# Pedestrian Detection on Mobile Devices

Lyne P. Tchapmi

## 1. Introduction

Pedestrian detection is an ongoing area of research with applications such as video surveillance, robotics navigation, and collision avoidance for autonomous vehicles. We propose the design of a pedestrian detection system that can run on mobile devices. Such a system can be valuable as a driving assistant in order to reduce the probability of accidents, and help keep drivers aware of nearby pedestrians.

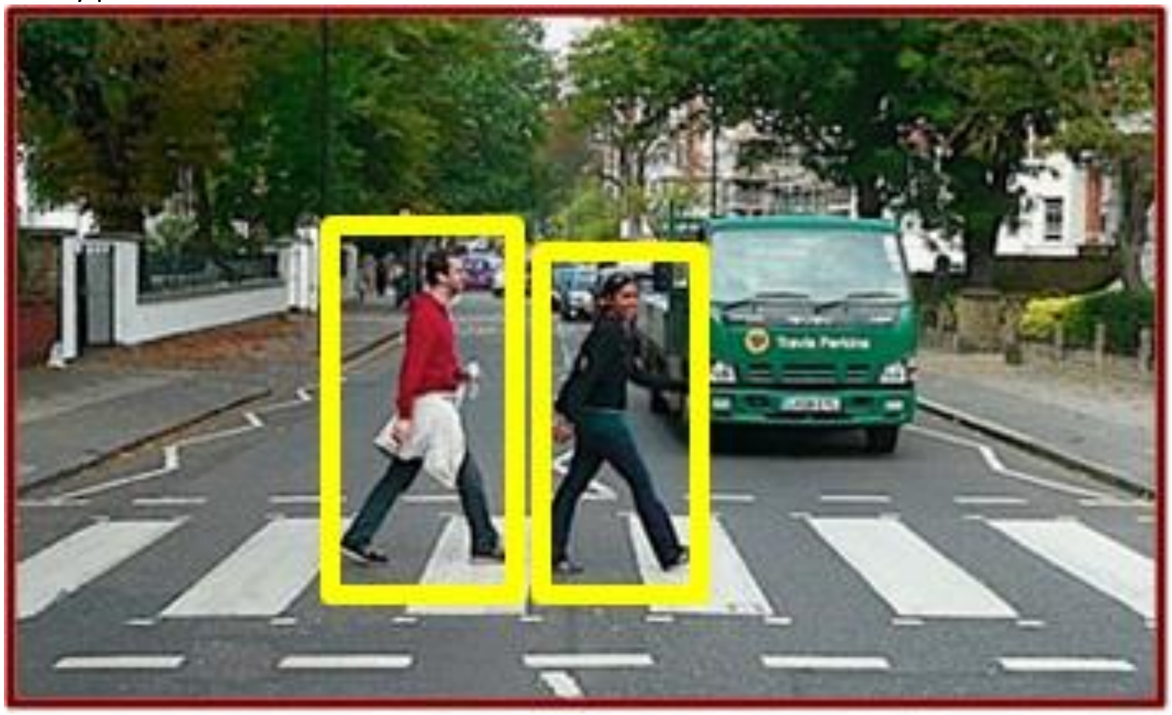

**Figure 1: Pedestrian detection**

## 2.1 Previous Works

A lot of studies have focused on the problem of making pedestrian detection systems, faster, more accurate and more efficient. In 2013, Varga et al. [2] introduced a gradient based method to efficiently select Regions of Interests (ROIs) for the location of people in an image. The method efficiently looks for pedestrian bounding boxes by a series of images filters which significantly reduce the number of potential candidates. To improve efficiency, the authors circumvent resizing images by choosing a few bounding box scales for training and classification. Integral channel features is chosen for that system due to their ease of computation.

In 2014, the same authors optimized the algorithm presented in their previous work, and ported the resulting system to an Android device, thus demonstrating its efficiency [1,2].

## 2.2. Contribution

We implement a pedestrian detection system similar to the work in [1,2], but we choose HOG features as they are well suited for the representation of people. We modify the bounding box scales reported in [1,2] to meet the requirements of the OpenCV modules for HOG Descriptors. And to reduce the number of false positives after processing, we perform non-max suppression based on bounding box overlap and relative SVM scores.

## 3.1. Technical Details

In pedestrian detection systems, an image is given as input, and the output is a bounding box or a set of bounding boxes representing the location(s) of pedestrian(s) in the image. These systems have multiple stages of execution. Based on work presented in [1] and [2] we took a 4-step approach: Pre-processing, ROI extraction, Features extraction and classification.

## 3.2. Implementation

***- Pre-processing***

The detection pipeline starts with image processing for noise reduction. We used Gaussian blurring for this stage. The filtered image is then processed to extract candidate regions for pedestrian locations.

**- ROI extraction**

After pre-processing, we need to extract potential rectangular regions in the image. We do so by looking for pixels that are potential candidates for the midpoint of the top of the bounding box. These candidates are found through a Sobel filter as shown in Figure 2.

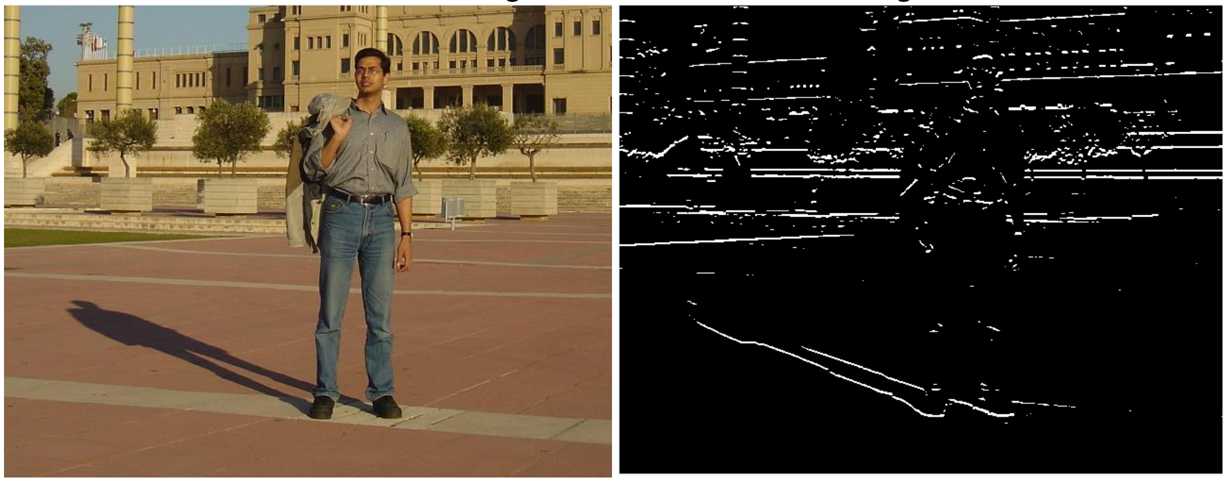

**Figure 2: Original Image(left), Top image(right)**

After top image candidates are found, we look for pixels that are candidate for the bottom midpoint of the bounding box. These candidates are found by applying a horizontal box filter to the top image (Figure 3). Candidates for top and bottom are then matched if they create a bounding box with one of our chosen scales (Figure 4). The width is deduce from a fixed width-to-height ration WHR =0.5.+ For each bounding box, we proceed to extract features that will be used for Classification.

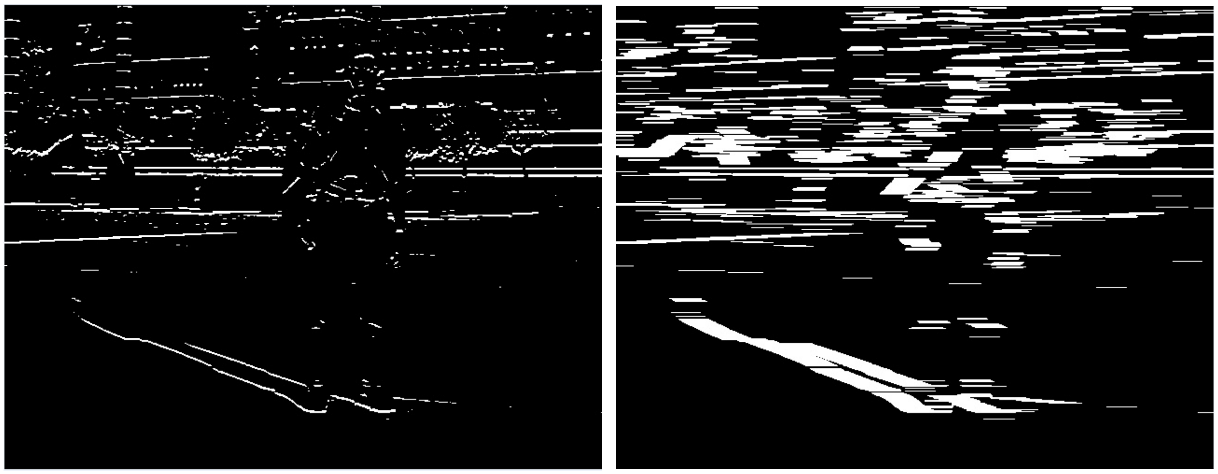
**Figure 3: Top image(left), Bottom image(right)**

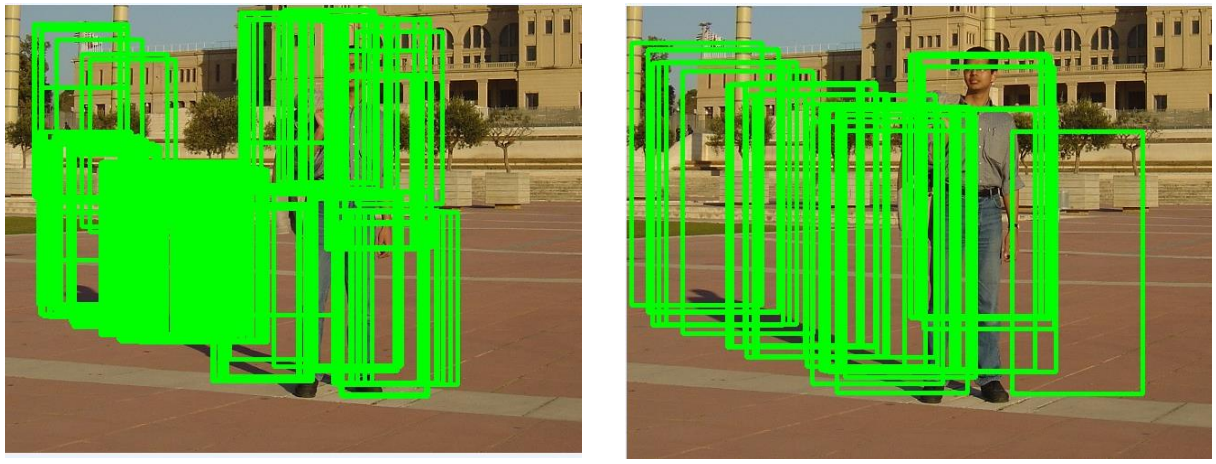
***Figure 4: Bounding boxes extracted at 2 different scales after matching top and bottom pixels***

***- Features extraction***

Popular features for pedestrian detection include HOG features, Haar wavelets, and Integral Channel features. We chose HOG as they have been demonstrated to have good performance for pedestrian detection. We experimented with the parameters for HOG descriptors including cell size, block size and block stride to find a balance between storage size and accuracy. Our final cell size and block stride were 4X4. And we chose an 8X8 block size.

***- Classifier***

In the final step of our system, a classifier determines whether a given patch represents a pedestrian or not. Linear SVMs are used to allow a relatively fast classification. A classifier is trained for each of the 6 scales chosen above so as to avoid an expensive image resizing step. After classification, a set of detections is returned and we use non-maximum suppression to remove multiple responses around the same objects. A bounding box is discarded if its Intersection over Union overlap with another bounding box with higher score is greater than a chosen threshold (0.2 in our implementation)

**-Target Mobile Platform**

We implemented our algorithm on an Android Nvidia Shield Tablet with Tegra K1 192 core Kepler GPU, 2.2GHz ARM Cortex A15 CPU with 2GB RAM. The native code was written in C++. Training was performed offline on a Mac OS laptop, and XML files were saved for each classifier. We loaded those files onto the tablet at runtime for detection.

## 4. Experiments

We trained the SVMs on the INRIA pedestrian dataset, and the total training time was around 15mins. All 6 XML files obtained take up 4.83MB of memory in total. We ran a few experiments on the laptop and on the mobile device. Runs on the mobile device had running times ranging from 10s to a few minutes per image including file processing. Results from the detection pipeline are shown in Figure 5. We notice that despite the presence of false positives, at least one of the final detections significantly overlaps with the location of a pedestrian. This suggests that by optimizing criteria for filtering bounding boxes, it is possible to narrow down the location of the pedestrian. These figures also illustrate a limitation of the algorithm which is the fact that using a fixed WHR and a small discrete set of bounding boxes can lead to partial detections or no detections depending on the size of the pedestrian. Figure 6 illustrates a case where the final detections are all far away from the pedestrians. This can potentially be caused by limitations from using a linear SVM, the lack of sufficient training data (of people facing back for example), as well as the simplifications we made.

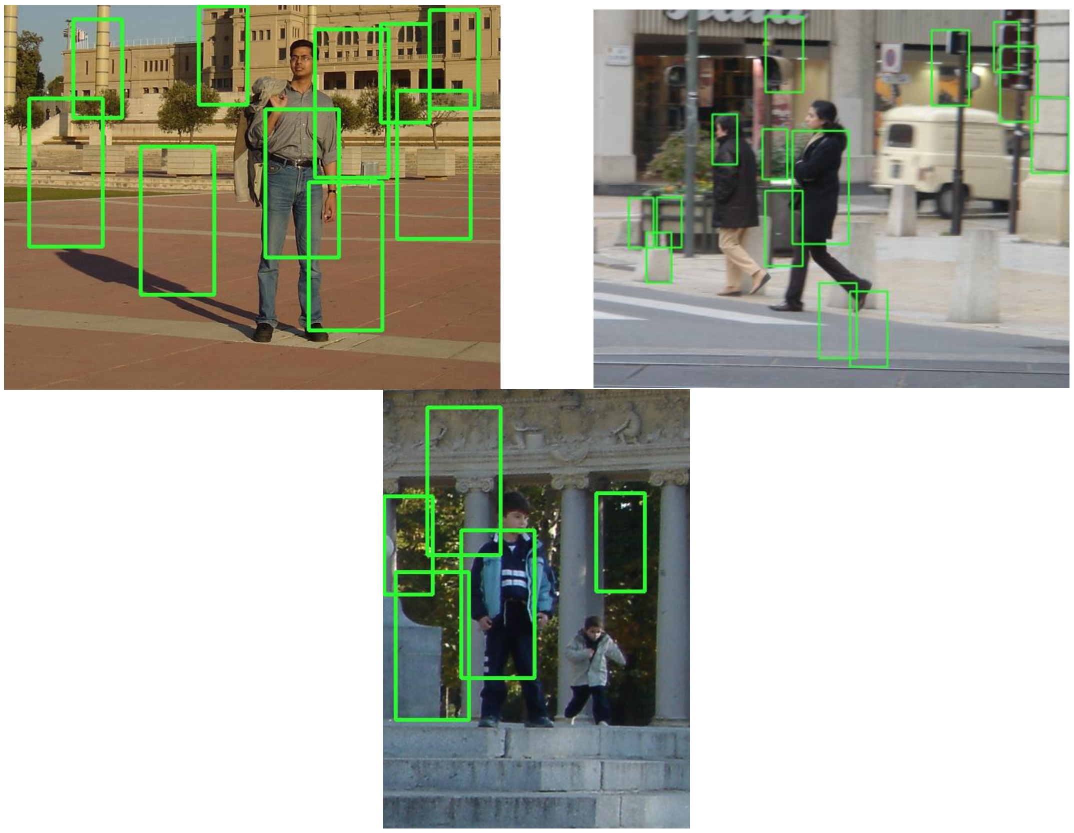
***Figure 5: Detections with a bounding box with significant overlap***

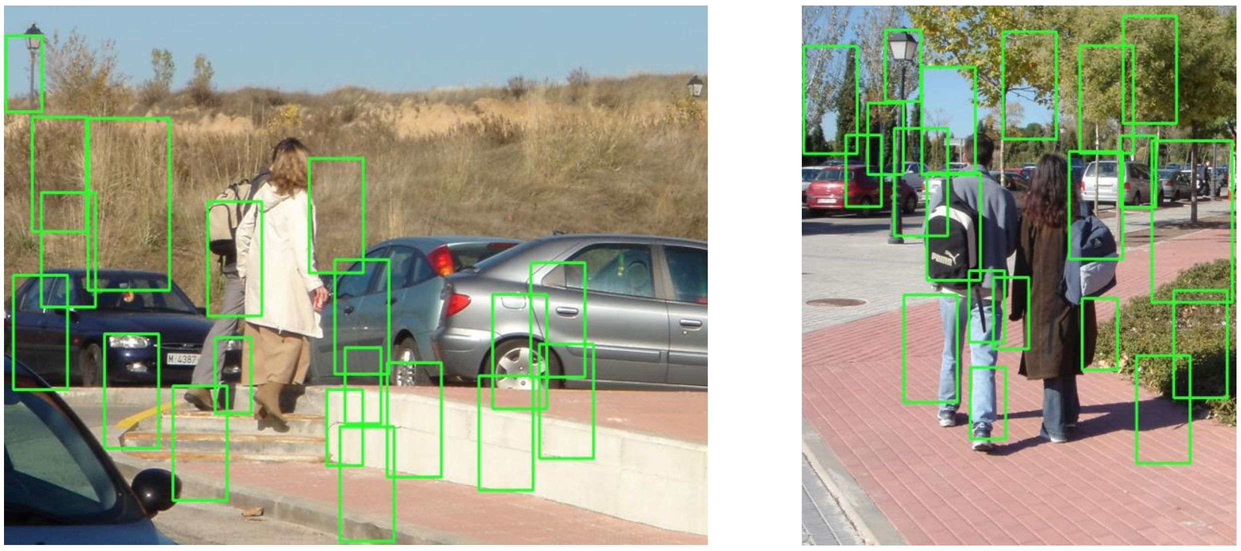
***Figure 6: False Positives and missed detections***

**5 Conclusions**
Overall, the simplifications done in the system by choosing a small number of scales, a fixed WHR, a linear SVM, HOG Features, and filter based ROI extraction, make the

pedestrian detection algorithm faster and possible to implement on constrained environments such as mobile devices. But those simplifications also results in a reduction in accuracy represented by multiple detections and several false positives. Clever implementation strategies for bounding box suppression, and better features could help alleviate this issue. There also is room for improvement in terms of speed on the mobile device. Future implementations can benefit from using efficient parallel computations and clever methods for loading, using and storing SVM XML files and images.

## 6 References

[1] Varga, Robert, et al. "Real-time pedestrian detection in urban scenarios."*Intelligent Computer Communication and Processing (ICCP), 2014 IEEE International Conference on*. IEEE, 2014.

[2] Varga, Robert, and Sergiu Nedevschi. "Gradient-based region of interest selection for faster pedestrian detection." *Intelligent Computer Communication and Processing (ICCP), 2013 IEEE International Conference on*. IEEE, 2013.