

# A Real-time Object Detection Framework using Resource Optimized Cascaded Perceptron Classifiers and its Application to US Speed Limits

**Armin Staudenmaier, Ulrich Klauck**

Hochschule Aalen  
E-Mail: armin.staudenmaier@gmx.de

**Ulrich Kreßel, Frank Lindner**

Daimler AG

**Christian Wöhler**

Technische Universität Dortmund

## Abstract

In this study we present a machine learning framework for object detection which consists of a training stage and a detection stage. We use a *cascaded classifier* which is composed of different *perceptron classifiers* differing in dimensionality, geometrical peculiarity, and feature types. In contrast to many other systems we do not use the classical AdaBoost approach and the concept of weak classifiers, but instead employ perceptrons of variable dimensionality and features, which are chosen in a classifier selection process and are added in each stage to the cascade classifier. The basic idea of the selection is that we start with a set of classifiers in a *very low resolution subspace*, which is achieved by geometrical constraints of the classifiers themselves rather than by resizing the image. A further contribution of this study is the introduction of a *cost-performance function* for rating a binary classifier which consists of a weighted ratio of the operational cost required for a single classification operation and the classifiers recognition performance. We explain how this cost-performance function models the probability of false operations and therefore can produce very effective cascaded classifiers when used as a *selection criterion* and as a *stopping criterion* for classifiers. We also show how this function can be used with cascaded classifiers for cost-performance prediction. Our cascaded classifier is composed of resolution subcascades. As long as the cost-performance function decreases, we stay in the current resolution subspace and add the classifier which yields the minimum value until the function increases. We thus obtain a cascade that consists of subcascades of increasing resolution with a decreasing cost-performance ratio for each new stage minimizing the probability of false operations. The training algorithm which is a Fisher Linear Discriminant Analysis yields a high generalization performance also for nonseparable data due to the effective computation of weights and margins by statistical modelling in feature space. The combination of linear classifiers of variable dimension and the integration of different resolution levels, which can be seen as a coarse-to-fine-search, yields very short computation times both for the training and for the detection stage.

## 1 Introduction

Object detection systems are used in a wide variety of applications. This is due to the growth of computational performance and at the same time decreasing dimensions of controllers. Especially, the field of Advanced driver assistance systems (ADAS) for cars offers a wide area of different detection tasks such as traffic sign detection, pedestrian detection or curb detection just to mention a few. Such systems can be used on the one hand as information advising systems to show the driver relevant information or, integrated into an adaptive cruise control system (ACC) assist the driver. Detecting potentially dangerous situations can help to minimize accidents. Even the strict adherence to speed limits for example can reduce the risk of accidents while at the same time saving energy. Most systems evolving in recent years use different stages during the detection. Our aim is to design a general system that automatically builds a classifier system being resource optimized and therefore working efficiently for a wide variety of detection tasks.

## 2 Related Work

Machine learning algorithms are increasingly applied to object detection tasks. One of the most famous systems is the face detection system presented by Viola and Jones [7, 6]. In that work, cascaded classifiers consisting of strong classifiers are trained. Each strong classifier consists of weak classifiers that have fast computation times due to the use of integral images for their computation. The strong classifiers are created by the AdaBoost algorithm [8, 9], where each weak classifier is assigned a weight dependent on its classification performance and therefore the decision of the strong classifier can be seen as a weighted majority vote. Many other authors introduce new fast features for the weak classifiers based on integral images. Dalal and Triggs introduce HOG features [12] which are used for example in combination with a SVM classifier for pedestrian detection [13] and with a cascaded classifier [11]. The detection of U.S. speed limits is presented in [18, 19], both using in the first stage a shape based detector followed by a Viola Jones classifier [18] and a neural network digit recognition module [19].

Nearly all the mentioned methods contain a classifier selection process where quality functions are needed for the rating of the classifiers. Mostly the cost functions depend on the type of misclassifications. The model described by Zhang [14] assigns different losses to different kinds of mistakes depending on the type of faces in a face recognition system. Another algorithm called stacking, which is adapted to the task of classifier learning for misclassification cost performance, is presented in [15]. A detailed introduction and overview is given in [16]. A cost model for a cascaded AdaBoost classifier which is solved there as a constrained numerical nonlinear optimization problem is presented in [17]. This model also incorporates the operational cost of the AdaBoost classifier.

While AdaBoost provides an effective learning algorithm ensuring strong bounds on generalization performance [9], Freund and Schapire also use large margins to achieve generalization performance with the perceptron algorithm [10].

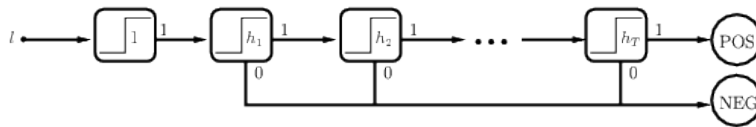
### 3 Base classifiers

The main algorithm described here is a greedy method. We have a pool of different binary classifiers  $h_i \in \mathcal{H}$  which we call *base classifiers*. In a selection process we iteratively choose a classifier from the pool and add this classifier to a *cascaded classifier* [7] by means of a quality function.

A cascade classifier is in principle a series of ordered binary classifiers. The decision function of a cascade classifier  $h_C$  with  $T$  stages containing  $h_1, \dots, h_T$  binary classifiers is as follows:

$$h_C^T = (h_1, \dots, h_T) = \begin{cases} 1 & \text{if } T = 0 \\ 0 & \text{if } h_j = 0, 1 \leq j \leq T \\ 1 & \text{else} \end{cases} \quad (1)$$

The first case ensures that if no classifier is contained in the cascade, it will always answer with “yes” and therefore never produce a false negative. If a sample is presented to the cascade, it will ask the first classifier. If it answers with “no”, no other classifier has to be considered. The cascade will only answer with “yes” if one classifier after the other answers “yes” including the last one. This is shown in figure 1. The famous detection system



**Figure 1:** Cascade classifier composed of binary classifiers.

of Viola and Jones uses so-called strong classifiers in the cascade which consist of several weak classifiers. The weight of each weak classifier is chosen depending on its classification performance. A weak classifier is one-dimensional and uses simple but fast features. Here we present an algorithm that only uses one type of base classifiers and automatically rates the classifiers depending on the classification performance and operational costs. Arbitrary features can be used for the base classifiers which can be multidimensional and the weighting of the features is done with a flexible learning method, therefore the algorithm is not only restricted to detection tasks in image processing.

The base classifiers are linear perceptron classifiers with  $n$ -dimensional weight vectors that are trained by a linear discriminant analysis procedure. The classification function is a perceptron[4]:

$$h = \begin{cases} 1 & \text{if } \vec{w}^T \vec{x} - \theta > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$(3)$$

where  $\vec{x}$  is the extracted feature vector,  $\vec{w}$  is the trained weight vector and  $\theta$  is a specific threshold. The perceptron classifier is the most simple feedforward neural network and consists of one output layer with one node. This classifier is linear, since the weight vector can be regarded as the normal of a multidimensional plane in feature space which separates the samples. The features  $x_i$  are the components of the feature vector  $\vec{x}$ . They are multiplied by the weights of the weight vector and summed up afterwards. The original perceptron learning rule iteratively updates the weights of a perceptron depending on

the classification of a sample. Since we claim specific characteristics concerning the classification and time performance we use a different learning method for the determination of the weight vector  $\vec{w}$  and the threshold  $\theta$ .

### 3.1 Base classifier training

The weight vector  $\vec{w}$  of the base classifiers can be interpreted as an axis with a specific direction in feature-space, which should have best discrimination properties. To compute this direction for each classifier we use Fisher’s Linear Discriminant Analysis [5, 3]. We have a list of positive feature vectors  $\vec{x}_{pos} \in \mathcal{P}$  generated with the feature extraction function from the positive samples containing objects and a list of negative feature vectors  $\vec{x}_{neg}$  generated by negative samples without objects which have approximately the same size. The negative examples are randomly chosen from a huge set of negative examples that do not contain any positive objects. Out of these vectors we compute the mean vectors  $\vec{m}_{pos}, \vec{m}_{neg}$  and the scatter matrices  $S_{pos}, S_{neg}$  and  $S_w = S_{pos} + S_{neg}$ . Then  $\vec{w}$  is computed by

$$\vec{w} = \alpha S_w^{-1} (\vec{m}_{pos} - \vec{m}_{neg}) \quad (4)$$

and normalized such that  $\|\vec{w}\| = 1$ .

For the computation of the threshold  $\theta$  there are three main aspects. First we should guarantee a high detection rate. Second we should be aware that not all positive samples must be *correct* positive samples. It could be the case that the human expert made some mistakes during the label process or that some very “difficult” labels are contained. For the classifier it could be better to leave out these perhaps wrong or difficult samples. And third we should incorporate a separability measure which tells us the difficulty of the separation. For easy problems we can set the threshold nearer to the positive samples.

We use the p-quantile to leave out positive samples which we expect to be unusual. The computation of the p-quantile  $Q_p$  of positive samples is done by projecting all positive features  $\vec{x}_i^T \vec{w}$  and sorting according to the projected value. The lower the values, the smaller is the distance to the projected negative mean. The sample with the lowest projected value that is contained in  $Q_p$  is  $\vec{x}_q$ . We compute the threshold according to

$$\theta = \vec{x}_p^T \vec{w} - g \quad (5)$$

where  $g \geq 0$  can be regarded as a generalization constant. The larger this value is, the more space is between the projected quantile  $\vec{x}_p$  and the threshold  $\theta$ .

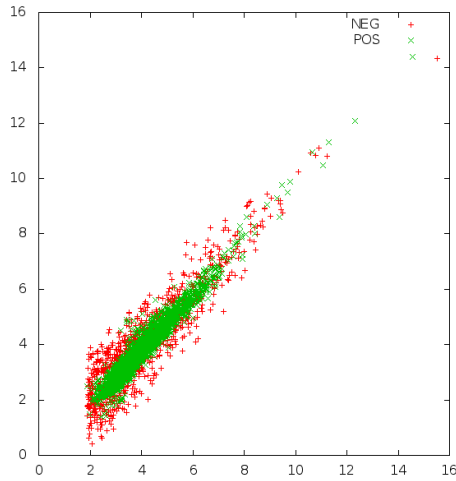
We compute the constant  $g$  based on the weighted variance of the projected positive samples and on an additional constant value  $q_2$ .

$$g = q_1 \sigma_{Q_p}^2 + q_2 \quad (6)$$

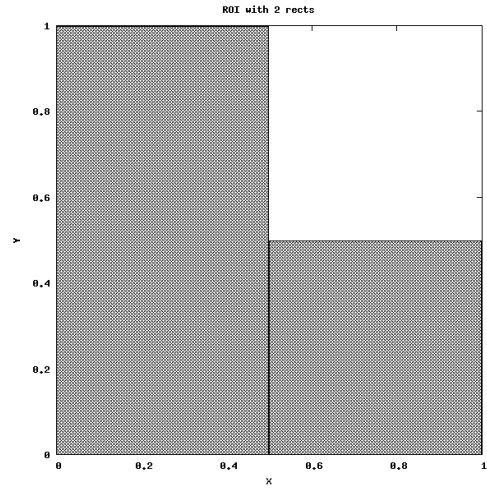
Figure 2 shows two scatter plots of the first two features from two perceptron classifiers. The green points are object points in feature space and the red points are non-object points.

## 4 Performance Measurement

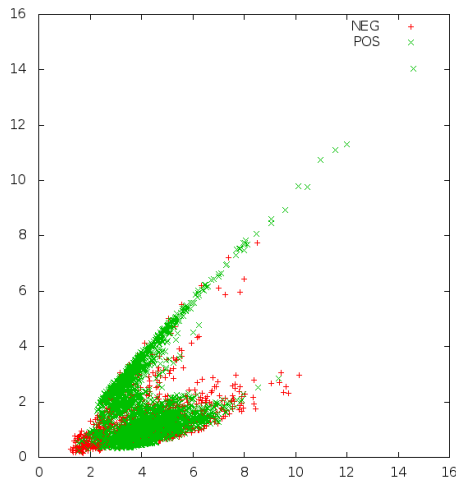
As described above we use perceptron classifiers with variable dimensions and a large variety of features which have differing operational costs. Therefore we introduce a power



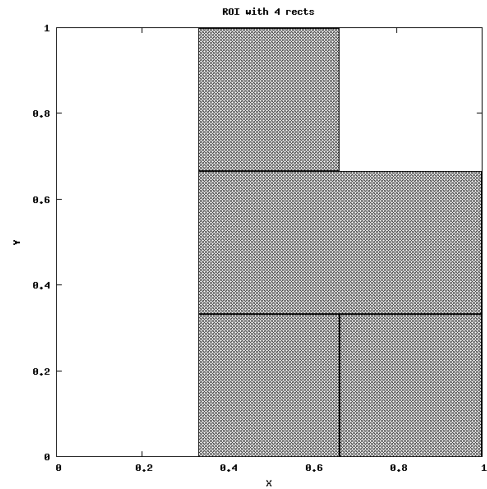
(a) Scatter plot of 4-dim classifier with  $\Delta k_{\frac{1}{2}}$



(b) Typical rectangle configuration with  $\Delta k_{\frac{1}{2}}$



(c) Scatter plot of 24-dim classifier with  $\Delta k_{\frac{1}{3}}$



(d) Typical rectangle configuration with  $\Delta k_{\frac{1}{3}}$

**Figure 2:** Scatter plots of two perceptron classifiers together with their corresponding rectangle configurations. Green points are object points, red points are non-object points. The features shown are average intensity and variance of pixel gray levels.

function that combines the operational cost for a classifier with its classification performance as a general performance measurement method for arbitrary classifiers. Since we always have the opportunity to add another classifier to the cascaded classifier the idea is that the classifiers should have a favorable ratio between classification performance and operational costs needed for a classification. We also show a way how the operational cost of a classifier can additionally be weighted by its classification performance. By doing this we can give good classifiers more computational resources. This cost-performance ratio should implicitly rate how strong or weak a classifier is, regarding its operational cost and classification performance. The cost-performance function measures “likely false operations per classification”. This function can be used on the one hand for the base *classifier selection* and on the other hand as a *stopping criterion* for the cascade training. We also show a way how a cost-performance prediction on a small sample set can be achieved.

## 4.1 Cost-Performance Function

We use the probability of a wrong decision  $P(h)$  of a classifier  $h$  as a *classification performance* measurement since we expect that a wrong decision will generate additional operational costs for the classifier for compensation or additional costs resulting from the misbehavior of the detection system itself. Therefore a low probability of a wrong decision yields a low probability of additional operational costs and vice versa. We use the operational cost  $C(h)$  for one classification as a *operational cost performance* measurement. The unit of the operational cost can be chosen arbitrary, for example in mips or as absolute time. Therefore we are independent of a specific architecture and the training and recall can run on different platforms, since the cost for each operation of a classifier can be assigned according to the target platform of the recall. We define the cost-performance function as

$$\Psi(h) = C(h)^X P(h). \quad (7)$$

The exponent  $X$  is a cost weighting factor. For the case where  $X = 1$  the costs have the unit [operations/classification], and multiplied with the probability of a wrong decision we can interpret the unit of  $\Psi$  as [likely false operations/classification] which we want to minimize since they will generate additional costs.

If we know in each stage of a cascade classifier  $h_C$  the probability of appearance of costs, “likely costs” of the cascade can be computed. We approximate the appearance probabilities of costs with the probability of a wrong decision of the previous stage since we claim that wrong classified costs will survive the actual stage.

The probability of costs in stage  $k$  in the cascade for example is  $(\prod_{j=0}^{k-1} p_j)c_k$ . The absolute expected costs of the cascade classifier per classification are [17]:

$$C(h_C) = \sum_{i=1}^T \left( \prod_{j=0}^{i-1} p_j \right) c_i \quad (8)$$

with  $c_i = c(h_i)$  the cost of the base classifier  $h_i$  in stage  $i$  and  $p_i = p(h_i)$ ,  $p_0 = 1$  the probability of a wrong decision.

Expected global costs of a classifier can then easily be computed with the total amount of classifications  $n$ :

$$C_G(h) = C_0 + nC(h) \quad (9)$$

where  $C_0$  are initialization costs for the computation of integral images for example. We measure the performance of a cascade classifier based on the wrong decision probabilities of the base classifiers:

$$P(h_C) = \prod_{i=0}^T p_i \quad (10)$$

The cost-performance function of the cascaded classifier can then be computed with equation 7.

## 4.2 Classification Performance

As described above, we measure the classification performance of a classifier regarding the probability of a wrong decision which is in our case the probability of a false classified sample. This can be measured for a negative sample with the false positive rate

$p(\text{false}_h|\text{neg}) = FP_R(h)$  and for a positive sample with the false negative rate  $p(\text{false}_h|\text{pos}) = FN_R(h)$ . In other words, the higher the false positive rate of a classifier, the higher is the probability that a negative sample is misclassified. Therefore, our classification performance function corresponds to:

$$P(h) = \alpha FP_R(h) + \beta FN_R(h) \quad (11)$$

The rating between false positives and false negatives is specified by the parameters  $\alpha$  and  $\beta$ . For typical detection tasks false negatives are often more disadvantageous than false positives. To assign higher costs for false negatives we could set  $\alpha \ll \beta$ .

### 4.3 Performance-dependent cost weighting

In the training phase of a cascade classifier, only misclassified samples are considered in the following stages. Therefore, the higher the stage, the more difficult are the samples presented to the base classifiers but the better is the performance of the cascade classifier. The idea is now to focus more on the classification performance the better the classifier is:

$$\Psi_W(h) = P(h)C(h)^{P(h)} \quad (12)$$

The higher the stage, the better the performance. This means a low probability of a wrong decision  $P(h)$ , and therefore the cost factor  $C(h)^{P(h)}$  approaching to 1. As a result we reduce the total number of base classifiers in higher stages.

### 4.4 Cost-performance prediction

Since we optimize our cascade in a greedy process we can predict the cost-performance function for the cascade based on our trained perceptron classifiers. The prediction with the current Cascade  $C_T = C(h_C^T)$ ,  $P_T = P(h_C^T)$  for the next  $k$  stages and base classifiers  $h_i$  for  $1 \leq i \leq k$  can be computed with:

$$\Psi_{Pred}(h_C, h_1, \dots, h_k) = (C_T + \sum_{i=1}^k (\prod_{j=1}^{i-1} P_T p_j(\mathcal{S}_j)) c_i(\mathcal{S}_j))^X P_T \prod_{i=0}^k p_i(\mathcal{S}_i) \quad (13)$$

Where  $p_0 = 1$  and  $\mathcal{S}_i = \{\mathcal{P}, \mathcal{N}_i\} = \{\mathcal{P}, FP(\mathcal{N}, h_C(h_1, \dots, h_i))\}$  are the misclassified samples in each stage. Again,  $X$  is the weight factor that can be set according to equation 12. We refer to the term prediction since the number of negative samples can be adjusted according to the available computational resources for the training by using an appropriately sized subset of randomly chosen samples from the original set of negatives. Also the amount of classifiers for the computation of  $\Psi$  can be set for the search. If we use one classifier, we have to evaluate  $\Psi(C_T, h_i)$  for all  $h_i \in \mathcal{H}$ . For two classifiers we have to evaluate  $\Psi(C_T, h_i, h_j)$  for all  $(h_i, h_j) \in \mathcal{H}$  which are all pairs. Using all classifiers with all combinations would then result in a global search which would require to evaluate the function on all combinations of classifiers .

## 5 Cascade Training

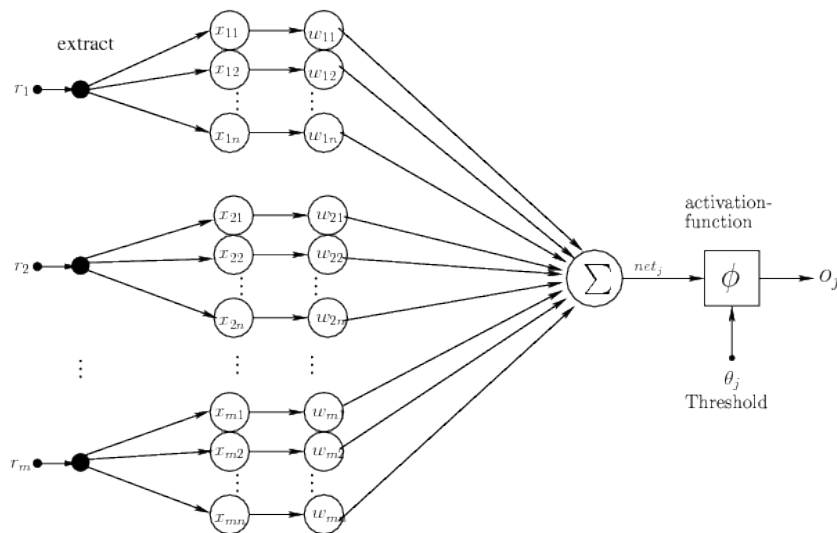
The overall performance of a detection system is heavily dependent on the type and amount of features. The first consideration is that the extraction of features in a rectangular regions is nothing else than a local transformation of the underlying signal into a specific resolution subspace which depends on the structure of that region. For example, taking the sum of pixel grey values and dividing by the area is in principle a convolution with a constant kernel mask. The next consideration is that the oscillation of the classifier signal depends on that transformation. Put differently, the lower the resolution the lower the oscillation of the signal of each classifier. We use this behavior to improve both the training and the recall.

### 5.1 Resolution dependent training

Our perceptron classifiers have assigned a list of rectangles, from which features are extracted. Each classifier is assigned a  $k$ -elementary subset of rectangles from a  $n$ -elementary set of rectangles which do not overlap:

$$\mathcal{R}_i = \{r_k | w(r_k) \geq \Delta k, h(r_k) \geq \Delta k\} \quad (14)$$

where  $w(r)$  is the width and  $h(r)$  is the height of a rectangle  $r$ . A perceptron classifier  $b_i$  extracts features only within the regions of rectangles  $\mathcal{R}_{b_i}$ . Since we train with a discriminant analysis, the overlap of rectangles will partly override the information contained and we remove rectangles that overlap with other rectangles. Figure 3 shows the extraction of features with a specific extraction function for each rectangle  $r_i$ .



**Figure 3:** Perceptron classifier. Features are extracted from within the rectangles  $r_1, \dots, r_m$ .

For the evaluation we hold a large set of samples  $\mathcal{G} = \{\mathcal{P}, \mathcal{N}_{\Delta k}\}$  where the negative samples are generated by moving detection boxes  $R_i$  at different sizes over all images  $I$  and collecting all samples that do not intersect with a positive sample. The step size depends on the actual size of the box and on the actual resolution  $\Delta k$ . Then we generate a

smaller set  $\mathcal{S}$  by random sampling of  $\mathcal{G}$  and filtering  $FP$ s and  $TP$ s by the actual cascade  $h_C^T$ :

$$\mathcal{N}^T = \{FP(h_C^T, \mathcal{N}_{\Delta k})\} \quad (15)$$

$$\mathcal{P}^T = \{\mathcal{P}\} = const \quad (16)$$

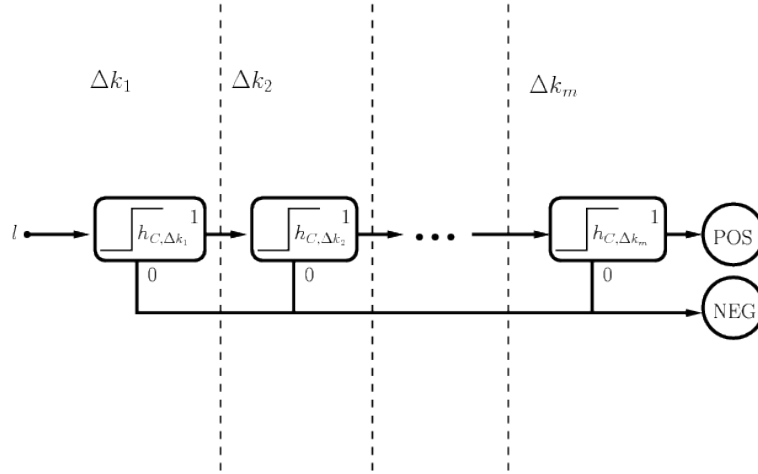
The size of  $\mathcal{N}^T$  is restricted to  $|\mathcal{N}^T| \simeq |\mathcal{P}^T|$ . We train each classifier  $h_i \in \mathcal{H}_{\Delta k}$  as described above on the set  $\mathcal{S} = \{\mathcal{P}^T, \mathcal{N}^T\}$ . The “best” base classifier  $h_{min}$  is the one with a minimum  $\Psi_{Pred}$ , which is the *selection criterion*:

$$h_{min} = \arg \min_i \{\Psi_{Pred}(h_C^T, h_i) \mid h_i \in \mathcal{H}_{\Delta k}\} \quad (17)$$

This classifier is added to the existing cascade  $h_C^{T+1} = h_C^T \otimes h_{min}$ . The selection process is repeated as long as the cost-performance function is decreasing on the global set  $\mathcal{G} = \{\mathcal{P}, \mathcal{N}_{\Delta k}\}$ :

$$\Psi(h_C^{T+1}, \mathcal{G}) < \Psi(h_C^T, \mathcal{G}) \quad (18)$$

which is our *stopping criterion* for the selection process. If this relation is violated, we increase the resolution by decreasing the minimum rectangle length  $\Delta k' < \Delta k$ . Then we generate a new set of perceptron classifiers with  $\Delta k'$  and new samples and go on with the selection of perceptron classifiers as long as equation 18 holds. Then the selection process can be divided into *resolution stages*, where only classifiers  $\mathcal{H}_{\Delta k}$  with constant resolution  $\Delta k = const$  are used and each scale stage has its subcascade  $h_{C, \Delta k}$ . If we use different resolutions  $\Delta k_1 > \Delta k_2 > \dots > \Delta k_m$ , each resolution is assigned a cascade. This is shown in Figure 4.

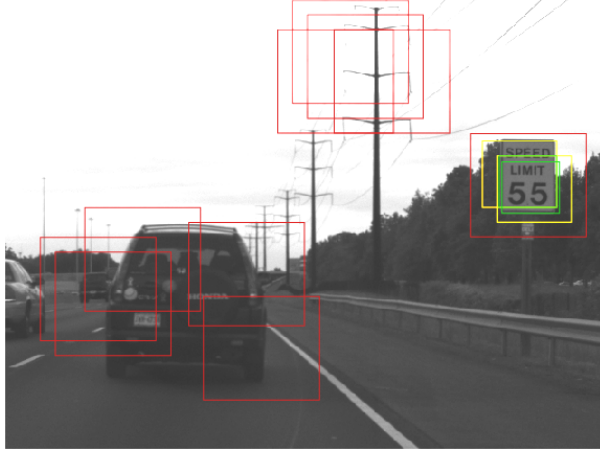


**Figure 4:** Resolution cascade containing resolution dependent subcascades.

## 6 Detection

The basic idea for the detection algorithm is that we use our resolution subcascades for a coarse-to-fine search in the image at increasing resolutions. A subcascade only triggers the transition to the next higher resolution inside the current box if it detects an object. This *recursive* coarse-to-fine search is very effective.

For the detection algorithm we start with the first resolution and generate all boxes  $\mathcal{R}_{\Delta k_1}$  inside the image. For each box we request the subcascade  $h_{C, \Delta k_i}$  for the current resolution level. If it rejects the box, we continue with the next box. Otherwise, if it detects an object, we pass this box to the next higher resolution subcascade  $h_{C, \Delta k_{i+1}}$  and generate a new set of boxes depending on  $\Delta k_{i+1}$  where we again start to classify each box. If the subcascade in the last resolution stage affirms an object, we have achieved the terminating condition of the recursion, and the detection is deemed to be successful. An example is given in Figure 5 with three different resolution stages.



**Figure 5:** Detection with three different resolutions and subcascades. First stage is red, second stage is yellow and third stage is green.

## 7 Experiments

The different parameters and the cost-performance function as selection and stopping criterion make it difficult to determine an appropriate parameter combination. This is why we trained 144 cascades with starting resolution 0.5 and 144 cascades with starting resolution 0.25 using different parameter combinations and we picked out those cascades with a false positive rate below 0.0005. For training we used 2800 positive samples extracted from about 250 Sequences of US-Speed-Limit signs containing 13 different numbers ranging from 10 to 70. The average training time for a cascade was about 4 hours. After each resolution stage we evaluated the resolution cascades and the contained base classifiers on a validation set. The following tables show important attributes of the cascades concerning their structure and parameters. We used a consistent numbering of the different cascades as unique id. Since for all cascades the overall stopping criterion for the training and for each resolution subcascade was done with equation 18 we show the configuration of cascades with the total amount of base classifiers and the amount of classifiers in each resolution stage in Table 1. For  $\Delta k_1$  only the rectangle over the whole detection area is assigned to the base classifiers. For  $\Delta k_{\frac{1}{2}}$  the detection area is divided by two and therefore there are 9 possible sub-rectangles. The cascades 6, 9, 10 and 13 were trained by skipping resolution  $\Delta k_{\frac{1}{2}}$  and  $\Delta k_{\frac{1}{3}}$ . The same basis features which are extracted from inside the rectangle regions are used for all experiments. To compare the performance we listed the detection rate and the false positive rate together with the operations per classification in table 3. The operations per classification are measured as average time needed for one classification in [microseconds/classification]. The values are averaged over 4000 classifications. Table

Id	Total	$\Delta k_1$	$\Delta k_{\frac{1}{2}}$	$\Delta k_{\frac{1}{3}}$	$\Delta k_{\frac{1}{4}}$
1	255	5	42	208	0
2	255	5	39	211	0
3	174	2	16	60	96
4	120	3	11	44	62
5	117	2	7	20	88
6	112	3	0	0	109
7	109	2	7	20	80
8	105	2	0	0	103
9	97	3	0	0	94
10	44	1	0	0	43
11	41	1	1	1	38
12	30	1	2	3	24
13	30	5	6	8	11
14	27	5	0	0	22

**Table 1:** Overview of the amount of base classifiers contained in the resolution stages of cascade with Id  $j$  where  $\Delta k_i$  is the current resolution.

Id	$\alpha$	$\beta$	$X$	$g$	$\mathcal{Q}$
1	1	1	$0.1P(h)$	$g_1$	1
2	1	1	0.1	$g_1$	1
3	1	1000	$P(h)$	$g_2$	1
4	1	1000	$P(h)$	$g_1$	1
5	1	1	$P(h)$	$g_4$	0.999
6	1	1000	$P(h)$	$g_3$	1
7	1	1	$P(h)$	$g_1$	0.999
8	1	1	$P(h)$	$g_3$	1.
9	1	1000	$P(h)$	$g_3$	1.
10	1	1	2	$g_3$	0.999
11	1	1	2	$g_2$	0.999
12	1	1	2	$g_4$	0.99
13	1	1	0.1	$g_5$	0.99
14	1	1	$0.1P(h)$	$g_4$	0.99

**Table 2:** Parameters used for the different cascades with  $g_1 = 0.001\sigma^2 + 0.001$ ,  $g_2 = 0.001\sigma^2$ ,  $g_3 = 0.01\sigma^2$ ,  $g_4 = 0.01\sigma^2 + 0.001$  and  $g_5 = 0.1\sigma^2 + 0.001$ .

Id	$TP_R$	$FP_R$	Operations
1	0.931	$2.7 \times 10^{-4}$	33.2
2	0.936	$2.3 \times 10^{-4}$	34.6
3	0.921	$2.5 \times 10^{-5}$	0.3
4	0.950	$2.2 \times 10^{-4}$	0.7
5	0.897	$1.4 \times 10^{-5}$	0.9
6	0.941	$3.7 \times 10^{-4}$	1.9
7	0.897	$1.6 \times 10^{-5}$	1.5
8	0.916	$2.6 \times 10^{-4}$	2.0
9	0.936	$2.6 \times 10^{-4}$	1.2
10	0.936	$2.7 \times 10^{-4}$	0.7
11	0.936	$2.8 \times 10^{-4}$	3.3
12	0.848	$4.0 \times 10^{-6}$	1.7
13	0.789	$3.0 \times 10^{-6}$	1.4
14	0.833	$4.0 \times 10^{-6}$	1.0

**Table 3:** Performance of each cascade measured by its detection rate, false positive rate and average operational costs in [microseconds/classification].

2 shows the different parameters used for training. For the quantiles with  $Q = 1$ . we did not sort out any positive objects, for  $Q = 0.999$  we sorted out the lower permille and for  $Q = 0.99$  we sorted out the lower percent of positive objects along the discrimination axis. Since the weighting of  $\alpha$  and  $\beta$  influences the selection function, we used  $\alpha_1 = \alpha_2 = 1$  which means that we rate false positives the same as false negative samples and therefore the selection function chooses classifiers dependent on wrong classifications equally rating positives and negatives. The detection rate is then dependent on the threshold of each base classifiers computation of each base classifiers threshold for which we used five combinations:  $g_1 = 0.001\sigma^2 + 0.001$ ,  $g_2 = 0.001\sigma^2$ ,  $g_3 = 0.01\sigma^2$ ,  $g_4 = 0.01\sigma^2 + 0.001$  and  $g_5 = 0.1\sigma^2 + 0.001$ .

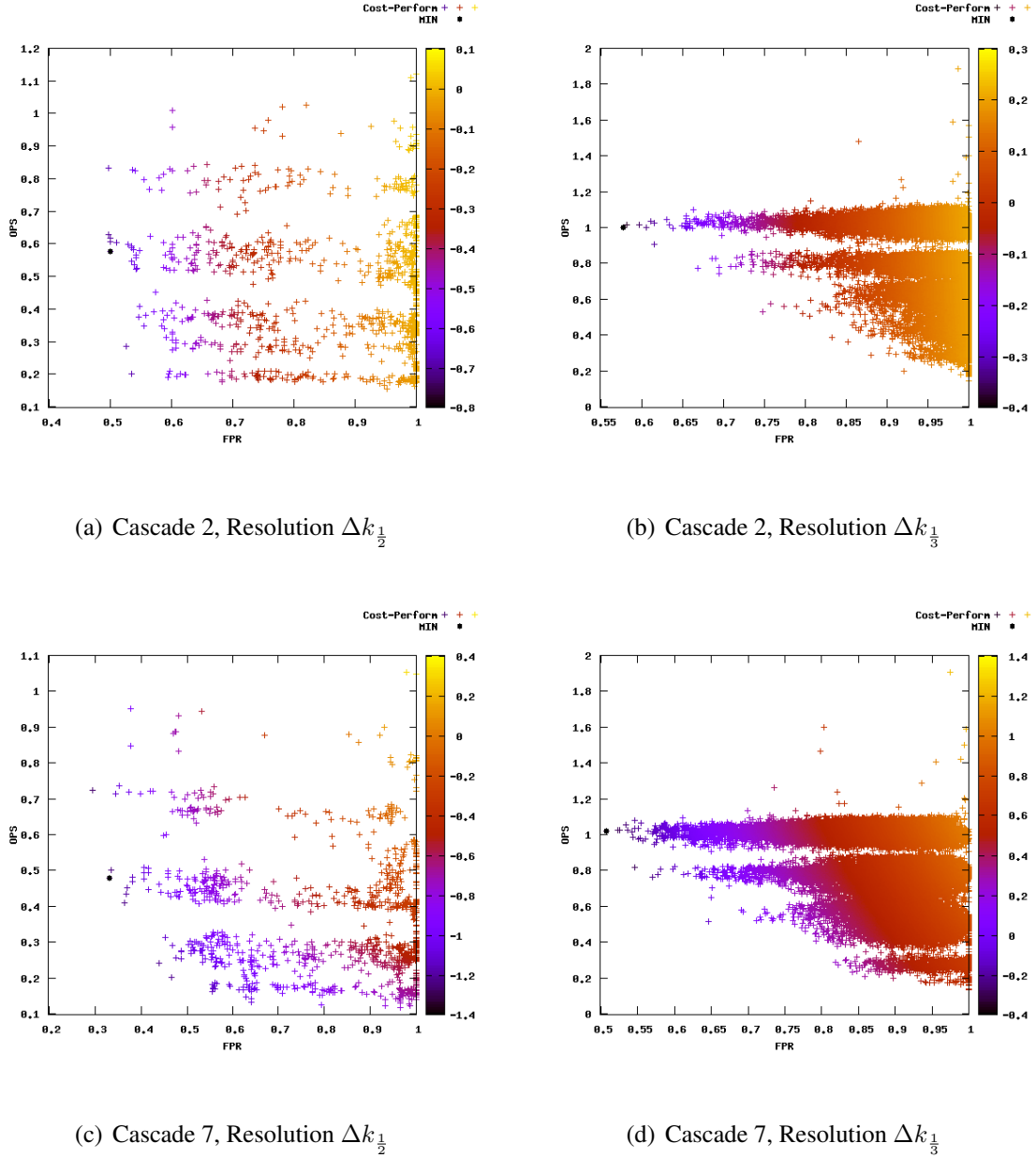
## 7.1 Performance discussion

Table 2 clearly shows that the cascades with a low cost weighting have much higher averaged operational cost than those with a higher weighting when claiming comparable detection rates. To illustrate the effect of different cost weighting we compare the values of the cost-performance function with the whole pool of classifiers during the training of cascade 2 and 7 since they have large differing average operational costs but the same performance weight parameters  $\alpha$  and  $\beta$ . Figure 6 shows 2D plots of the cost-performance function  $\Psi_{\text{Pred}}$  of all trained base classifiers  $\mathcal{H}_{\Delta k_{\frac{1}{2}}}$  on the training set in the first stage with resolution  $\Delta k_{\frac{1}{2}}$  of cascade classifier 7 and 2. The false positive rate is assigned to the  $X$  axis and the averaged operations are assigned to the  $Y$  axis. Since we use the same feature types in each resolution step the amount of base classifiers was the same for both cascades,  $|\mathcal{H}_{\Delta k_{\frac{1}{2}}}| = 1000$  and  $|\mathcal{H}_{\Delta k_{\frac{1}{2}}}| = 30000$ . Images 6(a) and 6(b) show a vertical color gradient while image 6(c) and 6(d) shows a more diagonal gradient since the costs have more influence.

Images 7(a) and 7(b) show 3D plots with an additional axis for the false negative rate. These plots show that the classifiers are clustered for  $\alpha = \beta$  and  $Q < 1$ . The selected classifier of cascade 7 produces a false negative on the training set. All positive samples sorted out according to the quantile criterion were stored. Figure 8 shows the first 6 quantile objects from a total of 303 objects that were sorted out from the cascade classifier 7 during the training. In each stage approximately three images have been sorted out. These images show all distortions regarding the average luminance being very dark or very bright. The images 8(b) and 8(d) are partially shaded and images 8(e) and 8(f) show a strong blurring effect.

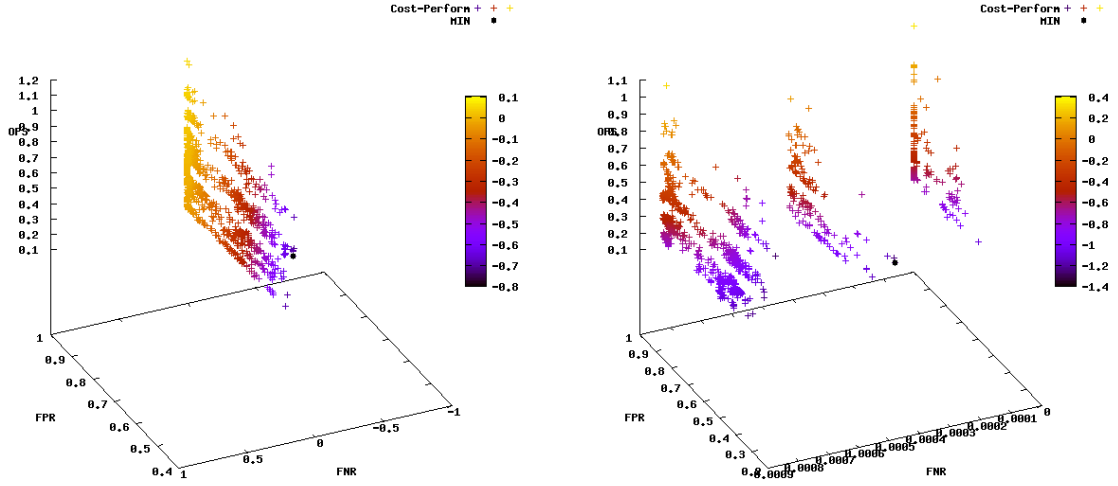
## 8 Results

The cascades already presented in the experiments section have been evaluated on 2981 single images extracted from 279 different real world tracks of US speed limit signs not contained in the training set. The signs also contain 13 different speed limits ranging from 10 to 70 mph, with large variations in illumination and contrast (see figure 8). A sign is considered as detected, if there is at least one detected rectangle that covers 30 percent area of the correct speed limit box. The detection algorithm was scanning the whole image which has the dimension 752x480 pixels while the processing time was measured in milliseconds on a fast 3.2 GHz processor including all steps needed for the complete detection.



**Figure 6:** Cost-performance function of classifiers  $\mathcal{H}$  using logarithmic color coding. Operational cost on  $Y$  axis and false positive rate on  $X$  axis. The black point is the selected base classifier in the first resolution stage.

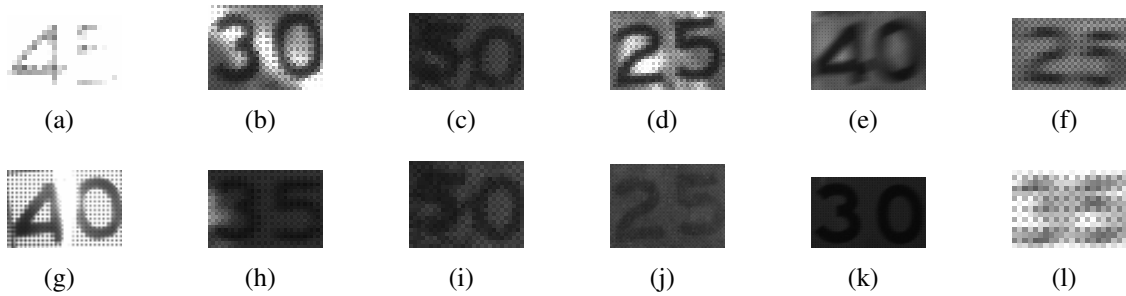
The detection boxes used range from 20 pixel width to 80 pixel width. Table 4 compares the total averaged cost and performance of each cascade with the computed costs during the training. Cascades 6, 8, 9 and 10 all have detection rates over 80% with comparable false positive rates. This indicates that the use of parameter  $g_3 = 0.01\sigma^2$  which was the largest weight for  $\sigma^2$ , works good for the application case. For speed limit detection single images from one object appear in a track. Therefore a detector must not necessarily detect all single images. We evaluated the detection rates dependent on the minimum detected images per track in table 5. This shows on the one hand that there are only few completely missed signs and that we achieve very good track based detection rates. The results show that very effective classifiers can be generated using the presented methods.



(a) Cascade 2, Quantile  $\mathcal{Q} = 1$

(b) Cascade 7, Quantile  $\mathcal{Q} = 0.999$

**Figure 7:** Cost-performance function of classifiers  $\mathcal{H}_{\Delta k \frac{1}{2}}$  using logarithmic color coding. Additional axis with false negative rate. Using  $\alpha = \beta$ , the classifiers are clustered for quantiles with  $\mathcal{Q} < 1$ .



**Figure 8:** Samples sorted out according to the quantile criterion.

The correct adjustment of the base classifiers parameters is crucial for the performance of the cascade in the final detection application. Nevertheless, the differing costs and performances between the validation during training and the final application clearly indicate overtraining.

## 9 Conclusion

In this study we presented a framework for object detection that consists of cascaded perceptron classifiers and applied the Fisher LDA method as a fast and effective learning method, since weights are specified with a statistical analysis of the data. On the other hand we showed how statistical methods can be applied to sort out unusual samples but nevertheless achieve a good generalization performance of the perceptrons. We introduced a quality function for the measurement of classifiers which incorporates the operational costs per classification and the classification performance of the classifier. This function is used as a selection criterion and as a stopping criterion for the selection process. Due to

Id	$TP_R$	$FP/I$	COST(ms)
1	0.634	18.0	19.4
2	0.611	16.2	18.0
3	0.553	3.6	13.5
4	0.697	47.0	18.1
5	0.564	2.6	13.8
6	0.826	18.8	25.0
7	0.566	2.2	13.8
8	0.806	13.0	24.3
9	0.810	14.1	27.4
10	0.844	10.9	19.7
11	0.915	122.3	105.9
12	0.581	1.2	13.2
13	0.323	0.9	7.9
14	0.603	0.5	20.2

**Table 4:** Performance of each cascade applied to whole images, average operational costs in [milliseconds], false positives per image.

Id	50 %	40 %	30 %	20 %
1	0.61	0.70	0.77	0.80
2	0.61	0.69	0.78	0.82
3	0.55	0.63	0.72	0.78
4	0.71	0.79	0.84	0.88
5	0.55	0.64	0.72	0.78
6	0.85	0.89	0.91	0.91
7	0.56	0.66	0.71	0.78
8	0.83	0.89	0.91	0.93
9	0.83	0.86	0.88	0.90
10	0.87	0.92	0.93	0.95
11	0.91	0.94	0.95	0.96
12	0.60	0.67	0.75	0.81
13	0.23	0.33	0.43	0.55
14	0.62	0.70	0.79	0.83

**Table 5:** Detection rate of each cascade dependent on the average detection rate per track.

this quality function our method handles arbitrary features differing in their operational costs. We showed that very effective detectors can be generated when applied to a difficult real world task of US speed limit detection. The experiments showed that a crucial step in building the classifier is the adjustment of parameters due to its cascaded structure.

## References

- [1] Tony Lindeberg: *Scale-Space for Discrete Signals*. In: PAMI, pages 234-254 1990.
- [2] Tony Lindeberg: *Feature Detection with Automatic Scale Selection*. In: International Journal of Computer Vision, pages 79-116. 1998.
- [3] Seung-Jean Kim Alessandro and Alessandro Magnani and Stephen P. Boyd: *Robust Fisher Discriminant Analysis*. In: Advances in Neural Information Processing Systems, pages 659-666, MIT Press. 2006.
- [4] Frank Rosenblatt: *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*. In: Psychological Review, pages 386-408. 1958.
- [5] Fisher, R.A.: *The Statistical Utilization of Multiple Measurements*. In: Annals of Eugenics, pages 376-386. 1938.
- [6] Paul Viola and Michael Jones: *Rapid object detection using a boosted cascade of simple features*. 2001.
- [7] Paul Viola and Michael Jones: *Robust Real-time Object Detection*. In: International Journal of Computer Vision. 2001.

- [8] Yoav Freund and Robert E. Schapire: *A decision-theoretic generalization of on-line learning and an application to boosting*. In: *Computational Learning Theory: Eurocolt 95*, pages 23-37. Springer-Verlag 1995.
- [9] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. *Boosting the margin: a new explanation for the effectiveness of voting methods*. In: *Ann. Stat.*, 26(5):1651-1686. 1998.
- [10] Yoav Freund and Robert E. Schapire: *Large Margin Classification Using the Perceptron Algorithm*. In: *Machine Learning*, 37(3):277-296. 1999.
- [11] Qiang Zhu , Shai Avidan , Mei-Chen Yeh , and Kwang-Ting Cheng: *Fast Human Detection Using a Cascade of Histograms of Oriented Gradients*.
- [12] Navneet Dalal and Bill Triggs: *Histograms of Oriented Gradients for Human Detection*. In: *CVPR*, pages 886-893 2005.
- [13] M. Bertozzi and A. Broggi and M. Del Rose and M. Felisa and A. Rakotomamonjy and F. Suard: *Pedestrian Detection using Infrared images and Histograms of Oriented Gradients*. In: *Intelligent Vehicles Symposium 2006*
- [14] Yin Zhang and Zhi-Hua Zhou: *Cost-Sensitive Face Recognition*. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, VOL. 32, NO. 10. 2010.
- [15] Cameron-Jones, R. Mike and Charman-Williams, Andrew: *Stacking for Misclassification Cost Performance*. In: *Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 215-224, Springer-Verlag. 2001.
- [16] Charles Elkan: *The Foundations of Cost-Sensitive Learning*. In: *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973-978 2001.
- [17] Brendan Mccane, Kevin Novins, Michael Albert and Pack Kaelbling: *Optimizing Cascade Classifiers*.
- [18] Christoph Gustav Keller and Christoph Sprunk and Claus Bahlmann and Jan Giebel and Gregory Baratoff: *Real-Time Recognition of U.S. Speed Signs*. In: *IEEE Intelligent Vehicles Symposium (IV 2008)* 2008.
- [19] Fabien Moutarde, Alexandre Bargeton, Anne Herbin, and Lowik Chanussot: *Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system*. 2007.