Real-time detection and classification of cars in video sequences

Alexander Gepperth, Johann Edelbrunner, Thomas Bücher

Abstract— **We present a system capable of detecting cars in gray-valued videos of traffic scenes based on easy-to-compute orientation selective features derived from gradient filter outputs. The car detection system consists of two processing stages (Initial detection and Confirmation) and is embedded into a comprehensive architecture of interacting modules optimized for various aspects of driver assistance applications. The Initial Detection stage uses a heuristic for generating hypotheses which are then presented to a single neural network (NN) classifier for Confirmation, which is trained on examples in a supervised way. We show that one can achieve approximate scale-invariance in the Confirmation stage by using approximately scale-invariant image features and training with differently sized examples. The NN used for Confirmation are optimized using a simple pruning algorithm. The dependence of detection accuracy and network complexity is investigated; we find that extremely simple networks give surprisingly good classification accuracies at very high speed.**

Index Terms—**Pattern classification, object detection/recognition, multiplayer perceptrons**

I. Introduction

In many applications in driver assistance systems, behaviourally relevant objects in traffic scenes must be reliably detected by image processing in order to generate high-level-representations allowing, e.g., behaviour planning. Most prominent among behaviourally relevant objects are certainly cars and pedestrians. Real-world problems such as, for example, visual pedestrian or car detection are considered to be very tough problems since objects can exhibit a very large amount of variability. Reasons include viewpoint dependency, intrinsic within-class variability, object occlusions and image transformations due to lighting changes or insufficient sensor performance. On the other hand, there are also advantages that can be exploited, namely the adherence to basic physical laws that is guaranteed, which makes sure that relevant objects behave in a predictable way.

For cars and pedestrians there exist numerous proposals for detection and confirmation strategies, many of which address different issues of those mentioned previously. A perfectly accurate and universally reliable object detection system either for pedestrians or cars remains, however, elusive. In the following paragraph, some interesting developments in these domains are reviewed, concentrating on car and pedestrian classification rather than detection since classification is the focus of the work presented here. Given the vast amount of research which is being done in this domain, however, this is not intended to be either a complete or a representative list, but merely a selection of interesting approaches which serves to highlight the intrinsic properties of our own approach in contrast or accordance to those that are mentioned here.

The most basic property that distinguishes different proposals from each other is the choice of features upon which the classification is based. In analogy to human and primate vision, classifications are performed using stereo information [16] as well as monocular image features such as the outline (shape) of objects [14], wavelet coefficients extracted by linear filtering [8],[15] or Haar wavelet decomposition [13], principal component analysis [18] and local orientation coding techniques [9],[19], to name just a few.

There are also systematic approaches to automatically select an optimal subset of image features from a given feature base using either evolutionary algorithms [18] or a procedure called "boosting" [13].

Another source of diversity are the methods employed to reach a classification decision: typical methods are template matching [15], neural networks of various architectures [17], [19] and support vector machines [18].

Driver assistance systems typically operate under real-time constraints, whereas high accuracy in object detection is nevertheless crucial. We propose a method to reliably and efficiently detect cars within the driver assistance framework developed at our research group. The framework consists of a number of independent but interacting modules each of which performs a specialized analysis task on a video sequence that is common input to all modules. Our classification approach is based on local orientation coding derived from local edge information; classification is performed on monocular, gray-valued video sequences. The feature base was designed, i.e. not constructed using an automated procedure as mentioned previously, and the classification is done by an artificial neural network which learns from examples in a supervised way.

An Initial Detection module for vehicles was already in existence within the general framework; goal of the work presented here is to show how hypotheses produced by the Initial Detection module ("regions of interest" – ROI) can be evaluated (i.e. classified) by a Confirmation module.

Manuscript received December 15th, 2004.
Alexander Gepperth and Johann Edelbrunner are with the Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany. E-mail: {Johann.Edelbrunner, Alexander.Gepperth}@neuroinformatik.rub.de
Thomas Bücher is with Viisage Technology AG, Universitätsstraße 160, 44801 Bochum, E-mail: Thomas.Buecher@neuroinformatik.rub.de

II. System Architecture

Advanced Driver Assistance applications are either safety oriented, such as lane departure warning, lane change warning and pedestrian detection, or comfort oriented, such as e.g. traffic sign detection. Among others, we have implemented specialized modules which carry out the aforementioned functions. Common input to all modules are pre-processed data obtained from the input video image; by using a common feature basis, we ensure that the image need only be pre-processed once per frame, thereby facilitating real-time application.

To make the car detection fast enough for real-time applications, the process of finding cars in image sequences is hierarchically organized. The different modules used for car detection are Initial Detection (different algorithms provide vehicle hypotheses, i.e. ROI), Confirmation (scale invariant evaluation of detected ROI) and Tracking (finds a given ROI in the following video frames). For a detailed description of the Tracking module, see [3], [10] and references therein. Obviously, the results of the different modules (except the feature extraction) are not independent and therefore a temporal coupling structure is implemented to increase the reliability of the car detection results. All specialized modules described use features calculated in the pre-processing. In spite of the complexity of the task, the Initial Detection guarantees fast data reduction.

The ROI generated by the Initial Detection module are presented to the Confirmation module which generates a confidence measure for each ROI indicating how likely it judges the ROI to contain a car. The number of initial hypotheses is thereby reduced depending on the "strictness" (which effectively expresses the minimum confidence that will still be interpreted as a "car present" decision) of the Confirmation module. This property is governed by a single threshold value which must be set according to the desired results. The remaining ROI are tracked; any incorrect hypotheses must be eliminated by heuristics governing the interaction of the Initial Detection, Confirmation and Tracking modules: The Tracking module output is continuously compared to hypotheses generated by the Initial Detection modules. A confidence value is maintained for each tracked ROI: the confidence is incremented if the tracked object coincides with a confirmed hypothesis, and decremented if it does not. When the confidence decays below a certain threshold, tracking is turned off for this particular ROI. It is then assumed that the hypothesis has become invalid or has been invalid all along. Thus, incorrectly confirmed hypotheses can be eliminated effectively. This heuristic takes several frames to discard incorrect hypotheses, besides the fact that tracking is a computationally expensive procedure. Therefore, it can make sense to use a more powerful classification function to save effort later.

To summarize, the use of a Confirmation module yields the advantage of increased speed due to fewer false detections on the one hand, and on the other hand it provides an effective way to obtain an independent quality measure on tracked ROI, which can be used to calculate more accurate confidence values for each tracked ROI.

III. Image analysis and feature selection

A. Preprocessing

The extraction of meaningful features is an important issue for any image processing algorithm. In contrast to the first driver assistance applications on the market (e.g. lane-departure-warning systems) where the preprocessing stage was directly linked to the application specific processing algorithms, the car detection and classification algorithms described in this paper are implemented in modules which run in the context of a whole driver assistance application. For such system architectures we propose to calculate a high-level feature basis which can be accessed by all image processing algorithms. This approach has a number of advantages: First of all, the development of robust task-specific algorithms is simplified significantly; secondly, assuming that the feature basis has certain invariance properties e.g. with respect to varying lighting conditions, such properties will be automatically incorporated in the task specific algorithms. And thirdly, depending on the number of modules, even the whole processing time can be reduced due to simplified processing in additional task-specific image processing algorithms. Furthermore the calculation of the feature basis can be implemented on dedicated hardware, so that a general purpose processing core can be used for the succeeding algorithms.

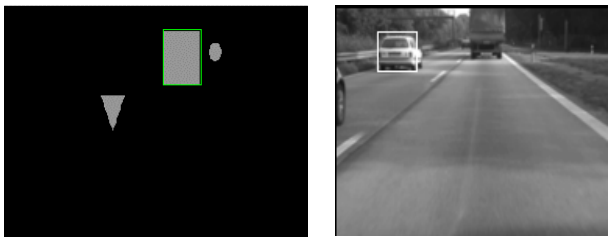
We chose to extract horizontal gradients, vertical gradients, energy of gradients, contour points including a quantized local orientation and line segments including the mean energy along each segment. The features up to the contour points are represented in terms of images and are calculated by the well known Canny edge detector [5]. For efficiency reasons, we use separable filters resulting in two 1D-convolutions for each gradient direction. It turned out that using a smoothing kernel of size 3 and gradient kernel of size 5 leads to good results.. For efficient access of sparse contour points, a linked data structure is used which is established during the non-maximum suppression algorithm of the Canny-Filter. The line-segments are obtained by a clustering algorithm that makes use of this linked representation; the involved calculations are not presented here as they are beyond the scope of this paper (but see [2]). Each line-segment is represented by the image coordinates of its end-points and the mean gradient energy along that line and therefore provides an extremely sparse coding of an image contour. For typical road scenarios the average number of line-segments obtained varies between 250 and 500 (image size: 496x256, minimal line-length: 5 pixels).

B. Features for Classification

1) Histograms

To allow for a real time decision process, the data provided by the pre-processing must be processed further, while still providing enough information for reliable classification. The outcome of this process shall be termed *feature set* in what follows. A sub-optimal choice of extraction procedure can significantly reduce classification performance; because of this, some effort was made to identify suitable features.

An upper boundary on the complexity of the feature extraction is set by the required calculation time, since we are inter-



Picture 1: Example of synthetic and real-world video sequences used for the testing of invariances. Typical objects that were used for feature extraction are in boxes.

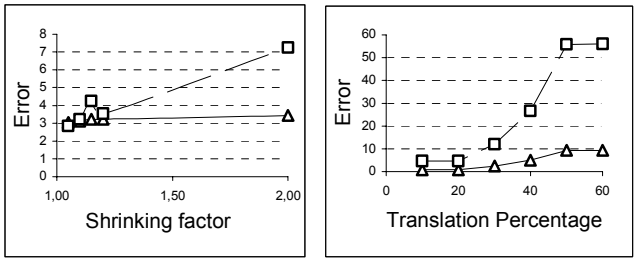


Fig. 1. Errors introduced into a feature set by scaling (left) and translation (right). Measurement was done on synthetic images Shown is the binary deviation (triangles) and the corrected MSE (squares). Both measures are explained in the text.

ested in real-time decisions. This excludes common methods in image processing like Gabor or Fourier transforms, which need not be a disadvantage provided it is possible to identify cars using only "primitive" features. We are not assuming this is true for more general object recognition applications.

In order to make our object recognition invariant with respect to scaling, we demand that this invariance is already incorporated into the generation of the feature sets. In the course our investigations, we identified an extraction method which is equally favourable in terms of processing time as well as suitability for classification. We will call this method the "**S**et of **O**rientation **E**nergies" or SOE method. It will be described in detail further below.

First of all, the ROI is subdivided into a fixed number of rectangular regions which we term *receptive fields* in analogy to biological image processing. From each receptive field a set of numbers is extracted by the chosen feature extraction scheme. The concatenated set of numbers extracted from all receptive fields in a ROI represents the feature set corresponding to that ROI. The point is that the number of receptive fields does not depend on ROI size: the larger the ROI, the larger the receptive fields. This incorporates already at a fundamental level the requirement of scale invariance: the feature set does not reveal the size of the ROI it was calculated from.

It is evident that a finer subdivision of a ROI gives higher spatial resolution and therefore more precise spatial information; on the other hand, it leads to greater processing costs and larger feature sets, which in turn slow down the classification process. We determined the optimal subdivision by increasing the number of receptive fields from 1 until classification results stopped improving.

The SOE method simply computes the sum of energies of all identically oriented edges in a receptive field and normalizes this sum by the total energy of edges within the receptive field. Orientations are quantized, that is, ranges of angles are mapped onto integers, thus effectively reducing the number of possible orientations. The output of the algorithm is a number for each orientation, indicating the ratio between oriented energies of that particular orientation and the total (= summed) energy of all orientations, and is therefore a number between 0 and 1. By analyzing the problem class and doing some experiments, we verified that four orientations are sufficient; the algorithm therefore produces 4 numbers per receptive field.

2) Invariance properties

To assess the invariance properties of SOE, we compute features from ROI in natural as well as artificial video sequences and from respective transformed versions, using a 7x7 subdivision of the ROI. The transformations are scaling (of ROI and image) and translation (of the ROI). Results are measured on ROI containing vehicles/objects in two video sequences. One sequence is synthetic, the other recorded on a highway. Results are averaged over all frames of each sequence.

For the investigation of translation invariance, ROI are shifted to the left and to the right by fixed percentages of their width. Without loss of generality, we consider only left/right shifts since by construction, ROI always have marked lower edges in the image. For testing scale invariance, we downsample and smooth the image by appropriate linear filters ([6]) and reduce ROI dimensions by the same factor.

We define two error measures describing the deviation between a feature set and its transform. One is essentially the mean squared error (MSE) normalized by the average of the original feature set, but correcting for the fact that the 4 numbers that are generated per receptive field are not independent (they must add up to 1.0). Therefore, the MSE is halved; we call this method *corrected MSE*. The other measure (we named it *binary deviation*) takes the relative ordering of the orientations into account. First, it transforms the feature sets to be compared into *binary feature sets* by substituting a value of 1.0 if an element is among the two (absolutely) strongest within a RF, and 0 if it is not. The MSE between the binary feature sets is given back as an error measure, producing a number between 0.0 and 1.0.

The results on synthetic images (a sample of which is shown in Picture 1) are shown in Fig. 1. Note that the method is almost invariant to translation until 20% of ROI width. The reason is that, with a 7x7 receptive field configuration, one receptive field corresponds to 28% of ROI width. As long as critical features stay within the same receptive fields, no changes can occur. Notable is furthermore the low error introduced by scaling.

Results on real-world images (Fig. 2) are poorer, as might be expected since translation introduces new and unpredictable content into an ROI instead of empty background in synthetic images. At first glance the method seems to perform poorly under scaling in particular, but at second glance one can perceive that the structure of a feature set remains quite unaffected. This becomes apparent when considering binary feature sets which contain 4 numbers per RF: a one if the angle index (running from 0 to 3) is among the two strongest in the RF or a zero if it is not. These binary feature vectors exhibit a deviation of only about 15 under 50% scaling, and this is, in our opinion, the reason why classification performance is quite independent of ROI size (see later sections, especially Fig. 4).

IV. Generation of initial hypotheses

The generation of vehicle hypotheses is the first processing stage within an architecture for car detection. In this section, the algorithms generating the ROI presented to the Confirmation module are briefly depicted.

We developed two different algorithms for generating initial vehicle hypotheses, which are based on different image features. One algorithm employs the line-segments calculated in the preprocessing stage for producing a list of potential vehicle positions: The middle of each approximately horizontal line-segment serves as a starting position for searching lateral vehicle boundaries. Based on the location of the line-segment in the image, the image region occupied by a vehicle at that location is estimated. In this region a one-dimensional signal is calculated by vertically projecting horizontal gradient information. Extraction of local maxima, analysis of maxima positions and signal values finally leads to either acceptance or rejection of that region. The other strategy for calculating potential vehicle positions utilizes estimates of lane borders provided by another module and is therefore based on higher-level knowledge. It evaluates a robust measure of mean gray-value in each row (up to a maximal predefined distance from the ego-vehicle) for each lane, resulting in hypotheses for vertical vehicle positions for each detected lane. In order to suppress false detections due to horizontal shadows, the same mechanism for evaluating the presence of horizontal vehicle boundaries as in the algorithm outlined first is employed.

V. 5 Requirements on a confirmation module

For the implementation of the Confirmation module, we demand the following properties:

--adaptivity: can be trained by examples

--flexibility: able to generalize when dealing with previously unseen data

--good performance: capable of real-time decisions

--robustness: invariant to scale, translation and a wide range of lighting conditions.

--independency: should not rely on other modules

In the light of the explanations given so far, it is evident that the last three constraints are fulfilled by construction: the performance constraint is taken care of by the way input features for the Confirmation module are generated, provided only that the confirmation itself can be implemented efficiently (which shall be shown later). The robustness constraint is satisfied on the one hand by using the results of the pre-processing stage which already exhibit a great deal of invariance properties, and on the other hand by the invariance properties of the feature sets extracted from each ROI which serve as inputs to the Confirmation module. Lastly, the independence property is ensured by the fact that only the input feature sets determine the decision of the Confirmation module.

The first two constraints are satisfied by our implementation of the classification function within the Confirmation module which will be described in the next section.

VI. The Confirmation module

The previously stated requirements on the Confirmation module do not specify a particular implementation, and indeed several alternatives present themselves, most prominently mul-

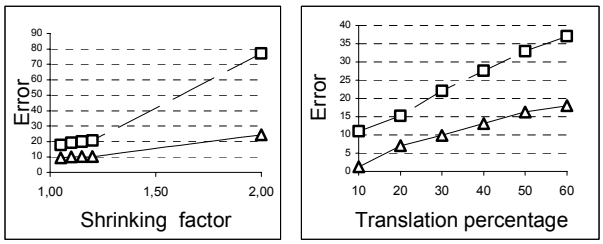


Fig. 2. Errors introduced into a feature set by scaling (left) and translation (right). Measurement was done on real-world images. Shown is the binary deviation (triangles) and the corrected MSE (squares).

tilayer perceptrons (MLP) and support vector machines (SVM). For now, we will focus on issues that are independent of this particular choice.

A. Training data

The Confirmation module is required to decide whether a ROI that is presented contains a car or not, therefore distinguishing two classes of objects (cars and not-cars) from each other. Training examples consisting of a ROI and a class label (0.0 or 1.0) were generated from a significant number of different videos of typical highway scenes. We generated four disjunct datasets termed D_{train}, D_{val}, D_{test} and D_{extern}, all containing 5000 examples, which are randomly taken from a larger database of over 100000 examples. 50% of the examples in each set are positive examples. The positive examples were labelled manually, the negative examples were produced using the Initial Detection module: for each frame of a video sequence, its output (a collection of ROI) was compared to the previously labelled positive examples. All ROI that did not coincide (within a certain tolerance range) were added to the database as negative examples. In this way, it is ensured that the negative examples used for training are comparable to those encountered during application. Furthermore, if the Initial Detection module should be altered, negative examples can simply be generated automatically by using the new Initial Detection module.

The positive examples must fulfill a number of specifications. Positive examples must have a certain width/height ratio (close to 1 for cars), contain only cars and no trucks; they must

include the pronounced lower edge of cars, enclose all of the rear view of a car tightly and vary over all scales. The last condition is a precaution: although the feature sets are roughly invariant to scaling, it is prudent to account for small deviations by including training examples of all sizes. For negative examples, all conditions are fulfilled by construction.

All our training data (the video sequences as well as the specification of the training examples) are available on our group's homepage (www.neuroinformatik.rub.de/group/Mesa), as well as the necessary software to manage the training data.

B. Decision making

We expect the output of the Confirmation module to be a continuum of values between the labels for the two classes. In order to interpret this as a decision in favour of a particular class, a threshold is defined, and the decision for a particular class is then expressed by a module output that exceeds or does not exceed the chosen threshold.

The underlying assumption that justifies this simple method is that the degree of match between objects and learned models is at least approximately encoded by the euclidean distance between the module output and the corresponding class label. If this assumption holds, we expect the module outputs to be strongly centered around the class labels (in this case we obtain a strongly bimodal distribution), and an optimal threshold can be applied that separates the outputs with minimal error. This assumption needs to be checked explicitely.

C. Implementation of the classification function

For the implementation of the classifier, a MLP is used which converges onto a single output neuron. We make this choice because we expect better performance than from SVM while obtaining comparable classification accuracies. We use one hidden layer, and all activation functions are of logistic sigmoidal type. For training, we use a modified form of the Rprop learning algorithm [4]. The layers are fully connected, and all neurons in the input layer are origins of shortcuts (connections that bypass one or more layers) to the output neuron.

Initially, we generate a population of NN with different parameter values and topologies. Each member of the population is subjected to an iterated optimization loop, a step of which consists of NN training and subsequent magnitude-based pruning [7]. This simple pruning heuristic was chosen because initial attempts using a sensitivity-based method yielded poorer results at higher computational cost and were therefore abandoned. For a review of pruning techniques refer to [7]. The goal of this procedure is to obtain a NN that has as few connections as possible (because execution speed scales linearly with the number of connections in our implementation, see Fig. 3) while still being capable of the best possible classification. Motivated by general statements about the learning capacity of MLP [7], the unoptimized NN have about 5000 connections. Since it is a difficult issue to show analytically which number and size of hidden layers is optimal for a specific problem [1], [7], we make no attempt to tackle it but leave this to an improved optimization technique. For this purpose, evolutionary NN optimization methods seem the methods of choice to us. For a comparision of magnitude-based pruning to

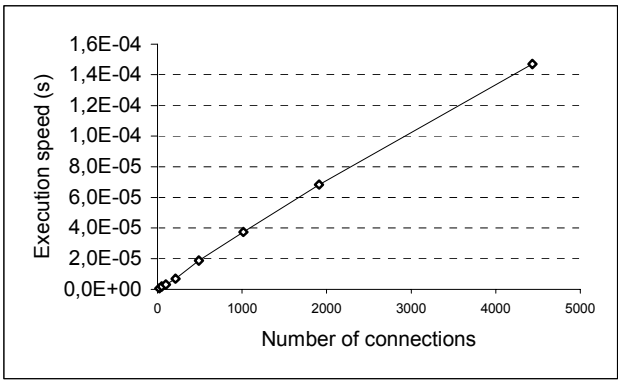


Fig. 3. Dependence of execution speed on NN complexity

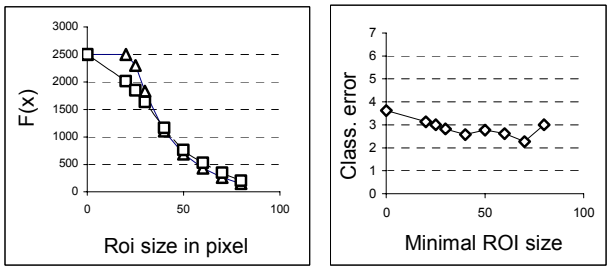


Fig. 4. Dependence of classification accuracy on ROI size. Left: size distribution of examples in D_{test}. Car ROI are depicted by squares, non-car ROI by triangles. F(x) denotes the cumulative distribution function, i.e. the number of ROI wider than x. Right: classification accuracy plotted against the lower bound on ROI width

an evolutionary NN optimization method combining the abilities to reduce and increase the size and complexity of the NN, see [20].

Inputs to the net are the feature sets that are generated in the previous processing step of feature extraction, so the input layer must have precisely that dimensionality.

For the implementation of the NN we use the ReClaM package of the freely available SHARK library (http://shark-project.sourceforge.net).

D. NN Training

For training, we use the MSE as an error measure. As usual, weights are initialized to random small values between –0.05 and 0.05 with each NN using a different seed value for the random number generator. MSE on D_{train} is minimized for 100 epochs, whereupon the net with the best MSE on D_{val} is selected as training result. Convergence is very fast; in general, no more than 10 epochs are needed until training achieved an overall error smaller than 10% on D_{val}.

E. Optimization

We set the size of the initial population of NN to 250. The size of the hidden layer is chosen to lie between 20 and 25. After every training step, we apply magnitude-based pruning to all NN. One pruning step consists of the elimination of a percentage of weights; we eliminate those 10% of weights that have the smallest absolute value.

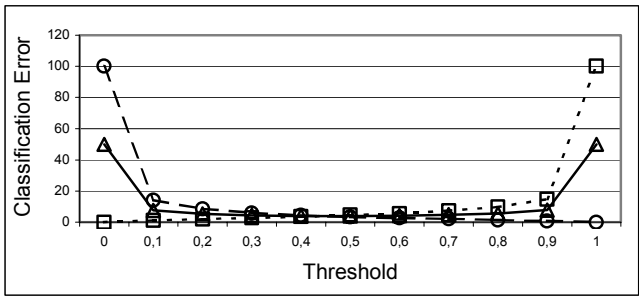


Fig. 5. Error measures depending on applied decision threshold. False positives re depicted by circles, false negatives by squares. The overall classification error is indicated by triangles.

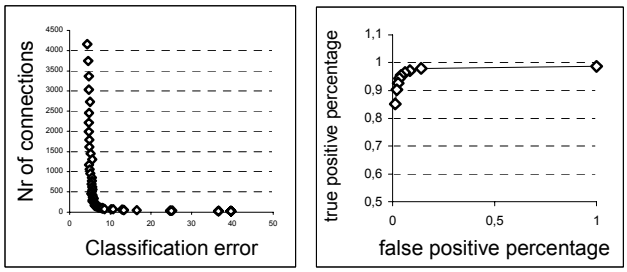


Fig. 6. Left: Typical optimisation results. Right: Receiver-operator characteristic of the best optimized NN

All evaluations of optimization results are performed using the classification error (CE) on D_{test}. The result of the optimisation is a statement about the network capacity needed for this particular problem class: it turns out that the number of connections can be reduced by approximately 55% while retaining optimal CE, i.e. comparable to those obtained by repeatedly training the unoptimized NN with different initializations and choosing the best one as reference. At 10% of the original connections, NN are still capable of an overall CE of about 95%. But, surprisingly, even NN with fewer than 2% of the original connections are able to produce CE of about 90%.

When talking about results, we clarify some terms first since they are not used coherently in the literature: We denote as *false positive/negative rate* the percentage of positive/negative examples that are classified wrongly. Analogously, the *true positive/negative rate* denotes the percentage of correctly classified positive or negative examples.

Fig. 4 (right) supports an assertion we made earlier: the approximate independence of the CE on ROI size. What one can furthermore perceive is that CE actually falls below 3% when excluding ROI smaller than 30 pixels. As shown by Fig. 4 (left), such ROI are quite common and their influence on the classification error is therefore notable, whereas they typically do not contain cars but artefacts produced by the Initial Detection. Fig. 5 shows the dependence of the classification error as well as the false positive and false negative rates as a function of the applied decision threshold. As can be perceived from the figures, the best overall classification error is 3,8%. Of course it can make sense to accept a higher overall error rate in order to minimize the false negative or false positive rates, depending on the demands of an application. Fig. 6 (left) shows representative optimization results. Notable is the clear trade-off between NN complexity and classification accuracy.

Fig. 6 (right) shows the receiver-operator characteristic (ROC) [11] of the best optimized NN.

VII. Performance, benefits

On a Pentium II PC with 1 Ghz under Windows NT, using the MS Visual C 6.0 compiler, the largest NN from the optimisation runs takes about 0,15 ms for one classification. The smallest optimized net that gives overall classification errors of under 6% takes 0,022 ms per classification. This results from the fact that the net has only about 10% the number weights compared to the unoptimized NN, with a corresponding order-of-magnitude increase in speed. The smallest NN that yields an overall classification error of under 10% needs 10^{-4} ms for a classification. These extremely fast classification times make the optimized NN applicable for brute-force search methods that scan the whole image at multiple scales to reliably detect all objects of interest. It should be noted, however, that we do not expect this result to be extensible to other objects than cars since our investigation suggests that the problem class is easily separable.

Together with the time needed for feature extraction, which is (averaged over all ROI in the training sets) 0.3 ms, the object detection module takes between 0,3 ms and 0,45 ms for one operation, depending on the complexity of the classification NN. Another point is that the optimisation procedure generates a sequence of classifiers that can be applied successively to candidate ROI, thus generating hypotheses of increasing reliability at negligible computational cost, since the feature set needs to be calculated only once. Classifiers of differing accuracy and complexity can be selected to enhance different modules of the system according to speed and accuracy requirements.

VIII. Conclusion

We have shown that car classification is a task that can be solved to very good accuracy and at high speeds by NN classifiers. From the front of trade-off solution we obtained, NN can be selected according to different demands on speed and classification accuracy. Extremal NN containing only a few dozen connections are still able to give reasonable accuracy at negligible computational cost. These findings make it plain that car detection and classification can easily operate even under real-time constraints. In the face of the enormous variety of possible traffic situations, it is, however, not to be expected that the approach described here can lead to perfect detection accuracy under all circumstances. Therefore, further studies will be devoted to the issue of how the performance of the whole driver assistance framework can be improved by the enhanced object detection, and how to make object detection more stable by considering additional information provided by other modules and information sources.

IX. References

[1] E Baum and D Haussler. What size network gives valid generalizations? *Neural Computation*, 1:151-160, 1989.

[2] T Bücher, C Curio, H Edelbrunner, C Igel, D Kastrup, I Leefken, G Lorenz, A Steinhage and W von Seelen. Image processing and behaviour planning for intelligent vehicles. *IEEE Transactions on industrial electronics*, 90(1):62-75, 2003.

[3] G.A. Klanderman D.P. Huttenlocher and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850-863, 1993.

[4] Christian Igel and Michael Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105-123, 2003.

[5] J.F.Canny. A computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679-698, November 1986.

[6] B. Jähne. *Digital Image Processing - Concepts, Algorithms, and Scientific Applications, 3rd Edition*. Springer Verlag, Berlin, 3rd edition, 1995.

[7] Russel D. Reed and Robert J. Marks. *Neural Smithing – Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, Cambridge, Massachusetts, 1999.

[8] H Schneiderman and T Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, 2000.

[9] A Shashua, Y Gdalyahu and G Hayun. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *IEEE Intelligent vehicle symposium (IV2004)*, 2004.

[10] M Werner. *Objektverfolgung und Objekterkennung mittels der partiellen Hausdorff-Distanz*. Number 574 in Reihe 10. Fortschritt-Berichte VDI, 1999.

[11] XH Zhou, NA Obuchowski, and DK McClish. *Statistical methods in diagnostic medicine*, pp. 15-164. John Wiley & Sons, 1 edition, 2002.

[12] P Viola, MJ Jones and D Snow. Detecting Pedestrians Using Patterns of Motion and Appearance. *Proceedings of the Ninth IEEE International Conference on Computer Vision (ICCV 2003),* 2003.

[13] P Viola and MJ Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2001)*, 2001

[14] D Gavrila and V Philomin. Real-time object detection for smart vehicles. In *Proceedings of the ICCV 1999*, 87-93, 1999.

[15] M Oren, C Papageorgiou, P Sinha, E Osuna, and T Poggio. Pedestrian detection using wavelet templates. In *Proc. Computer Vision and Pattern Recognition*, pp. 193-199, Puerto Rico, June 16-20 1997.

[16] U Franke and I Kutzbach. Fast Stereo Based Object Detection for Stop&Go Traffic. *Proc. Intelligent Vehicles Conf. '96*, pp. 339-344, 1996.

[17] C Wöhler and JK Anlauf. Real-time object recognition on image sequences with the adaptable time delay neural network algorithm - applications for autonomous vehicles. *Image and Vision Computing*, vol. 19, no. 9-10, pp. 593-618, 2001.

[18] Z Sun, G Bebis and R Miller. Object detection using feature subset selection. *Pattern Recognition* 37, pp. 2165 –2176, 2004.

[19] C Goerick, D Noll and M Werner. Artificial neural networks in real-time car detection and tracking applications, *Pattern Recognition Lett.*17, pp.335 –343, 1996.

[20] A Gepperth and S Roth. Applications of multi-objective structure optimization. To appear in *Proceedings of the 13th European Symposium on Artificial Neural Networks*, Bruges, April 27-29, 2005.