Methods for vehicle detection and vehicle presence analysis for traffic applications

Oliver Sidla*, Yuriy Lipetski
SLR Engineering GmbH, Gartengasse 19, A-8010 Graz, Austria

ABSTRACT

This paper presents our work towards robust vehicle detection in dynamic and static scenes from a brief historical perspective up to our current state-of-the-art. We cover several methods (PCA, basic HOG, texture analysis, 3D measurement) which have been developed for, tested, and used in real-world scenarios. The second part of this work presents a new HOG cascade training algorithm which is based on evolutionary optimization principles: HOG features for a low stage count cascade are learned using genetic feature selection methods. We show that with this approach it is possible to create a HOG cascade which has comparable performance to an AdaBoost trained cascade, but is much faster to evaluate.

Keywords: vehicle detection, genetic algorithm, HOG cascade, HOG detector

1. INTRODUCTION

This paper presents our work towards robust vehicle detection in dynamic and static scenes from a historical perspective up to current state-of-the-art methods. The robust detection of vehicles is of great importance for intelligent transportation systems (ITS), especially for law enforcement, traffic monitoring, and traffic flow analysis systems. Despite the progress made in the last 10 years, we are still some steps away from an optimal, real-time capable algorithm which is able to detect vehicles completely pose and rotation invariant (scale invariance can be handled relatively easy by all known approaches, although this may imply considerable more computing time).

The algorithms presented in this work are designed to run in real-time on small platforms like Intel Atom based machines, or small ARM based embedded systems. For the purpose of becoming faster we sometimes accept compromises in terms of detection quality, our results are nevertheless comparable to current methods. Furthermore, most our algorithms, data sets, and training results are tuned to vehicle front/real views with a deviation of no more than 30 degrees. The majority of traffic scenarios we have dealt so far are limited to those viewing angles and this limitation allows us to become more robust. Applications of the algorithms elaborated in this work include

- vehicle tracking for trajectory and flow analysis
- law enforcement
- parking lot management and monitoring
- verification of vehicle presence as supporting function
- vehicle detection for congestion analysis

Departing for the still widely used methods of background modeling and blob analysis, our efforts have from the beginning concentrated on appearance based methods. The first part of this work gives an overview of the different algorithms we have tested and deployed in the field over the past years. The second part of this work introduces a new training algorithm for HOG cascades with low stage count (<= 5) and very good runtime performance. These detector type can achieve recall and precision rates which are comparable to the state-of-the-art reported in vehicle detection.
2. RELATED METHODS AND STATE-OF-THE-ART

2.1 Related work

The detection principles used for vehicles are generally similar to the concepts applied for pedestrian detection, generally with modifications which try to exploit specific properties of vehicle shape. The more rigid structure and usually high symmetry of cars (see Sidla[6]) can be exploited in order to verify a detection. This approach is taken by Khairdoost et al [1] who employ PCA and feature selection using a genetic optimization framework for their detection algorithm. Zehang et al [3] use a GA framework for the creation of an optimized set of Gabor filters for vehicle detection. By clustering features they avoid redundancies and thus speed up detection.

The use of genetic algorithms (GA) has never been of great importance for the design or of object detectors or even their optimization. Our experience nevertheless shows that GA can be used very effectively to explore the high dimensional feature spaces encountered in computer vision. Over the last 15 years we have optimized stereo matching features (Paar [9]), and tuned feature configurations for object detectors with this approach (Wildling [8]) and make use of this experience in the second part of this work.

The HOG detector framework introduced by Dalal and Triggs [2] still represents a baseline (although not considered to be state-of-the-art anymore) in terms of feature creation, sample classification and detection. Over time, the classical HOG framework has been improved in several ways to achieve higher processing speed and increase detection robustness and accuracy. By employing a cascade of simple HOG detectors in a cascaded fashion, Zhu et al [10] can significantly speed up the detection process. Their work is distinguished by the fact that they learn important features and simple classifiers with the AdaBoost strategy and combine the results into a cascade of increasingly complex classifiers. They make also use of the concept of integral histograms to make the creation of local edge orientation histograms in overlapping windows much faster. They can speed up the HOG detection process 10-15 times compared to the original version of Dalal and Triggs.

Local Binary Patterns (LBP) as local texture descriptors which have until now mainly been used in machine vision applications, are combined by Wang et al [11] with HOG features in order to create stronger object descriptors. The advantage of the LBP approach is that it is computationally very efficient. In a similar fashion Armanfad et al [12] use Local Binary patterns in their texture-edge descriptor in blocks of pixels to segment background patches from object candidates with promising results. The addition of new feature dimensions by adding Local Binary Patterns also improves the recognition performance of our vehicle presence detector. We show that the combination of HOG and LBP can achieve almost 100% accuracy even in adverse and varying outdoor conditions.

2.2 Contribution of this work

The main contribution of this work is the introduction of evolutionary algorithms for the training of low stage count HOG detector cascades. Although a training algorithm using genetic optimization to create HOG cascades is not faster or demanding less CPU power than AdaBoost (the training of a cascade can take days to complete), we believe it is advantageous because it can better exploit the vast high dimensional space covered by HOG features.

Some work on the use of genetic algorithms for the training of HOG classifiers has been reported in the past (see Zehang [3], Khairdoost [1]), but generally feature selection using genetic algorithms is seldom used in favor of AdaBoost and its variations. This work tries to demonstrate that evolutionary computation can be brought to advantage for the design of low cascade count detectors which have similar, if not better, properties than detectors trained with AdaBoost.

The following Section 3 provides an overview of vehicle detection methods we have developed over the years and which have been proven useful for actual, real-world applications. Section 4 introduces the genopt evolutionary computation framework for the training of our current state-of-the-art vehicle detector.
3. METHODS FOR VEHICLE DETECTION

3.1 Principal Component Analysis

Our first appearance based approach for vehicle detection made use of the fact that cars exhibit a relatively similar basic structure which can be described and compressed into an efficient model using principal component analysis (PCA, see Sidla [6]). To further enhance the robustness of detection, we de-emphasized the outer regions of detection windows with a Gaussian weighting mask. Several classifiers approaches have been tested in this work, with Support Vector Machines (SVM) and k-NN delivering the best results.

A set of manually annotated training images \( V \) (typically several thousand) was prepared in the first step. Each of the \( V \) training images was then converted to a vector \( s' \) of size \( M \) by concatenation of all of its rows:

\[
s' = \left[ s'_1, s'_2, \ldots, s'_M \right] \tag{1}
\]

Then the average vector \( c \) of all training samples is subtracted from each \( s' \) which are then stacked column-wise into the matrix \( P \) of size \( M \times V \):

\[
c = \frac{1}{V} \sum_{i=1}^{V} s' \tag{2}
\]

\[
P = \{ s' - c, s' - c, \ldots, s' - c \} \tag{3}
\]

The Eigenvectors of the pattern training set are computed by solving the Eigen decomposition problem with \( Q \) being the covariance matrix of \( P \):

\[
Q = PP^T, \quad Qe_k = \lambda_k e_k \tag{4}
\]

Generally all of the Eigenvectors are necessary to fully reconstruct an image. But since they are placed so that the variation of the training samples along them is maximized, it is possible to use only a few in order to reconstruct an image sufficiently detailed for recognition. Practice shows that usually no more than 20 Eigenvectors are needed for good recognition rates. The first \( K \) Eigenvectors with the largest Eigenvalues form the Eigenspace, or image subspace, of dimension \( K \). Figure 1 shows the first five Eigenvectors as they have been computed from our training image set. The relative size of the Eigenvalues is a measure the importance of the Eigenvector for image reconstruction.

![Figure 1. The first five Eigenvectors with associated Eigenvalues e1, e2, e3, e4, e5 as computed from our training set.](image)

Each of the training images \( (s' - c) \) is projected into the Eigenspace giving a \( K \) coordinate vector \( f' \) - it represents the compressed image \( s' \) in the Eigenspace:

\[
f' = \left[ (s' - c)e_1, (s' - c)e_2, \ldots, (s' - c)e_K \right] \tag{5}
\]

The image subspace representation \( f' \) reduces the amount of storage needed for a training image from typically 128x128 pixels to only \( K \) (which is around 20) co-ordinates within Eigenspace. Image comparison respective the database search uses only the \( K \) size feature vector which results in a dramatic speedup compared to direct image based correlation methods.
An input image is sampled (possibly using different scales) by a sliding window approach. Each sub-image, or local pattern $p^n$ is projected into the Eigenspace of the training set, generating $R$ K-dimensional coordinates (note the average image vector $c$ is used again):

$$r^n = [(p^n - c)e_1, (p^n - c)e_2, \ldots, (p^n - c)e_K].$$  

(6)

The local pattern $r^n$ may now be classified with respect to the training database which is represented by the vectors $f^l$.

Our experiments have shown that classification using k-NN or a linear SVM result in the best recognition rates.

### 3.2 Classical HOG detection

The disadvantages of PCA, namely its sensitivity to illumination changes relative to the training data base and its sensitivity to scale and rotation variations makes it hard to deploy in real-world installations without considerable re-training efforts. The much better generalization capabilities of the HOG detection method introduced by Dalal and Triggs [2] makes use of local normalized histograms of gradient orientations. The histograms are collected from overlapping small patches and combined into large feature vectors for subsequent classification. Our first approach used polynomial SVMs for better classification performance, but was later modified to use linear SVMs for the much better runtime performance. Linear SVMs can, when optimized, be executed 2-3 orders of magnitude faster than SVMs with a more complicated kernel (a comparison of basic HOG and our PCA is given in Sidla [4]).

Figure 2 shows the arrangement of cells and blocks used in our classical HOG implementation. This system for car detection has been used in several experimental traffic analysis systems. The implementation from the year 2006 was fast enough to allow real-time operation in small regions of interest (ROI) of images. Note that the PCs used at the time were an order of magnitude less powerful than today’s systems. Figure 3 shows an image from one of the experimental installations.

---

**Figure 2:** The computation of the classical HOG. Cells of size 6x6 pixels are combined into blocks, overlapping blocks span the whole sample window. The resulting high dimensional feature vector is classified using a linear SVM.

**Figure 3:** The output of our HOG detector as used in an experimental setup for a traffic analysis system. Detection takes place only in the marked regions within the frame. Detected cars are drawn as green rectangles. In order to save CPU time, the scanning step for the sliding window is relatively large, therefore location accuracy is limited.
3.3 HOG combined with LBP

This section describes a real-world parking lot monitoring application in a demanding outdoor environment which combines HOG descriptors and local binary patterns (LBP) in order to create a stronger classifier (Lipetski [13]).

The first part of the classifier uses HOG feature descriptors trained with a linear SVM. The vehicle detection areas in the camera image are of several meters length (see Figure 4 below). Every area is split into 4x4 blocks, each of which is divided into 4x4 non-overlapping cells. We construct a gradient histogram with a resolution of 8 bins over a 180 degree range. Thus, the whole feature vector length is $4 \times 4 \times 4 \times 4 \times 8 = 2048$ elements.

The second part of the classifier computes LBP histograms within the detection ROIs which are classified using linear SVMs. The combination of HOG and LBP achieves a very good detection rate with very few false positives for several reasons:

- We take into account a history of $N$ successive detector answers to suppress single false detections.
- The system is made more robust by the combination of two different detectors. Only when both detectors, HOG and LBP, mark the presence of a vehicle, is the state of the parking lot changed to occupied.
- The site specific training optimizes the detector for the geometry of the situation. This is important for both HOG and LBP features. This drawback would need to be eliminated in a production system to avoid the overhead of manual training for every camera.

Two smart cameras in the City of Vienna have been installed at test sites and operated constantly day and night during a period of several months. The goal was to observe parking areas which are used for delivery of goods and need to be free at all times. The sensors had to handle two parking lots per field of view.

For evaluation purposes we have taken two arbitrary days for the training purpose and another two arbitrary days for evaluation. From the two training days we have labeled images with ten seconds intervals. By using only images between 6AM and 7PM we gathered 4680 training samples per parking place.

The results of our evaluation were encouraging: 97.4 % of all vehicles were detected, at a false positive rate of 0 %, despite the fact that the parking lost were constantly perturbed by pedestrians, shadows, and reflections. We explain this robustness against passing persons by the fact that people have mostly vertical edges, while vehicles also contain horizontal edges. All of the 2.6% of the vehicles that we did not detect were parked in such a way that they occupied only a fraction (mostly less than 50%) of the parking lot area.

![Figure 4: Typical detection results using the combined HOG-LBP detector. The disadvantage of camera specific training is offset by the excellent detection rate and very low false positive rate.](image)
3.4 3D Stereo system for detection and localization of vehicles

Recently we have used a new detection modality to measure the location and position of cars relative to tram tracks on the ground: a wide baseline stereo sensor can detect those cars in a parking area which may potentially be too close to the tracks. The basic geometric setup can be seen in Figure 5 below.

![Figure 5: The setup of the stereo detection system. A stereo camera rig with a vertical baseline of 1500 mm (symbolized left by the camera symbol) observes an area of parking lots from a distance of 33 m to 55 m.](image)

A vertical imaging sensor arrangement was chosen for this purpose because it can be mounted easily on the supporting mast. In addition, our experiments have shown that the computed stereo disparities from the vertical configuration seemed to be slightly more robust than those from the horizontal sensor configuration.

The stereo camera rig with a vertical baseline of 1500 mm (symbolized in Figure 5 left by the camera symbol) observes a stretch of parking lots from a distance of 33 m to 55 m. The accuracy $dz$ of depth estimation from stereo reconstruction depends on the camera baseline $B$, focal length $f$, and eventually the disparity resolution $D$ at an object distance $z0$ (for $f << z0$):

$$dz = \frac{z0^2 * D}{B * f}$$

For the system parameters (estimated) $D = 0.0010$ mm, working distance $z0 = 40000$ m, baseline $B = 1500$ mm, and focal length $f = 16$ mm we have an approximate accuracy of 67 mm in depth resolution. In practice we have observed a localization accuracy of the side of the cars of around 25–60 mm (between 33000 – 55000 mm working distance) which is roughly corresponding with the theoretical result – probably we can achieve a better disparity estimation accuracy than the assumed 1/4 pixel. In addition the lateral car position measurement is not only dependent on the accuracy of depth measurement but also on the horizontal point matching accuracy, so that problems in accurate absolute depth estimation are mitigated. The processing steps for vehicle detection from the stereo measurement can be summarized as follows:

1. Disparity estimation using a stereo matching algorithm on the rectified stereo image pair.
2. 3D reconstruction with known calibration of the camera rig.
3. The street level is computed using RANSAC plane fitting on the 3D points.
4. Keep only points in an elevation range 150–1500 mm above ground plane.
5. Project points from Step 4 to the ground and find best fitting rectangles. To filter out noise and outliers, only objects of a minimum size of 1500 mm length will be accepted.

6. From camera calibration, the ROI and border lines which make up the detection area and parking lot limits are known. Mark those objects detected in Step 5 and which cross the virtual boundary line as potential hazards.

Figure 6 depicts results of the detection algorithm outlined above. The system has been running at the test site for 3 months during day and night. The processing loop takes 5 Seconds on a motherboard equipped with an Intel Atom processor. Note the the quality of the 3D point cloud along the car structures is not very good due to the little texture available for stereo matching. Still it is sufficient in order to create a useful depth map and derive measurements.

This system works robust enough so that after upgrading the imaging sensors to higher resolution and providing a faster computing unit it can be commercialized in the next project phase.

Figure 6: The 3D processing steps. The numbers in the sub-images correspond to the according steps as mentioned in the algorithm description.
4. GENETIC OPTIMIZATION ALGORITHM

4.1 An Alternative to AdaBoost strategies

State-of-the-art HOG detectors employ deep cascades which are practically always trained using AdaBoost (see Dollar [5] for a list of current HOG pedestrian detection architectures). Our first HOG detector cascades also used AdaBoost training for cascades with increasingly more complex HOG descriptors from stage to stage.

The first 1-2 stages in a detector are designed to be lightweight and fast, so that processing time is saved. Subsequent stages have to test only the already filtered set of samples and can therefore be allowed to be more complex. Typically 80–90% of all false positives are rejected in the first two stages. Our multi-stage HOG detection cascade which has been trained using AdaBoost resulted in a very significant speedup as well as an improvement in recognition quality compared to the classical HOG detector which uses all cells within one stage. The detector resulting from AdaBoost training can run almost in real-time on a low power Atom processor and has been used in several real-world applications so far. Our car HOG cascade with AdaBoost training is detailed in Table 1. It consists of 13 stages with ever increasing feature numbers.

This section describes a method to further reduce the number of stages needed for the design of strong detectors. The concept of evolutionary algorithms for optimization tasks is rarely found in the literature. A genetic algorithm is a subset of evolutionary algorithms that model biological processes to optimize highly complex cost functions. A genetic algorithm allows a population composed of many individuals to evolve under specified selection rules to a state that minimizes a predefined cost function. The method was originally developed by John Holland and further improved over the years. Some advantages of genetic algorithms are: optimization with discrete or continuous parameters, it searches from a wide sampling of the cost function, it can deal with a large number of parameters and it optimizes parameters with extremely complex cost surfaces (can jump out of local minima).

We believe that genetic optimization is a valuable tool for finding features in cascaded detectors because it can better exploit the high dimensional search space encountered in this type of problems. To this end, to leverage the full potential of the combination of a HOG cascade and a linear SVM classifier, we have designed a genetic algorithm framework (GA) for the training of a HOG cascade which is described in the following Section 4.2. Section 4.3 gives results of the resulting detector cascade in comparison to our old AdaBoost detector.

4.2 Training using the genopt algorithm

This section describes our algorithm for training a cascaded HOG detector (see also Wildling [8] and Paar [9] for an overview). The genetic optimization algorithm, we call our version genopt, is an iterative procedure. Each (initially random) set of HOG descriptors for all stages make up an individual in this context. For every generation the quality (fitness function $F$) of each individual is evaluated on a test set. Good individuals are combined (mated) into new siblings, they form the basis for the next generation. The mating process takes two parents and merges them into two new siblings by splicing and merging their descriptor information at random points, this is called crossover. In order to allow for variation in the data set, mutation can change an individual HOG descriptor randomly in every generation. The best fitting individuals are selected for mating, they ‘survive’ the selection process with regard to good classification results on the car database.

Table 1: The 13 cascaded HOG detectors from Stage 1 to Stage 13 and the number of features used in each stage. The complexity of each HOG detector increases from stage to stage, but since also less and less candidate windows are being evaluated, the overall runtime performance of the cascade is very good.

<table>
<thead>
<tr>
<th>STAGE</th>
<th>STAGE 2</th>
<th>STAGE 3</th>
<th>STAGE 4</th>
<th>STAGE 5</th>
<th>STAGE 6</th>
<th>STAGE 7</th>
<th>STAGE 8</th>
<th>STAGE 9</th>
<th>STAGE 10</th>
<th>STAGE 11</th>
<th>STAGE 12</th>
<th>STAGE 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>352</td>
<td>416</td>
<td>544</td>
<td>1056</td>
<td>1664</td>
<td></td>
</tr>
</tbody>
</table>

This section describes a method to further reduce the number of stages needed for the design of strong detectors. The concept of evolutionary algorithms for optimization tasks is rarely found in the literature. A genetic algorithm is a subset of evolutionary algorithms that model biological processes to optimize highly complex cost functions. A genetic algorithm allows a population composed of many individuals to evolve under specified selection rules to a state that minimizes a predefined cost function. The method was originally developed by John Holland and further improved over the years. Some advantages of genetic algorithms are: optimization with discrete or continuous parameters, it searches from a wide sampling of the cost function, it can deal with a large number of parameters and it optimizes parameters with extremely complex cost surfaces (can jump out of local minima).

We believe that genetic optimization is a valuable tool for finding features in cascaded detectors because it can better exploit the high dimensional search space encountered in this type of problems. To this end, to leverage the full potential of the combination of a HOG cascade and a linear SVM classifier, we have designed a genetic algorithm framework (GA) for the training of a HOG cascade which is described in the following Section 4.2. Section 4.3 gives results of the resulting detector cascade in comparison to our old AdaBoost detector.

4.2 Training using the genopt algorithm

This section describes our algorithm for training a cascaded HOG detector (see also Wildling [8] and Paar [9] for an overview). The genetic optimization algorithm, we call our version genopt, is an iterative procedure. Each (initially random) set of HOG descriptors for all stages make up an individual in this context. For every generation the quality (fitness function $F$) of each individual is evaluated on a test set. Good individuals are combined (mated) into new siblings, they form the basis for the next generation. The mating process takes two parents and merges them into two new siblings by splicing and merging their descriptor information at random points, this is called crossover. In order to allow for variation in the data set, mutation can change an individual HOG descriptor randomly in every generation. The best fitting individuals are selected for mating, they ‘survive’ the selection process with regard to good classification results on the car database.

Table 1: The 13 cascaded HOG detectors from Stage 1 to Stage 13 and the number of features used in each stage. The complexity of each HOG detector increases from stage to stage, but since also less and less candidate windows are being evaluated, the overall runtime performance of the cascade is very good.

<table>
<thead>
<tr>
<th>STAGE</th>
<th>STAGE 2</th>
<th>STAGE 3</th>
<th>STAGE 4</th>
<th>STAGE 5</th>
<th>STAGE 6</th>
<th>STAGE 7</th>
<th>STAGE 8</th>
<th>STAGE 9</th>
<th>STAGE 10</th>
<th>STAGE 11</th>
<th>STAGE 12</th>
<th>STAGE 13</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
<td>FEAT.</td>
</tr>
<tr>
<td>32</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>128</td>
<td>192</td>
<td>256</td>
<td>352</td>
<td>416</td>
<td>544</td>
<td>1056</td>
<td>1664</td>
<td></td>
</tr>
</tbody>
</table>

This section describes a method to further reduce the number of stages needed for the design of strong detectors. The concept of evolutionary algorithms for optimization tasks is rarely found in the literature. A genetic algorithm is a subset of evolutionary algorithms that model biological processes to optimize highly complex cost functions. A genetic algorithm allows a population composed of many individuals to evolve under specified selection rules to a state that minimizes a predefined cost function. The method was originally developed by John Holland and further improved over the years. Some advantages of genetic algorithms are: optimization with discrete or continuous parameters, it searches from a wide sampling of the cost function, it can deal with a large number of parameters and it optimizes parameters with extremely complex cost surfaces (can jump out of local minima).

We believe that genetic optimization is a valuable tool for finding features in cascaded detectors because it can better exploit the high dimensional search space encountered in this type of problems. To this end, to leverage the full potential of the combination of a HOG cascade and a linear SVM classifier, we have designed a genetic algorithm framework (GA) for the training of a HOG cascade which is described in the following Section 4.2. Section 4.3 gives results of the resulting detector cascade in comparison to our old AdaBoost detector.
Critical for the success of the genopt algorithm are the following points:

- **The genome of individuals**: consists of HOG cell descriptors. The location and size of a block of 2x2 cells is part of the genome of each individual which is optimized.

- **The fitness function** defines the performance of the detector cascade in terms of recall and precision. Our fitness function is defined so that false positives are weighed in more than false negatives.

- **Samples** used for training and test: the training and test samples should cover as many imaging situations as possible to generalize the detector cascade as much as possible and to avoid over fitting.

- **Design of the detector cascade**: to reduce the complexity of the GA training procedure we pre-define the number of cascade stages as well as the number of features to be used in each stage. Currently we combine four stages ST_0, ST_1, ST_2, ST_3 into one HOG detector, with 4, 8, 16, and 32 cells each.

Figure 7 below depicts the sample selection and training algorithm for every generation. The HOG detector individuals are created and initialized using random cells. A basis set of manually annotated ground truth data TP/TN Train is used to train the detectors of the first cascade stage ST_0, their performance is evaluated on the ground truth data TP/TN Test. We extract additional pure background samples TN BG (true negatives background) from a set of street images so that the performance of the detector stages can be tested against a large set of diverse image data to avoid over fitting.

The following cascade stages ST_1, ST_2, ST_3 combine the TP Train data sets and the false positives from the classifier stage of the previous stage for training, and again all TP/TN Test samples and TN BG samples for testing.

The fitness function for each individual tests the quality of each genome (set of cell descriptors) by computing the metric given in equation (7) below. The number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), respective the sum of their scores $TP_{scores}$, $TN_{scores}$ are weighted so as to make the fitness function sensitive to false positives:

$$F = \frac{\sum TP_{scores} + \sum TN_{scores} - (#FP_{BG} + #FP) \cdot 8 - #FN \cdot 0.5}{TP + FN + TN + FP} \quad (8)$$

Figure 7: The genopt training algorithm for all cascade stages ST_0-ST_4 for one individual. Each stage uses a mixture of manually annotated ground truth data (TP/TN Train, TP/TN Test), as well as background samples extracted from scenes without vehicles.
4.3 The full car detection pipeline

The fully trained car detection cascade contains the 4 HOG cascade stages. The detection algorithm then consists of a dense sliding window detector over several scales with additional post-processing phases which are designed to reject remaining false positives. Our approach extracts image samples \( W \) in a dense grid, each sample is classified into vehicle/background window as follows:

1. For every sample \( W \), HOG cascade step \( C \), stage acceptance threshold \( T_C \) and total confidence threshold \( T_D \)
   a. Compute sum confidence \( D_W = D_W + S_C \)
   b. If \( S_C < T_C \) reject \( W \), test next sample

The algorithm checks every cascade stage against a stage threshold \( T_C \), only when a sample is accepted by all stages it is valid. It must then also have a sum score \( > T_D \) for final acceptance. The value of \( T_C \) allows us to control the recall rate of the detector, the introduction of \( T_D \) controls its precision. A higher \( T_C \) threshold makes the detector faster because a lesser number of samples will pass each cascade stage - at the cost of a reduced recall rate. Note that by optimizing the stage thresholds \( T_C \) the quality and speed of a cascaded detector can be further improved (see Bourdev [7]) – this will be investigated in the future.

Scale invariance is achieved by testing sample windows of different sizes at every image location. We use a scanning step of typically 0.07 of sample width, and a scale factor of 1.07 for 10-30 scales, depending on the application and imaging geometry.

The proposed car detection pipeline can run in real-time on a standard PC (see Section 5.3 below for details). Runtime is mainly affected by the scanning parameters which control how many samples are scanned per image, namely scanning step and number of scales per sample location.

5. EXPERIMENTS WITH THE GENOPT DETECTOR

This section describes experiments we have undertaken in order to evaluate the performance of the genopt HOG detector. The number and size of publicly car detection databases is limited and not very representative with regards to the possible performance of state-of-the-art detectors. We believe that, like for pedestrians, only a very and demanding set of images should be used for training, test and evaluation (see e.g. the Caltech data set [5] for pedestrians) and therefore have resorted to a sample set of our own.

5.1 Ground Truth Reference Data

Over the last years we have built a library of video files, images, and image samples which cover many different imaging situations, camera types, geometries, and climate conditions. We are continuously extracting car and pedestrian samples as solid ground truth data from these image sources. The following Table 2 lists the data sets for car detection we have used for training of our current detector. All samples are manually extracted and annotated as 72x72 pixel images.

Table 2: The SLR car samples used to train and test the genopt algorithm. In total a number of 11,924 annotated samples are used for training and test.

<table>
<thead>
<tr>
<th>Dataset</th>
<th># positive samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feldkirchen</td>
<td>300</td>
</tr>
<tr>
<td>Graz</td>
<td>200</td>
</tr>
<tr>
<td>REF2</td>
<td>2949</td>
</tr>
<tr>
<td>REF3</td>
<td>8475</td>
</tr>
</tbody>
</table>
5.2 Evaluation with ground truth data: genopt vs. AdaBoost

We have used a test video of a street scene in the City of Vienna for the evaluation of the detector properties. No car sample from this scene has been used for training.

Test runs have been made with i) the original genopt detectors of 4 stages, ii) a modified version which uses an additional 5\textsuperscript{th} stage in the form of a dense grid of cells/blocks as from a classical HOG detector, and iii) with our old AdaBoost detector.

The test video contains 1029 frames for which we have manually annotated every single car. In total this video contains 5300 cars. The scanning step was chosen as 0.08, the detection window scale step was 1.06 with 25 scales. Figure 8 shows the ROC curves for this setup. A detection is accepted if it overlaps more than 50\% with a manual annotation, as defined on the PASCAL evaluation criteria (Dollar [5]).

It can be seen that the AdaBoost detector performs worst, although it contains 13 stages in total. The new 4 stage genopt detector performs significantly better than AdaBoost, but it is superseded by the 5 stage genopt detector. The final verification of the low cost genopt HOG stage needs to process only very few samples (typically less than 100) so that the cost for this stage is low.

![Figure 8: The ROC curves for the evaluation test sequence from the City of Vienna. cars adaboost gives the ROC curve for our 13 stage AdaBoost detector, cars 4 stages shows the ROC curve for our detector trained using genopt with 4 stages in total, cars 5 stages adds a fifth classical HOG stage as verification step.](image)

5.3 Runtime performance

The low count of only 4 respectively 5 HOG stages lead to a fast execution time of the detector. Depending on the scale of the detection windows and the scanning steps, processing can easily be done in real-time (we consider everything > 15 fps real-time for our applications) on images of VGA resolution on a standard PC.

Table 3 lists processing times on the test sequence in Vienna for the 5 stage genopt detector with several configurations (scales, scanning steps). The tests have been made on an Intel laptop Core i7 processor @ 2.3 GHz, running on a single core without GPU support. They confirm that our 5 stage genopt detector runs easily in real-time over a detected vehicle size ranging from 60-200 px.
### Table 3: Runtime measurements for different configurations of the vehicle detector. The detector speed has been tested on a Video with 640x480 pixel resolution. The starting scale was 0.90 which corresponds to a sample window size of 65 pixels, the scale step was always 1.06 for 20 scale steps.

<table>
<thead>
<tr>
<th># scales</th>
<th>scan step size</th>
<th>runtime/frame (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.08</td>
<td>22</td>
</tr>
<tr>
<td>20</td>
<td>0.08</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>0.08</td>
<td>25</td>
</tr>
<tr>
<td>20</td>
<td>0.04</td>
<td>64</td>
</tr>
<tr>
<td>20</td>
<td>0.06</td>
<td>36</td>
</tr>
<tr>
<td>20</td>
<td>0.08</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>0.10</td>
<td>19</td>
</tr>
</tbody>
</table>

### 6. SUMMARY AND OUTLOOK

We have presented methods for the detection of vehicles in real-world environments for several different application scenarios. This includes an efficient implementation suitable for low power embedded systems. Building upon the experience gained from those systems we have presented a new algorithm for HOG cascade training which allows us to create shorter, and therefore faster, detection cascades without compromise in detection quality.

For vehicle detection there is still much to do in terms of robustness and stability for generic systems. In order to further improve the state-of-the-art, it is necessary to create large vehicle evaluation datasets which are publicly available, similar to the well known Caltech datasets for pedestrians. Our future work will concentrate on creating such a public database for scientific use.

### REFERENCES