The System Design of a High-Speed Object Detector

Tom Runia
Delft University of Technology*
Pattern Recognition Laboratory
tomrunia@gmail.com

Robert Lukassen
TomTom
Product Lab, Eindhoven
robert.lukassen@tomtom.com

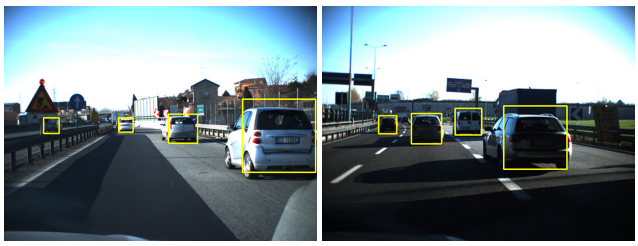
Lu Zhang Marco Loog
Delft University of Technology
Pattern Recognition Laboratory
{lu.zhang,m.loog}@tudelft.nl

Abstract

We discuss the design of a high-speed object detection framework. More specifically we focus on rear view car detection. However the methods discussed are not limited to this object class. Our baseline detector uses integral channel features in a boosting framework reminiscent of Viola-Jones. With our baseline detector as starting point we consider the implementation and study of two approximation techniques that result in a significant detection speedup. The first approximation focuses on decreasing the time spent on classification of subwindows. The second approximation technique targets the multiscale aspect of our object detector. We show that these techniques result in a $17.5\times$ *speedup without sacrificing on detection rate. Based on these approximation techniques and a fast GPU implementation for extracting channel features we report detection speeds up to* 55 *fps without exploiting scene geometry or reducing the search space.*

1. Introduction

The automotive and navigation industry shows increasing interest in the development of computer vision systems [5, 13]. More and more cars are equipped with cameras that can sense the world around the vehicle for example by capturing video and running complex computer vision algorithms to analyze the data. In the context of *Advanced Driver Assistance Systems* (ADAS) computer vision can prevent collisions or fatal injuries by forward collision warnings [17], pedestrian recognition [6] or car detection [5]. The process of automatic map making can benefit from aggregating computer vision observations from cars driving around. Vision-based localization of traffic signs, traffic lights, road closings, accidents and parking places can be of great value for drivers and advanced driver assistance. While the detection of static objects such as traffic signs is closely related, in the current work we focus on the detection of other vehicles on the road. More precisely we focus on highway scenarios in which *rear view car detection* (Figure 1) is one of the important problems in computer vision for highly automated driving.



(a) Occlusions **(b)** Difficult lighting

Figure 1: Frames from the TME Motorway Sequence [5]. Only cars and vans are labeled; annotations are rescaled to square size.

In this work we offer a look behind the scenes for designing a high-speed object detector. Based on an exhaustive review of existing literature on time constrained object detection we select methods that find the right balance between speed and detection rate. The detection framework we propose and implement builds upon the work on integral channel features [8], WaldBoost classification [22] and publications on feature scaling approximations for fast multiscale detection [7, 1]. The core contribution is the integration of these high-performance vision algorithms into a full detection framework and the analysis of the contributions of these techniques to high-speed performance. In other words, we show what algorithms yield the most important contributions to achieving real-time detection speeds and potential sacrifices in detection accuracy. We also discuss the technical aspect of our detection framework by giving a number of implementation details.

Similar to many modern object detectors, our work utilizes rigid templates trained at a certain size (*e.g.* 64×64) that at detection time are applied in a sliding window fashion to full images. As mentioned, we focus on the development of a rear view car detector, however we emphasize that all methods could easily be extended to other object classes.

*This research was conducted at TomTom Product Lab in Eindhoven.

Regarding the vision system, we opted for a monocular camera, mounted for example on the dashboard or just below the rear view mirror. In this work we exclude the components of tracking, depth-estimation, temporal information, geometric scene information and additional prior knowledge. Each image is analyzed "as is" meaning that no information from preceding frames is used during the detection process.

The paper is organized as follows. In Section 2 we discuss a number of recent contributions on high-speed object detection that are related to our work. Section 3 introduces our feature presentation, learning framework and baseline detector. This is followed by two sections focusing on algorithmic speedups for our baseline detector: Section 4 describes WaldBoost for speeding up the classification, while Section 5 introduces the idea of feature rescaling for fast multiscale detection. Our complete detection framework, hardware setup, implementation and experimental results are presented in Section 6. In Section 7 we discuss and summarize our results.

2. Related Work

Two object detector types that are known for its excellent detection rates are the works by Felzenswalb *et al*. [10] on deformable part based models (DPM) featuring pictorial structures, and recent advances on convolutional neural networks [14]. However these methods tend to focus more on achieving the highest detection accuracy while neglecting the need for speed. For example we report processing times of more than 2 seconds per 1024×768 image using the DPM detector on a modern laptop. For the current work we are interested in running object detection in real-time on embedded devices, hence we do not consider these two detection frameworks.

Providing an exhaustive overview of publications on high-speed object detection is beyond the scope of this paper. However we will summarize a number of techniques that are related to our work and we believe are important to achieving the highest possible detection speeds. Speed improvements for existing object detection frameworks can roughly be categorized into (1) improving the feature representation, (2) speeding-up the classification of subwindows, (3) focusing on fast multiscale detections and (4)computational improvements. Below we summarize important contributions to these topics in relation to sliding-window based object detector.

Feature Representation. Over the last couple of years there tends to be more focus on improving features rather than on improving classification [3]. Meaningful and efficiently extractable features are crucial to good detection rates. The first generation of object detectors frequently employed Haar features. While fast to compute using integral images, the popularity of Haar features decreased mainly due to the introduction of histograms of oriented gradient (HOG) features. A recent survey paper on pedestrian detection [2] shows that many of the high-performing detectors use a HOG-like feature representation. It seems like gradient orientation-based features are highly efficient in encoding edge, corner, key-point and local-shape information, while being reasonably fast to extract. Although HOG features with SVM classification [6] are not particularly slow, the specific normalization scheme and high dimensional feature vectors hinder computational speedups. For our detector we use integral channel features [8], which are versatile, information rich and efficient to compute.

Classification Speedups. Inspired by the cascade structure of classifiers in increasing complexity by Viola and Jones, a number of papers appeared on so-called *soft-cascades* [22, 28, 27] for speeding up boosted classification. These works focus on pruning negative subwindows as early as possible in the detection process. During the training process in addition to selecting the best features and thresholds also rejection thresholds are learned. When at evaluation time the cascade score drops below this threshold the subwindow is immediately pruned and labeled as non-object window. Typically non-promising windows are rejected early in the cascade resulting in a significant speedup when the number of windows to evaluate is large as with full image detections. We implement and study WaldBoost [22] for speeding up the classification of subwindows.

Multiscale Models. Typically for detecting objects of various size, the input images are rescaled a number of times (*e.g*. 30 scales) and features are extracted for each scale. Computing the image pyramid and extracting features at every scale requires a significant amount of computational resources. To overcome this computational burden some authors propose interpolation of features over scales to reduce the number of scales in the image pyramid. Dollár *et al*. [7] introduce a method for approximating various feature types over adjacent scales using the so-called power law for feature scaling. Since then this technique has been employed with great success for pedestrian detection [1] and general object detection using deformable part models by Saghedi *et al*. [20]. This technique is also an important aspect of our detector and dramatically reduces computation time spent on image rescaling and feature extraction (see Section 5).

Computational Speedups. In a different line of work, several improvements focusing on computational speedups have been proposed. Zhu *et al*. [29] show that HOG features can be rapidly computed using integral images. Furthermore various GPU implementations for speeding

up computer vision algorithms are described [18], and a number of low-level operations are optimized with vector operations, multiple cores and CPU cache management [20].

The work by Beneson *et al*. [1] shares the most overlap with our work. Similar to our work the authors propose a very fast object detector for pedestrian detection using integral channel features, boosting and an approximation scheme for multiscale detections. Core component for reaching high detection speed in their detection framework is the ground plane estimation technique for reducing the search space. The authors report detection speeds of 100 fps, using the ground plane estimation technique and stereo-camera. Using the stereo-camera setup and the ground plane estimation technique the authors are able to reduce the search space to 640×60 pixels regions over 10 scales, hence this allows them to run the detector at such high speeds. Our work is different since we do not consider ground plane estimation but rather focus on speeding-up the classification of subwindows.

3. Baseline Detector

Rear view car detection is a classic instance of rigid object detection. We train an object detector of fixed size and perform detection in full images using the *sliding window* methodology. Our detection framework is based on *integral channel features* [8], which combine the information richness of HOG features with the computational efficiency of Haar features. We motivate this decision by observing that – with a proper set of channels – the work on channel features is closely related to HOG features and at the same time has been shown to outperform other features computationally [8, 16].

3.1. Integral Channel Features

The general idea behind integral channel features is that multiple registered image channels $C(i, j)$ are computed using shift-invariant transformations Ω applied to an input image, *i.e*. $C(i, j) = \Omega[I](i, j)$. First-order integral channel features f_Ω are defined as the sum of pixels in a fixed rectangular region R in a single image channel C:

$$f_\Omega(I) = \sum_{i,j \in R} C(i, j) \tag{1}$$

Using integral images for each channel such features are extremely fast to compute. Experiments in [8] show that the combination of color + gradient magnitude + six gradient orientation channels (Figure 2) typically result in the best detection performance. The LUV color space is superior compared to other color spaces and the addition of more than six gradient orientation channels shows no improvements in detection rate for pedestrian detection [8].

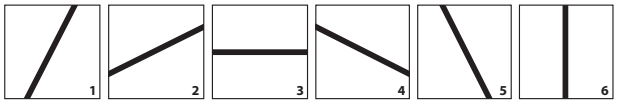


Figure 2: Gradient orientations ("unsigned" versions are utilized)

3.2. Learning Framework

The training process of our detector is similar to Viola-Jones [25] and the work by Dollár *et al*. [8]. Rather than carefully designing features we generate a large feature pool of random features and adopt **Discrete AdaBoost** [11] for greedy feature selection. Each feature is described by a randomly generated rectangle with minimum area of 25 pixels and a randomly selected channel index 0–9. In each training stage t the best *weak learner* h_j restricted to a single feature j is trained by optimizing the decision threshold and polarity. The strong classifier H_T returned by the training process is an ordered sequence of all T weak learners in which the associated weight α_t assigned to each weak learner is inversely proportional to the training error of that stage. Deciding upon the label $y = \{-1, +1\}$ of an object x is done by thresholding the cascade score $H_t(x)$ with is linear combination of the weak classifiers:

$$H_t(x) = \sum_{t=1...T} \alpha_t \cdot h_t(x) \tag{2}$$

For all experiments reported in this paper we set $T = 400$ (feature dimensionality). For 640×480 images at 25 scales our baseline detector spends approximately 4.5% of the processing time on image rescaling and channel extraction, 32% on feature extraction and 63% on classification of subwindows. However the actual contribution to the total processing time per image is different since feature extraction and classification run in a parallelized loop.

4. Soft-Cascade Classification

Cascade detections have been shown to operate extremely fast making them a good choice for high-speed object detection. Fast rejection of non-promising subwindows is crucial to high detection speeds, motivated by the observation that the majority of tested subwindows does not contain the object of interest. Since the original introduction of the cascade structure by Viola-Jones [25] many publications have been devoted to improving cascade learning and detection. One particular problem of the original cascade structure is determining the optimal target detection rates for each stage of the cascade. This is one of the problems addressed by *soft-cascades*.

The structure of soft-cascades is different to the cascade proposed by Viola-Jones since only one fine-grained classifier of T weak learners is constructed. Various soft-cascade

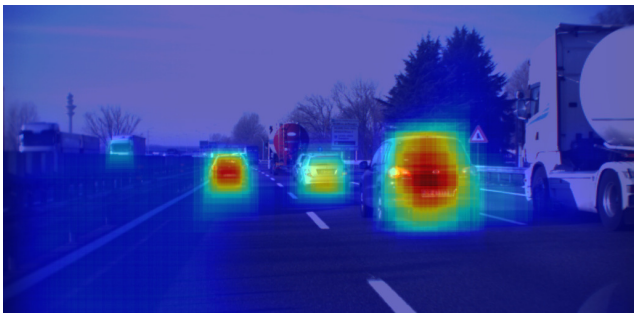


Figure 3: WaldBoost attention heatmap. For each pixel in the images we visualize the number of weak classifiers that was evaluated. For blue regions the subwindows were rejected almost immediately, while for red regions the entire cascade was evaluated.[1]

approaches [22, 28, 27, 4] haven been described, each focusing on learning stage thresholds $\theta(t), t \in \{1,\ldots,T-1\}$ for rejecting non-promising subwindows at the earliest stage possible. Zhang and Viola [28] formulate the process of learning stage thresholds as multiple instance learning problem (MIP). This approach is successfully adopted in detection frameworks [8, 7] due to its simplicity and effectiveness. For our work we choose to adopt WaldBoost as soft-cascade approach to speedup classification. Experiments [28] have demonstrated that WaldBoost is faster than MIP for rejecting subwindows, and we appreciate the statistical foundation of WaldBoost, *i.e.* the sequential probability ratio test used for calculating stage thresholds is proven optimal, while the other methods are more designed based on intuition and empirical observations.

4.1. WaldBoost

WaldBoost is an extension of AdaBoost and formulates the classification problem as a *sequential decision process*. Based on the current cascade score H_t (*trace*) the sequential decision process S_t offers three possible options:

$$S_t : (h_1, h_2, \ldots, h_t) \rightarrow \{-1, +1, \sharp\} \tag{3}$$

Where $\sharp$ means 'continue' when the object is undecided from the first t measurements. The sequential decision strategy is formulated to minimize the average decision time $\overline{T}_S = E[T_S(x)]$, where the decision time $T_s(x)$ denotes the minimal t for which $S_t \neq \sharp$. To meet system requirements, the user specifies the allowed false negative rate α (*e.g.* $0.5 \cdot 10^{-4}$) and allowed false positive rate β (typically set to 0 for object detection). Wald [26] has proved that the optimal strategy for this decision problem can be expressed

[1]For a video visualization of the WaldBoost attention heatmap constructed for this article, see: `https://www.youtube.com/watch?v=4OlITWa1bTY`

Table 1: Comparison in the number of weak classifiers evaluated for AdaBoost and WaldBoost per subwindow. For AdaBoost the label for each subwindow is determined after processing all $T = 400$ stages. On average WaldBoost only requires 2.1 weak classifier evaluations per subwindow. Statistics are computed over 200 images 1024×768.

	AdaBoost	**WaldBoost**
Windows per image	$2.3 \cdot 10^5$	$2.3 \cdot 10^5$
Tot. features extracted	$9.4 \cdot 10^7$	$4.8 \cdot 10^5$
Avg. stages per window, $\overline{T}_S$	400	**2.08**

in terms of the likelihood ratio $R_t(x)$:

$$R_t(x) = \frac{P(h_1(x),\ldots,h_t(x) \mid y=-1)}{P(h_1(x),\ldots,h_t(x) \mid y=+1)} \tag{4}$$

Given the likelihood ratio $R_t(x)$ and constraints on the error-rates, Wald proposes the *sequential probability ratio test* (SPRT) for setting rejection thresholds $\theta_A^{(t)}, \theta_B^{(t)}$ in each stage of the decision process. To avoid the computation of $R_t(x)$ involving a high dimensional density estimation, Šochman and Matas [22] suggest projection of the t-dimensional space into a one-dimensional space using the strong classifier score up until the current stage (Equation (2)). This simplifies the density estimation of $P(h_1(x),\ldots,h_t(x) \mid y=\pm 1)$ in Equation (4) to a one-dimensional density estimate for which Parzen windows are adopted. After estimating the probability densities, stage thresholds can be determined using α, β. For all experiments in this paper we set $\alpha = 0.005$ and $\beta = 0$, *i.e.* we allow 0.5% false positives and no false negatives. See Figure 4 for an illustration of the stage threshold estimation process. During the detection phase, stage thresholds $\theta_A^{(t)}, \theta_B^{(t)}$ are applied for rejecting subwindows as early as possible:

$$S_t^* = \begin{cases} +1 & \text{if } H_t(x) \geq \theta_B^{(t)} \\ -1 & \text{if } H_t(x) \leq \theta_A^{(t)} \\ \sharp & \text{if } \theta_A^{(t)} < H_t(x) < \theta_B^{(t)} \end{cases} \tag{5}$$

Using this strategy most of non-promising subwindows are rejected after evaluating only one or two stages. The visualization in Figure 3 illustrates this effect, while Table 1 shows significant reduction in the average number of weak classifiers $\overline{T}_S$ evaluated in comparison to AdaBoost. Extending our baseline detector with WaldBoost decreases the time spent on feature extraction to 22%, classification to 40% and consequently increases the fraction of time spent on channel computation to 38%.

5. Fast Multiscale Detections

Conventional object detectors construct an image pyramid to detect objects with variable sizes. The computation

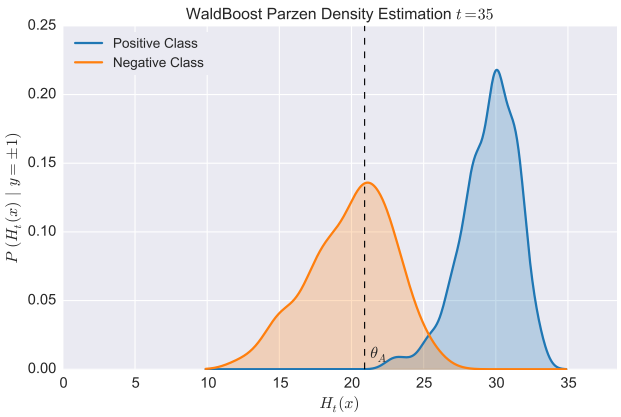


Figure 4: Parzen density estimate of $H_t(x)$ distribution computed on the validation dataset (500 examples). Wald's SPRT is used for computing stage threshold θ_A given the user-specified α.

of the image pyramid and feature computation for each scale typically is the bottleneck in modern object detectors. Based on important works on scale space [15] and experiments on scale behavior of image statistics [24], Dollár *et al*. [7] propose an elegant technique for multiscale feature approximation to speedup multiscale detections.

Early studies [19, 23] on image statistics show that natural image spectra follow a power law. In context of object detection using integral channel features, let I be an image available at scales s_1 and s_2 denoted by I_{s_1} and I_{s_2} and a channel feature f_Ω. The power law describing scale behavior of image statistics over an ensemble is the foundation of the work by Dollár *et al*. describing the power law for feature scaling which expresses that the ratio between features at two different scales is only a function of the relative scale difference:

$$\frac{f_\Omega(I_{s_1})}{f_\Omega(I_{s_2})} = \left(\frac{s_1}{s_2}\right)^{-\lambda_\Omega} + \varepsilon \tag{6}$$

Where ε indicates the deviation from the power law and λ_Ω is the power law parameter characteristic for a certain channel Ω. Originally the validity of this power law was found to hold for *ensembles* of images, however it was shown by [7] that this also holds for individual images by decomposing the image into an ensemble of patches (small images). For fast multiscale detections Dollár *et al*. [7] rescale the input image two times per octave (*base scales*) and exploit the feature approximation relationship for interpolation to nearby scales. This significantly reduces the number of rescalings and time spent on feature extraction. For each channel λ_Ω was computed using the methodology as explained in [7].

5.1. Object Detection without Image Resizing

Beneson *et al*. [1] built upon the multiscale feature approximation techniques by proposing object detection without image resizing. Their core idea is to move resizing of the image from test time to training time. Similar to their detection framework we train our models at four different scales, *i.e*. $s = 0.5, 1, 2, 4$ – While this is not practically feasible for many object detectors due to excessive training times, our detector trains in approximately 40 minutes. Consequently training the detector at four different scales can be done in 2.5 hours on a fast desktop using a parallel implementation. At runtime we use the four models at different scales as base scales and apply the power law for feature scaling (6) to approximate intermediate scales.

One problem that arises when training the detector at multiple scales is the availability of training examples in high resolution. In our current setting the training examples are available at 80×80, but our largest detector model is trained at 256×256. This unavoidably causes negative upscaling effects certainly resulting in a decrease in detection performance. In the ideal case, training examples at all scales should be available although we recognize the difficulty of putting such dataset together.

6. Detection Framework and Experiments

Bringing together all the parts discussed in the previous sections we put together our object detector. The architecture of our full detection framework is displayed in Figure 5. Upon receiving a video frame from the camera the image is sent to GPU memory. OpenCL filter kernels compute the 10 registered image channels and send these back to CPU memory. Surprisingly enough computing the *integral* channels on the GPU and then pushing these back to CPU was found to be slower than our current implementation. This can be explained by the fact that 10 integral channels are quite large memory objects hence the data bandwidth between CPU and GPU becomes the bottleneck. Once computation of the integral channels is finished, the classification of all subwindows over all scales is started in parallel. We emphasize that *no* image rescaling is performed but rather we use the models trained at 4 different scales and use these 'base' scales for our approximations to other scales. After processing all scales, nearby detections are combined using a simple non-maxima suppression method described in [8].

Hardware. Experiments for this paper are performed on a MacBook Pro (late 2013) running OS X 10.9 – This laptop contains a dual-core Intel i5-4258U processor running at 2.4GHz and is supported by an integrated Intel Iris 3000 GPU running at 1.2GHz over 280 processing units. The detector proposed in this work was designed to run on an

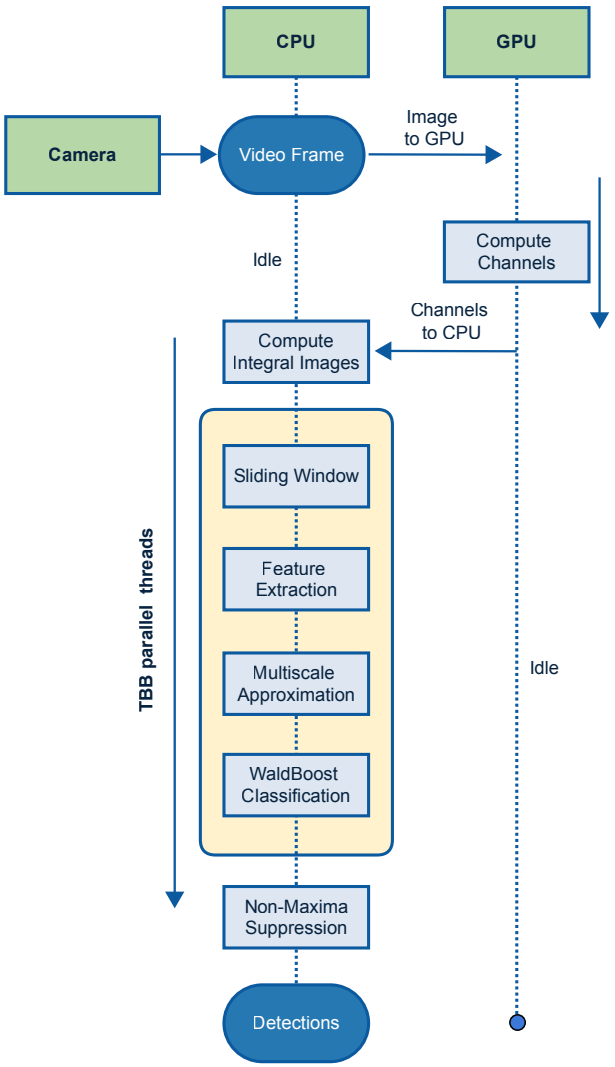


Figure 5: Architecture of our object detection framework. In our current implementation the GPU only applies filter kernels for computing the 10 image channels. Feature extraction, multiscale approximations and classification are performed on the CPU in a parallel implementation.

embedded platform, although unfortunately we have not yet performed experiments on an embedded device. Our specific embedded platform is the Jetson TK1 platform featuring the NVIDIA Tegra K1 processor. However NVIDIA has not released OpenCL support for the Tegra K1 platform (Linux for Tegra) making it impossible to run our algorithms relying on OpenCL.

Implementation. Our detector is written from scratch in C++11 building on top of OpenCV 3.0.0, although we only use its core functionality.

Figure 6: Top: positive training samples, in total 2000. **Bottom:** randomly selected negative training examples not containing cars and vans, in total 4000.

Intel's Task Building Blocks (TBB) was selected for CPU parallelization. For GPU parallelization we choose OpenCL in favor of CUDA. The flexibility and portability of OpenCL is more important to us than the minor speed increase CUDA has to offer at the cost of vendor lock-in.

Image Channels. For extraction of our channel images we utilize fast OpenCL kernels to run image processing operations. In our implementation we use the unsigned gradient orientations displayed in Figure 2, extracted using simple $[-1,0,+1]$ and $[-1,0,+1]^T$ filter kernels without prior smoothing [6]. The cube root $\sqrt[3]{x}$ calculations required for LUV color conversion are speedup using Halley's method [12] for providing a fast approximation with small error. The gradient orientation computation $\Theta = \text{atan2}(G_y, G_x)$ using outputs G_x, G_y from the derivative masks is speedup using an approximation for the inverse tangent described in [21]. The extraction of all 10 channels using our GPU implementation and downloading the channels back to CPU memory runs at 40 fps for 1024×768 images and 92 fps for 640×480 images on our hardware. Note that this is almost five times as fast as the original implementation [8] reporting channel extraction running at 40 fps on 320×240 images.

Training Data. Our training data consists of 2000 rear view car samples extracted from panorama images captured by TomTom mobile mapping (MoMa) vehicles. Uniformly at random we have sampled a large collection of panoramas captured by the fleet of MoMa cars within The Netherlands. From these panoramas all rear view cars and vans are manually labeled and bounding boxes are extracted at size 80×80 to serve as training data for our detector. Additionally 4000 random patches, not containing cars or vans, are extracted as negative training data. Figure 6 displays a subset our training examples.

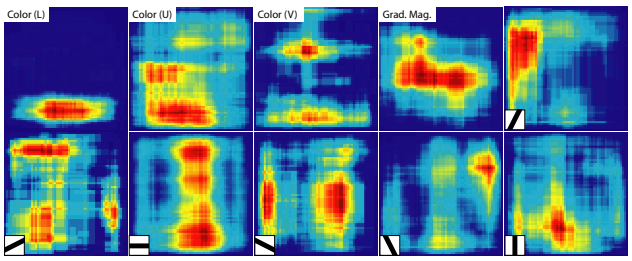


Figure 7: Feature selection by AdaBoost for rear view car detection. The gradient orientation channel corresponding to horizontal lines accounts for 25% of all features, far more than the other channels. These gradients have the strongest response around the license plate and near the roof of the car. The first color channel focuses on the dark shadow on the ground just below the car.

Test Data. For the evaluation of our object detector we report detection results for the TME Motorway Sequence [5]. The videos are captured by a dashboard mounted camera while driving highways in Northern Italy at 1024×768 (20 Hz). The total of 9 daylight sequences include various traffic situations, number of lanes, road curvature and lighting conditions. Two example images are displayed in Figure 1. For evaluation we computed statistics over 465 video frames each time with an interval of 50 intermediate frames. Cars and vans with more than 50% occlusion are excluded as ground truth.

Evaluation Metrics. The PASCAL VOC overlap criterion [9] is used for matching detections to ground truths and for determining the number of true positives (TP), false positives (FP) and false negatives (FN). To be considered a correct detection, the overlap ratio between the predicted bounding box and the ground truth bounding box must exceed 50% according to PASCAL VOC definition [9].

Training Time. Training of our baseline detector and learning stage thresholds using WaldBoost is fast in comparison to other detectors: training the detector at a single scale with 600 weak classifiers and an initial feature pool of 40.000 features takes about 40 minutes on a powerful desktop (Intel Core i7-2600). Note that the training process runs completely on the CPU and is parallelized using TBB. Prior to the training process all feature values for all training samples are precomputed and loaded into RAM, resulting in fast training stages. Training our multiscale detector on 4 scales finishes in 2.5 hours.

Learned Features. An interesting observation made from analyzing the features learned for rear view car detection is the fact that over 25% of the channel features correspond to vertically oriented directional derivatives, *i.e.* they correspond to horizontal edges (see Figure 2). Remaining color and gradient magnitude and orientation channels each account for 5–12% of the total number of features.

Figure 8: Detection quality comparison on TME Motorway Sequence. As comparison we show the quality of rear view car detections for the Felzenswalb detector [10] (OpenCV implementation).

Table 2: Speed benchmark over 640×480 images analyzed at 30 scales. Runtimes are averaged over 200 video frames; for all settings the strong classifier consists of 400 weak learners.

	Relative Speed	**Absolute Speed**
Baseline (ICF + AdaBoost)	1.0×	1.3 fps
+ WaldBoost	2.6×	3.2 fps
+ Multiscale Approximations	17.5×	56.0 fps

Detector Performance. The detection quality of our detector on the TME Motorway Sequence is given as precision-recall curve in Figure 8. In Table 2 we report the detection speed of our detector. The speedup on per-component basis shows that both the soft-cascade and multiscale approximations are essential for reaching these speeds. Not surprisingly adding more scales is very cheap since by using multiscale approximations no image rescaling is required.

7. Discussion and Conclusion

In this work we have described the system design of our object detector. We show that it is possible to perform high-speed object detections at 56 fps on a laptop without including prior scene information or reducing the search space. Both WaldBoost and multiscale approximations are imperative for reaching these speeds. We acknowledge a decrease in detection rate of about 10% with our implementation of multiscale approximations. Possible explanations for this drop in performance are (1) the fact that our trainings images

are not available in high resolution, (2) some features are too small causing the feature approximations to be inaccurate and (3) detection thresholds for the strong classifier are not optimized and are currently the same for each scale.

For additional detection speedups we believe two strategies are the most promising: (1) streamline the detection pipeline and (2) for further work we could incorporate search space techniques to further speed up. For streamlining the detection pipeline we propose moving the classification task to the GPU. For devices without *shared memory* between CPU and GPU this has the advantage of not pushing images between CPU and GPU. Additionally this idea is motivated by the fact that subwindows can be classified independently hence massive parallelization offered by the GPU processing units is beneficial. An alternative idea for speeding up the detector would be to use the WaldBoost heatmap displayed in Figure 3 over a number of frames to build a *occurrence probability map* and include this as prior information for reducing the search space.

Beneson *et al*. [1] use stereo-vision to build a "stixel world" to reduce the search space dramatically. Their stixel world reduces the search space to 640×60 pixels $\cdot\, 10$ scales. At this particular setting the authors report speeds of 135 fps on a high-end laptop featuring Intel Core i7 and dedicated GPU (faster than ours). As comparison we also reduced the search space by simply excluding top and bottom part of the image to 640×130 pixels $\cdot\, 25$ scales. After this search space reduction our detector runs at 150 fps without sacrificing on detection quality. To our best knowledge our detector is one of the fastest object detectors reported in computer vision literature. For future work we are interested in a direct comparison in detection quality and explore different approaches for search space reduction.

References

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

[2] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3666–3673. IEEE, 2013.

[3] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? *arXiv preprint arXiv:1411.4304*, 2014.

[4] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 236–243. IEEE, 2005.

[5] C. Caraffi, T. Vojir, J. Trefny, J. Sochman, and J. Matas. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera. In *ITS Conference*, pages 975–982, Sep. 2012.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[7] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(8):1532–1545, 2014.

[8] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC*, volume 2, page 5, 2009.

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. 88(2):303–338, June 2010.

[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[11] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[12] W. Gander. On halley's iteration method. *American Mathematical Monthly*, pages 131–134, 1985.

[13] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994.

[16] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool. Traffic sign recognition—how far are we from the solution? In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–8. IEEE, 2013.

[17] K.-Y. Park and S.-Y. Hwang. Robust range estimation with a monocular camera for vision-based forward collision warning system. *The Scientific World Journal*, 2014, 2014.

[18] V. Prisacariu and I. Reid. fastHOG-a real-time gpu implementation of HOG. *Department of Engineering Science*, 2310, 2009.

[19] D. L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814, 1994.

[20] M. A. Sadeghi and D. Forsyth. 30hz object detection with dpm v5. In *Computer Vision–ECCV 2014*, pages 65–79. Springer, 2014.

[21] J. Shima. DSP Trick: Fixed-Point Atan2 With Self Normalization. 1994.

[22] J. Sochman and J. Matas. Waldboost-learning for time constrained sequential detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 150–156. IEEE, 2005.

[23] D. Tolhurst, Y. Tadmor, and T. Chao. Amplitude spectra of natural images. *Ophthalmic and Physiological Optics*, 12(2):229–232, 1992.

[24] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: computation in neural systems*, 14(3):391–412, 2003.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[26] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[27] R. Xiao, L. Zhu, and H.-J. Zhang. Boosting chain learning for object detection. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 709–715. IEEE, 2003.

[28] C. Zhang and P. A. Viola. Multiple-instance pruning for learning efficient cascade detectors. In *Advances in Neural Information Processing Systems*, pages 1681–1688, 2008.

[29] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 1491–1498. IEEE, 2006.