https://doi.org/10.2352/ISSN.2470-1173.2017.4.SRV-350

A combined HOG and deep convolution network cascade for pedestrian detection

Yuriy Lipetski, Oliver Sidla; SLR Engineering GmbH; Graz, Austria

Abstract

For the analysis of the interaction patterns of traffic participants, a robust visual detector and tracker for pedestrians and vehicles has been developed. The resulting implementation is currently being used to analyze hundreds of hours of recorded videos. This work concentrates on the detector for pedestrians, which combines several key concepts into a processing framework, which can run close to real-time even without GPU acceleration: a fast and efficient HOG detector cascade is combined with a deep convolutional network to combine the advantages of both algorithms. In addition to the detector, this work covers also aspects of camera calibration, which is used to control the scale of detection windows based on the viewing geometry.

The evaluation of our detector on the CALTECH database as well as on real world ground truth videos and manually annotated sample data demonstrates the effectiveness of our approach.

Introduction

Visual based object detection is a classic task in computer vision. Foremost, the detection of humans and vehicles has a large number of practical applications – both in video surveillance and intelligent transportation systems. The state-of-the-art of visual object detection has made great progress over the last years with regard to accuracy and efficiency of implementation [16]. Nevertheless, practically all methods i) tend to have problems in difficult scenarios (dense crowds, poor image quality etc.), and ii) they are still too slow for many real-world applications. In contradiction to i), and ii), many applications have extremely strict demands in terms of acceptable false positive rate and number of missed detections. For example, Zhang et al in [17] find that human performance in terms of miss rate is still an order of magnitude better than the best available computer vision methods. Thus, further improvement of state-of-the-art of detection quality and performance is required, and a goal worth pursuing.

The work presented here emerged from the need for an accurate evaluation of hundreds of hours of outdoor videos, which were used as data source for automated traffic analysis. To this end a visual pedestrian detector and tracker has been developed, which needs to be i) fast, so that processing time remains manageable, and ii) accurate enough to allow for the solid analysis of tracking data for quantitative pedestrian-vehicle interaction analysis.

Cascaded HOG detectors [1][5] and deep convolutional networks [7] are both very powerful approaches, and each of them has its advantages and drawbacks – the merging of these two methods into one detector can potentially be beneficial. Our work presents the result of merging the speed of HOG with its fairly good detection rate, and a deep convolutional network. The resulting combined detector is fast and robust enough so that the automatic and accurate evaluation of the large amount of traffic video for our intended application is possible. The novelty of our approach lies

1. in the method for training of the primary HOG detector cascade, which is practically unique,
2. in the approach of using genetic optimization methods to select the optimal HOG features,
3. in the combination of a HOG cascade with a deep learning detector as post-processing filter

The performance of our detector is evaluated on a large corpus of approximately 500,000 manually annotated samples. In addition, we present results of the evaluation of the detector on the public Caltech dataset [13], which has been introduced in 2012 by P. Dollár. The size of the video corpus, which we have processed in the course of several projects so far, is considerable.

Related Work

Beginning with Viola et al. [21] the concept of detector cascades has proven to be extremely worthwhile. The simple Haar like features used by Viola et al. have been superseded in the following years. The most significant progress in terms of detection capability has been made with the introduction of the Histograms of Oriented Gradients (HOG) idea by Dalal and Triggs in 2005 [1]. HOG features have since then set the quality standard for rigid object detection for the coming years, and with modifications, the lifespan of HOG does not seem to be over yet [4].

Felzenszwalb introduced a method of training model parts and creating a combined classifier [10] in 2010. It has gained some popularity, but is being used less in recent years. One reason may be that the classifier is complex to implement, and hard to optimize in terms of runtime performance.

The development of the channel feature concept by Dollar et al. [2] in 2009 has again dramatically improved detection performance as well as detection efficiency. Runtime performance improvements have been achieved by exploiting the redundancies in local image structure using soft cascades [6], in a work done by Bourdev et al. The basic channel feature approach has been improved upon by Zhang et al. [17] by combining low-level filter banks in an intermediate processing layer with boosted decision trees. Zhang et al. report excellent results on the Caltech dataset, and other variants of the channel feature approach still deliver top results on the Caltech benchmark. [20]

By selecting HOG features for detection with a shallow cascade of detectors, Lipetski et al. [22], see also [5] for a similar concept, have shown that a robust and real-time capable object detector can be trained using genetic optimization methods. This is achieved by using simpler, but less discriminative HOG features in the primary cascades stages, and by testing the more complex and computationally demanding HOG descriptors in the final stages of the cascade.

Although a training algorithm using genetic optimization to create HOG cascades is not faster or requiring less CPU power than AdaBoost (the training of a cascade can take days to complete), it seems to be advantageous because it better exploits the vast high dimensional space covered by the HOG features.

Some work on the use of genetic algorithms for the training of HOG classifiers has been reported in the past, see Zehang [9], but typically feature selection is done with AdaBoost and one of its many variations. This work tries to demonstrate that evolutionary computation can be brought to advantage for the design of shallow cascaded detectors, which have similar, if not better, qualities than detectors trained with AdaBoost.

The field of deep learning/neural networks (NN) is currently a very active field of research [7], [11]. Especially the sub class of convolutional neural networks (CNN) developed into a success story [8][12], [18]. Explicit pedestrian detection is in many cases based on existing network architectures, which are refined using pedestrian specific training datasets. Today deep learning detectors achieve top results on the Caltech benchmark [19].

Methodology of Pedestrian Detection

The detector presented in this work uses a HOG detector and a CNN detector combined into a shallow cascade. Each of the two classifiers has its advantages and drawbacks. Over the years HOG has recommended itself as a detector, which is fast to compute and fairly robust to illumination changes, image quality and partial occlusions. The core idea of our work is to merge the two detection concepts into one framework: we localize object candidates with a fast HOG cascade and then verify each candidate with a trained CNN. The threshold for the HOG detector is set to a relatively low value so that we can detect practically all objects of interest within an image. The task of the CNN then is to weed out the false positives. In this way, the slow CNN classification is applied only on a very small amount of detection candidates instead of many thousands classifications necessary for full image detection with a sliding window approach.

Detection Pipeline

Different types of descriptors have their own unique properties of computational complexity, locality, invariance to geometric transformations, etc. By combining descriptors into cascades, a balance between detection accuracy and computational resources can be achieved.

Using a lightweight and fast HOG detector as the first stage in a cascade significantly reduces processing time of the overall detector, because the majority of the background samples can be rejected early. The following stages of the detector can then have a more complicate structure because they need to test much fewer samples.

Even though a fine tuned HOG detector can provide good detection capabilities, it has its limits. We estimate that the useful number of training samples for HOG lies at about 50,000 samples. After that the classification performance of HOG cannot be improved further, the authors of this work even see that the introduction of too many background samples can reduce the quality of the trained HOG detector.

On the contrary, deep CNNs show unlimited training potential with the sample numbers currently in use. In almost each modern detection benchmark, deep network based algorithms show by far superior detection results. The drawback of CNNs still is the relatively slow offline training and relatively slow online processing time, especially when used on mobile or embedded systems with limited hardware capabilities. The best quality is usually achieved with the largest networks, which have millions of connections in total.

In order to be able to optimize the use CNNs, we verify the result of the last HOG detector stage with a CNN only. In this way a tiny fraction of samples needs to be processed, thereby saving CPU time.

To make the detection practically useful, scale invariance is achieved by testing sample windows of different sizes at every image location. We use a scanning step of (0.07-0.10)*sample width, and a scale factor of 1.07 for 4-30 scales, depending on the application and imaging geometry. When available, geometric calibration information on a known ground plane can improve detection by limiting the scale range. The following Figure 1 depicts our detection pipeline.

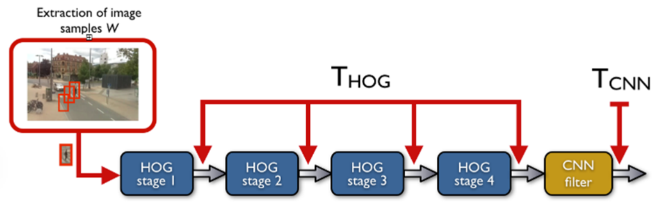


Figure 1. The structure of the proposed HOG/CNN detector cascade. Guided by calibration, overlapping sample windows W are extracted for each video frame. The 4-stage HOG cascade pre-selects pedestrians using acceptance threshold T_{HOG} ,the final decision is made with a CNN using a decision threshold T_{CNN} .

HOG Detector Cascade

Dalal and Triggs [1] introduced the HOG feature descriptor concept in 2005, which is conceptually somewhat similar to SIFT and SURF, but is easier to compute and therefore requires less computational power. Until the introduction of channel features by Dollar, the HOG descriptor represented a very good baseline in terms of feature creation, sample classification and detection performance.

In our implementation HOG features for a sample window W are computed and classified using the following algorithm:

1. Divide W into non-overlapping $N\times M$ square cells of size $C\times C$ pixels ($C = 6$, 8 typically)
2. Compute the L2 normalized histogram H_c of edge orientations, for each cell, the orientation bins for the histograms are 22.5 degrees wide, so that each cell histogram has 8 bins
3. Threshold and saturate H_c , a typical threshold value is 0.2
4. Combine 2×2 neighboring cell histograms H_c into a block histogram B and normalize B using L2 norm
5. Combine all block histograms B into a feature vector FR, and normalize it using L2 norm
6. Classify the resulting feature vector FR using a linear SVM

The concept of integral histograms to speed up the creation of local edge orientation histograms in overlapping windows makes HOG real-time capable. The computation of edge orientation histograms is done using quantized edge orientation channels OH for each possible orientation bin. The 0-180 degree raw edge orientations from the Sobel operator are for this purpose quantized into the typical 8 HOG orientation bins OH. The computation of an integral histogram IO from an edge orientation map E is done as follows:

For each orientation channel O:

$$IO(x, y) = IO(x-1, y) + IO(x-1, y) - IO(x-1, y-1) + MO(E(x, y)) \quad (1)$$

MO() is a function which maps an edge orientation value *E(x,y)* to a discretized edge histogram entry which can be added to the summed edge histogram. This map can be a lookup table for computational efficiency.

The Histogram of Gradients for a rectangular region at image coordinates from *L(xl, yl)* (left upper point) to *R(xr, yr)* (right lower point) can then be computed from the integral histogram as follows:

For each orientation channel O:

$$HO = IO(xl, yl) + IO(xr, yr) - IO(xr, yl) - IO(xl, yr) \quad (2)$$

By combining integral histograms with low level program optimizations to make use of modern CPU instructions (SIMD instructions like SSE for x86 architectures, and NEON for ARM CPUs) HOG features can be computed an order of a magnitude faster compared to conventional implementations. As can be seen from Equation (2), a local histogram can be computed using 4×(Histogram Size) memory accesses only and the remaining normalizing and scaling operation of the feature vectors can be efficiently parallelized using SIMD vector instruction techniques.

Scale invariance of the HOG feature descriptor is achieved by either scaling the original input image and running the detector iteratively with identical cell/block size. Alternatively, the HOG detection window and its corresponding cells/blocks can be scaled and tested, with the advantage that the integral histogram makes this approach especially easy to apply. Only the sample co-ordinates in the integral histogram image HO from Equation (2) need to be modified in order to obtain the scaled HOG features.

The resulting high dimensional normalized HOG feature vector is classified using a linear SVM [3], which again is very well suited for optimization using SIMD programming techniques.

When using HOG, the question remains, which cells resp. blocks within a sample window *W* should be used to obtain the maximum discriminative power with the least number of features. To answer this question, we have devised a variant of a genetic optimization procedure, which tries to solve this selection process for a given number of pre-determined features per sample window. This process is described in the next section.

Feature Optimization using Genetic Algorithms

The best state-of-the-art HOG detectors employ deep cascades which are practically always trained using AdaBoost (see Dollar [2] for a list of HOG pedestrian detection architectures). AdaBoost and its variants have been very successfully used by Viola et al. in combination with Haar like features. Our method uses a completely different approach - we propose the use of a genetic optimization procedure (GA) instead. Our reasoning is that the GA is better able to sample the high dimensional space represented by HOG features, and so can more reliably find generalized robust solutions. The following paragraphs describe our approach in detail.

The genetic optimization algorithm, we call our version *genopt*, is an iterative procedure. To make the selection process easier to parametrize and to speed up computation, we define a set number of stages, 4 in our application, and a fixed number of cells for each detector stage. It would be possible to also consider both for optimization, but the resulting computation times are prohibitive.

The GA algorithm, see also [14], tries to optimize the properties of a set of classifiers by treating them as population with individual genomes (*the HOG features*), which undergo a test of fitness (*classifier accuracy*) and reproduction (mixing of HOG features to create new ones) and mutation (random changes in the features to explore the feature space). By mating (merging) the fittest individuals and by removing lesser ones, the overall fitness (accuracy) of the classifier population should increase over time. The optimization procedure works as follows:

1. *Initialization*: each initially random set of HOG classifiers for all stages make up an 'individual' with a specific *genome*.
2. For every generation the quality (fitness function *F*) of each individual is evaluated on a test set.
3. The best fitting individuals are selected for mating, they 'survive' the selection process with regard to good classification results on the test database. They 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*.
4. In order to allow for variation in the data set, mutation can change an individual HOG descriptor randomly in every generation with a certain given likelihood *L*. This mutation frequency defines how fast the procedure can converge to a good solution, and how aggressive the feature space is being samples.

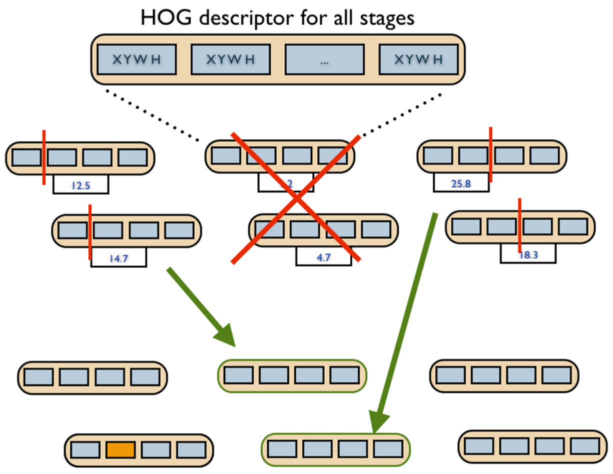


Figure 3. The GA algorithm depicted from top to bottom. HOG cells, denoted by co-ordinates x,y and their dimension w,h (top) are tested and selected based on their fitness function. The best individuals (classifiers) are chosen and mated (middle) to create new classifier individuals, possibly also being modified by random mutation (last bottom).

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.

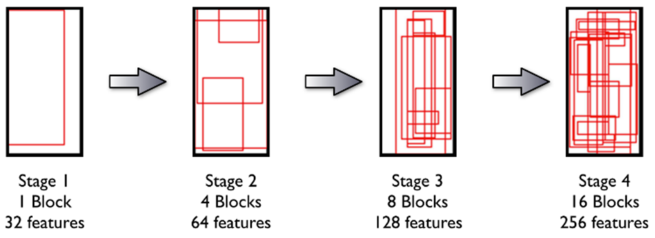


Figure 4. The optimized HOG detector cells for each cascade stage.

For the training process, which can run for several days, we have used 47000 samples in total, of which 20000 are background samples, and 27000 are foreground samples. The sample database has been manually annotated and double checked.

The algorithm checks every cascade stage against stage specific 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 albeit 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, this is very similar to the soft cascade approach from Bourdev [6]).

The resulting cascaded detector is very well suited for very fast and relatively accurate (although not state-of-the-art) localizing of objects within the image – even the highly overlapped full scanning of an entire HD image (sometimes up to 1,000,000 of HOG detections and SVM classifications) can be done close to real-time.

The following table below shows the rejection rate for each of the HOG classifier stages. We keep the detection thresholds such that the true positive rate is as high as possible so that no pedestrians are missed in the next following verifying/filter stage, which is described in the next section.

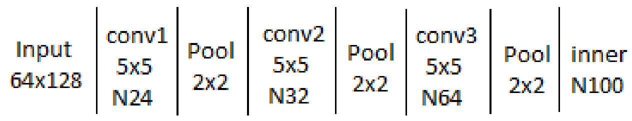
Typical values for the number of samples windows W for an image used in our traffic surveillance applications. The rejection rates for each stage show how effective the cascade can reduce processing times.

HOG stage	Number of HOG blocks	Feature vector length	Candidates left
1	1	32	17.58%
2	4	128	3.43%
3	8	256	0.35%
4	16	512	0.12%
5	fully connected	5760	0.07%

Adding the Deep Learning Stage

Although a well-trained HOG detector can pass as a good baseline detector, it is not to be considered as state-of-the-art anymore. We observe in our applications resp. data sets that HOG creates a relatively high number of false positives, which cannot be removed by using more training samples. In fact the introduction of too many background samples can be counterproductive and decrease the quality of a HOG detector.

A solution to this problem is the introduction of a further detector stage after the primary cascade. We apply a deep convolutional detector as the final detector stage. Our convolutional neural network (CNN) contains 3 convolutional layers with kernel size equal to 5x5 in all layers. The full network structure is depicted in Figure 5.



Input 64x128	conv1 5x5 N24	Pool 2x2	conv2 5x5 N32	Pool 2x2	conv3 5x5 N64	Pool 2x2	inner N100

Figure 5. The structure of our CNN detector. N is the number of feature maps for convolutional layers, and resp. the number of neurons for inner product layer.

The specific feature of deep networks is that they can be successfully learned from practically unlimited number of training samples. To get full usage of the deep network approach, training set has to be as big and as various as possible. We enhanced our initial training set with many new samples – most of them are false positives that were produced by the HOG cascade. Some typical examples are shown in Figure 6.

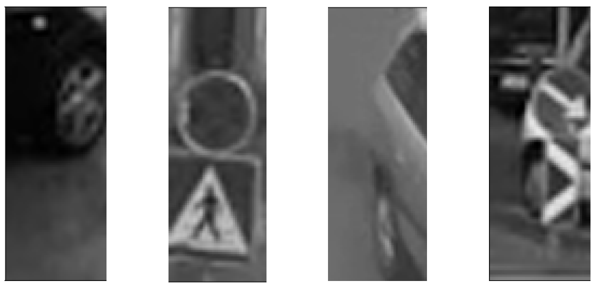


Figure 6. Typical false positives of the HOG detector, which we added to the training set of the CNN detector.

The resulting annotated database has grown to around 510,000 samples in total. This database was randomly split to train and test sets having *70%* and *30%* of the whole data respectively. After the training process has finished, the detector showed *99.4%* recognition performance on the test set.

Evaluation of the Detector

The proposed detector has been integrated into our tracking system which goal was to automatically analyze interaction behavior between different groups of road users. Various statistics were generated afterwards using object trajectories data, such as distances and velocities during overtakes, potentially dangerous situations on crosswalks etc. The whole set of road users includes i) pedestrians, ii) cyclists, iii) motorcyclists, iv) cars, v) trucks, vi) busses and vii) trams. Many hours of recorded videos for various outdoor scenarios were processed offline within two different research projects. In this work, we evaluate only the pedestrian detector. The following table gives an overview of the amount of video, which we have processed in the course of two projects.

Amount of total processed video data.

Project name	Total places	Total video hours	Resolution (px)	Pedestrian detection time (ms)
Observe	86	3918	800 x 600	57.8
Bicycle	10	332	1920 x 1080	150.4

To be able to handle such huge amount of data in a feasible time, the runtime performance of the detection algorithm plays one of the core roles. The presented detection times were obtained on Intel Core i7-6700K CPU. The detector runs on 4 cores in parallel; no GPU has been used so far. Using also the GPU for computing features and the classifiers, would certainly decrease the processing times significantly.

The following table represents detection statistics and processing times for typical scenes for each video resolution.

Detection statistics for typical video resolutions.

Project	Resolution (px)	Sliding windows per frame	HOG candid. per frame	CNN candid. per frame
Observe	800 x 600	212 500	13.9	1.2
Bicycle	1920 x 1080	920 000	121.4	37.0

As it can be seen, our HOG detector rejects more than 99.9% of all sliding windows. Only the remaining tiny part has to be verified with a computationally heavy CNN detector. The CNN detector further rejects about 91% and 70% of HOG candidates for *Observe* and *Bicycle* videos respectively. The relatively small number of output candidates for the *Observe* video can be explained by the nature of the chosen scenario – the focus has been taken to the busy road junctions with a heavy car traffic and relatively small pedestrian traffic.

To evaluate detector performance, 20 minutes of one video sequence has been manually annotated. The detector performance and detection examples can be seen on Figures 7 and 8.

In the example shown in Figure 8, the combined HOG + CNN detector has filtered out both false positive detections. There is also one missed detection – it cannot be recovered by the CNN stage anymore.

To compare our implementation with other state-of-the art detectors, we performed the detector evaluation on the Caltech dataset. The Caltech dataset [13] is a large corpus of video material, which has been captured in VGA resolution from through the front window of a moving car. The original test set, which has been used for our evaluation, contains 4024 video frames with about 1000 annotated pedestrians. The videos contain samples of medium and good quality, but also many hardly visible pedestrians in low resolution and low contrast. The benchmark therefore is divided into several categories with varying levels of difficulty. Our evaluation falls into the category "medium scale" meaning that only pedestrians, which have more than 50 pixels in height, were considered. Those pedestrians could also have a certain amount of occlusion.

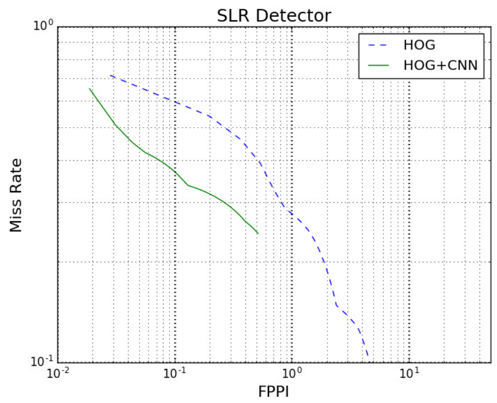
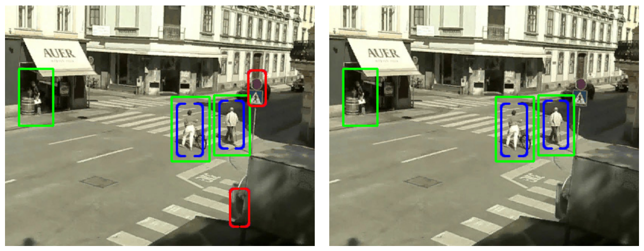


Figure 7. Detector performance on the Observe dataset. The blue dashed line represents the HOG detector alone, the green solid line represents the combined HOG/CNN detector. The latter is almost one order of magnitude better in terms of FPPI for a given miss rate.



a) HOG detector b) HOG + CNN detector

Figure 8. Visualization of the detector outputs. Rectangles represent the ground truth, and the brackets the detector outputs.

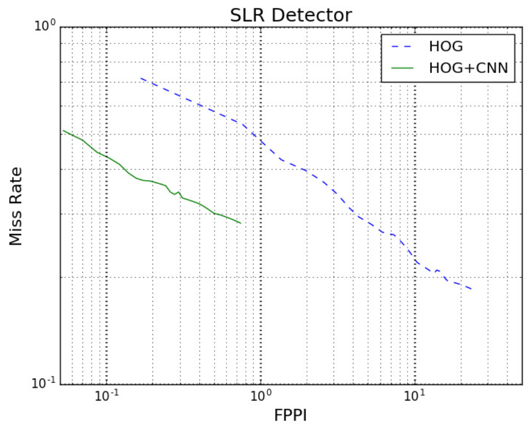


Figure 9. Detector performances on the Caltech dataset. The blue dashed line represents the HOG detector alone, the green solid line represents the combined HOG/CNN detector. The latter is almost one order of magnitude better in terms of FPPI for a given miss rate.

Again, the combined HOG+CNN detector shows significantly better performance than the pure HOG detector. One can note that the combined detector has its limit in achieving lower miss rate – this is because the CNN threshold is set to a fixed value of 0.0. The presented detector has slightly worse performance then the best pedestrian detectors shown in [13]. However, the average processing time of our detector is just 40.8 ms per frame (measured on an Intel Core i7-6700K CPU), meaning that the detector can operate fully real-time – even without GPU utilization.

Calibrating the Detection Results

To support the detection process and to be able to create useful measurements from the detection results, a geometric calibration of the camera view is undertaken. By referring to an expected size range for each object class (e.g. the known size range of pedestrians) in an image, we can limit the detector size window to reasonable values and thus save computing time and enhance the detection accuracy. In this sense, we see the camera calibration as integral part of our detection pipeline. Furthermore, the calibration will be needed for the analytics after detection (e.g. the automatic measurement of distances and velocities), and can therefore be reused.

The calibration process is based on Tsai's method of lens camera calibration [15], which uses a set of corresponding point sets from camera to ground plane and basic information about the used sensor in order to establish the following camera parameters: the focal length *f*, a radial lens distortion coefficient *k*, co-ordinates of the center of the lens distortion *Cx, Cy*, the translation of the camera between camera and world co-ordinates *Tx, Ty, Tz*, and the rotation of the camera to transform between camera and world co-ordinates *Rx, Ry, Rz*.

The computed camera calibration allows us to project every image point to its corresponding world point on the ground plane and vice versa – as mentioned before, this is a pre-requisite for all following traffic analysis measurements.

Due to the monocular setup, a calibration can only take place on one elevation above the ground plane, which is typically the street level. For calibration using Tsai's method it is necessary to manually establish ground control points in metric co-ordinates in an arbitrary co-ordinate system and relate them to the corresponding image points. The use of available satellite image data for this purpose has proven feasible, but can meet limitations when the imagery is not up to date, or the quality of the satellite images is not sufficient for finding usable ground control points.

Depending on the camera resolution and the distance of the object to the camera, measurement accuracies of a few centimeters are possible using the proposed calibration method. This is generally sufficient for our type of applications. Error sources are mainly

- Errors in object detection resp. the object bounding boxes. These errors can be reduced in post-processing by smoothing the trajectory of object bounding boxes.
- Errors in projection of the bounding boxes to the ground plane, these are essentially also detection inaccuracies.
- Camera shake and movement. Small amounts of vertical displacement of the camera viewing angle can create large distance measurement errors.
- Inaccuracies in the measurement of the ground control points and image points used for calibration can lead to imprecise or distorted mapping of the ground plane to image co-ordinates.

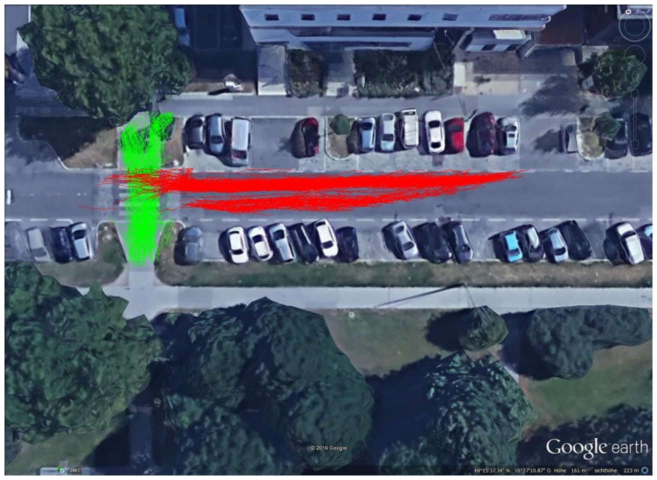


Figure 10: Trajectories of pedestrians and cars projected onto a Google Maps section for visualization of detection and calibration results. The co-ordinates of detected objects have been transformed into the Google maps image using the calculated Tsai calibration. red lines indicate cars, green lines indicate pedestrians.

References

[1] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", in Conference on Computer Vision and Pattern Recognition, pp. 886-893, San Diego, 2005.

[2] P. Dollár, Z. Tu, P. Perona and S. Belongie,"Integral Channel Features", BMVC, London, 2009.

[3] N. Cristianini and J. Shawe-Taylor, "An Introduction to Support Vector Machines and other kernel-based Learning methods", Cambridge University Press, 2000.

[4] H.K. Ragb and V.K. Asari, "Histogram of oriented phase (HOP): a new descriptor based on phase congruency ", in Proc. SPIE 9869, 2016.

[5] Q. Zhu, S. Avidan, M.-C. Yeh and K.-T. Cheng, "Fast Human Detection Using a Cascade of Histograms of Gradients", Conference on Computer Vision and Pattern Recognition, New York, 2006.

[6] L. Bourdev and J. Brandt, "Robust object detection via soft cascade", in IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 236-243 vol. 2, 2005.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks", in Advances in Neural Information Processing Systems 25, pp. 1106-1114, 2012.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions", in *arXiv*, 2014.

[9] S., Zehang, G., Bebis, R., Miller, "On-Road Vehicle Detection using Evolutionary Gabor Filter Optimization", IEEE Transactions on Intelligent Transportation Systems, Vol 6, Nr 2, 2005.

[10] R. Felzenszwalb, D. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 32, No 9, pp 1627-1645, 2010.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", Proceedings of the IEEE, November 1998.

[12] P. O. Pinheiro, T.Y. Lin, R. Collobert, P. Dollàr, "SharpMask: Learning to Refine Object Segments", in European Conference on Computer Vision, 2016.

[13] https://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians/

[14] http://lancet.mit.edu/ga/

[15] R.Y. Tsai, "An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision",.in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Miami Beach, 1986.

[16] R. Benenson and M. Omran and J. Hosang and B. Schiele, "Ten years of pedestrian detection, what have we learned?", in European Conference on Computer Vision, CVRSUAD workshop, 2014.

[17] S. Zhang and R. Benenson and B. Schiele, "Filtered channel features for pedestrian detection", in Conference on Computer Vision and Pattern Recognition, 2015.

[18] J. Hosang and M. Omran and R. Benenson and B. Schiele, "Taking a deeper look at pedestrians", in in Conference on Computer Vision and Pattern Recognition, 2015.

[19] S. Zhang and R. Benenson and M. Omran and J. Hosang and B. Schiele, "How Far are We from Solving Pedestrian Detection?", in Conference on Computer Vision and Pattern Recognition,, 2016.

[20] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art", TPAMI, 2011.

[21] P. Viola and M. Jones,"Robust Real-time Object Detection", International Journal of Computer Vision, 2001.

[22] Y. Lipetski, G. Loibner, O. Sidla. "Close to real-time robust pedestrian detection and tracking". In Proceedings SPIE 9407, Video Surveillance and Transportation Imaging Applications, 2015.

Author Biographies

Yuriy Lipetski *attained his master degree in telematics at the National Technical University "Kyiv Polytechnic Institute" in 2002. Since then he works as a researcher in the computer science field. His main focus is i) development and implementation of object detection and tracking algorithms, and ii) development of an automatic number plate reading system. Since 2011 Yuriy Lipetski is senior researcher in SLR Engineering.*

Oliver Sidla *attained his master degree in computer science at the Technical University of Graz in 1991. Until 2007 he worked as a researcher and project manager in Graz. His main focus was the development of machine vision systems for industrial applications, and from 2002 on, development of algorithms and systems for object tracking and detection. Furthermore Oliver Sidla has experience in the development of classification algorithms, feature descriptors, especially the HOG detector, and their integration into real-time systems.*

In 2008 Oliver Sidla founded the company SLR Engineering, which is based in Graz, Austria. The main focus of SLR Engineering is i) the implementation of machine vision systems for industrial inspection and ITS systems, and ii) the design and implementation of object detection strategies which are suited for embedded systems and smart cameras.