2015 IEEE 18th International Conference on Intelligent Transportation Systems

Fast Pedestrian Detection for Mobile Devices

Arthur Daniel Costea Andreea Valeria Vesa Sergiu Nedevschi

Image Processing and Pattern Recognition Research Center
Computer Science Department, Technical University of Cluj-Napoca, Romania
{arthur.costea, andreea.vesa, sergiu.nedevschi}@cs.utcluj.ro

***Abstract*—In this paper we present a fast and robust solution for pedestrian detection that can run in real time conditions even on mobile devices with limited computational power. An optimization of the channel features based multiscale detection schemes is proposed by using 8 detection models for each half octave scales. The image features have to be computed only once each half octave and there is no need for feature approximation. We use multiscale square features for training the multiresolution pedestrian classifiers. The proposed solution achieves state of art detection results on Caltech pedestrian benchmark at over 100 FPS using a CPU implementation, being the fastest detection approach on the benchmark. The solution is fast enough to perform under real time conditions on mobile platforms, yet preserving its robustness. The full detection process can run at over 20 FPS on a quad-core ARM CPU based smartphone or tablet, being a suitable solution for limited computational power mobile devices or embedded platforms.**

I. Introduction

Nowadays, pedestrian detection has become a major topic within the Computer Vision research community. There are multiple approaches that appear every year, each trying to outperform the previous state of art. This becomes even more challenging if we take into consideration the availability of several pedestrian detection benchmarks [10], [4] that allow an easy comparison between these different approaches. Over the years several surveys [11], [12], [14] were published reviewing and evaluating the proposed solutions. Considering the high interest in developing real-time pedestrian detection applications there are still few approaches that manage to achieve state of art detection with high accuracy, but low execution time. Within the top 10 approaches evaluated on the Caltech-USA dataset, only one approach succeeds in running at over 30 FPS [8], while the other ones perform at considerably lower FPS rates. Most of these solutions run only on CPU, but further improvements have been suggested using a GPU implementation. However little attention has been given to their portability on mobile platforms. Smartphones, tablets or embedded systems have all invaded the market during the last years. Even though these devices are rapidly evolving, there are still limitations in terms of memory and processing power when addressing their architecture. Therefore a portable pedestrian detection scheme has to take into account also these limitations.

The main contribution of this paper is a pedestrian detection solution that can run at a rate of over 100 FPS on a desktop CPU and is suitable for real-time applications even on limited computational power mobile devices. The proposed approach competes well with the current state of art in terms of detection accuracy and precision, and provides outstanding results on Caltech pedestrian detection benchmark. The fast execution time at high detection accuracy was achieved using:

- a novel multiscale detection scheme with 8 multiresolution pedestrian models
- computationally efficient multiscale features
- several implementation optimizations

The proposed detection scheme has the advantage that image features can be computed only once per half octave scale and there is no need for approximation operations for intermediate scales due to the multiple multiresolution pedestrian models.

The approach was integrated into a driver assistance application (Figure 1) for mobile devices. The application can run at over 20 FPS.

Fig. 1. Pedestrian detection on mobile devices

II. Related work

Considering monocular vision, most top performing approaches detect pedestrians using a sliding window at multiple scales over the input image. The most common features used for classification are color and histogram of oriented gradient (HOG) features organized in 10 image channels, as proposed by Dollar et al. in [9]. Several multiscale detection schemes have been proposed relying on these 10 channels. The Integral Channel Features (ICF) approach [9] used a single classifier for pedestrians having a height of 96 pixels and resized the image multiple times. Due to the time-consuming feature computation at each scale, the approach achieved around 1-2

978-1-4673-6596-3/15 $31.00 © 2015 IEEE
DOI 10.1109/ITSC.2015.382

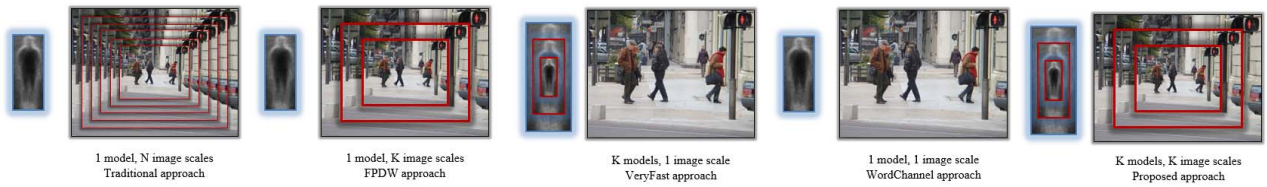


Fig. 2. Various multiscale pedestrian detection approaches

FPS. The Fastest Pedestrian Detector in the West [7] improved the frame rate of the previous solution to 8 FPS by computing the features only for the half octave scales (eight times less) and approximating the features at the intermediate scales. Benenson et al. [1] proposed a solution using a single image scale and multiple classifier scales. Five pedestrian classifiers were similarly trained in half octaves and the features were approximated for the intermediate scales. This approach run at 50 FPS using a GPU implementation and had the lowest miss rate on the INRIA pedestrian benchmark at the time it was proposed. An improvement in accuracy and execution time over ICF was achieved by Aggregate Channel Features (ACF) proposed in [8]. Aggregate channels are computed instead of integral images for the 10 channels by computing mean values for 4×4 pixel aggregates. This way classification features become simple pixel lookups. This approach runs at 30 FPS and is the fastest top-performing CPU solution currently on the Caltech benchmark.

Detection rate improvements were achieved by using optical flow [20], deformable part-based models [22], locally decorrelated channel features [16], deep neural networks [15], however at a significant increase in computational costs.

In order to optimize the overall detection time, each step of the detection process should be parallelized on the available CPU or GPU cores in a memory bandwidth friendly manner, which is not an easy task. GPU-based solutions achieved 50 FPS [1] and 17 FPS [3] running on desktop GPUs. Parallelization on mobile devices was considered in [21].

III. Proposed solution

Figure 2 presents different implementation choices for multiscale detection. The trivial approach is to train a classifier for detecting a pedestrian with a fixed size and to resize the image multiple times as in [4] and [10]. The image features need to be recomputed after each resizing, thus being a computationally costly approach. Other approaches rely on approximating the classification features from K image scales to a single pedestrian model (Dollar et al. [8]), or from one image scale to K pedestrian models (Benenson et al. [1]). In [3] a single classifier model was used with a single feature scale using higher order Wordchannel features.

In our solution we train 8 classifier models in order to detect pedestrians with heights between 50 and 100 pixels. First, these classifier models are applied over the original image scale. To detect larger pedestrians, we halve the image multiple times and reapply the same classifier models. The proposed multiscale detection scheme uses 3 half octave image scales and 8 classifiers to detect all scales inside a half octave. Using multiresolution pedestrian models there is no need for feature approximation. The classifier uses only features that are computed once per half octave. In our experiment we use 3 half octaves, resulting in 24 detectable pedestrian scales.

We detect pedestrians having a size of at least 50 pixels in the case of a 640×480 pixel image size. We use the aspect ratio of 0.41 as recommended in [11]. For a pedestrian of size 50×20.5 we use a detection window of 64×32 pixels in order to include also some pedestrian context [4]. The eight pedestrian models defined for half octaves have an aspect ratio of 0.5 and have the following heights: 64, 72, 80, 88, 96, 104, 112, 120.

We use the 10 channels proposed in [10] for generating the classification features. The channels consist of three color channels (LUV color space), one gradient magnitude and six oriented gradient magnitudes (HOG). For a faster computation we use lookup tables for generating the channel values based on RGB values or derivative values.

As classification features for pedestrian models, we use the sums for 2×2, 4×4 and 8×8 pixel groups as seen in Figure 3, at each position in the pedestrian model (features can overlap). After grouping all pixels from the 10 channels into 2×2 cells, 4×4 and 8×8 cells can be computed using the previous scale. We store the sum of each such feature at each position for each half octave, and avoid the use of integral images, which reduces further computational costs. The classification of bounding boxes is efficient because the classification features are only pixel lookups as in [8], but we use multiple aggregation scales.

We train 8 classifiers for the 8 pedestrian models using AdaBoost [13] with two-level decision trees as weak learners. We follow the training protocol from [8], [1] using 3 rounds of bootstrapping and a final classifier consisting of 4096 weak learners, but with half as many negative samples. The whole training procedure for all 8 classifiers on the Caltech training dataset took around two hours on an Intel Core i7 CPU using a multicore implementation.

IV. Implementation details

There are two main time-consuming processes: image channel computation and bounding box classification. In order to reduce the execution time we several optimizations. All of the processes are optimized to run on all available CPU cores.

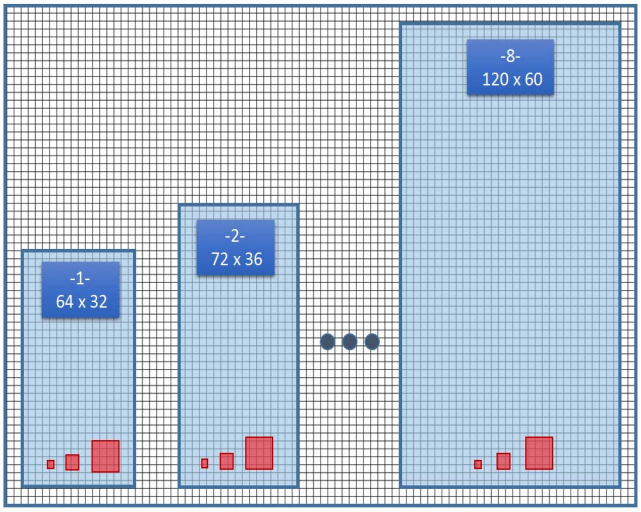


Fig. 3. Multiresolution models

In order to generate the image channels we use lookup tables for LUV color space conversion (indexed by RGB values) and for gradient magnitude and orientation (indexed by derivative values). We compute at each pixel the gradient on each L,U and V channels and use the one with the maximum magnitude. Then, the gradient is normalized with the smoothed gradient similarly to [9]. However, we use a 11×11 pixel size box filter to obtain the smoothed gradient, which is computationally simpler than a triangle filter and provides similar results. The 2D box filtering can be computed using two 1D filters in constant time (independent of filter size). The normalized gradient is used to generate the gradient magnitude channel and the 6 orientation channels.

Three aggregate channels are computed for each of the 10 image channels, one for each of the three block sizes seen in Figure 3. In each position the value of an aggregate channel represents the average over the corresponding pixel block. The aggregate values are computed for all possible block locations using a step of 2 pixels. This way the aggregate channels will have two times smaller height and width. In the case of 4×4 and 8×8 aggregates, the pixel blocks will overlap. Finally, a post smoothing is applied over each aggregate channel using a 3×3 box filter.

For fast sliding window classification we use soft-cascading [6] with a rejection threshold of -1. The evaluation of the 4096 weak learners is halted if the classification cost drops below -1. It is difficult to ideally parallelize the classification of all bounding boxes because the number of weak classifiers that are evaluated is not the same. In our experiments after evaluating the first 32 weak learners over 90% of the bounding boxes are rejected. A pedestrian needs to pass all 4096 weak classifiers. We distribute the initial bounding boxes on all available CPU cores and evaluate only the first 32 weak classifiers. The remaining bounding boxes are redistributed over the CPU cores in order to evaluate all remaining weak classifiers. For a faster feature evaluation we precompute the positions (memory location) of all necessary classification features for each individual bounding box.

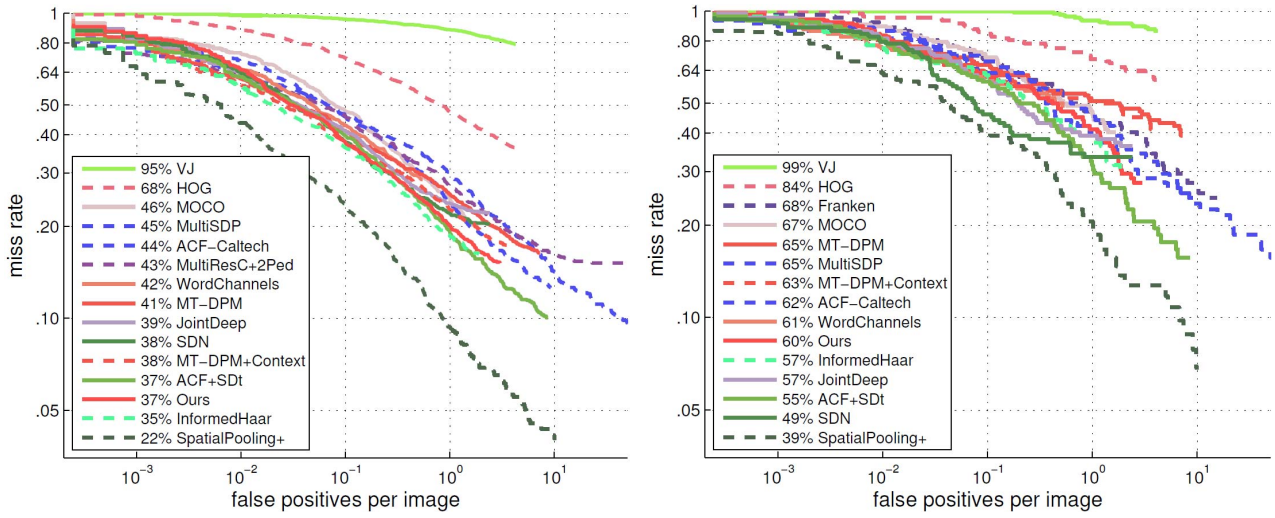
We consider that two bounding boxes are overlapping if the ratio between the common area and the unified area is larger than 50%. In order to eliminate overlapping bounding boxes we use the greedy solution for non-maximum suppression proposed in [10]. The bounding boxes are selected in decreasing order of classification confidence. A bounding box is valid only if it does not overlap a bounding box with a higher confidence. We apply non-maximum suppression after evaluating the first 512 weak classifiers and after evaluating all detection models. The former can reduce up to ten times the remaining bounding boxes which have a high chance for being evaluated by most of the 4096 weak classifiers, resulting in reduced computational costs.

V. Experimental results

In order to validate the proposed solution and to provide a comparison with the current state of art we use the Caltech USA pedestrian detection benchmark [11]. Using the Matlab toolbox provided at [5] we are able to evaluate and to compare our detector with 43 other detectors. We use the standard Caltech training dataset consisting of around 4000 fully annotated images in order to train the 8 pedestrian classifiers for our experiments. We use 3 half-octaves for the 640×480 pixel images, that permits the detection of 24 pedestrian scales with sizes ranging from 50 to 375 pixel height. The dataset contains a large variety of occlusion cases, several pedestrians having more than 50% occlusion. Detailed statistics are provided in [11]. A simple search space reduction [21] is applied by considering only bounding boxes having their center between rows 140 and 300 (out of 480 rows). Over 99% of the pedestrians annotated in the Caltech database have their centers in this range, thus the search space can be reduced by 35%.

The best way to represent the detection performance is using ROC curves. These curves represent the detection accuracy as a function of detection precision and are generated using different detection thresholds for the same approach. Figure 4 presents the detection performances on the Caltech "reasonable" and "partial occlusion" test scenarios for our detector and the current top 10 approaches (approaches using the same training dataset): SpatialPooling+ [18], InformedHaar [24], ACF+SDt [20], MT-DPM+Context [22], SDN [15], JointDeep [17], MT-DPM [22], WordChannels [3], MultiResC+2Ped [19], ACF-Caltech [8], MultiSDP [23], MOCO [2]. Approaches are ordered based on the log-average miss rate for the precision range of 10^{-2} to 10^{0} false positives per image (FPPI). The metric is obtained by computing the average miss rate at 9 evenly spaced FPPI rates in log-space for the specified precision range. Our approach compares well with the current state of art.

Our experiments were carried out on a system equipped with an Intel Core i7 4960X (3.0 GHz) CPU. The following average



(a) Reasonable (b) Partial occlusion

Fig. 4. Evaluations on the Caltech dataset

execution times were obtained on the Caltech evaluation set for 640 × 480 pixel images:

- Feature computation: 8.8 ms (113 FPS)
- Classification of each bounding box: 0.8 ms
- Total detection time: 9.6 ms (105 FPS).

We also tested our approach using an older generation Intel Core i7 2600K (3.4 GHz) CPU, realeased in 2013, obtaining a detection time of 13.9 ms (72 FPS). Table I provides a comparison of our approach with other approaches that gave details regarding their execution time. Except WordChannels (GPU based solution) the other approaches were CPU only solutions. Some of them were implemented in Matlab, while other used a C/C++ based solution. The ACF approach, running at 30 FPS, was mostly implemented in Matlab and used C/C++ code for the time-consuming parts.

VI. Porting to Mobile Platforms

The proposed solution was ported and tested on the following mobile devices:

- Samsung Galaxy Tab Pro T325 tablet (Quad-core 2.3 GHz Krait 400 CPU)
- Sony Xperia Z1 smartphone (Quad-core 2.2 GHz Krait 400 CPU).

The main challenge was to have a detection algorithm that runs fast enough on mobile devices, yet still preserving its robustness. This is not an easy task since current mobile architectures are quite limited with respect to processing power, memory space and camera frame rate. To overcome these limitations speed and memory efficiency are a must.

On the scope of this, Java Native Interface (JNI) programming framework was used, that allows an easy interaction of native (C/C++) code and Java code. To take full benefit of the processing power of such devices, the code was parallelized using the Qualcomm Multicore Asynchronous Runtime Environment (MARE) library. Where applicable, multiple instructions were executed in parallel in a multicore processing manner.

We obtained an average frame rate of around 8 FPS on both testing devices when detecting pedestrians with heights of at least 50 pixels and 20 FPS with heights of at least 100 pixels. In comparison with the unparallelized version we succeeded in more than halving the computation time with multicore execution. Examples of some pedestrian detections in traffic scenarios are provided in Figure 5. Images were obtained by capturing the tablet's screen while driving.

The proposed solution was integrated into a driver assistance application. The goal is to detect pedestrians in traffic

TABLE I
Miss rate vs. execution time

Approach	Miss rate	FPS
HOG	68.46%	0.05
FPDW	57.40%	2.6
ChnFtrs	56.34%	0.2
CrossTalk	53.88%	14
ACF-Caltech	44.22%	30
WordChannels	42.30%	17
SDN	37.87%	10
SquaresChnFtrs	34.81%	1
InformedHaar	34.60%	0.62
LDCF	24.90%	2
SpatialPooling+	21.89%	0.5
Ours	37.33%	105

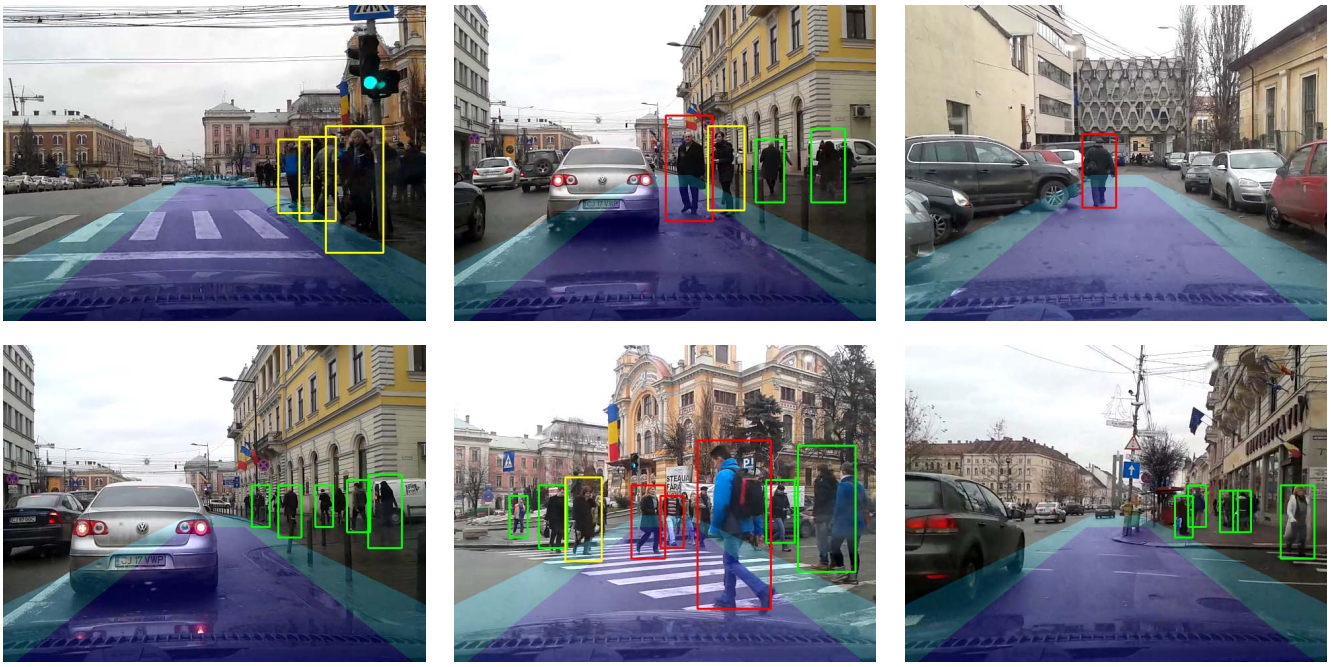


Fig. 5. Various detection scenarios

scenarios and alert the driver when there is a potential danger. The pedestrians are displayed in three colors: green - is far; yellow - is approaching the car; red - is on the car's trajectory. The blue lane in Figure 5 is a projection of an estimated lane in front of the car and is used to determine if a pedestrian is on the car's trajectory. The projection is adapted to the pitch angle measured with the help of the gravity sensor available on the mobile device. To avoid the driver's necessity to pay attention to the screen, the application emits a warning sound that alerts the driver when there is a potential danger. Therefore, the main purpose of the application is to increase the alertness of the drivers.

VII. Conclusion

We presented a real-time pedestrian detection scheme targeted at mobile devices. A novel detection scheme is proposed that relies on 8 multiresolution models, applied over each half octave, and uses multiscale aggregation for obtaining classification features. The algorithm can run at over 100 FPS using CPU implementation, being the fastest top-performing pedestrian detector tested on Caltech pedestrian detection benchmark. Such a fast yet robust detector proves to be a suitable solution to be ported on mobile devices or embedded platforms, which are more limited in terms of processing power. We report an average frame rate of around 20 FPS on mobile devices.

The proposed solution can still be improved. The current approach does not use any information outside the bounding box. Motion, context information and tracking can further increase detection robustness. The execution time on mobile devices can be improved further by adopting a GPU implementation using a powerful and energy efficient GPU, such as the NVIDIA Tegra K1 mobile processor with 192 CUDA cores.

As future work we aim to obtain better risk assessment for the detected pedestrians. All available information has to be taken into consideration, such as pedestrian orientation, attitude and moving pattern, in order to recognize any dangerous situation and to avoid false warnings.

Acknowledgment

This work has been partially supported by SmartCoDrive project (PNII-PCCA 18/2012) and MULTISENS project (PNII-ID-PCE-2011-3-1086), funded by the Romanian Ministry of Education and Research, UEFISCDI.

References

[1] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. "Pedestrian detection at 100 frames per second". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 2903–2910.

[2] G. Chen, Y. Ding, J. Xiao, and T. X. Han. "Detection evolution with multi-order contextual co-occurrence". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 1798–1805.

[3] A. D. Costea and S. Nedevschi. "Word Channel Based Multiscale Pedestrian Detection without Image Resizing and Using Only One Classifier". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 2393–2400.

[4] N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection”. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005, pp. 886–893.

[5] P. Dollar. *Piotr’s Image and Video Matlab Toolbox (PMT)*. http://vision.ucsd.edu/toolbox/doc/index.html.

[6] P. Dollar, R. Appel, and W. Kienzle. “Crosstalk cascades for frame-rate pedestrian detection”. In: *IEEE European Conference on Computer Vision (ECCV)*. 2012, pp. 645–659.

[7] P. Dollar, S. Belongie, and P. Perona. “The Fastest Pedestrian Detector in the West”. In: *British Machine Vision Conference (BMVC)*. Vol. 2. 3. 2010, p. 7.

[8] P. Dollar, R. Appel, S. Belongie, and P. Perona. “Fast feature pyramids for object detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 36.8 (2014), pp. 1532–1545.

[9] P. Dollar, Z. Tu, P. Perona, and S. Belongie. “Integral Channel Features”. In: *British Machine Vision Conference (BMVC)*. Vol. 2. 3. 2009, p. 5.

[10] P. Dollar, C. Wojek, B. Schiele, and P. Perona. “Pedestrian detection: A benchmark”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 304–311.

[11] P. Dollar, C. Wojek, B. Schiele, and P. Perona. “Pedestrian detection: An evaluation of the state of the art”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 34.4 (2012), pp. 743–761.

[12] M. Enzweiler and D. M. Gavrila. “Monocular pedestrian detection: Survey and experiments”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 31.12 (2009), pp. 2179–2195.

[13] J. Friedman, T. Hastie, R. Tibshirani, et al. “Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)”. In: *The annals of statistics 2000* 28.2 (2000), pp. 337–407.

[14] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf. “Survey of pedestrian detection for advanced driver assistance systems”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 32.7 (2010), pp. 1239–1258.

[15] P. Luo, Y. Tian, X. Wang, and X. Tang. “Switchable deep network for pedestrian detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 899–906.

[16] W. Nam, P. Dollar, and J. H. Han. “Local Decorrelation For Improved Pedestrian Detection”. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 424–432.

[17] W. Ouyang and X. Wang. “Joint deep learning for pedestrian detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 2056–2063.

[18] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. “Pedestrian Detection with Spatially Pooled Features and Structured Ensemble Learning”. In: *arXiv preprint arXiv:1409.5209* (2014).

[19] D. Park, D. Ramanan, and C. Fowlkes. “Multiresolution models for object detection”. In: *IEEE European Conference on Computer Vision (ECCV)*. 2010, pp. 241–254.

[20] D. Park, C. Lawrence Zitnick, D. Ramanan, and P. Dollar. “Exploring weak stabilization for motion feature extraction”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 2882–2889.

[21] R. Varga, A. V. Vesa, P. Jeong, and S. Nedevschi. “Real-time pedestrian detection in urban scenarios”. In: *IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. 2014, pp. 113–118.

[22] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li. “Robust multi-resolution pedestrian detection in traffic scenes”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 3033–3040.

[23] X. Zeng, W. Ouyang, and X. Wang. “Multi-stage contextual deep learning for pedestrian detection”. In: *IEEE International Conference on Computer Vision (ICCV)*. 2013, pp. 121–128.

[24] S. Zhang, C. Bauckhage, and A. B. Cremers. “Informed Haar-like features improve pedestrian detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 947–954.