Manuskript

Online Vehicle Detection using Haar-like, LBP and HOG Feature based Image Classifiers with Stereo Vision Preselection

Daniel Neumann[1], Tobias Langner[1], Fritz Ulbrich[1], Dorothee Spitta[1] and Daniel Goehring[1]

***Abstract*— Environment sensing is an essential property for autonomous cars. With the help of sensors, nearby objects can be detected and localized. Furthermore, the creation of an accurate model of the surroundings is crucial for high-level planning. In this paper, we focus on vehicle detection based on stereo camera images. While stereoscopic computer vision is applied to localize objects in the environment, the objects are then identified by image classifiers. We implemented and evaluated several algorithms from image based pattern recognition in our autonomous car framework, using HOG-, LBP-, and Haar-like features. We will present experimental results using real traffic data with focus on classification accuracy and execution times.**

I. Introduction

Driver assistance systems are becoming more and more common in modern cars, including e.g. parking assistants, autonomous cruise control (ACC) or lane departure warning systems (LDW). All these systems provide a certain degree of autonomy for safer, more comfortable, and more efficient driving. It is likely that fully autonomous cars may take part in everyday life in the near future. At the Freie Universität Berlin, the project *AutoNOMOS* has focused on autonomous driving for many years. Currently there are two test vehicles, *e-Instein* an electrically powered Mitsubishi i-MiEV and *MadeInGermany* a Volkswagen Passat Variant 3c, see Fig. 1.

MadeInGermany provides drive-by-wire technology to control the engine, brakes, and gears as well as other mechanical components directly via the connected CAN bus. For *e-Instein* there is another drive-by-wire technology which uses additional actuators. Our vehicles are equipped with a variety of sensors, see Fig. 2.

*This work was supported in part by the Federal Ministry of Education and Research (BMBF) as a part of the Project *Kombinierte Logik für Energieeffiziente Elektromobilität (KLEE)*.

[1]Department of Computer Science, Freie Universität, Berlin, Germany, email: { daniel.neumann | tobias.langner | fritz.ulbrich | dorothee.spitta | daniel.goehring }@fu-berlin.de

Fig. 1: Our test vehicles e-Instein and MadeInGermany

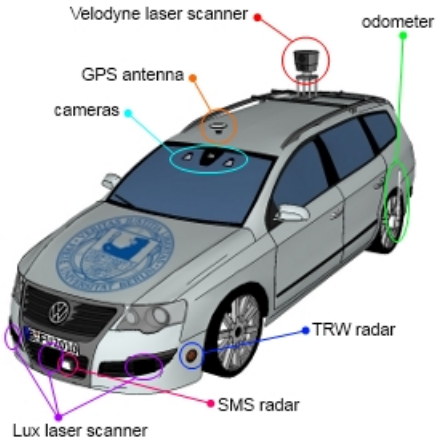


Fig. 2: Sensor configuration, camera, lidar-, and radar sensors on our Volkswagen Passat named MadeInGermany.

In this paper, we present a vehicle detection system based on a stereo camera system located behind the car's front window. Relying on camera images alone allows the creation of a fast and accurate system to detect rear views of cars. Furthermore, the system can easily be installed in normal cars. It is only important that the two cameras are precisely calibrated to obtain realistic results.
Our idea is to combine two image-based methods. On the one hand, we use stereoscopic vision to localize objects. On the other hand, we classify the objects as vehicles by using classical computer vision algorithms.

II. Related Work

Our current AutoNOMOS software framework provides a number of implementations for obstacle and object detection using radar or lidar sensors to localize objects precisely. However, identifying objects in specific scenarios such as traffic jam detection still remains a challenging task. Therefore, the goal of the here presented object recognition system is to reliably detect any trained pattern in camera images while using stereoscopic vision to localize object candidates in 3D space.

Moqqaddem et. al. [9] proposed an approach for object detection with stereo vision based on spectral analysis and k-means clustering. Similarly, [1] and [4] use clustering on 3D point clouds created by stereo vision to detect objects in camera images.

To identify objects, there exist popular image based computer vision algorithms. In [11], P. Viola and M. Jones

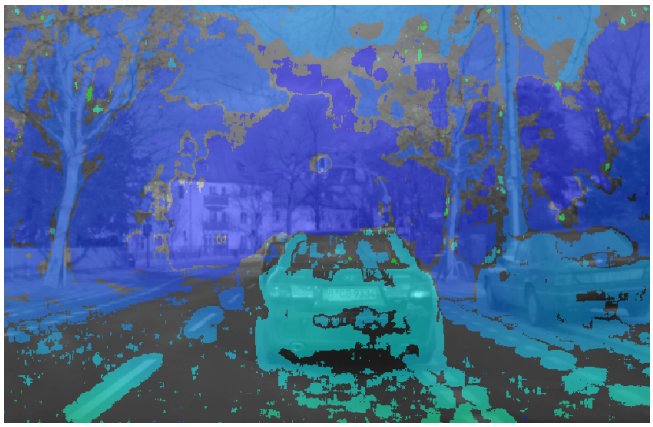


Fig. 3: Disparity map, points with no correspondence found are not colored.

showed how Haar-like features in combination with AdaBoost can be applied for face detection. N. Dalal and B. Triggs used histogram of oriented gradients (HOG) features [3] to detect pedestrians. P. F. Felzenszwalb's extension is able to detect deformations of patterns with a star model [5].

III. Application of stereo vision and image classifiers for Realtime-Vehicle Recognition

A. Stereo vision

In a stereo vision system, the environment is usually recorded by two cameras capturing different perspectives of a given scene. Spatial depth for a given point is then computed by searching for point correspondences in both camera images. To do so, we first rectify both images. We then determine the point correspondences using Konolige's Block Matching algorithm [7] which compares the *Sum of Absolute Differences* (SAD) between two blocks $B1$ and $B2$ of pixels from the image.

$$SAD = \sum_{x_i} \sum_{y_i} |B_2(x_i, y_i) - B_1(x_i, y_i)| \qquad (1)$$

Alternatively, we also experimented with Semi-Global Matching [6] and Variational Matching [8] which provide a smoother disparity map. But they consume more computation time and lead to the same results for our purpose.

The resulting offset of a point in one camera image with respect to the other image is called disparity which serves as a basis for a disparity map as shown in Fig. 3.

For object detection, we use a clustering approach based on the disjoint-set data structure [2] where nearby points are joined together as a graph. First, the 3D space is voxelized. If a minimum number of points are located inside a block, then the bock is filled. For our use case - the detection of other vehicles - we are only searching inside a height range of $0.8m$ to $2.0m$ above the ground and count filled blocks in two-dimensional layers with fixed heights as shown in Fig. 4. As a fill threshold, we choose $\geq 5\%$ per block. After that, we look for adjacent blocks with a maximum Manhattan distance of $2.0m$ and merge them together as a cluster.

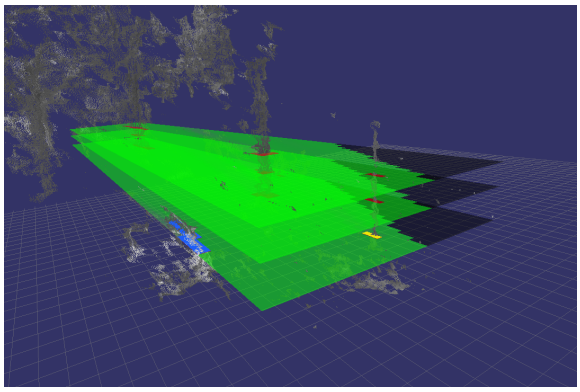


Fig. 4: Searching for filled blocks in two-dimensional layers with a fixed height above the ground

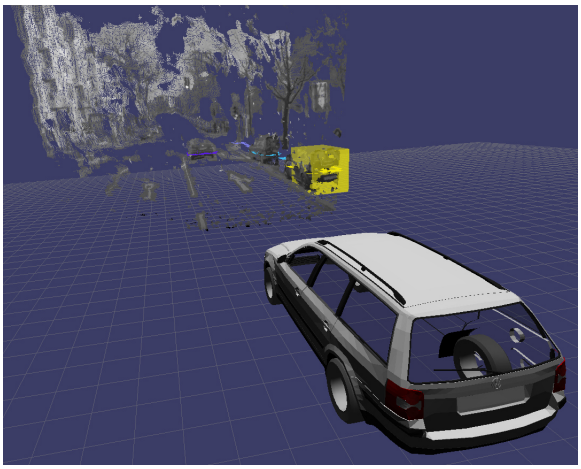


Fig. 5: 3D bounding boxes computed by stereo vision clustering.

To be tolerant against mismatches, the block size has to be chosen carefully. With a size that is too small there can be a lot of empty blocks because either none, or too few point correspondences are inside of it. That leads to missing parts or gaps inside a detected object. If block sizes are too large, the environment is not represented authentically and objects may be merged into only one cluster. Therefore, we observed $0.5m^3$ as a good block size. With clustering, we obtain potential car objects as 3D bounding boxes, see Fig. 5. Spatial coordinates of block clusters are then reprojected on the image plane to obtain two-dimensional bounding boxes.

B. Image classifier

To be able to identify vehicles, we use image classifiers and train them on images of rear views of vehicles. We then employ a variety of efficient computer vision algorithms.

One of them is the combination of HOG features with a support vector machine (SVM) by Dalal and Triggs [3]. In order to compute HOG features, the image is divided into blocks consisting of several cells. We use rectangular blocks and cells (R-HOG). For each cell, a histogram with gradient directions of all containing pixels is created. All directions ranging from $0° - 360°$ are discretized into a fixed number of directions. The resulting histograms of all feature blocks

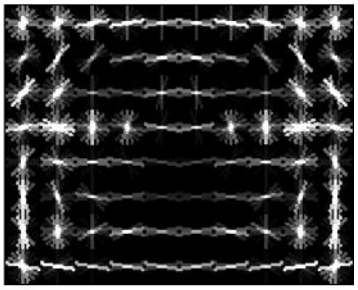
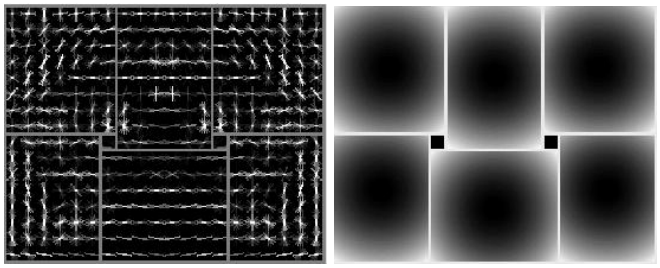
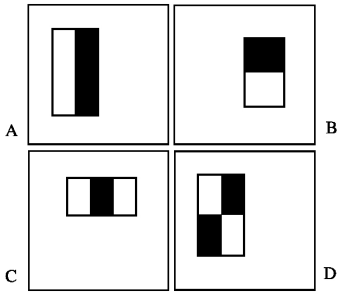


Fig. 6: HOG features of cars' rear view

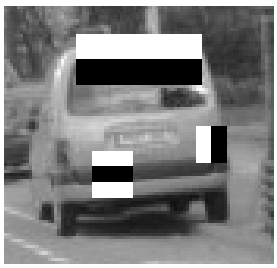


(a) HOG features of a car with part filters (b) Part filter positioning of a car

Fig. 7: HOG features of DPM detector [5]



(a) Types of Haar-like features [11]



(b) Example of learned Haar-like features

Fig. 8: Haar-like features

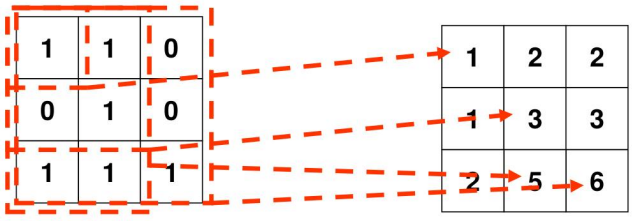


Fig. 9: Creation of an integral image

are concatenated into one vector, yielding the so-called HOG descriptor, illustrated in Fig. 6. For classification, an SVM compares the HOG features of an input image with the features of the trained images.

As an extension to [3], we applied the DPM (*Deformable Parts Model*) detector by P. F. Felzenszwalb et al. [5], where parts of the image pattern are trained with increased resolution by part filters, illustrated in Fig. 7a.

The parts and their order are computed during training of the SVM:

$$f(x) = sgn(\ max_{z \in Z(x)} < w,\ \theta(x,z) > + b\ ) \quad (2)$$

where set Z includes possible positions of part-filters, vector w is the trained HOG descriptor and $\theta(x,z)$ describes the concatenation of HOG features in input image x in addition to possible part-filter positions z. An optimal position z of the part-filters is searched such that $< w, \theta(x,z) >$ is maximized. For classification, a deformation cost function describes how far part-filters deviate from trained positions. An illustration of possible positions is given in Fig. 7b. The nearer the parts are in their respective dark regions, the more optimal the result.

Another applied image classifier is the algorithm of Viola and Jones [11], whereby Haar-like features are extracted from the image. Rectangular regions with shaded and clear areas are determined as shown in Fig. 8a. The resulting value of the feature is the sum of all pixels within clear rectangles minus the sum of the shaded rectangles. Positioning, scaling and rotation (respectively around 90 degrees) of the feature is arbitrary.

For each Haar-like feature j, a score f_j is computed as the difference between clear and shaded rectangles. A fixed threshold θ_j and a polarity $p_j \in \{-1, 1\}$ yield a weak classifier $h_j(x)$ where x is an input image:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

A strong classifier can be computed as a linear combination of weak classifiers, weighted by α_t and projected to either 1 or 0:

$$h(x) = \begin{cases} 1 & \text{if } \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

To improve the computation time for a score $f(x)$, integral images can be used. Thereby all values in the rectangle from position $(0,0)$ to (i,j) are summed up as shown in Fig. 9, where 0 denotes a white and 1 a black value.

An arbitrary rectangle D can be computed in constant time by using the bottom right values of the neighboring rectangles A, B and C (Fig. 10).

$$D = P4 - P2 - P3 + P1 \quad (5)$$

For further optimization, it is possible to evaluate several classifiers in a cascade. Instead of one complex strong classifier which decides directly whether a pattern in an image is found, we apply several weak classifiers in a decision tree, see Fig. 11.

Besides Haar-like features, we also apply HOG and LBP (*Local Binary Patterns*) features for our cascade method. LBP features [10] are similar to HOG features. The image is divided into cells of e.g. 24×24 pixels. For each pixel, all 8 neighbouring pixels are considered in a fixed order. If the value of a neighbouring pixel is greater or equal, it will marked with either 1 or 0, resulting in an 8 bit binary number

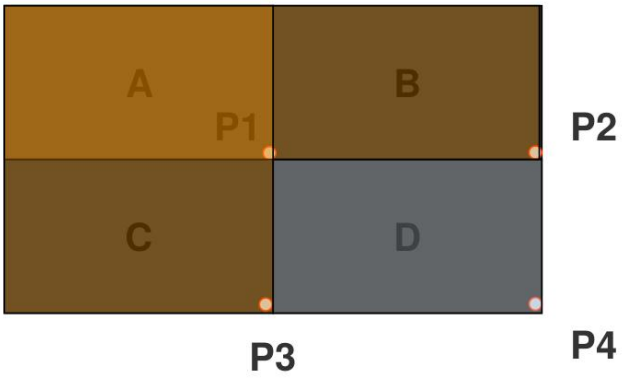


Fig. 10: Values of D are given by $P1$, $P2$, $P3$ and $P4$

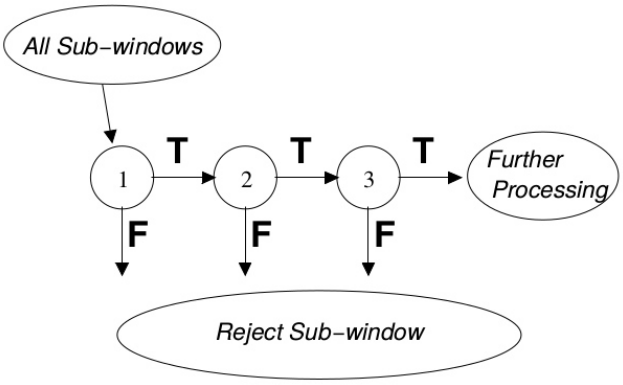


Fig. 11: Cascade of classifiers [11]

Fig. 13: Pixelwise-LBP of a car

(a) Positive (b) Negative

Fig. 14: Data set examples

as illustrated in Fig. 12 and 13. With the determined binary numbers, a histogram for each cell is created. All histograms together represent a LBP feature.

C. Data Set

Our image classifier is trained with images of vehicle rear views with about 1000 positive and 1500 negative examples with resolutions of either 64×64, 128×128, or 256×256 pixels. To obtain positive examples, we extracted vehicles from videos we recorded during test drives with our autonomous car as well as random vehicles from web search images. To obtain negative examples, we created clippings of streets, buildings, traffic signs, trees and other objects from our recorded videos (Fig. 14).

To evaluate our image classifier, about 150 gray-scaled images with a resolution of 750×500 pixels were used. The vehicle rear views were annotated each with two bounding boxes, whereby a small box was placed inside and another one well-spaced outside of the vehicle, see Fig. 15. The classified result must be around the small box as well as inside of the large one. Furthermore, in order to be more tolerant against mismatches due to partial occlusion or distant, and hence small, vehicles in the images, we specified for each candidate whether it needs to be detected or if detection is optional.

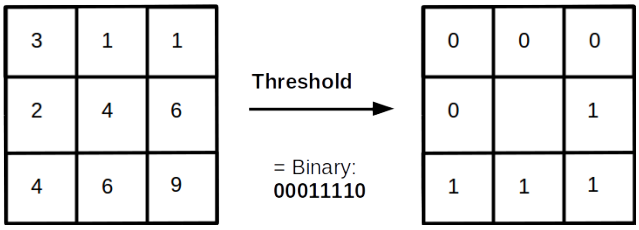


Fig. 12: LBP feature thresholding

D. Detection Module

The implemented vehicle detection module was integrated into our existing autonomous driving framework. An overview of the data flow is given in Fig. 16.

The module's input data are 16-bit gray-scaled images of the two on-board cameras with a resolution of 768×488 pixels each. Based on the images, the module creates a three dimensional pointcloud by a stereo vision submodule. 3D bounding boxes of objects are detected by clustering in fixed horizontal layers. The bounding boxes are projected to the two dimensional camera image afterwards. We then have a preselection of rectangular regions which may potentially contain vehicles. The advantage of this approach is that not the whole image has to be classified but only those areas where objects are detected - leading to higher frame rates and faster execution times. To ensure that the bounding boxes contain entire vehicles, the rectangular regions were increased by a constant percentage of 7%.

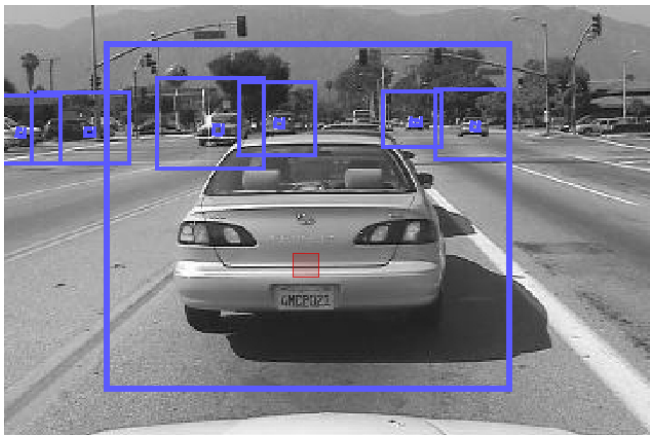


Fig. 15: Example image from test data set with annotations

In the next step of our vehicle detection module, the image classifiers are applied to the preselected regions. We use the HOG descriptor [3], DPM detector [5] and Cascade Classifier (with Haar-, LBP- and HOG-Features) [11], which are provided by the OpenCV software library. In the resulting image, the detected vehicles are marked by red bounding boxes as can be seen in Fig. 17. In addition, the vehicle distance in meters and the probability for a true positive are printed on top of the result rectangle.

IV. Experimental Results

All tests run on a system with an Intel i7-4710HQ 64-bit processor, 16 GB of RAM and an Nvidia GeForce GTX 860M graphics device.

We measured precision and recall for our image classifiers on the given test set, as shown in Fig. 18.

$$Recall = \frac{True\ Positives}{Overall\ Positives} \tag{6}$$

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \tag{7}$$

The HOG feature based approaches (blue, purple, orange and yellow line on the graph) yield the best results, especially when used in DPM detector. With LBP features (green lines), most of the vehicles can be found as well, but many false positives are detected too. Haar-like features (violet and brown line) performed worst, which may possibly have been due to a high variety of vehicle models and colors, preventing good matches for the clear and shaded areas of the pattern. Some implementations also provide swapping computations to the GPU. Because of different adjustable parameters, the results between GPU and CPU implementations of the algorithms vary to a certain degree.

In live tests with our autonomous car, the stereo vision clustering is able to reliably detect objects at distances up to 30 meters in real time (> 25 fps). On recorded video files, we could even run it with an average frame rate of 160 fps. Because we are using preselected areas of the image for our

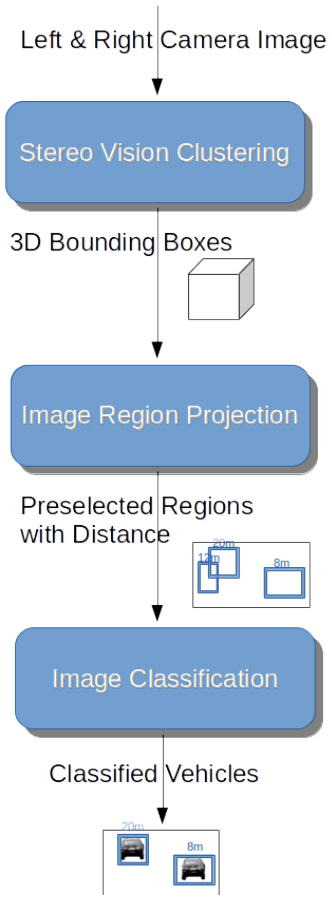


Fig. 16: Basic data flow of vehicle detection system

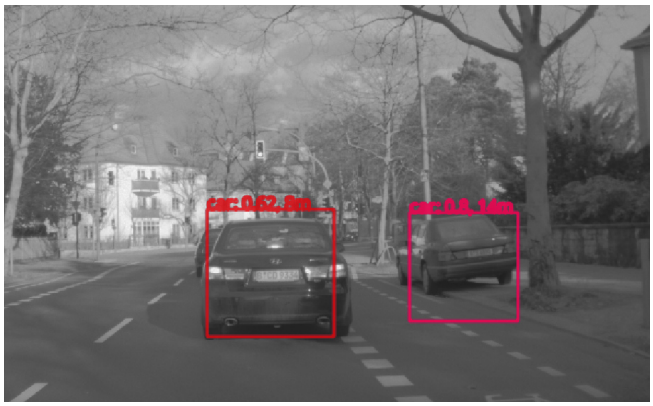


Fig. 17: Example result in live tests

classifiers, the frame rate of them was always also above 25 fps in our live tests. Running them on the recorded data, the frame rate varied from 30 to 100 fps. Since the GPU implementations were in the upper frame rate range, even we were able to scan with HOG / Haar-like feature based classifiers the whole camera image (768×488 pixels) in real time.

Furthermore, for a more precisely measurement of the performance of the different image classifiers, we classified our test set. The results are shown in Table I. All the 750×500 pixels of an image were given as the input of

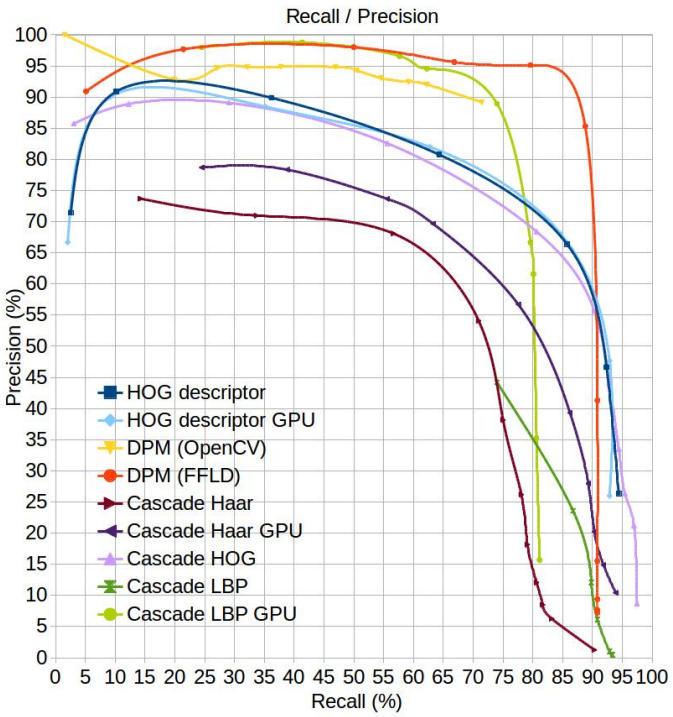


Fig. 18: Precision and recall of the applied image classifiers on our test data set

TABLE I: Performance results of the applied image classifiers on a test data set of 138 images

Algorithm	Runtime in sec.	Frames per sec.
HOG descriptor (OpenCV)	11.59	11.9
HOG descriptor (OpenCV GPU)	3.167	43.57436
DPM (OpenCV)	98.545	1.40038
DPM (FFLD)	23.062	5.98387
Cascade Haar (OpenCV)	25.249	5.46556
Cascade Haar (OpenCV GPU)	5.172	26.68213
Cascade HOG (OpenCV)	45.338	3.0438
Cascade LBP (OpenCV)	41.793	3.30199
Cascade LBP (OpenCV GPU)	8.465	16.30242

the classification algorithms leading to lower frame rates, since we did no preselection with stereo vision clustering. The GPU implementations reached significant higher frame rates here as well. The basic HOG descriptor performed best, followed by Cascade Haar classifier, Cascade LBP classifier and DPM detector.

V. Conclusion

In this paper, a reliable approach for image based vehicle detection and localization in the environment of our autonomous car was demonstrated. Stereo vision clustering was used to determine three dimensional vehicle coordinates. The system is structured in abstract object detection modules which provide the possibility to adapt it to other use cases such as the detection of pedestrians or traffic signs.

With the usage of the GPU for our applied pattern matching algorithms, our system is able to detect vehicles in realtime. In our experiments we determined the HOG descriptor as the best trade-off between computation time and detection quality. With the DPM detector, we could increase the quality but to the expense of performance.

In future work, higher detection rates might be possible by increasing the training set and camera image resolution, or by using newer computer vision algorithms such as neural network approaches for example.

References

[1] L. Cai, L. He, Y. Xu, Y. Zhao, and X. Yang. Multi-object detection and tracking by stereo vision. *Pattern Recogn.*, 43(12):4028–4041, Dec. 2010.

[2] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[4] R. K. Dewan and R. K. Barai. Fast kernel based object detection and tracking for stereo vision system.

[5] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, Sept. 2010.

[6] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*, CVPR '05, pages 807–814, Washington, DC, USA, 2005. IEEE Computer Society.

[7] K. Konolige. Small vision system: Hardware and implementation. In *Proc. of the Intl. Symp. of Robotics Research (ISRR*, pages 111–116, 1997.

[8] S. Kosov, T. Thormählen, and H.-P. Seidel. Accurate real-time disparity estimation with variational methods. In *Proceedings of the 5th International Symposium on Advances in Visual Computing: Part I*, ISVC '09, pages 796–807, Berlin, Heidelberg, 2009. Springer-Verlag.

[9] S. Moqqaddem, A. Sbihi, R. Touahni, and Y. Ruichek. *Objects Detection and Tracking Using Points Cloud Reconstructed from Linear Stereo Vision*. INTECH Open Access Publisher, 2012.

[10] T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proc. 12th International Conference on Pattern Recognition (ICPR 1994), Jerusalem, Israel*, volume 1, pages 582–585. IEEE, 1994.

[11] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.