLinear is an open-source library for large linear classification developed by the machine learning group at National Taiwan University [2]. It implements linear Support Vector Machines (L-SVM) and a coordinate descent algorithm trained regression models. It was written in C++ but can be accessed with a C API.

The model was trained by running the descriptors across a high number of positive images and supply the model with their final feature vectors as a positive label. These descriptions are to help the model identify a pedestrian on a 64x128 image patch. A high number of negative images are sent through the descriptors as well; they serve as negative labels for the model.

These 2000 descriptions are for the trained model to be able to differentiate the two labels. The model can also be tested by running a wide range of images other than the ones used to train to check its accuracy.

A saved model can be finally used by us as a classifier for our new unknown image. The feature vectors of the unknown image are supplied to the model and the SVM processes to return the label and a decision value to inform us how much of a match it is to one of the labels to determine whether the image contains a pedestrian. Using this information and OpenCV library a highlighted rectangle as shown in Figure 3 has been built upon the image to clearly enclose the location of the pedestrian.



Figure 3. 64x128 images with highlighted rectangles enclosing pedestrians. [13]

### 3.3 ANDROID APPLICATION

For the purpose of building an android application and testing an application was built which would grant the user access to the camera as well as the gallery of the device to choose the image to be processed. The application has a simple user interface built using Google's Android Studio and Android Software Development Kit (SDK) [16].

The algorithm, descriptors and classifiers were all built using C++ programming language. As Android works with Java, the whole source code was required to be rebuilt in order to provide its functionality within the application. For this purpose, Android Native Development Kit (NDK) was installed [16]. The Android NDK let the developer implement parts of his application using native languages like C and C++.

The entire LibLinear library, along with the C++ code for the algorithms and descriptor needed to be wrapped by Android NDK to overcome the incompatibility between C++ and Java before any of its functionality could be made available to the application. The wrapped C++ code and functionality provided by the NDK forms the backend of the application. The user would provide an input image to the application and it would provide an output after processing it using the backend functionality. If the output was positive it would display the image with highlighted rectangle enclosing the pedestrian; if negative, it would just provide a message to notify the user that it could not detect any pedestrian on the image.

### 4 DATASET

The images that were processed by the descriptors and used to train the model were a subset of the actual dataset downloaded from the KITTI Vision Benchmark Suite website [3]. The subset was selected by choosing at random pedestrians that were clear and visible. The dataset downloaded had images of street views or footpaths where pedestrians would most likely be found. It was already divided into two parts with one being images meant for training while the other was meant for testing. It also contained labels for each image with the coordinates, height and width of the rectangular window



that would enclose each of the pedestrians. Each image could have none, one or multiple pedestrians. The people are usually standing, but appear in any orientation and against a wide variety of background image including crowds. Many are bystanders taken from the image backgrounds, so there is no particular bias on their pose. Since our detection window was 64x128 (Ratio of 1:2), pedestrians patches were extracted and the patch was resized to a ratio identical to the required ratio of 1:2. The labels were used to find the exact patch that can enclose each pedestrian and depending on the dimensions of the patch height or width are added to make the patch size match the exact ratio of the detection window without having to deform the structure of the pedestrian. Once having attained the right ratio, all images were resized to a patch size of 64 x 128. Some of these images are shown in Figure 1.

The training part of the dataset had 4780 images that yielded to more than 10000 pedestrians. Having extracted these patches and having corrected their width to height ratio, 1000 images with pedestrians were taken and kept as positive images. Another 1000 images were extracted with patches other than pedestrians and were stored to be used as negative images. These images, each of the same size namely, 64 x 128, were used together to train the SVM model for classification. The model was then tested by repeating the above process on the images present in the testing part of the dataset. The testing part contained 7517 images but only some of them were used to extract and stored as positive and negative images. 2000 different images were run against the model and it was continuously improved by replacing images that were unclear or did not contain the full body of the image; until it attained an accuracy of 95% in most cases. For practical purposes, the Samsung Galaxy S6 was used as the testing device throughout the project.

## 5 RESULTS AND DISCUSSIONS

The HOG-SVM provided correct results with rare occurrences of false positives as shown in Figure 5. This is mostly due to the fact that in very few cases, combined objects in images resemble the features of a human body closely and hence are accepted by the classifier as a pedestrian. These objects include stop signs, trees and pillars. Pedestrians were detected mostly in standing poses but also included some instances while captured cycling. Pedestrians overlapping other pedestrians were detected in most cases, with exceptions in cases where the features of the body of the other person could not be seen in the images. The detection worked well on images with cluttered as well as badly illuminated background, so long as the shape and size of the person was visible, the detector could detect it. Moreover, the application built using android supported the wrapped C++ source code as well library and hence helped reuse and increase efficiency of the overall process. An alternative to such an approach would involve rewriting code as well as the library in java and then using it with android, but this would not take advantage of NDK and would lead to a more time consuming detection process.



Figure 5: Left image shows successful detection while right image shows a false positive detected by the detector. [13]

## 6 CONCLUSION

In this paper, an approach for pedestrian detection using HOG-SVM was described. Histograms of oriented gradients(HOG) were the descriptors that provide a feature vector of an image and LibLinear Support Vector Machine (SVM) was used to create and save a trained model. This model would identify if an image contained a pedestrian when the feature vector was supplied to it. Several images were taken and the classifier model was trained to an acceptable accuracy of above 90%. The final trained model was developed using 1000 positive images and 1000 negative images.

The main result was that even though the HOG-SVM methodology had some false positive results it was successful in accurately identifying pedestrians. The simplicity of the method and source code also allowed its successful implementation in the smartphones operating system and hardware.

In future work, the HOG descriptor could be divided into separate models and fine tune them to different body parts so that it could handle a wider range of poses such as sitting, lying and bending.

## ACKNOWLEDGMENT

*We wish to acknowledge the funding support for this project from Nanyang Technological University under the Undergraduate Research Experience on Campus (URECA) programme.*

*Also wish to thank Professor Felix Klanner, Mr. Alexander Hanel and BMW@NTU Future Mobility Research Lab for providing me with the opportunity, guidance as well as facilities to work on this project under their supervision.*

## REFERENCES

- [1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005.
- [2] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, *Journal of Machine Learning Research* 9(2008), 1871-1874.
- [3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [4] R. Ronfard, C. Schmid, and B. Triggs. Learning to parse pictures of people. *The 7th ECCV, Copenhagen, Denmark*, volume IV, pages 700–714, 2002.
- [5] P. Felzenszwalb and D. Huttenlocher. Efficient matching of pictorial structures. *CVPR, Hilton Head Island, South Carolina, USA*, 2000.
- [6] S. Ioffe and D. A. Forsyth. Probabilistic methods for finding people. *IJCV*, 43(1):45–68, 2001.
- [7] D.M.Gavrila and V.Philomin.Real-timeobjectdetectionfor smart vehicles. *CVPR, Fort Collins, Colorado, USA*, 1999.
- [8] D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: the protector system. *Proc. of the IEEE Intelligent Vehicles Symposium, Parma, Italy*, 2004.
- [9] C. Papageorgiou and T. Poggio. A trainable system for object detection. *IJCV*, 2000
- [10] A. Mohan, C. Papageorgiou, and T. Poggio. Example-based object detection in images by components. *PAMI*, April 2001.
- [11] K. Mikolajczyk, C. Schmid, and A. Zisserman, *Human detection based on a probabilistic assembly of robust part detectors. Computer Vision - ECCV 2004* vol. 3021, 2004.
- [12] J. Howse, *Android Application Programming with OpenCV 3*: Packt Publishing, 2015.
- [13] "The KITTI Vision Benchmark Suite," *The KITTI Vision Benchmark Suite*. [Online]. Available: [http://www.cvlibs.net/datasets/kitti/eval\\_object.php](http://www.cvlibs.net/datasets/kitti/eval_object.php). [Accessed: 29-Jun-2016].
- [14] R. -E. Fan K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin., "LIBLINEAR -- A library for large linear classification,". [Online]. Available: <https://www.csie.ntu.edu.tw/~cjlin/liblinear/>. Accessed: Jun. 30, 2016.
- [15] Weisstein, Eric W. "L<sup>2</sup>-Norm." From *MathWorld*--A Wolfram Web Resource. <http://mathworld.wolfram.com/L2-Norm.html> Accessed: Jun. 30, 2016.
- [16] Google, and Open Handset Alliance n.d. Android APIGuide.<http://developer.android.com/guide/index.html>. Accessed Jun 30, 2016.
- [17] Image And Video Descriptors Advanced Topics In Computer Vision Spring 2010 Weizmann Institute Of Science Oded Shahar And Gil Levi. - Ppt Download". *Slideplayer.com*. N.p., 2016. Web. 10 July 2016.