

CZECH TECHNICAL UNIVERSITY IN
PRAGUE

FACULTY OF ELECTRICAL ENGINEERING



BACHELOR THESIS

**Car Recognition in Cross-traffic
Video-sequences**

Prague, May 2011

Aram Simonian

Acknowledgement

Foremost, I would like to thank to Ing. Martin Matoušek, PhD. for his excellent leadership, improving ideas and valuable advice. I would also like to express great gratitude to my family for their support, patience and calm study environment they provided.

Abstract

This thesis deals with passenger car recognition in individual frames of cross-traffic video sequences. Firstly, we have suggested characteristic visual features of the car object class that are suitable for detection from the side-view. Secondly, we have designed, implemented and trained detectors of these individual features. Thirdly, we have suggested a structural model of car side-view, which allowed us to integrate the detectors of the individual features together. Finally, we have proposed a probabilistic fusion of the visual features and structural model. The probabilistic fusion is then used for final detection of cars as whole objects.

Abstrakt

Tato práce se zabývá rozpoznáváním osobních automobilů v jednotlivých rámcích videosekvencí z příčného dopravního provozu. Nejdříve jsme navrhli optické rysy charakteristické pro boční pohled na automobil, které by byly vhodné k detekci. Pro tyto jednotlivé rysy jsme navrhli, naimplementovali a naučili elementární detektory. Dále jsme navrhli strukturní model bočního pohledu na automobil, který nám umožnil propojit dohromady detektory jednotlivých rysů. Nakonec jsme vytvořili pravděpodobnostní model spojující optické rysy a strukturní model. Tento pravděpodobnostní model jsme následně použili pro výslednou detekci automobilů jako celků.

BACHELOR PROJECT ASSIGNMENT

Student: Aram Simonian
Study programme: Electrical Engineering and Information Technology
Specialisation: Cybernetics and Measurement
Title of Bachelor Project: Car Recognition in Cross-traffic Video-sequences

Guidelines:

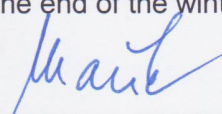
1. Perform a bibliography search on the topic, learn about the state-of-the-art and relevant publications.
2. Study the current method for detection of cars in video-sequences. The method is based on detection of features – wheels – using an AdaBoost classifier, and uses the features in structural model for car side-view.
3. Suggest additional features suitable for the detection and engagement in the structural model for side-view.
4. Implement detection of individual features (including the creation of adequate training and test sets). All implementations should be done in Matlab. Focus on passenger cars only.
5. Use the particular feature detectors in structural model for car detection. Propose and implement the final detector of car side-views.
6. Verify experimentally the performance of particular feature detectors and the final detector as well.

Bibliography/Sources:

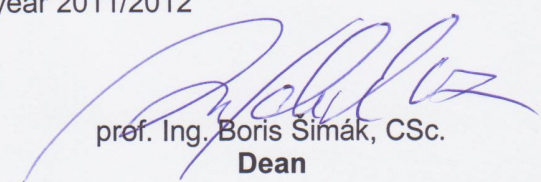
- [1] Duda, R.; Hart, P.; Stork, D.: Pattern Classification. John Wiley & Sons, 2001.
- [2] Bishop, C.: Pattern Recognition and Machine Learning. Springer Science+Business Media, 2006.

Bachelor Project Supervisor: Ing. Martin Matoušek, Ph.D.

Valid until: the end of the winter semester of academic year 2011/2012


prof. Ing. Vladimír Mařík, DrSc.
Head of Department




prof. Ing. Boris Šimák, CSc.
Dean

Prague, February 2, 2011

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Aram Simonian
Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný
Obor: Kybernetika a měření
Název tématu: Rozpoznávání automobilů ve videosekvencích z příčného dopravního provozu

Pokyny pro vypracování:

1. Provedte bibliografickou rešerši na zadané téma práce, seznamte se se stavem problematiky a relevantními publikacemi.
2. Seznamte se se stávající metodou detekce automobilů ve videosekvencích: detekce typických rysů - kol - detekovaných klasifikátorem AdaBost a jejich využitím ve strukturním modelu popisujícím boční pohled na automobil.
3. Navrhněte další rysy vhodné k detekci a k zapojení do strukturního modelu pro boční pohled. Zaměřte se pouze na osobní automobily.
4. Naimplementujte detektory jednotlivých rysů (včetně vytvoření dostatečných trénovacích a testovacích množin). Veškerou implementaci provádějte v prostředí Matlab.
5. Zapojte jednotlivé detektory rysů do strukturního modelu pro detekci automobilu a navrhněte a implementujte celkový detektor automobilů.
6. Experimentálně ověřte výkonnost jednotlivých detektorů rysů a výkonnost celkového detektoru automobilů.

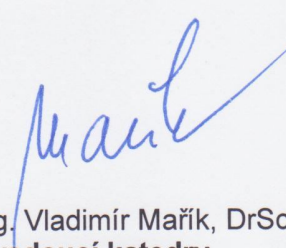
Seznam odborné literatury:

- [1] Duda, R.; Hart, P.; Stork, D.: Pattern Classification. John Wiley & Sons, 2001.
[2] Bishop, C.: Pattern Recognition and Machine Learning. Springer Science+Business Media, 2006.

Literaturu [1] a [2] poskytne vedoucí práce. Další literatura vzejde ze studentem vypracované rešerše.

Vedoucí bakalářské práce: Ing. Martin Matoušek, Ph.D.

Platnost zadání: do konce zimního semestru 2011/2012


prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry




prof. Ing. Boris Šimák, CSc.
děkan

V Praze dne 2. 2. 2011

Contents

1	Introduction	1
1.1	State of the art	2
1.2	Suggested approach overview	5
1.3	Notation	7
2	Visual appearance features	9
2.1	Detection at a single scale	9
2.2	Components for detection	9
2.3	Wheel detection	10
2.3.1	Training samples for wheel detection	11
2.3.2	Building a strong classifier with AdaBoost	12
2.3.3	Masks for wheel detection	14
2.3.4	Detection of wheel candidates	15
2.3.5	Learning process and optimisation	18
2.3.6	Transforming wheel detector responses to probability	20
2.4	B-pillar detection	21
2.4.1	B-pillar detector training	21
2.5	A-pillar detection	22
2.5.1	AdaBoost based detection	23
2.5.2	Edge based detection	25
2.5.3	Transformation of oriented angles to probability	26
2.6	Side panel verification	28
2.7	Contribution	31
3	Structural model	33
3.1	Learning the structure	34

3.2	Probabilistic representation of the structure	35
3.3	Contribution	35
4	Final detector	37
4.1	Labelling of detections and its probability	38
4.2	Labelling evaluation optimisation	40
4.3	Multi-scale detection	40
4.3.1	Merging detections across scales together	41
4.4	Contribution	42
5	Experiments	43
5.1	Data for experiments	43
5.2	Experimental results	44
6	Conclusion	49
A	Contents of the attached CD	I

Chapter 1

Introduction

Various assistance systems are being integrated into cars these days to support the driver. The aim of these systems is to contribute to the traffic safety, either in passive (information/warning) or active (intervention) way. In addition, the assistance systems can share information and cooperate with each other (car-to-car systems). In order to be able to deal with the environment, the intelligent vehicles must be equipped with various sensors allowing them to observe their surrounding. These sensors include radar, lidar, sonar, stereo vision or monocular vision systems. We focus on the latter one.

The monocular vision systems have the advantage that they are passive. The monocular camera is fairly small and encapsulated to be easily integrated into the vehicle. In addition, these systems are cheap enough to be used in wide range of car models. However, it is still a problem for a machine to understand the obtained data in global context. In our case, this means to give a desired interpretation to an image – not to take the image only as a set of pixels, but also recognise some objects it contains. This is a typical problem of pattern recognition, easy for a human being, difficult for a machine.

In this work, we describe design of a detector of cars in cross-traffic video-sequences. A car equipped with a camera and such detector could “see” the other cars and therefore would be able to actively prevent collisions. We will focus on side crashes, which are quite frequent in street traffic. Following a typical scenario depicted in Figure 1.1, a car approaches to crossroads and

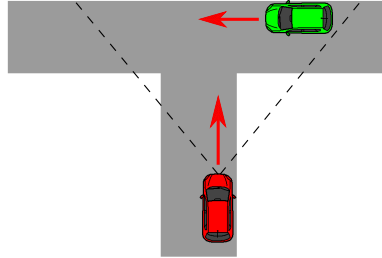


Figure 1.1: Typical scenario for car side-view detector application

monitors the traffic situation ahead. If the system detects another car coming in cross-direction it can assess velocities of both vehicles and activate brakes in case of imminent danger of collision.

The fact that we focus only on cross-traffic and side crashes simplifies the recognition problem. We will only have to detect the cars from side-view. If we didn't know the orientation of the car in the image a priori, the problem would be far more difficult, because we would have to deal also with rotation in space beside the variability of the appearance of cars.

1.1 State of the art

Car detection is an object recognition problem, whose purpose is to determine, if the input image contains a car(s), or not. If there are some cars in the image, we also want to mark their position. The problem can be solved using different approaches that vary in the representation of the detected object, representation of the input image and machine learning methods for detector training.

There are two main types of object representation. The global one, which treats the detected objects as compact units, and the component- or part-based one that understands object as a geometrically organised set of component features. The part-based models consist of a set of component detectors that are encapsulated into a higher level detector. The two approaches can be combined like in [3], where they use a global root filter which covers whole detected object plus a set of part filters and spatial model for the location of each part with respect to the root. The score of detector

is then given as sum of scores of the root and part filters minus the deformation cost. The part-based models are more robust against pose and shape changes than the global ones ([4]). The challenge of the part-based models is to choose the components suitable for detection. The components must be at once characteristic for the whole detected object class and distinguishing from the background. The authors of [3] prefer training the models on images labeled only with bounding boxes around the objects of given class. They claim that training with automatic component labelling has potential to achieve better performance than training with manually labeled components. Richer labelling can result in better training, but it can also lead to worse performance of the final classifier, if the labeled components are not optimal. The problem of the automatic component labelling is that it needs large training sets. There are annotated databases of car images available, such as the PASCAL Object Detection Challenge Dataset or StreetScenes Dataset, but these databases are not useful for our purpose, because they contain cars in various poses or partially occluded. In addition, we have a wide field of view (FOV) in our application, which yields lower image resolution per a single car. Therefore we decided to choose and annotate the components in training data manually, so we also had to think of the components for detection.

A detector classifies the input image evaluating a feature vector that represents the information carried in it. We have to choose a suitable feature space, into which the input will be transformed before classification. There are several commonly used feature spaces including grayscale pixel intensities, histograms of oriented gradients (HOG) or Haar wavelet coefficients. Each of them fits a different object class. HOG features were successfully applied in [3] for detection of different object classes including cars. Their advantage is that they cover also the statistic information between adjacent pixels of the image. According to [5], grayscale pixel intensity values work well enough for the car class and for some data sets even better than HOG features. The image transformed to chosen feature space is subsequently classified on the principle of sliding window, which is based on application of a filter at all positions and scales of the image.

When designing an object detector, we also have to choose a classifier that will be used for detection. According to [1], the classifiers can be divided into two major groups: generative and discriminative. The two groups differ in the method of solving the decision problem, which requires determining the posterior probability $p(y|x)$ of labelling y given observed data x . A generative model learns the joint probability $p(x, y)$ and prior distributions $p(x), p(y)$ from training data and then uses the Bayes' rule to assign observed data to a class with highest posterior probability. However, large training sets are needed to find the joint distribution $p(x, y)$ and priors $p(x), p(y)$ with sensible accuracy, especially for high dimensional x . If we only want to make classification decisions (object is either included or excluded from a class), the posterior probabilities $p(y|x)$ are sufficient and we do not need the joint distribution at all. A discriminative model determines the posterior probability directly from the observed data without any assumption about their probability. The discriminative models are more popular than the generative ones, because they are less difficult to train.

The most common discriminative training methods for object detection are Support Vector Machines (SVM) and boosting algorithms including the AdaBoost. These supervised learning methods can be used for training a global or component detector, as well as for composition of component detectors into a part-based model. SVM algorithm constructs an optimal hyperplane in a high-dimensional feature space of training samples, which separates positives and negatives. If the problem is not linearly separable, the samples are mapped to a higher-dimensional feature space in which the separability is ensured. Further description of SVM can be found in literature [1, 2]. The boosting algorithms construct a strong (boosted) classifier from a set of weak classifiers (weak learners). The weak classifiers can be based on different principles (linear classifier, k-nearest neighbour, etc.). The only condition for a weak learner is, that it has to perform at least better than chance. Note that if the weak learners have high accuracy on the training data set, the boosting effect will be small. The most popular variant of boosting is AdaBoost, which is described in Section 2.3.2.

There already exist sophisticated systems for object detection including car, pedestrian or face recognition. There are even general detection systems

that can be trained to recognise different object classes, like the one presented in [3]. These systems provide an excellent detection accuracy and also certain robustness against class variances. However, they are also very complex and the question is, whether our problem is worth of such complexity. We attempt to design a simpler detection method that could be implemented for real-time video processing, which is necessary for the intended area of application. In addition, we are not so interested in highly precise object location in the input image as the authors of [6]. It is not that important for us if we detect a car a few pixels next to its real position, but the more precise detection we get, the better, because the detector could be later extended with some kind of object tracking.

1.2 Suggested approach overview

We decided to adopt a component-based detection approach using a structural model, which is often referred to as bag-of-features in literature. This approach is based on detection of several characteristic features that are relatively invariant within the object class and their subsequent integration into the structural model. Its advantage is that it makes use of the information that is carried in the structure of the car, i.e., not only in the visual appearance of its components, but also in their relative displacement.

In comparison with the “whole object” detection, this approach has better ability of generalisation when being learnt from a relatively small training sets. Similarly to the systems like [3], it provides some robustness against class variability, which is directly what we need, because we want to detect cars and the cars are quite variable in shape, colour and size. This is obvious even if we only consider types of car body - limousine, hatchback, waggon, SUV, MPV, coupé etc.

We have transformed the problem of car detection in a video-sequence to the problem of car detection in every single video frame. Our detector does not use any mutual information between the adjacent video frames. The detector consists of three major modules:

1. visual appearance of components,

2. structural model of car side-view,
3. probability model for final detection.

The visual appearance of components is represented by component detectors, whose principles and learning are described in Chapter 2. The structural model presented in Chapter 3 captures the information carried in the structure of the car and allows us to build and use the probability model for final car side-view detector described in Chapter 4.

The proposed detection method is divided into several consecutive stages:

- **Step 1:** Detection of single wheel candidates. We use wheel component detector to locate wheel candidates in the input image.
- **Step 2:** Formation of car candidates. We form pairs of wheel candidates using geometrical constraints given by the structural model. These pairs act as whole car candidates in further stages of detection.
- **Step 3:** Verification of the car candidates. We evaluate the rest of components (the A-pillar and side panel) corresponding to every car candidate.
- **Step 4:** Probabilistic fusion. We transform all measurements performed on the input image to probabilities and use the probability model to find the best labelling of car candidates.
- **Step 5:** Merging of different scales. The detection in previous steps is done for vehicles that have one size in the image – i.e., for each particular scale separately. The final interpretation of the input image is acquired by merging the optimal labellings across all inspected scales.

The whole process described above is performed independently on every single video-frame. By measurements mentioned in **Step 4**, we mean the component detector responses, as well as relative displacement of component detections. The probabilities of component detector responses are learnt from distributions of the responses on annotated training samples, whereas the probabilities of relative displacements of the components represented by the structural model are learnt from displacements of the annotated training samples.

1.3 Notation

We will use lowercase bold Roman letters such as \mathbf{x} to denote vectors. Uppercase bold Roman letters such as \mathbf{X} will denote matrices. When referring to size of matrices or images, as well as to coordinates of an element of matrix or image, we will preserve the usual convention $x \times y$ for size and (x, y) for coordinates, where x denotes horizontal size or column index and y denotes vertical size or row index. The origin of system of coordinates is always located in the upper left corner of matrix or image.

Chapter 2

Visual appearance features

In this chapter, we will describe the component detectors we have designed. We will show the principles and training of the component detectors, as well as subsequent component detection and transformation of detector responses to probabilities that will be used in probability model for final detection described later in Chapter 4.

2.1 Detection at a single scale

The component detectors are learnt on training samples normalised to a defined size. Nevertheless, when using the detector, we do not have granted that all cars in the scene are in the same viewing distance, so we have to process each frame at multiple scales, too. To preserve simplicity, all examples of detections provided in this chapter are computed at the same scale, at which we have trained the detectors. The multi-scale detection is described in Chapter 4.

2.2 Components for detection

Undoubtedly, the best component for detection are the wheels. Other suitable features proposed in [5] include roof, rear view mirror and the "side panel", which is the area between wheels containing lower part of doors, usually a strong horizontal edge and the shadow under the car. We had to reject

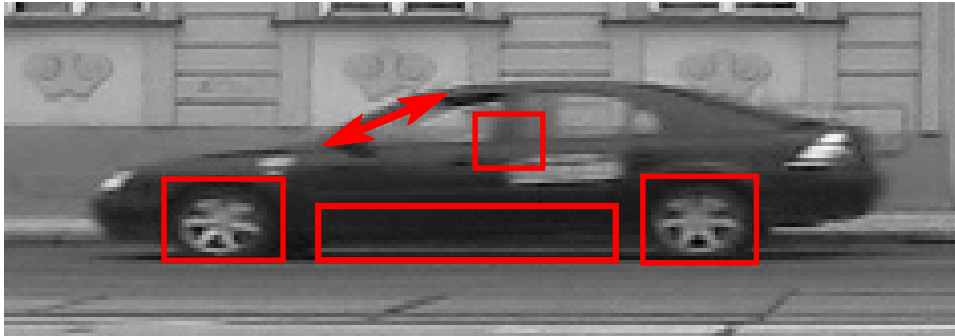


Figure 2.1: Components suggested for detection: wheels, A-pillar, lower part of B-pillar and side panel

the rear view mirrors, because we have low resolution per one car due to the wide FOV, so the mirrors are too small to be detected. At first, we had not chosen neither the roof, neither the side panel, because they are both characterised by horizontal edges, which are quite common in the scene, especially in the urban environment. However, we later returned to the side panel and used it for verification of detections.

Instead of the rejected components we suggested the A-pillar and the lower part of B-pillar with the upper part of doors in addition to wheels. The main advantage of the A-pillar is that sloping edges are not very usual in the scene, so it is quite a well distinguishing feature. During experiments, it came clear that B-pillar is problematic, because it is hardly visible in many cases. The components we suggested for detection are shown in Figure 2.1.

2.3 Wheel detection

Wheels are probably the most characteristic attribute of the side view of a car. They are constant in shape and if we consider a fixed viewing distance, they have nearly the same size, too. Therefore, they are ideal for detection. We use the AdaBoost algorithm to learn the classifier, which is a substantial part of the proposed wheel detector. The algorithm is closely described in [1] or [2], we will give only a brief description in Section 2.3.2.

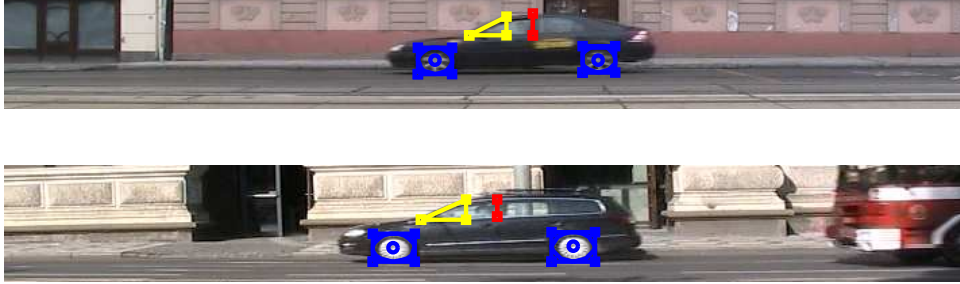


Figure 2.2: Examples of annotated video-frames (blue – bounding boxes and positions of wheels, yellow – angle and location of A-pillar, red – location of B-pillar)

2.3.1 Training samples for wheel detection

To build up the training set, we extracted a set of frames from a few video-sequences and manually annotated the components suggested for detection. Each car has been annotated twice in a sequence. We have annotated location and size of wheels (as bounding boxes), as well as all the other components (described later) belonging to just one car in each annotated frame. The annotations were later reused for learning the structural model. Two examples of annotated video-frames are shown in Figure 2.2.

The training samples for wheel detector were cut from the annotated video frames. The samples were scaled during cutting, so that the wheels at all samples had a uniform size, which we defined as 20 pixels in diameter. As the training samples contain also certain surrounding, their final size is 30×35 pixels. It is worth mentioning that the wheels are not vertically centred in the positive training sample, because the area above the wheel contains the mudguard, which carries more information than the area below the wheel, which usually contains only the shadow of the car on the road.

Our positive and negative training samples differ in relative location of the wheel centre within the sample. Location of the wheel centre within positive and negative samples is shown in Figure 2.3. The negative samples were generated by shifting the centre of the wheel to the points lying on the border of positive wheel location. These points are marked red in Figure 2.3. This

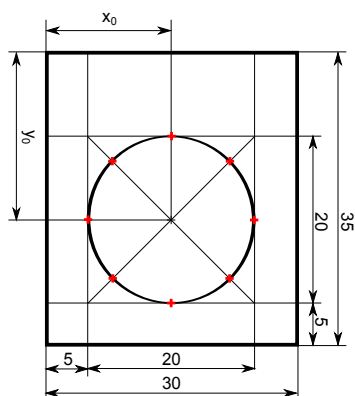


Figure 2.3: Location of the wheel centre within training samples (centre of the circle for positive, red crosses on its perimeter for negative samples)

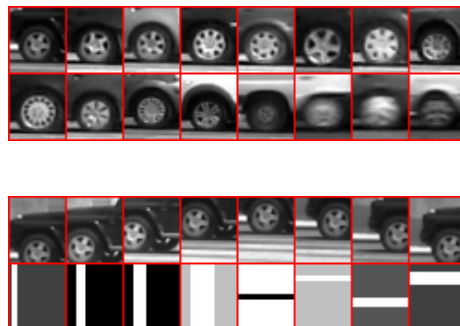


Figure 2.4: Examples of positive (top) and negative (bottom) wheel training samples

way, we generated one positive and eight negative training samples from each annotation. In fact, the negative samples do not necessarily have to contain wheels at all – they could be cut randomly from the background or contain other car parts than wheels. However, this way, we reach strong positive detector response around the wheel centre and strong negative response on its borders simultaneously. We will take advantage of this effect and will detect the wheels as positive peaks in wheel classifier responses using Mexican hat filtration, which is described in Section 2.3.4. Beside the mentioned negative samples, we have automatically generated additional negatives containing horizontal and vertical stripes of various shades in different positions to decrease the detector sensitivity to strong contrasts. Examples of positive and negative training samples are shown in Figure 2.4.

2.3.2 Building a strong classifier with AdaBoost

AdaBoost is an iterative algorithm that composes multiple weak classifiers into a strong one. Inputs of the AdaBoost are a labelled set of training samples and a set of weak classifiers to choose from, output is a set of chosen weak classifiers, their parameters and weights. The key idea of the AdaBoost

algorithm is weighting of the samples and minimizing the weighted training error. The weights are updated in each iteration, so that the learning process focuses on wrongly classified samples. We can influence the learning process by setting the initial weights of the training samples. These weights are usually set uniform for all samples. However, our training sets are unbalanced (the wheel training set contains 8 times more negative than positive samples, as described in previous section), so we have to set uniform weights for positives and negatives separately, so that sum of weights in both groups was 0.5. Otherwise, the learning would rather focus on negative samples, which would be undesirable.

Classified samples are 2D regions of the input image. Our weak classifiers consist of a mask \mathbf{M}_i of the same size as training samples and an associated threshold t_i . The classification $h_i(\mathbf{X})$ of a sample or detection window \mathbf{X} using mask \mathbf{M}_i is

$$h_i(\mathbf{X}) = \begin{cases} 1, & \text{if } \sum_u \sum_v |X(u, v)M_i(u, v)| \geq t_i \\ -1, & \text{if } \sum_u \sum_v |X(u, v)M_i(u, v)| < t_i. \end{cases} \quad (2.1)$$

The choice of particular masks and threshold values is made by the AdaBoost learning algorithm.

Numeric value of response $r(\mathbf{X})$ of the strong classifier to detection window \mathbf{X} is

$$r(\mathbf{X}) = \sum_{i=1}^{i_{max}} \alpha_i h_i(\mathbf{X}), \quad (2.2)$$

where i_{max} is number of weak classifiers chosen by AdaBoost and α_i are weights of the weak classifiers learnt by AdaBoost as well. Classification $c(\mathbf{X})$ of detection window \mathbf{X} is then given depending on the value of $r(\mathbf{X})$ as

$$c(\mathbf{X}) = \begin{cases} 1, & \text{if } r(\mathbf{X}) > 0 \\ -1, & \text{if } r(\mathbf{X}) \leq 0, \end{cases} \quad (2.3)$$

where $c(\mathbf{X}) = 1$ signifies membership in class and $c(\mathbf{X}) = -1$ excludes the sample from the class.

2.3.3 Masks for wheel detection

All the masks available for weak classifiers are generated as matrices of the same size as the size of training samples. We use three different types of masks: circular, radial and edge masks.

Circular masks \mathbf{M}^c are special case of radial masks with zero circular frequency k (see (2.5) for definition of radial masks). They are generated as matrices containing ones in a circle around the expected centre of the wheel and zeros elsewhere. They differ in the radius of the circle r_{max} , which is graded from 3 to 12 pixels with step 1. Elements of masks are defined as

$$\begin{aligned} M_i^c(x, y) &= \begin{cases} 1, & \text{if } r \leq r_{max_i} \\ 0, & \text{otherwise,} \end{cases} & (2.4) \\ r &= \sqrt{(x - x_0)^2 + (y - y_0)^2}, \\ r_{max_i} &\in \{3, 4, \dots, 12\}. \end{aligned}$$

Complete set of circular masks is shown in Figure 2.6a. These masks give good response for wheels with full naves, but their response to wheels with spokes is worse.

The pixel intensity values of wheels with spokes form a periodic function along the perimeter, so we are able to detect them using radial periodic masks \mathbf{M}^r , whose elements are defined in complex domain as

$$\begin{aligned} M_i^r(x, y) &= \begin{cases} e^{jk_i\varphi}, & \text{if } r \in \langle k_i/2; r_{max_i} \rangle \\ 0, & \text{otherwise,} \end{cases} & (2.5) \\ r &= \sqrt{(x - x_0)^2 + (y - y_0)^2}, \\ \varphi &= \tan^{-1} \left(\frac{y - y_0}{x - x_0} \right), \\ r_{max_i} &\in \{3, 4, \dots, 12\}, \\ k_i &\in \{5, 6, \dots, 12\}. \end{aligned}$$

The responses of radial masks are invariant to the rotation of detected wheel thanks to the application of absolute value in (2.1). Radial masks vary in radius r_{max} and circular frequency k . The nonzero elements of the masks form an annulus with centre identical with the expected centre of the wheel.

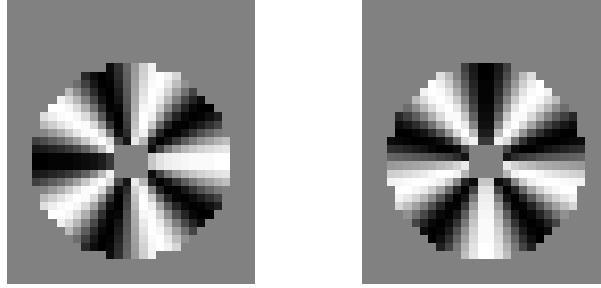


Figure 2.5: Detail of real (left) and complex (right) part of a radial mask ($r_{max} = 12$, $k = 5$)

Outer radius of the annulus r_{max} is graded in the same way as in case of the circular masks. Reason for the presence of the inner circle is the aliasing caused by limited resolution of the processed images. Its radius depends on the circular frequency k . Figure 2.5 shows example of real and complex part of a particular radial mask. Real parts of complete set of radial masks are shown in Figure 2.6b.

The last type of masks used for wheel detection are the edge masks. An edge mask contains ones in a rectangle in its upper left corner and zeros elsewhere. Particular edge masks differ in the dimensions of the rectangle,

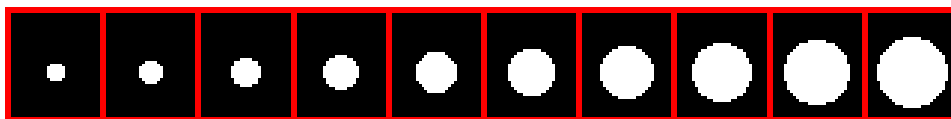
$$\begin{aligned}
 M_i^e(x, y) &= \begin{cases} 1, & \text{if } x \in \langle 1; x_{max_i} \rangle \wedge y \in \langle 1; y_{max_i} \rangle \\ 0, & \text{otherwise,} \end{cases} & (2.6) \\
 x_{max_i} &\in \{1, 2, \dots, 30\}, \\
 y_{max_i} &\in \{1, 2, \dots, 35\}.
 \end{aligned}$$

We get 1050 edge masks in total. Examples of edge masks are shown in Figure 2.6c.

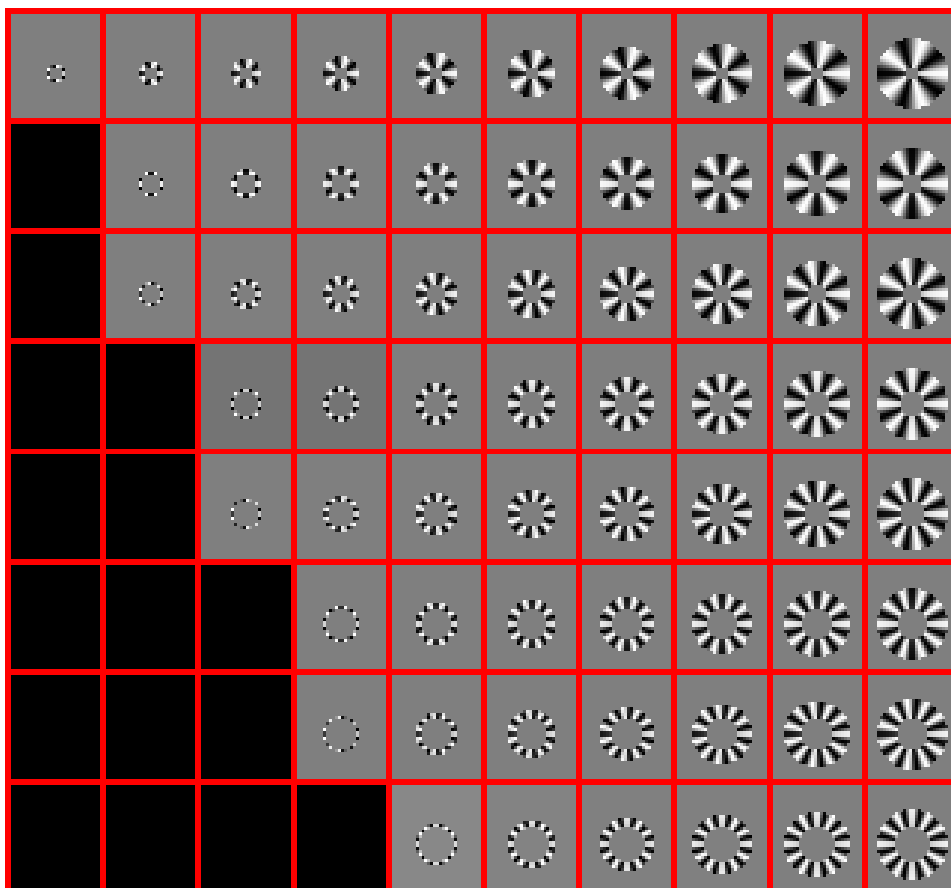
In summary, the AdaBoost learning selects masks \mathbf{M}_i for weak classifiers from the union of all masks \mathbf{M}^c , \mathbf{M}^r and \mathbf{M}^e .

2.3.4 Detection of wheel candidates

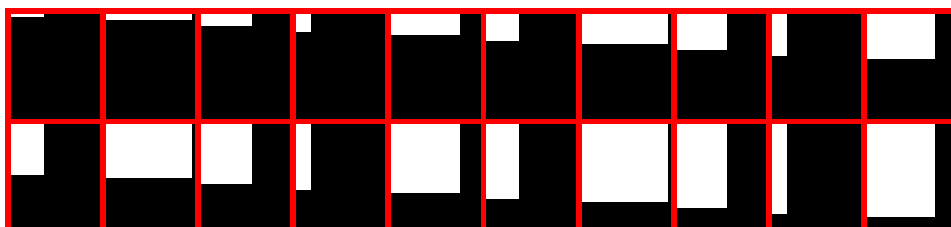
We have built a strong classifier, that can classify a detection window of size 30×35 pixels into two classes – *wheels* and *non-wheels*. To compute detector responses for the whole video-frame \mathbf{G} , we would need to place the detection



(a) Set of circular masks



(b) Set of radial complex masks, real part displayed



(c) Examples of edge masks (only 20 of 1050 shown)

Figure 2.6: Masks for wheel detection

window above each pixel of the frame and store the value $r(\mathbf{X})$ from (2.2) as the strong classifier response for that pixel. This principle can be effectively implemented using 2D convolution with the selected masks. For each mask \mathbf{M}_i , we compute 2D convolution with the whole frame. By thresholding the absolute values of the convolution result \mathbf{Z} , we get matrix \mathbf{H}_i that contains values $h_i(\mathbf{X})$ from (2.1) for each pixel of \mathbf{G} :

$$\begin{aligned} \mathbf{Z} &= \mathbf{G} * \mathbf{M}_i, \\ H_i(x, y) &= \begin{cases} 1, & \text{if } |Z(x, y)| \geq t_i, \\ -1, & \text{if } |Z(x, y)| < t_i. \end{cases} \end{aligned}$$

Resulting matrix of responses of the strong classifier $\mathbf{R}(\mathbf{G})$ is given according to (2.2) as weighted sum over masks

$$R(x, y) = \sum_{i=1}^{i_{max}} \alpha_i H_i(x, y). \quad (2.7)$$

As it is shown in Figure 2.7a, the strong classifier gives positive response not only for wheels, but also for the background, thus generating false positives. Here we can make profit of the fact that there is usually an area with strong positive response around the centre of the wheel and another area with strong negative response around its border, as mentioned in Section 2.3.1. We can suppress the false positive responses in the background and emphasise the wheel responses by convolving the AdaBoost response \mathbf{R} with 2D Mexican hat wavelet (see [7]), which is negative of normalised second derivative of a Gaussian,

$$\psi(x, y) = \frac{2}{\sqrt{3}\sigma\pi^{1/4}} \left(1 - \frac{x^2 + y^2}{\sigma^2} \right) e^{-\frac{x^2 + y^2}{2\sigma^2}}. \quad (2.8)$$

The value of $\sigma = 7$ has been found experimentally using numeric responses of the strong classifier to the positive training samples. The result \mathbf{R}' of the Mexican hat filtration is shown in Figure 2.7b,

$$\mathbf{R}' = \mathbf{R} * \psi. \quad (2.9)$$

In the next step, the filtered AdaBoost responses \mathbf{R}' are thresholded and we then use the Non-Maximal Suppression algorithm (NMS) [7] to find local

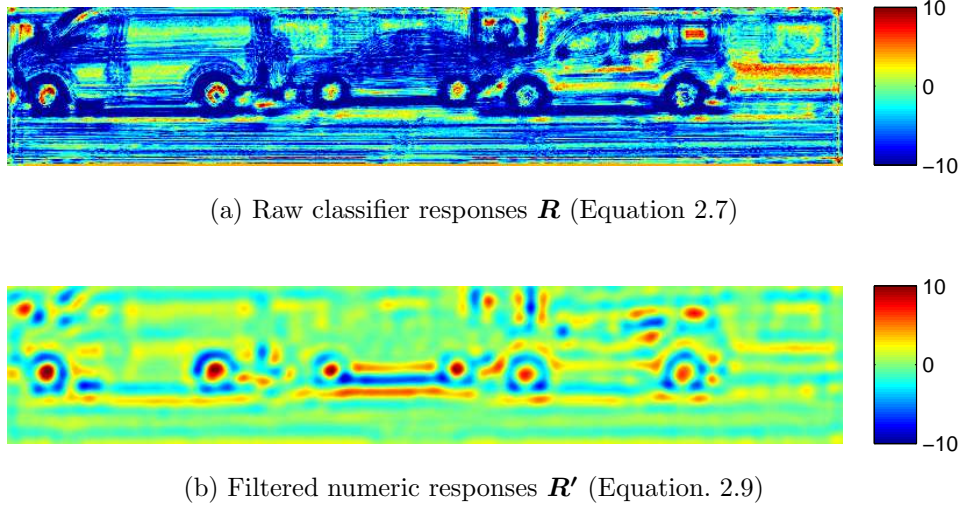


Figure 2.7: Comparison of raw and filtered wheel classifier responses

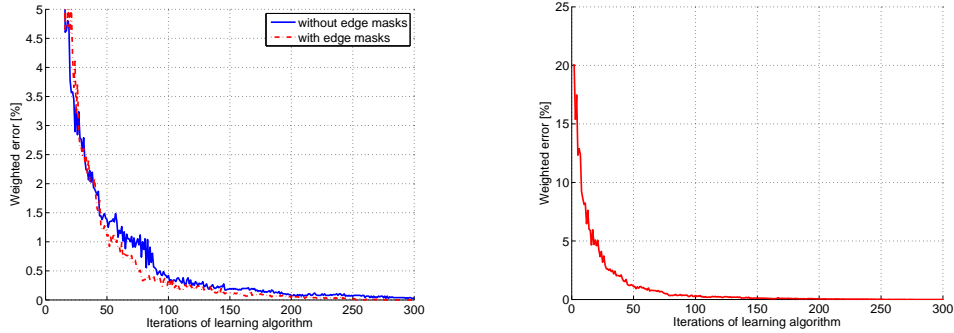


Figure 2.8: Wheel detections after thresholding and NMS. The detection was run on scale = 1, i.e. the size of the squares is 20 pixels.

maxima of the thresholded values, which represent the centres of detected wheel candidates. Each wheel candidate is then characterised by its coordinates (x_w, y_w) in the image and numeric value of the response function $R'(x_w, y_w) = w$. We can adjust sensitivity of the detector by setting the threshold level and size of the non-maximal suppression window. Example of wheel detections is shown in Figure 2.8.

2.3.5 Learning process and optimisation

Adding edge masks significantly lengthens the learning process, because the learning algorithm has to choose from over 1100 masks in each iteration instead of 74 masks without the edge masks. However, only the learning



(a) Comparison of learning with and without edge masks (zoomed). (b) Learning with edge masks (overview)

Figure 2.9: Weighted training error (weighted by initial weights of the training samples) depending on number of iterations of the learning algorithm

process is slowed down. The computational complexity of the strong classifier stays the same, because it depends on the number of masks chosen, which is limited by number of learning iterations. Both learning and subsequent classification and detection are implemented in Matlab.

Without edge masks, we reached training error 0.07% after 300 iterations. The error was caused by false positives and with the use of edge masks dropped down to 0%. Usefulness of the edge masks can be seen from distribution of kinds of masks. AdaBoost has chosen 94 circular, 106 radial and 90 edge masks in 300 iterations, so the contribution of edge masks is not negligible. Figure 2.9 shows training error weighted by initial weights depending on number of iterations of the learning algorithm.

Unlike the A-pillar and side panel detector responses, the computation of wheel detections is lengthy. It is given by the principle of the classifier algorithm, which requires convolving the input image with all the masks chosen during learning and subsequently thresholding the convolutions. We can accelerate both of these two stages.

Let's have a look at the convolution computation first. The masks we suggested for wheel detection contain large "inactive" areas of zeros, which can be cut off without any impact on the results of convolutions. We must

only be aware of the fact, that the Matlab function `conv2` we are using stores every partial result of convolution in the pixel corresponding to the centre of the mask, so we must take care of the relative offset of the mask centre caused by cutting. The acceleration of computation cannot be predicted in advance, because different masks have different relative size of inactive areas and we do not know, which masks will be chosen during the learning process. We have reached speed-up about 49.6% for the learnt wheel detector.

In case of the convolution thresholding, we cannot improve the computation speed, but we can minimise the number of calls of the thresholding function, because AdaBoost sometimes chooses the same mask with the same thresholds more times, only with different weights. Such group of weak classifiers can be merged into a single one with the same mask and threshold and weight equal to sum of weights over the group. This trick reduces multiple thresholding calls needed when evaluating the group of weak classifiers one by one to just one call. On our wheel detector, the difference makes another 7.2%.

Both trimming of the masks and mask merge are performed at the end of the learning process.

2.3.6 Transforming wheel detector responses to probability

In the probabilistic model described later in Chapter 4, we will need conditional probabilities $P(w|L)$ of wheel detector numeric responses for wheels and non-wheels, where L denotes label of a wheel candidate, which can be 0 for a non-wheel or 1 for a wheel. We can learn these probabilities from distributions of detector responses to positive and negative training samples. Figure 2.10 shows histograms of wheel detector numeric responses after Mexican hat filtering. The histograms can be fitted by Gaussian distributions quite precisely,

$$\begin{aligned} p(w|L = 0) &= \mathcal{N}(\mu_{neg}, \sigma_{neg}), \\ p(w|L = 1) &= \mathcal{N}(\mu_{pos}, \sigma_{pos}). \end{aligned} \tag{2.10}$$

We identified the constants of the Gaussian distributions for the learnt wheel detector as $\mu_{neg} = -2.19$, $\sigma_{neg} = 1.67$, $\mu_{pos} = 6.32$ and $\sigma_{pos} = 2.83$.

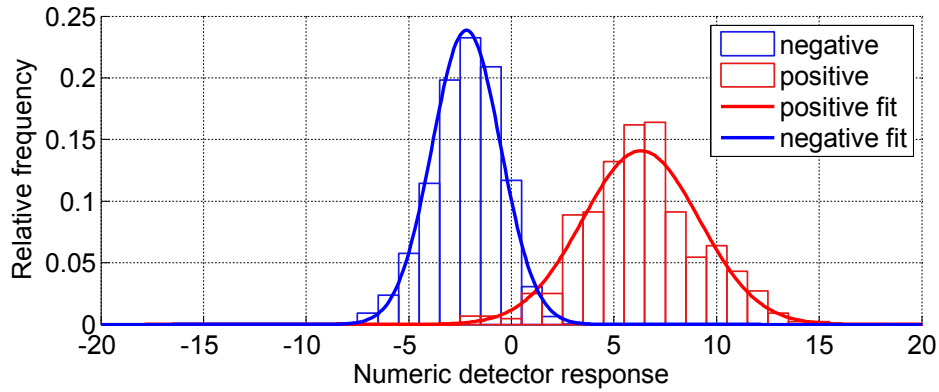


Figure 2.10: Conditional probability of wheel detector response R' (blue – responses to negative training samples, red – responses to positive training samples)

2.4 B-pillar detection

The next feature we suggested for detection is the lower half of the B-pillar with its surrounding. Having implemented the AdaBoost learning, we tried to apply the same method as for the wheels. We generated different training samples and provided AdaBoost only with the edge masks described in Section 2.3.3.

2.4.1 B-pillar detector training

The training samples were again cut from annotated video frames. As well as in the case of wheels, it was essential to normalise the scale to reach effective training. However the B-pillar is not so invariant in shape and size as the wheels, so this appeared to be a slight problem. Finally, we decided to pick the pillar height as a measure of its scale. The normalised height of the pillar is defined as 20 pixels. As we want the positive training sample to contain lower half of the pillar and its surrounding, we chose training sample size 30×20 pixels. Similarly to the wheel training samples, we generated negative samples by shifting the lower end of the pillar to different relative positions within the detection window. Location of the pillar in positive and negative training samples is shown in Figure 2.11, a few examples of positive and negative training samples are depicted in Figure 2.12.

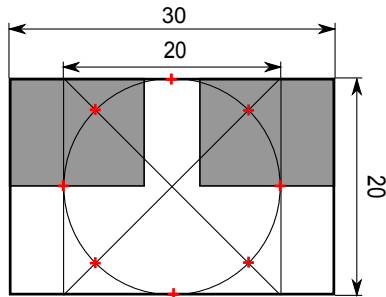


Figure 2.11: Location of the B-pillar within training samples (red crosses denote location of lower pillar end for negative samples, positive location is drawn directly)

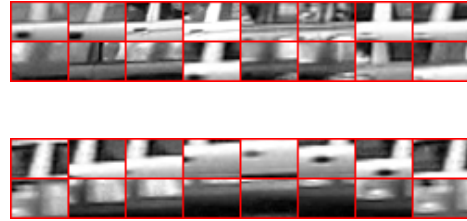


Figure 2.12: Examples of positive (top) and negative (bottom) B-pillar training samples

In spite of the fact that the positive and negative samples should be easily distinguishable using the edge masks, we have not reached expected results. The learning algorithm had not reached zero training error even in 1000 iterations, having problem with high false positive rate, shown in Figure 2.13. The reason is that the B-pillar is nearly or completely unrecognisable in approximately 50% of cases. The problem may be also caused by other factors, which include strong motion blur, variety of possible combinations of colours (dark pillar/light windows, dark pillar/dark windows, etc.) or non-optimal training samples and masks. The last and most probable possibility is presence of reflections and visibility of passengers and car background through the window, which can confuse the learning algorithm. The feature we proposed is therefore probably too varied to be detected effectively. We have finally excluded the B-pillar detection from our method due to the poor performance of the detector.

2.5 A-pillar detection

Whereas horizontal and vertical edges are very common in the scene (especially in urban environment), sloping edges are not so usual. They appear in shadows, reflections, they arise as a result of perspective, but in most cases,

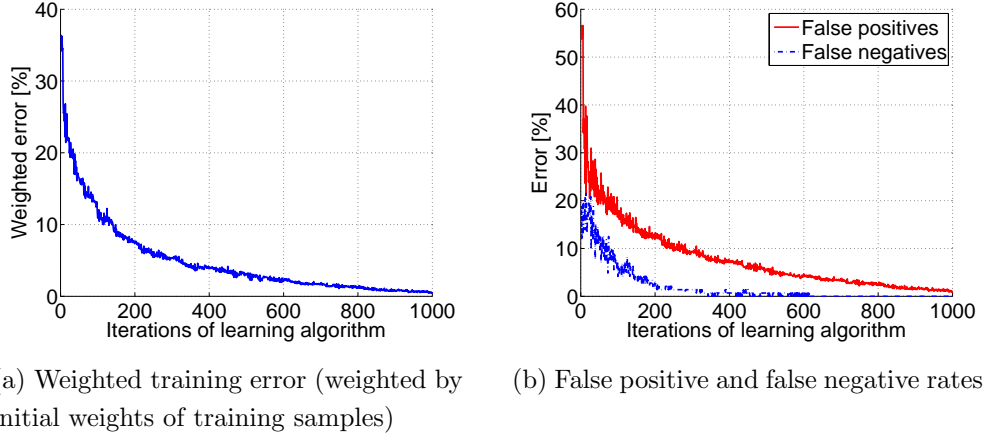


Figure 2.13: B-pillar detector learning process

they characterise the A-pillar. In addition, the angle of A-pillar of passenger cars lays within a relatively bounded interval, so it is a well distinguishing feature. On the top of that, we are able to determine the direction of car movement depending on the side, where we detect the A-pillar and the angle, under which it is detected.

2.5.1 AdaBoost based detection

At first, we tried the AdaBoost approach again. We restricted the detection problem only to cars going from right to left. The other case is axially symmetric with respect to the vertical axis, so we transform a trained detector to the opposite traffic direction by flipping the chosen masks.

Size of training samples was set to 15×15 pixels. We generated slope masks M^s , which contain ones and zeros below and above a sloping edge, respectively,

$$\begin{aligned}
 M_i^s(x, y) &= \begin{cases} 1, & \text{if } y \geq (\sin \varphi_i)x + k_i \\ 0, & \text{otherwise,} \end{cases} & (2.11) \\
 \varphi_i &\in \{25^\circ, 30^\circ, \dots, 50^\circ\}, \\
 k_i &\in \{-15, -14, \dots, 15\}.
 \end{aligned}$$

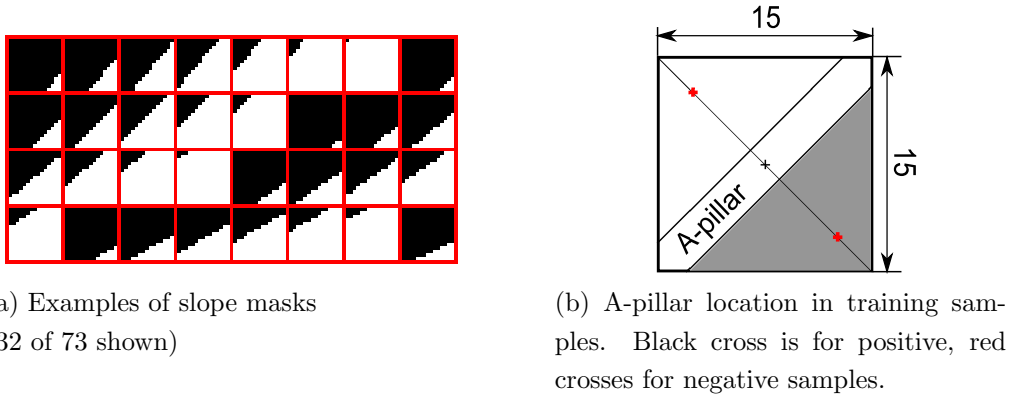


Figure 2.14: Masks and training samples for A-pillar detection

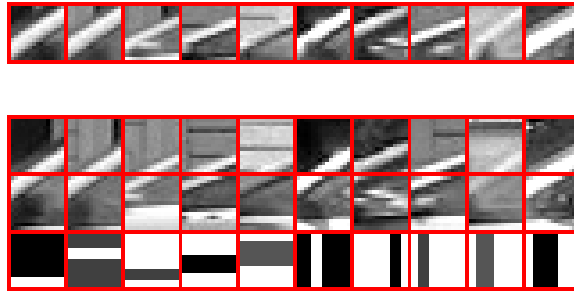


Figure 2.15: Examples of positive (top) and negative (bottom) A-pillar training samples

The masks vary in the angle and relative position of the edge. A few examples of slope masks are shown in Figure 2.14a. Except for the slope masks, we have added masks \mathbf{M}^e (2.6) containing horizontal and vertical edges to enable better elimination of potential false positives.

The training samples were cut from annotated video-sequences. A positive sample has the middle of the pillar located in its centre, as it is shown in Figure 2.14b. Negative training samples were generated by shifting the middle of the pillar along the diagonal about 14 pixels, so that both horizontal and vertical offsets were 10 pixels. Just as in the case of wheel detector training, we have added automatically generated negative samples with horizontal and vertical stripes. Examples of training samples are shown in Figure 2.15.

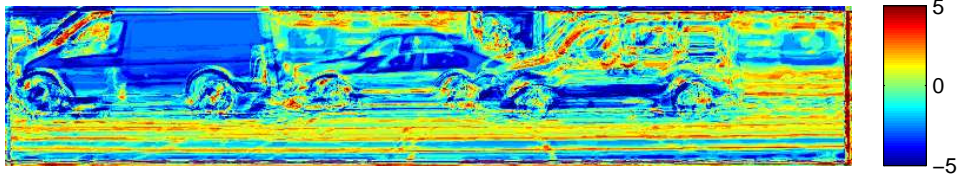


Figure 2.16: Responses of the AdaBoost A-pillar detector

The experimental results were not satisfactory. We were unable to reduce the high false positive rate of the detector, which is shown in Figure 2.16. We therefore proposed a different solution based on edge detection.

2.5.2 Edge based detection

The advantage of detecting the A-pillar as a sloping edge is its simplicity and smaller computational complexity in comparison with the AdaBoost approach. The A-pillar is characterised not only by one, but in many cases by two parallel edges. This depends on the differences between intensity levels of pillar, background and driver's window.

The edge in each pixel $g(x, y)$ of an image \mathbf{G} is described by its magnitude m and direction ϕ identical to the magnitude and direction of gradient.

$$m = |\text{grad } g(x, y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2}, \quad (2.12)$$

$$\phi = \arg\left(\frac{\partial g}{\partial y}, \frac{\partial g}{\partial x}\right). \quad (2.13)$$

Gradient is computed as 2D convolution of the image with a gradient operator. There is quite a big variety of gradient operators like Sobel, Laplace or Prewitt operator described in literature [7]. We use two operators \mathbf{d}_x , \mathbf{d}_y approximating first derivative of the image function in vertical and horizontal direction, respectively.

$$\mathbf{d}_x = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix}, \quad (2.14)$$

$$\mathbf{d}_y = \mathbf{d}_x^T. \quad (2.15)$$

Horizontal and vertical elements of gradient are then

$$\mathbf{G}_x = \mathbf{G} * \mathbf{H} * \mathbf{d}_x, \quad (2.16)$$

$$\mathbf{G}_y = \mathbf{G} * \mathbf{H} * \mathbf{d}_y, \quad (2.17)$$

where \mathbf{H} is additional Gaussian filter for noise suppression.

Magnitude and direction of gradient in each pixel is given by

$$m(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \quad (2.18)$$

$$\phi(x, y) = \arg(G_y(x, y), G_x(x, y)). \quad (2.19)$$

Consequently, we limit the computed directions $\phi(x, y)$ to interval $\langle -90^\circ, 90^\circ \rangle$, because the angle of A-pillar can be for example -45° as well as 135° when the car goes from right to left and we want to process these two possibilities as equivalent in further stages of detection. After this transformation, the cars going from left to the right are characterised by positive A-pillar angles, whereas the cars in the opposite traffic direction are characterised by negative ones.

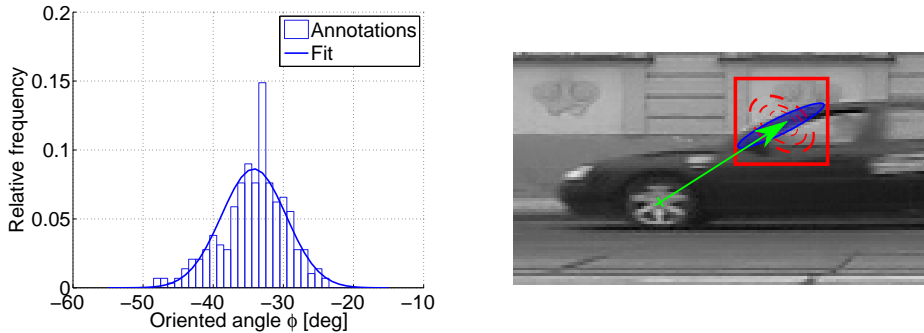
2.5.3 Transformation of oriented angles to probability

In the next step, we transform the oriented angles to probability of A-pillar presence. For this purpose, we extract A-pillar angles from the annotations and approximate their probability distribution. Figure 2.17a shows histogram of the extracted data with fitted Gaussian distribution:

$$p(\phi_p) = \mathcal{N}(\mu_{angle}, \sigma_{angle}), \quad (2.20)$$

where $\mu_{angle} = -34.2^\circ$ and $\sigma_{angle} = 4.6^\circ$ were estimated from the annotations of video-sequences with right-left direction of traffic. The distribution for opposite traffic direction differs only in the sign of mean value. Using the distribution (2.20), we can get the probability of A-pillar presence $P_{pil}(x, y)$ for each pixel. We are in fact interested only in strong edges, so we use a simple gradient magnitude thresholding:

$$P_{pil}(x, y) = \begin{cases} 0, & \text{if } m(x, y) < t, \\ P(\phi_p = \phi(x, y)), & \text{if } m(x, y) \geq t, \end{cases} \quad (2.21)$$



(a) A-pillar angle distribution for right-left traffic direction. (The distribution for opposite direction differs only in sign of the mean value.)

(b) A-pillar response probability computation. Green – pillar displacement, blue – high probability of pillar presence P_{pil} , red – 2D Gaussian distribution.

Figure 2.17: A-pillar response probability

where the threshold $t = 10$ has been chosen experimentally.

Another problem we had to solve was the fact that the pillar can generate one or two edges, whereas we would need always a single strong response, ideally located near the middle of the pillar. Therefore, after transforming the edge directions ϕ to probabilities \mathbf{P}_{pil} , we expand maximal probabilities using a square sliding window \mathbf{W} . The value of every element $P_{pil}(x, y)$ is set to maximum over the sliding window placed with its centre above this element,

$$P_{pil}(x, y) = \max\{\mathbf{W}_{xy}\}. \quad (2.22)$$

This way, we ensure that the responses of two detected edges are joined into one compact response. Regarding that the usual thickness of the A-pillar at normalised scale is around 3 pixels, sufficient sliding window size is the same. Figure 2.18 shows the detected edge directions transformed to probabilities \mathbf{P}_{pil} . Results reached with the edge based detector are significantly better than the ones with the AdaBoost detector.

The A-pillar detector responses are not localised very well, so we combine the A-pillar detector response with the structural model to get probability $P(a)$, that will be used in probabilistic model during final detection. We evaluate a detection window around the expected pillar position according to the

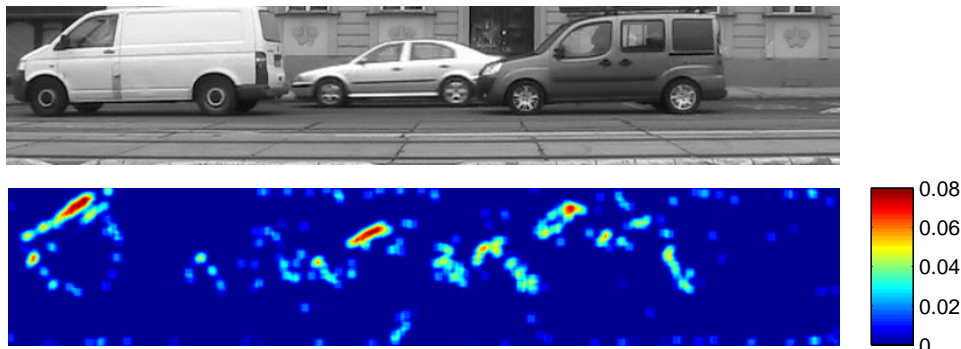


Figure 2.18: Input image (top) and detected edge directions transformed to probability of A-pillar presence P_{pil} (bottom)

corresponding front wheel detection. The process is schematically depicted in Figure 2.17b. The green arrow marks mean value of pillar displacement given by the structural model, the blue ellipse represents high values of probability P_{pil} and the red square with dashed ellipses represents detection window containing 2D Gaussian distribution of A-pillar relative displacement described later in Section 3.2. We are searching for A-pillar location with maximal joint probability with respect to both structure and visual appearance. We take maximum of the Hadamard product (\boxtimes) of detection window and the overlaid part of probability P_{pil} as the joint probability of the A-pillar,

$$P(a) = \max \{P_{pil} \boxtimes \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\sigma}_a)\}. \quad (2.23)$$

Our implementation of final detector allows also automatic detection of driving direction of detected car using the A-pillar detection. We can evaluate the A-pillar detector response with structural models for both directions and choose the one with higher probability, which determines the driving direction of the car.

2.6 Side panel verification

We have rejected the side panel as a primary component for detection, because we are unable to locate it in the image precisely. However, we can use it for verification of car detections – when we detect a potential pair of

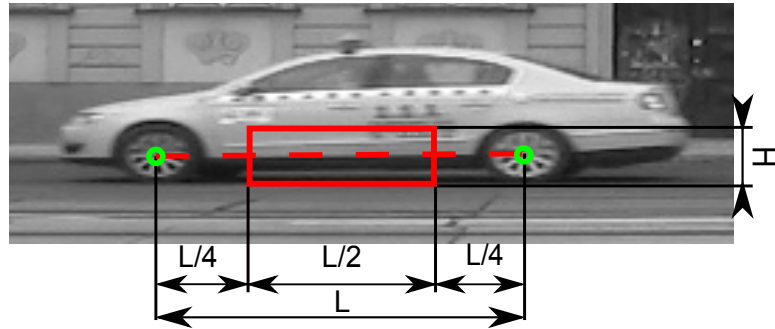


Figure 2.19: Location and dimensions of the side panel (red box). Wheel base L is shown as dashed line connecting wheel centres (green circles).

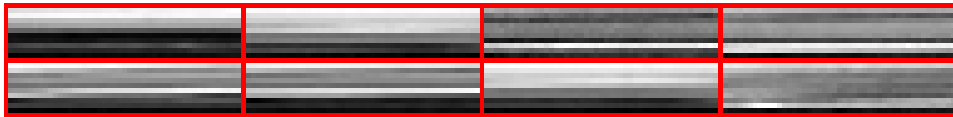


Figure 2.20: Examples of side panel area

wheels, we can inspect the area between them, whether it contains something similar to the side panel, or not.

We define the side panel as rectangular area of the same height as normalised wheel diameter (20 px for scale = 1) and width equal to one half of the wheel base, centred between the wheels, as it is shown in Figure 2.19. Several examples of side panel areas are depicted in Figure 2.20.

The key idea of the verification is that the side panel has typically quite low variance of intensity levels along the direction of the wheel base and remarkably higher variance along the perpendicular direction. Our camera is not inclined, so we can assume that the wheel base is horizontal, which simplifies the evaluation. To eliminate influence of small deviations of the wheel base from horizontal direction, we subsample the side panel area to half vertical resolution before evaluation, which ensures that the pixel rows are parallel with the edges of the side panel.

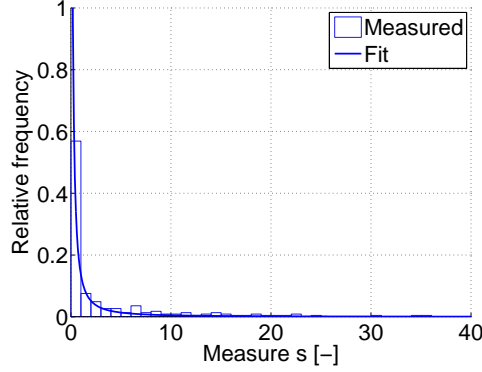


Figure 2.21: Conditional probability of side panel measure s

We define measure s that will evaluate a rectangular side panel region D of size $x_{max} \times y_{max}$ in a following way:

$$\begin{aligned}
 s_h &= \sum_y \left\{ Std_x \{ D(x, y) \} \right\}, \\
 s_v &= Std_y \left\{ \frac{1}{x_{max}} \sum_x D(x, y) \right\}, \\
 s &= \frac{s_h^2}{s_v^2}.
 \end{aligned} \tag{2.24}$$

The last thing we have to do is to transform values of the measure s to probability $P(s)$ that the inspected area contains side panel. This probability is used for final detection. The measure s is used to verify the assumption that a pair of wheel candidates forms a car. However, it cannot be used to verify that some wheel candidate is not a real wheel. This is a significant difference from the usage of the AdaBoost classifier response \mathbf{R}' defined in Section 2.3.4, which has meaning for both cases. Therefore, we only have to find the distribution of the measure s on positive samples cut from annotated video frames. Figure 2.21 shows histogram of measure responses on positive samples with fitted rational function. The sought probability of side panel measure s was experimentally identified as:

$$p(s) = \frac{1}{2\sqrt{10}} (s + 0.1)^{-\frac{3}{2}}. \tag{2.25}$$

The shift in s by 0.1 is used in order to obtain proper probability density function (which has to integrate to 1 over all possible values of s).

2.7 Contribution

We have implemented the AdaBoost learning algorithm, which is reusable for different components. It is possible to generate different training samples and masks to learn a strong classifier for any component, although it has certain limitations as we could have seen when trying to learn the B-pillar detector.

We have suggested additional components for detection and annotated them in the video-sequences. We have introduced new types of masks for AdaBoost weak classifiers and automatically generated negative training samples for AdaBoost learning.

We have proposed detectors based on different principles than AdaBoost. The edge based A-pillar detector does not only give better results than AdaBoost, but is less computationally demanding, too. The AdaBoost detector was finally used only for wheels.

Chapter 3

Structural model

The structural model interconnects single component candidates into whole car candidates. Without the structural model, we could not interpret any relationships between the components and we would not be able to detect cars as sets of components.

The structural model also increases the robustness of detection. Single component detectors described in Chapter 2 are quite susceptible to false detections on visually similar objects, e.g. the wheel detector can have positive response on a traffic sign. However, when using more component detectors, we can detect cars as combinations of components. This is much more resistant to the false detections, because the car is not characterised only by visual appearance of its components, but also by their relative displacement – the structure. If we consider the mentioned example with false wheel detection caused by a traffic sign, there probably won't be any suitable detection of another wheel, A-pillar and side panel, that would fit the structural model.

Another advantage brought by the multi-component detection is, that if we obtain strong detector response for some components and weaker responses for the others, we will still capture the car as whole object, because the overall probability of complete car detection will be sufficient.

Regarding experimental results, we have decided to integrate only the wheels, A-pillar and side panel components into the final detector.

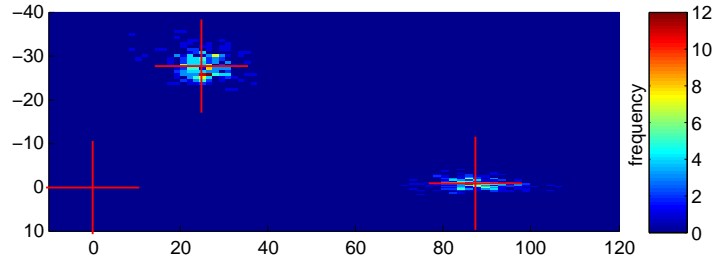


Figure 3.1: 2D histogram of relative displacements of wheels and A-pillar for right-left traffic direction, front wheel is the anchor position. Red crosses mark mean values of displacements.

3.1 Learning the structure

In this section, we will describe the structural model of side-view of a car going from right to left. The model is learnt at normalised scale, at which the wheels measure 20 pixels in diameter. The model for opposite traffic direction is axially symmetric with respect to vertical axis, so it does not have to be learnt separately.

The structure of the car is given by relative positions of the components, so we need an anchor position to compute the relative displacements of the other components with respect to it. We have chosen the front wheel centre as the anchor, because our wheel detector gives the best results in comparison with the other component detectors, which do not localise the components so precisely.

As the position of the side panel detection window is given as the centre of a wheel pair, we need to learn only two parameters of the structure, the displacement of the A-pillar centre and the displacement of the rear wheel centre with respect to the front wheel centre. We used the annotations of video-sequences where we know the direction of traffic for learning, so we could determine, which of the two wheel annotations present in an annotated frame denotes the front wheel. Figure 3.1 shows 2D histogram of relative displacements of the components for right-left traffic direction. Red crosses mark the mean values of positions of the wheel and A-pillar centres.

3.2 Probabilistic representation of the structure

Similarly to the component detector responses, we will need to get probabilities from the structural model for the final detection. We approximated the histograms in Figure 3.1 by two-dimensional Gaussian probability density functions

$$p(\mathbf{d}) = \frac{1}{2\pi|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{d} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{d} - \boldsymbol{\mu})\right), \quad (3.1)$$

where $\mathbf{d} = (\Delta x, \Delta y)^T$ is vector of relative displacement, $\boldsymbol{\mu} = (\mu_x, \mu_y)^T$ is vector of mean displacement values and Σ is covariance matrix. We get probability density $\mathcal{N}(\boldsymbol{\mu}_A, \Sigma_A)$ for A-pillar displacement and $\mathcal{N}(\boldsymbol{\mu}_d, \Sigma_d)$ for rear wheel displacement. The constants were identified as

$$\boldsymbol{\mu}_A = \begin{bmatrix} 25 \\ -28 \end{bmatrix}, \quad \Sigma_A = \begin{bmatrix} 16 & 2 \\ 2 & 5 \end{bmatrix}, \quad \boldsymbol{\mu}_d = \begin{bmatrix} 87 \\ -1 \end{bmatrix}, \quad \Sigma_d = \begin{bmatrix} 32 & 1 \\ 1 & 1 \end{bmatrix}.$$

We directly use only the wheel displacement probability density $p(d_{ij}) = \mathcal{N}(\boldsymbol{\mu}_d, \Sigma_d)$ in the final detector, whereas the A-pillar displacement probability is merged with the probability of A-pillar detector response as described in Section 2.5.3.

3.3 Contribution

We have proposed a structural model that integrates wheels, A-pillar and side panel into a compact set. We have extracted parameters of the structural model from the video-sequence annotations and suggested its probabilistic representation for use in the final car detector. It greatly improves robustness of detection – at once, it helps to avoid false detections and covers the variability of car body types.

Chapter 4

Final detector

In this chapter, we will describe the final detector of car side-views. Inputs to the final detector are wheel candidates, component detector numeric responses transformed to probabilities and structural model (probabilities of component displacements). Its task is to label the wheel candidates.

Because the A-pillar and side panel detector responses are not localised so well as the wheel candidates, we use following strategy:

1. detect single wheel candidates in the scene and form all possible wheel pair candidates that could foreshadow cars,
2. use A-pillar and side panel component detector responses in expected areas for verification of the wheel pair candidates.

As a result, every wheel candidate is either assigned to a wheel pair or marked as non-wheel. We assume that a car side-view must have just two wheels, an A-pillar and a side panel, i.e. must be fully visible.

For description of the probabilistic model, we will use following notation:

- n – total number of wheel candidates detected in the processed frame,
- k – number of wheel pairs (car detections) contained in a labelling \mathbf{L} ,
- \mathbf{L} – labelling assigning wheel candidates to pairs,
$$\mathbf{L} = \{(i_1, j_1), (i_2, j_2), \dots, (i_k, j_k)\},$$

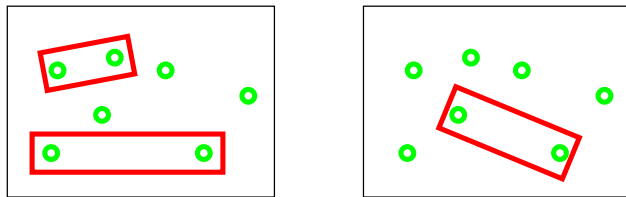


Figure 4.1: Two examples of possible labellings of wheel candidates. Wheel candidates are symbolised by the green circles. The red rectangles denote assignment of wheel candidates to a pair.

- L_i – label of i -th wheel candidate (1 for wheel or 0 for non-wheel),
- \mathbf{w} – vector of numeric values of AdaBoost responses characterising the wheel candidates, $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$,
- \mathbf{d}_{ij} – vector of relative displacements of wheel detections in pairs,
 $\mathbf{d}_{ij} = \{d_{i_1j_1}, d_{i_2j_2}, \dots, d_{i_kj_k}\}$,
- \mathbf{a} – vector of A-pillar displacement and detector responses,
 $\mathbf{a} = \{a_1, a_2, \dots, a_k\}$,
- \mathbf{s} – vector of side panel detector responses,
 $\mathbf{s} = \{s_1, s_2, \dots, s_k\}$.

The final detector considers and evaluates all possible labellings \mathbf{L} of the wheel candidate detections and chooses the optimal one, \mathbf{L}^* , as the best interpretation of the image. In Figure 4.1, we can see two examples of possible labellings of one image.

4.1 Labelling of detections and its probability

The decision about the optimal labelling \mathbf{L}^* is based on probabilistic model. When processing a video frame, we necessarily have to interpret all detections in the whole frame at once, which allows us to consider our measurements being statistically independent within one concrete labelling. Optimal labelling \mathbf{L}^* is given as

$$\mathbf{L}^* = \arg \max \left(P(\mathbf{L} | \mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s}) \right). \quad (4.1)$$

Using Bayes' theorem and assuming statistical independence of every particular measurement given a labelling \mathbf{L} , we can express the probability as

$$\begin{aligned} P(\mathbf{L}|\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s}) &= \frac{P(\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s}, \mathbf{L})}{P(\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s})} = \\ &= \frac{P(\mathbf{w}|\mathbf{L})P(\mathbf{d}_{ij}|\mathbf{L})P(\mathbf{a}|\mathbf{L})P(\mathbf{s}|\mathbf{L})P(\mathbf{L})}{P(\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s})}. \end{aligned} \quad (4.2)$$

Note, that the components of the joint probability $P(\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s})$ are not statistically independent while the components of conditional probability $P(\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s}|\mathbf{L})$ are assumed to be independent.

Because values of measurements (and therefore also their joint probability) are the same for all labellings of one frame, we can leave out the joint probability and get the optimal labelling as:

$$\begin{aligned} \mathbf{L}^* &= \arg \max \left(P(\mathbf{w}|\mathbf{L})P(\mathbf{d}_{ij}|\mathbf{L})P(\mathbf{a}|\mathbf{L})P(\mathbf{s}|\mathbf{L})P(\mathbf{L}) \right) = \\ &= \arg \max \left(\left[\prod_{i=1}^n P(w_i|L_i) \prod_{l=1}^k P(d_{ijl}) \prod_{l=1}^k P(a_l) \prod_{l=1}^k P(s_l) \right] P(\mathbf{L}) \right). \end{aligned} \quad (4.3)$$

We have already determined all probabilities from (4.3) in Chapter 2, except for the prior probability of labelling $P(\mathbf{L})$. We choose $P(\mathbf{L})$ dependent on the probability of wheel candidate label $P(L)$,

$$\begin{aligned} P(\mathbf{L}) &= \prod_{i=1}^n P(L_i), \\ P(L = 1) &= \text{const.}, \\ P(L = 0) &= 1 - P(L = 1). \end{aligned}$$

As we are maximising in (4.3), we can divide $P(\mathbf{L})$ by a constant $P(L = 1)^n$ without impact on the result. We substitute $P(\mathbf{L})$ by $Q(\mathbf{L})$,

$$\begin{aligned} Q(\mathbf{L}) &= \frac{P(\mathbf{L})}{P(L = 1)^n} = \prod_{i=1}^n \frac{P(L_i)}{P(L = 1)} = \\ &= \prod_{i:L_i=0} \frac{P(L = 0)}{P(L = 1)} = \left(\frac{P(L = 0)}{P(L = 1)} \right)^{(n-k)}. \end{aligned} \quad (4.4)$$

The probabilities are typically quite small numbers and we have only limited numeric precision, so we'd better work with logarithmic probabilities. Equation (4.3) after substituting $Q(\mathbf{L})$ for $P(\mathbf{L})$ from (4.4) changes to

$$\begin{aligned} \mathbf{L}^* = \arg \max & \left(\sum_{i=1}^n \log(P(w_i|L_i)) + \sum_{l=1}^k \log(P(d_{i_jl})) + \right. \\ & \left. + \sum_{l=1}^k \log(P(a_l)) + \sum_{l=1}^k \log(P(s_l)) + (n - k) \log \left(\frac{P(L = 0)}{P(L = 1)} \right) \right). \end{aligned} \quad (4.5)$$

We do not know the exact value of the prior probability $P(L = 1)$, so we have to find it experimentally. It acts as detector sensitivity adjustment parameter. The higher probability $P(L = 1)$ we set, the more candidates are likely to be chosen as positive car detections.

4.2 Labelling evaluation optimisation

In the beginning of this chapter, we have said that our final detector evaluates all possible labellings \mathbf{L} of the wheel candidates. However, we usually reach tens of wheel candidates at one scale of the processed image, so there is an enormous number of possible labellings and it would be impossible to really evaluate each of them. We therefore use hard constraints on the relative displacement of wheels in a pair to form all potential pairs of wheels in the image, whose number is substantially smaller than the number of all pairs of wheel candidates. We then generate labellings \mathbf{L} of the wheel candidates in an exhaustive way as all possible combinations of the potential wheel pairs and evaluate (4.5). Even after this improvement, the number of possible labellings is 2^N , where N is number of considered wheel pairs.

4.3 Multi-scale detection

During the training process of component detectors and structural model, we use normalised samples, so they are learnt only on a single scale. In practice, we need to detect cars in different viewing distances, so we have to process the input images at multiple scales. We use the scale-image-pyramid as presented in [3]. Every processed video frame is resized to defined scales, so

we get several images with different resolution from one frame. The choice of scales depends on the viewing distance, at which we want to detect the cars. The scales we use are graded by 10% from 1 to 2.14 (i.e. 9 different scales). Each of the scaled images is then processed separately, at normalised scale. Processing includes component detector response computation, transforming detector responses to probabilities and finding the best evaluated image labelling. From this point onwards, a labelling is characterised by two parameters:

1. its probability $P(\mathbf{L}|\mathbf{w}, \mathbf{d}_{ij}, \mathbf{a}, \mathbf{s})$,
2. list of positive car detections given by the labelling,

where every car detection is identified by the coordinates of its wheels and A-pillar detections. After choosing the optimal image labelling at given scale, we transform the coordinates of the component detections back to common coordinate frame corresponding to the scale 1 for further processing.

4.3.1 Merging detections across scales together

Independent processing of different scales leads to overlapping detections after being merged into the common coordinate frame, because a car is typically detected at more than one scale at once. Therefore, we remove the overlaps after the merge.

First, we put all detections from all scales into one set. Then we start to reject the overlapping detections whose labellings have smaller evaluation until there are no overlaps. We have to avoid undesired cascade rejections.

Figure 4.2 shows scheme of overlapping detections with numbers marking the evaluations e of labellings they originally belonged to. If we started removing overlaps from left to right, we would finish only with the blue one preserved, because $e_{red} < e_{green} < e_{blue}$, so we would reject the red detection in the first step and in the following step the green detection, too. However, it would be sufficient to reject only the green detection to remove all overlaps. Therefore, we first order the detections according to their evaluation and then start solving overlaps with the best evaluated detection, then with the second best etc. This way, any cascade rejection cannot occur.

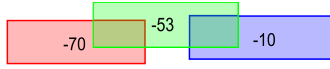


Figure 4.2: Overlapping detections example

The presented method of overlaps removal is not optimal. To reach optimal solution, we would have to evaluate possible labellings across all scales, not only at each scale separately. Nevertheless, the number of evaluated labellings would be so high then, that we would not be able to compute solution in a sensible time.

4.4 Contribution

We have successfully integrated the suggested component detectors into the final one with usage of structural model. We have proposed a probabilistic model for the final detector, which allows making optimal decision about the wheel candidate labelling at each inspected scale. We have implemented the final car side-view detector with optional automatic detection of the traffic direction. We have also implemented the multi-scale detection including the removals of the overlapping detections originating from independent processing at different scales.

Chapter 5

Experiments

In this chapter, we will give brief information about the data sets that were used for experiments and we will try to demonstrate experimental results. The attached CD contains testing video-sequences with detections, which can give additional information about the performance of the final detector.

5.1 Data for experiments

The video-sequences used for training and testing were taken with ordinary digital camera. They cover both traffic directions and also different scales, because some of the sequences were taken at multi-track roads. We have put aside two video-sequences for testing and used the rest for training. We have one testing video-sequence for each traffic direction.

We have completely annotated 289 cars in right-left traffic for structural model learning. In addition, we have also annotated 262 extra wheels in the sequences with opposite traffic direction, so we have 840 wheel annotations in total. However, some of the annotated wheels are at too big scale or too close to image border to be used for training. The wheel training data set therefore contains only 475 positive samples, the data sets for A-pillar and B-pillar AdaBoost detector training count 177 and 127 positive samples, respectively. The data set that was used to determine side panel measure distribution, contains 225 positive samples.

5.2 Experimental results

We present results of detection at multiple scales on sample video frames. We emphasise the contribution of each detection stage.

At the first stage, we detect single wheel candidates. Figure 5.1a shows result of single wheel AdaBoost detection at multiple scales with a very high false positive rate. The wheel candidates are marked with squares, whose size corresponds with scale at which the particular candidate was detected. We could eliminate majority of the false detections by setting a higher threshold for the local minima values, but the results of consequent detection stages wouldn't be comparable then.

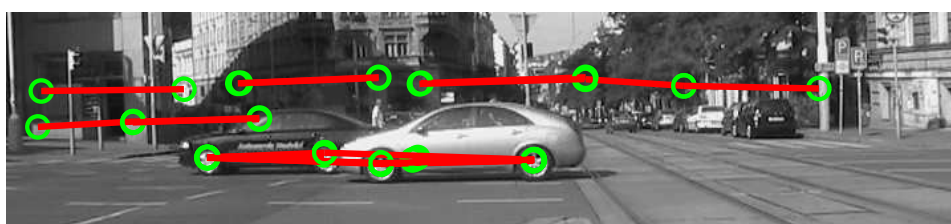
At the second stage, we form potential pairs of wheel candidates from stage one using the hard constraint on the relative displacement of wheels in a pair mentioned in Section 4.2. Figure 5.1b demonstrates, how the constraint can eliminate a great part of false wheel candidates. The potential wheel pairs are denoted as wheel candidates (green circles) connected by a line.

The results improve, when we use the final detector with probabilistic model at the third stage. The cars in Figure 5.1c were detected as pairs of wheels – the detector evaluated only wheel detection responses \mathbf{R}' and relative displacements of wheels within the pairs and did not use the A-pillar nor the side panel evaluation. As it can be seen from Figure 5.1d, we are able to eliminate all false positives with the use of all integrated component detectors in this particular case. The last step of detection is the removal of the overlapping car detections described in Section 4.3.1, whose results are shown in 5.1e.

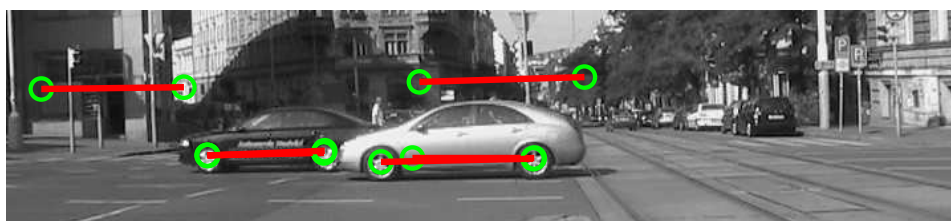
Figure 5.2 shows the same process as Figure 5.1 for the opposite traffic direction.



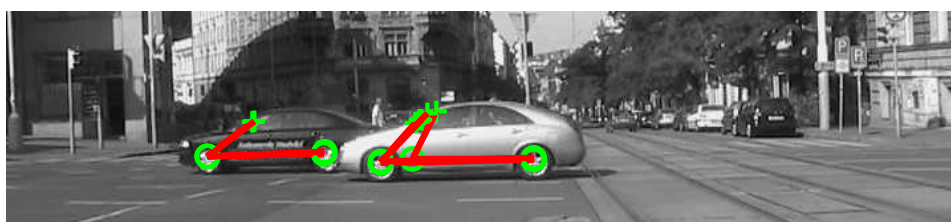
(a) Multi-scale detection of single wheels (size of squares denotes scale of detection)



(b) All wheel pairs complying the wheel displacement constraint, denoted as green circles (detected wheel centres) connected by red line



(c) Final detector output using only wheel pair evaluation (only wheel detector responses R' and geometrical displacement of wheels were evaluated)



(d) Final detector output using all components evaluation (green crosses denote A-pillar detections pertaining to particular wheel pairs)

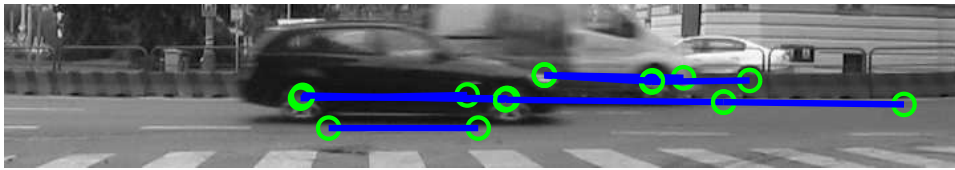


(e) Final detector output, removed overlaps

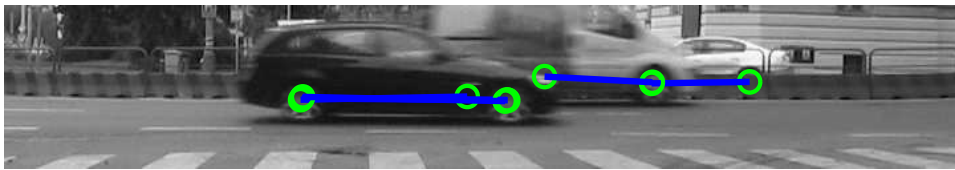
Figure 5.1: Consecutive stages of detection, given right-left traffic direction



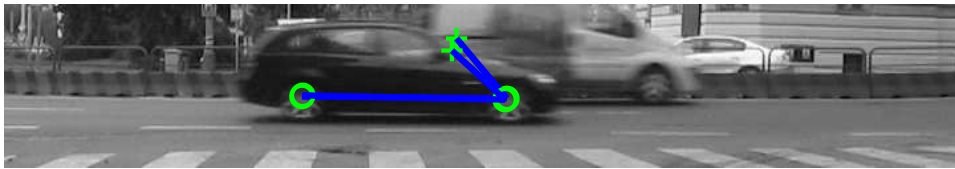
(a) Multi-scale detection of single wheels (size of squares denotes scale of detection)



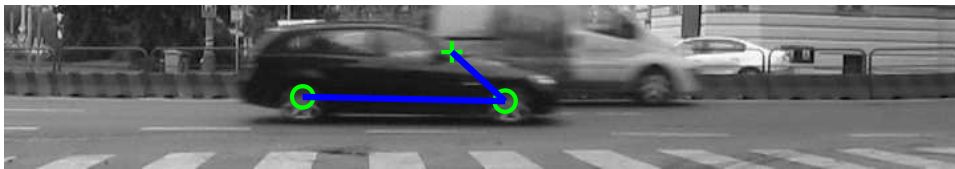
(b) All wheel pairs complying the wheel displacement constraint, denoted as green circles (detected wheel centres) connected by blue line



(c) Final detector output using only wheel pair evaluation (only wheel detector responses R' and geometrical displacement of wheels were evaluated)



(d) Final detector output using all components evaluation (green crosses denote A-pillar detections pertaining to particular wheel pairs)



(e) Final detector output, removed overlaps

Figure 5.2: Consecutive stages of detection, given left-right traffic direction



Figure 5.3: False positive example

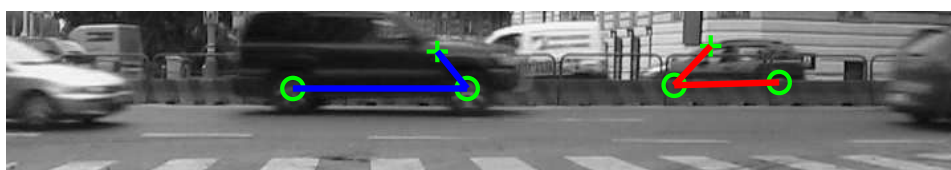


Figure 5.4: Automatic detection of driving direction

Even with the final detector, we cannot avoid all false detections, because the false positives sometimes fit the structural model quite well, as it is shown in Figure 5.3. The shadow on the building looks like the edge of an A-pillar, whereas the horizontal edge of the side panel is perfectly substituted by the roof of the car beneath the false detection.

The last thing we would like to present is the automatic detection of the driving direction in Figure 5.4. Although it is not very reliable, because the edge-based A-pillar detector is sensitive to reflections on the car bodies and sometimes is also activated by the background that can be seen through the car windows, we consider it to be an interesting feature.

Chapter 6

Conclusion

We have designed several detectors for suggested components of the car side-view. Having collected and annotated data for experiments, we have implemented the component detectors in Matlab.

It has been shown during experiments that the AdaBoost algorithm, though powerful, is not suitable for some features and that detectors based on different and simpler principles can work better and more efficiently. The AdaBoost works fine for the wheels, but it is outperformed by the edge direction detection in the case of the A-pillar. In addition, we cannot neglect that AdaBoost detector is quite computationally demanding.

After experimental testing of particular component detector performance, we have chosen wheels, A-pillar and side panel for integration into the structural model. Consequently, we have learnt the structural model and implemented the final detector. Our final detector is a general framework based on a probabilistic model. Additional or different component detectors and measures can be integrated into the detector in the same way as the current ones. The more powerful component detectors we use, the better will be the performance of the final detector. The detector can return not only position of the car in the input image, but also its driving direction.

In conclusion, we have developed and tested a working detection method for car side-views. For a practical use, however, it would have to pass through a thorough optimisation, because current computation time for one frame with resolution 120×720 pixels at 9 scales is around 50 seconds when mea-

sured on a notebook with Intel Core 2 Duo T8100 @ 2 GHz. The most computationally demanding part of detection is the computation of convolutions with masks. It takes up about 93% of the computation time. Fortunately, convolution computation is very effectively implemented on FPGA, which is commonly used in automotive industry.

Bibliography

- [1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [2] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. John Wiley, 2nd edition, 2001.
- [3] P. F. Felzenszwalb et al. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), September 2010.
- [4] B. Heisele, P. Ho, J. Wu, and T. Poggio. Face recognition: component-based versus global approaches. *Computer Vision and Image Understanding*, 91:6–21, 2003.
- [5] B. Leung. Component-based car detection in street scene images. Master's thesis, Massachusetts Institute of Technology, 2004.
- [6] M. Uřičář and V. Franc. Detector of landmarks on human face. *16th Computer Vision Winter Workshop*, February 2011.
- [7] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. PWS Publishing, 2nd edition, 1999.

Appendix A

Contents of the attached CD

A CD with Matlab source codes and other materials is attached to the thesis. The content of the CD is organised in following directories:

- *Thesis/*: contains electronic version of the thesis text
- *Codes/*: contains Matlab source codes
- *Data/*: contains learnt detectors, video-sequence annotations and training data
- *Video/*: contains sample videos with detections