Evolutionary Optimization of Wavelet Feature Sets for Real-Time Pedestrian Classification

Jan Salmen, Thorsten Suttorp, Johann Edelbrunner, Christian Igel
Institut für Neuroinformatik
Ruhr-Universität Bochum
44780 Bochum, Germany
{jan.salmen, thorsten.suttorp, hannes.edelbrunner, christian.igel}@neuroinformatik.rub.de

Abstract

Computer vision for object detection often relies on complex classifiers and large feature sets to achieve high detection rates. But when real-time constraints have to be met, for example in driver assistance systems, fast classifiers are required. Here we consider the design of a computationally efficient system for pedestrian detection. We propose an evolutionary algorithm for the optimization of a small set of wavelet features, which can be computed very efficiently. These features serve as input to a linear classifier. The classification performance of the optimized system is on par with recently published results obtained with support vector machines on large feature sets, while the computational time is lower by orders of magnitude.

1 Introduction

Pedestrian detection in images from car-mounted cameras is a highly relevant and difficult vision-based pattern recognition problem. The large number of traffic accidents involving foot passengers reveals the need for driver assistance systems that help to identify potentially dangerous situations that may lead to collisions with pedestrians. The difficulties for pattern recognition arise from partial occlusion, highly variable size, shape, and movement patterns of pedestrians, cluttered backgrounds, and the strict real-time requirements. Here, we focus on the last aspect and consider the design of computationally efficient, but still accurate classifiers for pedestrian detection.

Most preceding work in pedestrian detection has been done using visual gray scale cameras. Usually a stepwise procedure is implemented. First an initial segmentation is done on the whole image to generate a list of regions of interest (ROIs). These ROIs are then classified based on more complex features. Additionally, temporal integration and tracking can be employed to improve the classification results.

The initial segmentation can be done based on range information [15], optical flow [12] or contour features [4], just to name some of the approaches. For classification, often neural networks [15] or support vector machines [11, 10] are used. A shape-based method for classification is applied in [1]. In [2] a hybrid approach for pedestrian detection is presented, which evaluates the leg-motion and tracks the upper part of the walker.

In this study, our goal is to speed up the classification step. We propose evolutionary optimization of a set of wavelet features, which serves as input to a linear classifier. In order to draw a comparison between our algorithm and alternative methods in a straight-forward manner, we determine the performance using the pedestrian classification database provided by [10]. In that article, the authors evaluate different combinations of feature calculation techniques and classification algorithms in a benchmark scenario.

This article is organized as follows. In the next section, we introduce the features and the classifier used. In Section 3, the evolutionary optimization of feature sets is presented. Our experiments are described in Section 4, and finally the results are discussed.

2 Pedestrian Classification

2.1 Wavelet Features

Wavelet features are very popular for object recognition, for example in face detection [14] or pedestrian detection [11]. In the following, we present the type of wavelet features, often referred to as *Haar-like features*, that is used in this work.

2.1.1 Feature calculation.

Given a gray scale image of arbitrary size (usually defined by an ROI in a camera image), we refer to its upper left point as $(0, 0)$ and to its lower right point as $(1, 1)$. A single wavelet feature is defined by either two, three or four sub-rectangles, which are specified by their upper left and lower right corners. As the whole feature representation does not depend on absolute values, features can be calculated for images with arbitrary sizes and aspect ratios.

The sub-rectangles mentioned above are grouped into *white* and *black* ones. For a given gray scale image and a given wavelet feature, let s_B be the sum of all gray values from pixels covered by a black region and s_W be the sum for pixels covered by a white region. The key idea of the wavelet features is to consider the difference $s_B - s_W$. Let A_B and A_W denote the total number of pixels covered by black and white regions, respectively. Then, if $A_B \neq A_W$, it is reasonable to normalize s_B and s_W before calculating the feature response, and we define $r = \frac{s_B}{A_B} - \frac{s_W}{A_W}$. Thus, the output r of a single feature is a value in the range $[-g_{\max}, g_{\max}]$, where $g_{\max}$ is the maximum gray value in the input image.

The output of each feature is transformed by a non-linear function, which is defined by two thresholds x_t and x_s. For a given feature response r, either the activation function

$$f_1(r, x_t, x_s) = \begin{cases} -g_{\max} & r \leq -x_s, \\ (r + x_t) \cdot \frac{g_{\max}}{x_t - x_s} & -x_s < r \leq -x_t, \\ 0 & -x_t < r < x_t, \\ (r - x_t) \cdot \frac{g_{\max}}{x_t - x_s} & x_t \leq r < x_s, \\ g_{\max} & x_s \leq r \end{cases}$$

or its absolute value $f_2(r, x_t, x_s) = |f_1(r, x_t, x_s)|$ is calculated. One example of a resulting activation function is shown in Fig. 1.

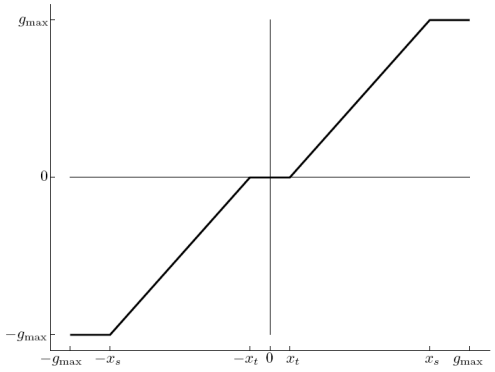


Figure 1. Non-linear function transforming the feature response.

Typically, for a single input image a set of some hundreds of features is calculated (e.g., see [10]). This can efficiently be realized by first precomputing a so called *integral image* [14]. Afterwards, the calculation of each feature requires only some look-ups in the integral image. For the basic types defined here, only six (basic types 1 and 2 from Fig. 2) up to ten (basic types 9 and 10) look-ups are needed.

2.1.2 Initial feature representation.

We initially distinguish ten different basic feature types. One basic feature F is characterized by its type $t^F \in \{1, 2, \ldots, 10\}$, upper left corner $c^F \in [0, 1]^2$, width $w^F \in [0, 1]$ and height $h^F \in [0, 1]$, thresholds $x_t^F, x_s^F \in [0, g_{\max}]$, and an indicator parameter $a^F \in \{1, 2\}$ specifying whether f_1 or f_2 is used.

The coordinates of c^F are relative to the given image and the width and height of the feature are relative to the width and height of the image. Figure 2 shows some examples of basic features.

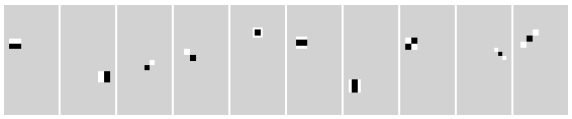


Figure 2. Examples of basic features. Left to right: types 1 to 10 with different parameter settings.

2.2 Linear Classification Based on Linear Discriminance Analysis

Linear discriminant analysis (LDA) is based on a maximum a posteriori estimate of the class membership under the assumption that the class conditional densities are multivariate Gaussians having a common covariance matrix. Despite its simplicity, LDA gives surprisingly good results in practice, of course crucially depending on the representation of the input patterns. For a detailed description, we refer to the literature [6].

2.3 Ensemble Classifier

To improve classification results, one common procedure is to combine a set of several classifiers. We combine linear classifiers, which are based on feature sets from independent evolution trials, to an ensemble classifier. The ensemble response is calculated as the mean of all normalized single responses.

3 Evolutionary Optimization of the Feature Set

Overcomplete dictionaries of Haar wavelets are popular features for classification. In order to speed up classification without loosing performance, a lot of different techniques for feature selection are applicable [9]. Evolutionary algorithms have frequently been considered for feature set generation. We only name a few of them using similar techniques as presented in this study: A set of filter masks and rules, how to apply them for classification, are evolved in [5]. In [13], evolutionary optimization of features is integrated into the AdaBoost framework and improves the final classification performance.

We propose to use an evolutionary algorithm to generate a set of wavelet features specialized for both the classification problem at hand and the classification technique used. Each individual represents an a priori fixed number of n_{feat} wavelet features serving as inputs to a linear classifier. Generating an offspring from a given number of parents is done by cross-over, where the same amount of features is randomly chosen from each parent to form a new feature set. This offspring is mutated as described below. Selection for survival uses EP-Tournament selection [3].

To calculate a fitness value Φ for a given feature set, we use n-fold cross-validation. The training data is partitioned into n disjoint subsets. For each of the subsets, the classifier is trained using the union of the $n-1$ other sets and a test error is computed on the left-out subset. The final cross-validation error e is the average of the n test errors. For convenience purposes, we maximize the fitness $\Phi = 1 - e$.

In the following we describe the mutation of individuals in detail.

3.1 Mutation Operators

Mutating an individual relies on the successive application of m basic mutation operators. The variable m is determined in each generation $t \geq 0$ anew according to the Poisson distribution with mean $\lambda = n_{\text{feat}} \cdot 10^{-1-\frac{t}{2000}}$ where n_{feat} is the number of features per set. We provide a set of 13 mutation operators and adopt their probabilities during the evolution (see below). For each mutation operator, one feature in the set is chosen uniformly at random.

The basic mutation operators are:

- *Shift:* The x and y coordinates of all corner points of all sub-rectangles are shifted by the same amount $\Delta_x \sim \mathcal{N}(0, 0.01)$ and $\Delta_y \sim \mathcal{N}(0, 0.01)$, where $\mathcal{N}(0, 0.01)$ denotes a normal distribution with zero mean and variance 0.01.
- *Increase size:* The size of the whole feature is scaled by factor 1.2 while its center is fixed.
- *Decrease size:* The size of the whole feature is decreased by factor 1.2 while its center is fixed.
- *Flip a^F:* The indicator parameter a^F is switched.
- *Increase x_t^F:* The threshold x_t^F is increased by a factor of 1.2.
- *Decrease x_t^F:* The threshold x_t^F is decreased by a factor of 1.2.
- *Increase x_s^F:* The saturation threshold x_s^F is increased by a factor of 1.2.
- *Decrease x_s^F:* The saturation threshold x_s^F is decreased by a factor of 1.2.
- *Mutate individual:* The center, width and height of all sub-rectangles are independently mutated by adding normally distributed variables with mean 0. Thus, after applying this mutation operator, the feature need not necessary belong to one of the basic types.
- *New initialization "small":* The feature is replaced by a new randomly created basic feature as described in Sec. 4.1.1. Its width is randomly chosen $w^F \sim \mathcal{N}(0.25, 2.5)$.
- *New initialization "medium":* Same as above, but $w^F \sim \mathcal{N}(0.5, 2.5)$ instead.
- *New initialization "large":* Same as above, but $w^F \sim \mathcal{N}(0.75, 2.5)$.
- *No operation:* The feature is not changed at all.

After mutation, *repair operators* assure that all values remain in their valid domains. Figure 3 illustrates the effect of some mutation operators.

Figure 3. Mutating a feature: original, after "Shift", after "Increase size" and after "Mutate individual" (left to right).

3.2 Adaption of Mutation Operator Probabilities

The search strategy is mainly determined by the variation operators and the probabilities of their application, the so called *operator probabilities*. Because the optimal operator probabilities depend on the problem at hand and may change during the course of evolution, we adapt them automatically using the algorithm proposed in [7, 8]. The adaptation mechanism implements the rule of thumb that

variation operators that led to comparatively large fitness improvements recently will also be beneficial in following generations.

Let Ω denote the set of variation operators and $P_o^{(t)}$ the probability that $o \in \Omega$ is chosen at generation t. Further, let $O_o^{(t)}$ contain all offspring produced in generation t by application of operator o. The case when an offspring is produced by applying more than one operator is treated as if the offspring has been generated several times, once by each of the operators involved. The operator probabilities are updated every τ generations. The average performance achieved by operator o in the last τ generations is measured by

$$q_o^{(t,\tau)} = \frac{\sum_{i=0}^{\tau-1} \sum_{g \in O_o^{(t-i)}} \max\left\{0, \Phi(g) - \max_{g' \in \mathrm{parents}(g)} \Phi(g')\right\}}{\sum_{i=0}^{\tau-1} \left|O_o^{(t-i)}\right|} ,$$

where the set $\mathrm{parents}(g)$ contains the parents of g. The operator probabilities $P_o^{(t+1)}$ are adjusted every τ generations according to

$$s_o^{(t+1)} = \begin{cases} c_\Omega q_o^{(t,\tau)} / q_{\text{all}}^{(t,\tau)} + (1 - c_\Omega) s_o^{(t)} & \text{if } q_{\text{all}}^{(t,\tau)} > 0 \\ c_\Omega / |\Omega| + (1 - c_\Omega) s_o^{(t)} & \text{otherwise} \end{cases}$$

and

$$P_o^{(t+1)} = P_{\min} + (1 - |\Omega| P_{\min}) s_o^{(t+1)} \Big/ \sum_{o' \in \Omega} s_{o'}^{(t+1)} .$$

The factor $q_{\text{all}}^{(t,\tau)} = \sum_{o' \in \Omega} q_{o'}^{(t,\tau)}$ is used for normalization. The learning rate $c_\Omega \in (0, 1]$ is set to $c_\Omega = 0.3$. The operator probabilities $P_o^{(t+1)}$ are bounded from below by $P_{\min} < 1/|\Omega|$. Initially, we set $s_o^{(0)} = P_o^{(0)} = 1/|\Omega|$ for all $o \in \Omega$.

4 Experiments

4.1 Experimental Setup

We considered the pedestrian classification benchmark dataset introduced in [10], where $29,400$ ($14,400$ pedestrian and $15,000$ non-pedestrian) examples are available for training. The test part of the database consists of $19,600$ examples ($9,600$ positive and $10,000$ negative examples). The training examples are split in three sets $D_{\text{Train},1}$, $D_{\text{Train},2}$ and $D_{\text{Train},3}$, the test examples in two sets $D_{\text{Test},1}$ and $D_{\text{Test},2}$. Figure 4 shows some samples from the database.

For optimization, only the three training sets were used for 3-fold cross-validation, whereas the test examples were exploited for generating the ROC curve of the final classifiers.

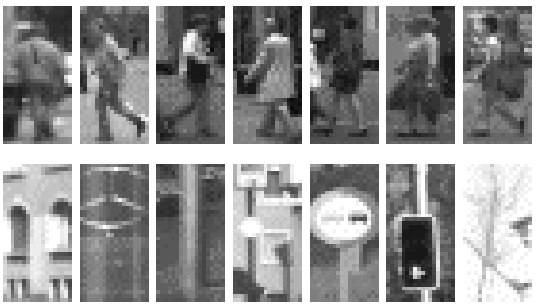


Figure 4. Samples from the pedestrian classification benchmark dataset [10].

As a baseline for comparison, one linear classifier was also trained based on the Haar feature set proposed in [10].

4.1.1 Creating random features.

In the first generation of the EA and during evolution, new features have to be created randomly.

To create one feature with given width w^F, the parameters t^F and c^F were drawn uniformly distributed at random. The height was randomly chosen: $h^F = w^F \cdot 1/2 \cdot z$ with $z \sim \mathcal{N}(1, 0.01)$. As the height/width ratio of the images in the considered dataset is 2, the features tended to be quadratic. The thresholds x_t^F and x_s^F were chosen normally distributed with mean 5 and 150, respectively (as $g_{\max} = 255$), and $a^F = 2$ with probability 0.6. These values had been found empirically to provide good results.

4.1.2 Feature set optimization.

Experiments with an EA for different feature set sizes were performed. The population size was set to 25, and the same amount of offspring was generated, where always two parents were randomly chosen to create one offspring by cross-over. Nine trials were conducted for each $n_{\text{feat}} \in \{50, 100, 150, 200\}$, the individual with the highest fitness from each run was stored and used for performance evaluation.

4.1.3 Evaluation of results.

For generating a ROC curve, the same procedure as in [10] was used: Three classifiers were trained, each on two out of the three training sets $D_{\text{Train},1}$, $D_{\text{Train},2}$ and $D_{\text{Train},3}$. Then, for each of the three classifiers, two ROC statistics were calculated, one on $D_{\text{Test},1}$ and another one on $D_{\text{Test},2}$. The six resulting ROC statistics were combined to give the final ROC curve.

4.2 Results

4.2.1 Feature set optimization.

Figure 5 shows the mean fitness of the best individuals in each generation from the nine trials with $n_{\text{feat}} = 50$.

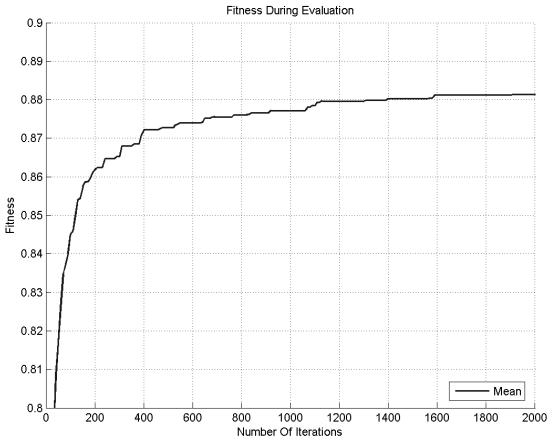


Figure 5. Fitness during evolution.

Table 1 shows the fitness of the best individual obtained for different feature set sizes.

Number of features	Fitness of best individual
50	89.23%
100	90.75%
150	91.68%
200	92.69%

Table 1. Best individuals from trials with different feature set sizes.

Figure 6 illustrates the best feature set with 100 features. Many of the evolved features look different from our basic types and typical manually designed features.

Figure 7(a) compares the ROC curves of the best optimized classifiers for different feature set sizes and the ROC curve of the linear classifier based on the feature dictionary from [10]. In accordance with Munder er al. [10], this shows the advantage of adapted features compared to non-adapted ones.

For each feature set size, we selected the very best solution of each trial to form an ensemble. Figure 7(b) compares the ROC curves of the different ensemble classifiers to the results of a quadratic support vector machine (SVM) from [10]. This SVM was the best classifier based on Haar features found by Munder et al. without increasing training sample size (the best classifiers in that study are based on local receptive fields).

Additionally to the good detection rates of the evolved classifiers, execution times are faster by orders of magnitude compared to the SVM: Linear classification corresponds to the calculation of one scalar product, whereas the classification with the quadratic SVM requires to calculate one scalar product per support vector (several thousands in this case).

Classification of 1000 test samples took 0.76 ms ($n_{\text{feat}} = 100$) on an Intel Pentium M processor with 1.8 GHz.

5 Conclusion

Object detection for real-world applications is a challenging task, and in many cases good classification performance comes along with slow execution times.

We considered the design of highly optimized classifiers and presented an architecture that uses evolutionary optimization for automatically generating a small feature set specialized for the given task. The feature set is adapted to both the classification problem at hand and to the classification technique used.

The proposed architecture was applied to the problem of pedestrian classification for driver assistance systems. Linear discriminant analysis was used for the decision whether or not a given example is a pedestrian. We obtained a set of specialized wavelet features and showed that its performance is comparable to results obtained with support vector machines on similar features while the execution time is lower by orders of magnitude.

Designing a system for real-time object detection requires finding an appropriate trade-off between classification accuracy and computational complexity. The hybrid approach of evolutionary optimization of specialized feature sets for rather simple learning machines addresses this challenge. The performance of the resulting classifier may not be better than the performance of complex classifiers, but the computational complexity is reduced drastically.

References

[1] M. Bertozzi, A. Broggi, A. Fascioli, and M. Sechi. Shape-based pedestrian detection. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, pages 215–220, 2000.

[2] C. Curio, J. Edelbrunner, T. Kalinke, C. Tzomakas, and W. von Seelen. Walking pedestrian recognition. *IEEE Transactions on Intelligent Transportation Systems*, 1(3):155–163, 2000.

[3] D. B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE Press, 1995.

[4] D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection: the PROTECTOR system. In *Proceedings of the IEEE Intelligent Vehicles Symposium*, pages 13–18, 2004.

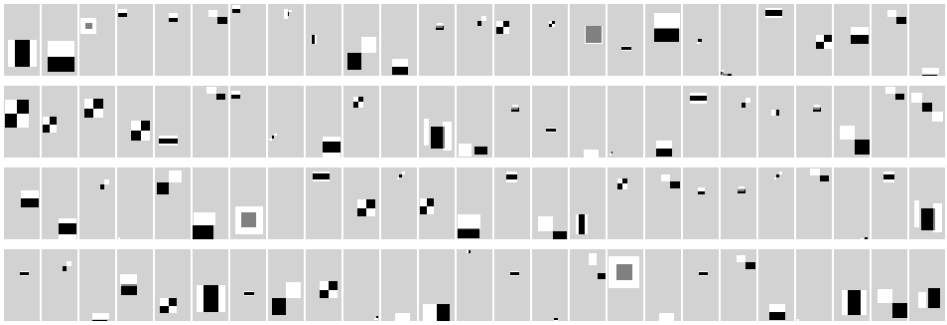
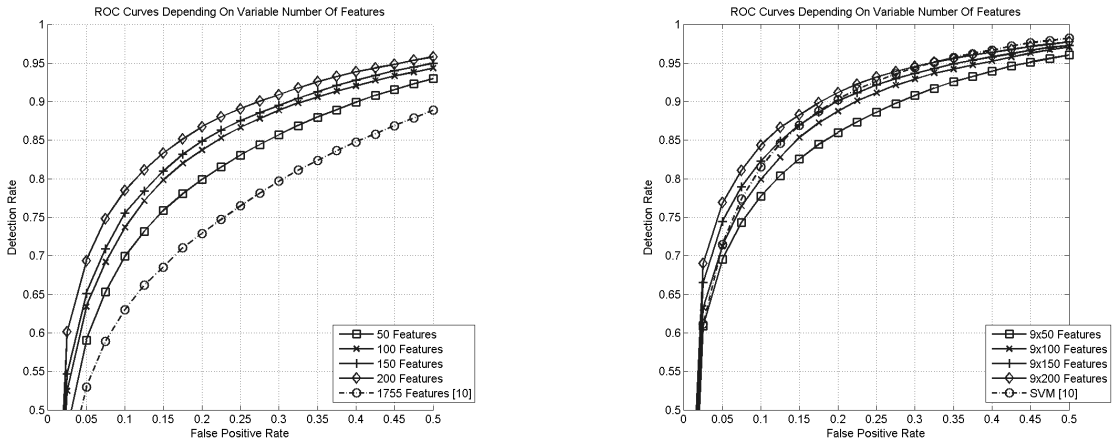


Figure 6. Optimized feature set (100 features).



(a) **(b)**

Figure 7. In subfigure (a), ROC curves of the best classifiers with 50, 100, 150 and 200 features are compared to the ROC curve from [10]. In subfigure (b), the ROC curves of ensemble classifiers are compared to ROC curves from [10].

[5] A. Guarda, C. L. Gal, and A. Lux. Evolving visual features and detectors. In *Proceedings of the International Symposium on Computer Graphics, Image Processing, and Vision*, pages 246–253, 1998.

[6] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[7] C. Igel and M. Kreutz. Operator adaptation in evolutionary computation and its application to structure optimization of neural networks. *Neurocomputing*, 55(1–2):347–361, 2003.

[8] C. Igel, S. Wiegand, and F. Friedrichs. Evolutionary optimization of neural systems: The use of strategy adaptation. In M. G. de Bruin, D. H. Mache, and J. Szabados, editors, *Trends and Applications in Constructive Approximation*, volume 151 of *International Series of Numerical Mathematics*, pages 103–123. Birkhäuser Verlag, 2005.

[9] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[10] S. Munder and D. M. Gavrila. An experimental study on pedestrian classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(11):1863–1868, 2006.

[11] C. Papageorgiou, T. Evgeniou, and T. Poggio. A trainable pedestrian detection system. In *Proceedings of the IEEE Intelligent Vehicles Symposium 1998*, pages 241–246, 1998.

[12] R. Polana and R. Nelson. Low level recognition of human motion. In *Proceedings of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 77–82, 1994.

[13] A. Treptow and A. Zell. Combining AdaBoost learning and evolutionary search to select features for real-time object detection. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 2107–2113, 2004.

[14] P. Viola and M. Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[15] L. Zhao and C. Thorpe. Stereo- and neural network-based pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(3):148–154, 2000.