Robust traffic state estimation on smart cameras

Felix Pletzer, Roland Tusch, Laszlo Böszörmenyi, Bernhard Rinner
Alpen-Adria-Universität Klagenfurt and Lakeside Labs
Klagenfurt, Austria
{felix.pletzer, roland.tusch, laszlo.boeszoermenyi, bernhard.rinner}@aau.at

Abstract

This paper presents a novel method for video-based traffic state detection on motorways performed on smart cameras. Camera calibration parameters are obtained from the known length of lane markings. Mean traffic speed is estimated from Kanade-Lucas-Tomasi (KLT) optical flow method using a robust outlier detection. Traffic density is estimated using a robust statistical counting method. Our method has been implemented on an embedded smart camera and evaluated under different road and illumination conditions. It achieves a detection rate of more than 95% for stationary traffic.

1. Introduction

Traffic information systems aim to provide reliable and fast traffic messages to advise drivers against obstructions, such as impeded road conditions and congestions. To generate traffic messages, different data sources such as drivers' reports, traffic sensors, or travel-times recorded by toll systems [12] can be used. Moreover, traffic operators employ video surveillance to recognize interesting traffic situations. Automatic video analysis is mainly used to detect interesting situations such as traffic jams or impeded road conditions. The video data can either be analyzed centralized on a remote server or decentralized on smart cameras or roadside units. Many motorway operators maintain a huge installation of simple surveillance cameras that provide video streams to remote servers, where video analysis can be applied without additional road side installations. However, centralized solutions have the drawback of high communication requirements and low scalability. In contrast, decentralized solutions perform local analysis of the captured video data and streams are only transmitted for verification if an interesting traffic situation has been detected.

Smart cameras integrate video sensing, video processing, and communication in a single device [3]. The acquired video data can be processed onboard in real-time. If the smart camera has detected an interesting traffic state, e.g., stationary traffic, the smart camera delivers an event description and a video sequence of the event as visual proof.

In this paper we present a fast and accurate method for estimating the average traffic speed and traffic density on the motorway for smart cameras. In contrast to most other methods, we use feature statistics to estimate the mean speed and traffic density. The mean speed is obtained from optical flow analysis and the traffic density is calculated from edge information. Speed and traffic density values are combined to compute the so-called *level of service* (LOS) that describes the prevailing traffic state. The presented method has been implemented on a smart camera and evaluated under real-world conditions.

The sequel of the paper is organized as follows. Section 2 provides a short overview of the background and related work. Section 3 presents the system architecture of the traffic state detector. In section 4, our video-based traffic state detection algorithm is described, including the camera calibration and the video-based estimation of traffic speed and density. Section 5 presents the evaluation results of our method using a 156 hours video data set. Finally, section 6 concludes this paper.

2. Background and related work

Traffic state detection in the uncompressed and compressed video domains has been a well-studied research area for more than ten years. Traffic speed and density estimation is typically based on vehicle tracking or vehicle detection. Vehicle tracking either relies on motion analysis using background models or on feature tracking. However, despite significant research efforts, accurate real-time background modeling for embedded computer vision is still a major challenge.

Segmenting moving foreground objects from the background is the main approach used in background modeling. In [14] and [16], background models have been proposed which adapt to changing light and weather conditions. Applying frame differencing with probability density functions to estimate the segmentation of background objects is proposed in [6].

Feature-based tracking methods typically employ corner features for vehicle tracking. In [1], an algorithm employing Kalman filtering and correlation testing for tracking the features is described. A grouping module groups the features in order to segment the individual vehicles. In [9], the authors describe a method for classifying the traffic state based on optical flow-based motion features and edge-based density features. The proposed method uses a Gaussian radial basis function network for traffic state classification and is also designed to run on smart cameras in real-time. The method reaches an average classification accuracy of 86.2%, but has not been evaluated under different weather conditions. Employing texture and edge features for traffic density estimation is proposed in [5]. Using a 21-dimensions feature, Hidden Markov Models are trained to estimate the traffic density state. The method was evaluated for different weather conditions and shows an average accuracy of 95.6%. However, this method estimates only the traffic density and is not designed for real-time execution on smart cameras.

Vehicle trackers utilizing vehicle detectors have shown promising results. In [11, 10], the authors employ classifier grids with adaptive online learning for detecting cars. The feature-based KLT algorithm is used to calculate the velocity of vehicles. In [13] the authors discuss the principal component analysis (PCA) and histogram of gradients (HoG) approaches for vehicle detection. However, the detection rate of these approaches is usually affected by occlusions and difficult weather conditions. Therefore, in [2] multiple sensor data is exploited and classification rates are improved by co-training.

3. System overview

Our video analysis method is designed to run on a smart camera. Our smart camera includes a 1280x1024 (SXGA) color CCD sensor that is connected to an embedded processor board. The processor board is equipped with an Intel Atom Z 530, 1.6 GHz processor, 1024 MB RAM and a 32 GB solid state disk for internal storage. The processor runs a standard Linux distribution that provides flexible integration of additional hardware components and external software libraries. The smart camera is connected to the video network through a Gigabit Ethernet interface. Every time the camera detects a change in traffic state, it uses a web service interface to send an event description and the associated video sequence to the back-end traffic information system.

Traffic state detection is carried out using the video analysis method described in section 4. Figure 1 shows the data flow of the video-based traffic state detection that is implemented on the smart camera. The image sensor acts as a data source providing the raw video data with a frame rate of 19 fps. The captured frames are written to the video ring buffer and passed to the video analysis module. For each frame, the video analysis module computes the estimated speed and traffic density values and stores them in the statistics buffer. At periodic intervals, e.g. every minute, the data aggregation module reads the data from the statistics buffer and computes the current mean traffic speed and density estimates. These traffic speed and density estimates are used to calculate the prevailing level of service (LOS).

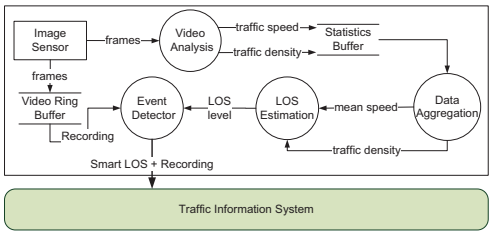

Figure 1. Data flow of the traffic state detection.

Table 1. National traffic state classes for a single lane[4].

Level	1 Lane	
	Mean speed (km/h)	Density (vehicles/km)
1 (free flow)	[80,∞)	[0,20]
2 (heavy)	[80,∞)	(20,50]
3 (queuing)	[30,80)	[0,50]
4 (stationary)	[0,30)	(50,∞)

Figure 2. Calibration points (left) and 1-by-1 meter grid, computed from obtained camera parameters (right)

LOS is a qualitative measure that describes the operational conditions of a segment or traffic stream. According to [4], four LOS classes are defined: free-flow, heavy, queuing, and stationary traffic. Table 1 shows the LOS definition for a single lane, depending on the measured mean speed and traffic density.

4. Video-based traffic state detection

4.1. Camera calibration

For our video analysis method described in sections 4.2 and 4.3, we use the length of motion vectors for speed estimation. To model the perspective projection of the camera from 3D world coordinates to 2D image coordinates, we use the well known pinhole camera model [7] in conjunction with a polynomial lens distortion model. Camera parameters are obtained from known point correspondences using Zhang's calibration method. The point correspondences (calibration points) are obtained from known distances of lane markings, as well as lane length and width. Figure 2 (left) depicts the calibration points used for calibration of a

Figure 3. Vehicle in ROI with KLT motion vectors (left) and KLT feature matching (right)

Figure 4. Incorrect KLT feature match (green line)

smart camera, which is located on a national motorway. The y-axis of the world coordinate system is chosen in direction of the lane markers (i.e., towards driving direction). Figure 2 (right) shows a 1-by-1 meter grid that was generated from the computed camera parameters.

4.2. Speed estimation

The mean speed is estimated from the length of motion vectors and the frame rate of the image sensor. To obtain the motion vectors, we use the Kanade-Lucas-Tomasi (KLT) [8] feature tracking method. Using corner detection, the KLT method selects a number of distinctive image points $p_{i,j}$ and searches for the corresponding matched feature points $\acute{p}_{i,j}$ in the subsequent frame. Using a feature pair $(p_{i,j}, \acute{p}_{i,j})$, we define the j-th motion vector from frame i to frame $i+1$ as:

$$\vec{m}_{i,j} = \overrightarrow{p_{i,j}, p_{i,j}^{\acute{}}} \tag{1}$$

Figure 3 shows an example for the matched features and the corresponding motion vectors. Although KLT feature tracking is considered relatively robust, incorrect feature matches (outliers) occur. To avoid outliers disrupting the speed estimations, we apply a cascaded outlier detection. Features points are only computed within a predefined ROI (region of interest) and cannot be tracked outside the ROI. Since appearing and disappearing feature points often lead to incorrect feature matches, any feature points within a given margin to the border of the ROI are ignored.

Moreover, the direction of the motion vectors is also well suited for detecting outliers. As discussed in section 4.1, we defined the y-axis of the world coordinate system heading towards the driving direction of the vehicles. Ideally, a valid motion vector should be parallel with the y-axis. In practice, however, the limited accuracy of the calibration parameters induces small deviations. Therefore, we calculate the deviation angle to the y-axis for all motion vectors and omit all vectors, where the deviation angle exceeds a certain threshold.

In the final stage of our outlier cascade, we take advantage of the rigidity of the moving objects. Due to the rigidity of objects, the relative order of feature points (with respect to the x-axis and y-axis) cannot change.

Figure 4 shows an example, where the relative order of feature points is changed due to an incorrect feature match. The green line (incorrect feature match) crosses a number of valid matchings. In a valid set of motion vectors, for any pair of feature points $(p_{i,j}, p_{i,k})$ and their corresponding feature matches $(p_{i,j}^{\acute{}}, p_{i,k}^{\acute{}})$, the following condition must be true.

$$\begin{aligned} c_{i,j,k} = [(X(p_{i,j}) \geq X(p_{i,k})) \rightarrow (X(p_{i,j}^{\acute{}}) \geq X(p_{i,k}^{\acute{}}))] \wedge \\ [(Y(p_{i,j}) \geq Y(p_{i,k})) \rightarrow (Y(p_{i,j}^{\acute{}}) \geq Y(p_{i,k}^{\acute{}}))] \end{aligned} \tag{2}$$

$X(p_{i,j})$ represents the x-coordinate, and $Y(p_{i,j})$ the y-coordinate of feature point $p_{i,j}$. Using equation 2, the outlier detector creates a consensus matrix C (upper triangular matrix), that contains the consensus $c_{i,j,k}$ for feature matches in the ROI. Further, for each $(p_{i,j}, p_{i,k})$, the number of contradictions (i.e., the number of $c_{i,j,k}$ that evaluate false) are written to the main diagonal of matrix C. After that, the algorithm iteratively rejects the motion vectors with the maximum number of contradiction until all elements in the main diagonal are zero (i.e., consensus is reached).

The speed is calculated from the median length of motion vectors. For that reason, the coordinates of the valid motion vectors are back-projected to metric world coordinates on a predefined plane. For reasons of efficiency, the mapping to of the individual pixels to world coordinates is stored in a lookup table. Using the metric length of motion vectors along with the known frame time τ, for every motion vector $\vec{m}_{i,j}$ the corresponding speed $v(\vec{m}_{i,j})$ is computed as follows:

$$v(\vec{m}_{i,j}) = \frac{Y(\omega(\acute{p}_{i,j})) - Y(\omega(p_{i,j}))}{\tau} \tag{3}$$

In equation 3, $Y(\omega(p_{i,j}))$ denotes the y-component of the back-projected feature point $p_{i,j}$. Since the direction of the y-axis was chosen in driving direction, the x-component of the world coordinates can be ignored. To compute the mean speed estimation $\tilde{\bar{v}}(r)$ for the r-th observation period, the median of calculated speed values is used.

$$\tilde{\bar{v}}(r) = \underset{r \cdot N_f \leq i < (r+1) \cdot N_f}{\text{median}} (v(\vec{m}_{i,j})), \forall \vec{m}_{i,j} \in \mathcal{R}_i \tag{4}$$

In equation 4, N_f denotes the number of frames for the observation time (e.g., number of frames per minute) and $\mathcal{R}_i$ is the set of valid motion vectors in frame i.

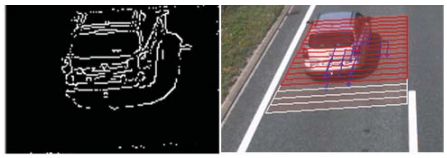

Figure 5. Binary edge image (left) and occupied stripes (right)

4.3. Traffic density estimation

For traffic density estimation, we use edge information within the ROI to obtain a statistical vehicle count. Therefore, we further subdivide the ROI in a number of disjoint stripes s_k, which are equally sized with respect to world coordinates. For each frame, our method applies canny edge detection to obtain a binary edge mask $e(x,y)_i$ for the frame i. Using the binary edge masks of the current and preceding frame, the moving edge mask $E(x,y)_i$ is calculated as described in equation 5.

$$E(x,y)_i = e(x,y)_i - [e(x,y)_i \wedge e(x,y)_{i-1}], i < 0 \quad (5)$$

In equation 5, $\wedge$ denotes the bit-wise AND operation. An example for a moving edge mask is shown in Figure 5 (left).

Using the moving edge mask $E(x,y)$, our traffic density estimation method calculates the number of edge pixels (edge count) for the individual stripes s_j. To obtain a metric unit ($pixel/m^2$), we define the edge density $E(s_j)_i$ of stripe s_j, as the number of moving edge pixels divided by the (predefined) area of the stripes A_{stripe}, as described by equation 6.

$$\hat{E}(s_j)_i = \frac{1}{A_{stripe}} \cdot \sum_{\forall (x,y) \in s_j} E(x,y)_i \quad (6)$$

Using the edge density $\hat{E}(s_j)_i$, the occupation state $occ(s_j)$ of a stripe s_j is defined in equation 7. In this equation, E_{thres} denotes a predefined threshold parameter.

$$occ(s_j)_i = \begin{cases} 1 & \text{if } \hat{E}(s_j)_i > E_{thres} \\ 0 & \text{else} \end{cases} \quad (7)$$

For ideal conditions (i.e., low noise), the state transitions of $occ(s_j)$ can directly be used for statistical vehicle counting. However, especially for difficult environment conditions (e.g., reflecting lights, moving shadows), it shows that simple transition counting of $occ(s_j)$ does not lead to accurate results. To increase the robustness of the counting method, we introduce additional constraints to the valid state transition of the individual stripes. These constraints make use of a number of assumptions: (1) vehicles move towards a (predefined) driving direction; (2) all vehicles within the ROI move at constant velocity; (3) all vehicles move through the entire ROI; (4) all vehicles have a minimum length (of 1.5 meters); and (5) vehicles do not move faster than 1 stripe per frame. Especially for stationary traffic situations (e.g., stop and go traffic) constraint (2) is violated, which leads to potential overestimation of traffic density. However, as our evaluation shows, this overestimation does not have a major impact on the accuracy of stationary traffic detection.

Table 2. Definition of stripe transition state $t(s_j)_i$.

$t(s_j)_i$	$occ(s_j)_{i-1}$	$occ(s_j)_i$
0	0	0
1	0	1
2	1	0
3	1	1

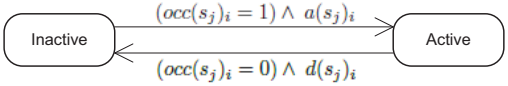

Figure 6. State diagram for stripe states

To implement robust statistical counting, we introduce the two stripe states *active* and *inactive*. The state diagram for the two stripe states is shown in Figure 6 . The state transitions not only depend on the occupation state $occ(s_j)_i$, but also on the constraints $a(s_j)_i$ and $d(s_j)_i$, defined in equation 8.

$$\begin{aligned} a(s_j)_i &= (j = 1) \vee (occ(s_{j-1})_i = 1) \\ &\quad \wedge inCluster(s_j, i) \\ d(s_j)_i &= (t(s_{j+1})_i \neq 0)) \wedge (t(s_{j-1})_i \neq 1) \end{aligned} \quad (8)$$

The $inCluster$ function returns true if the stripe is part of a cluster of active stripes with a predefined cluster size C_s. Furthermore, the constraints also use the temporal transition state $t(s_j)_i$ of the neighboring stripes. The four possible transition states are listed in table 2 . To compute the vehicle count, the algorithm maintains a transition count $c(s_j)$ for each stripe. The transition count is incremented on every *Active* to *Inactive* state transition and reset after each observation period. The traffic flow estimation $\tilde{q}$ is computed as median transition count of all stripes s_j, as described in equation 9.

$$\tilde{q} = \underset{C_s \leq j \leq (N_s - C_s)}{\text{median}} c(s_j) \quad (9)$$

where N_s denotes the number of stripes in the ROI. Using the flow estimate $\tilde{q}$ and the corresponding speed estimate $\tilde{\bar{v}}$, the traffic density estimate $\tilde{K}$ is calculated as shown in equation 10.

$$\tilde{K} = \frac{\tilde{q}}{\tilde{\bar{v}}} \cdot \lambda_s \quad (10)$$

Here, λ_s denotes a scaling constant that accounts for the observation period and unit conversions.

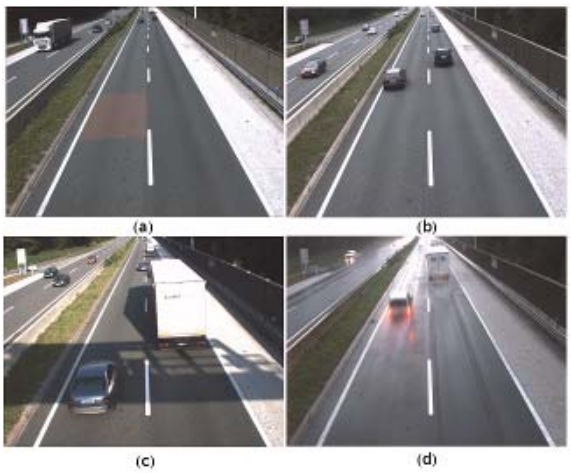

Figure 7. (a) Region of interest used for evaluation, (b) dry road with small shadows, (c) large shadows, (d) water on the road

5. Evaluation and results

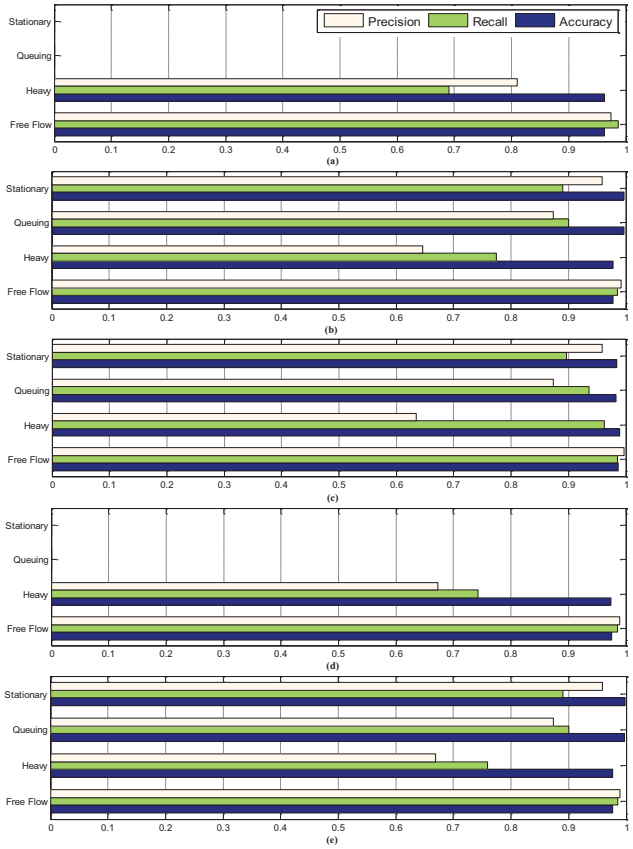

Figure 8. Evaluation of the traffic state detection method for different conditions based on precision, recall and accuracy of the LOS classifier: (a) Large shadows, (b) small shadows/no shadows, (c)wet road/water on the road, (d) dry road, (e) entire test set

For evaluating our video-based traffic state detection we used 156 hours video data recorded at successive days in July and August 2011 by our smart camera mounted at a gantry on the national motorway. The video sequences were recorded at daylight conditions at VGA resolution and a frame rate of 16 frames per second. All recordings were taken with the same camera orientation at the same camera location. For the speed estimations, we used the exact frame times obtained from the image sensor driver. The video sequences include traffic situations with different traffic states (all LOS levels) as well as different weather and illumination conditions (i.e., sunny, cloudy, light rain, and heavy rain). The ROI was defined manually on the left lane as shown in Figure 7(a) (red area) and 0.5m was used as stripe width. Mean speed and traffic density values were computed over one minute intervals. The results were compared to the measurements of a triple-tech traffic sensor [15] mounted next to the smart camera. The triple-tech traffic sensor combines doppler radar, passive infrared, and ultrasound for speed measurement and vehicle counting. To evaluate the robustness of our method for different conditions, the video data were manually annotated regarding weather and lightning conditions. The classification results were clustered in the following four (non-disjoint) categories: (1) No shadows or light shadows smaller than the lane, (2) large shadows, larger than the lane, (3) wet road or water on road, and (4) dry road. Figure 7 shows examples for the different weather and lighting conditions. The video analysis results were compared to the reference measurements, and the precision, recall, and accuracy values were calculated for the individual LOS classes. For practical reasons (i.e., more disk space, less computation time), the video data for this evaluation was analyzed on a server using the same implementation that runs on the smart camera. Live tests on the smart camera showed a mean execution time in the range of 30ms per frame.

Table 3. Distribution of test data (number of samples for different road conditions/traffic states).

Traffic State	Small Shadow	Large Shadow	Dry Road	Wet Road
Free Flow	7916	804	7628	1092
Heavy	347	58	364	41
Queuing	138	0	4	134
Stationary	144	0	0	144

Table 3 shows the distribution of the test data for the different road conditions and traffic states. The evaluation results are shown in Figure 8 and indicate a high precision and robustness of our method. The precision for the individual LOS levels does not show significant variations with respect to the different weather and lighting conditions. Stationary traffic is detected reliably also at difficult weather conditions (precision $> 95\%$). Free flow traffic is also detected very reliable (99% precision on the entire test set).

The evaluation shows a lower precision (67% on the entire test set) for heavy traffic. The discrimination between

Table 4. Mean absolute error (MAE) of traffic speed and traffic density.

Traffic State	MAE speed (km/h)	MAE density (vehicles/km)
Free Flow	3.01	1.11
Heavy	3.05	2.33
Queuing	1.79	4.78
Stationary	2.35	19.14

free flow traffic and heavy traffic is only related to traffic density (cp. definition of LOS in Table 1). Detailed analysis of reference data indicates two major reasons for the lower detection rate of heavy traffic. First, heavy rain can lead to significant image blur and lowers the edge density, which sometimes leads to lower vehicle counts. Second, the test set contains a high amount of borderline cases, where small deviations lead to classification errors (44% of misclassified heavy traffic samples differ only by 1 vehicle/km from free flow traffic).

In addition to the LOS-based evaluation of our traffic state detection method, we used the reference data from a triple-tech sensor to obtain the mean absolute error (MAE) for the described mean speed and traffic density estimation methods. Table 4 shows the MAE for the different traffic states, evaluated on the entire test set. The figures indicate the high quality of the presented methods for automatic video surveillance. The accuracy of the traffic density estimation decreases only for very high traffic density values (e.g., with stop and go traffic), which has no impact on the quality of the traffic state detection.

6. Conclusion

In this paper we presented a novel vision-based traffic state detection method that was implemented on an embedded smart camera. Our method uses optical flow and a robust cascaded outlier detection for speed estimation, and a robust statistical counting for traffic density estimation. The camera calibration parameters are obtained from the known length and distances of lane markings. Evaluations on a comprehensive test video set have shown the robustness of our method for different weather and illumination conditions. Stationary traffic is detected with a precision of 95.8%. As ongoing work, our prototype is currently evaluated for monitoring temporary maintenance areas on motorways.

Acknowledgments

This work was funded in part by the Austrian Research Promotion Agency under grant 825840 and was supported by Lakeside Labs GmbH Klagenfurt, Austria with funding from the European Regional Development Fund and the Carinthian Economic Promotion Fund (KWF) under grant KWF-20214/17097/24774.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik. A real-time computer vision system for measuring traffic parameters. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 495–501, 1997.

[2] H. Bischof, M. Godec, C. Leistner, B. Rinner, and A. Starzacher. Autonomous audio-supported learning of visual classifiers for traffic monitoring. *Intelligent Systems, IEEE*, 25(3):15–23, 2010.

[3] M. Bramberger, A. Doblander, A. Maier, B. Rinner, and H. Schwabach. Distributed embedded smart cameras for surveillance applications. *Computer*, 39(2):68 – 75, 2006.

[4] Bundesanstalt für Straßenwesen (BASt). Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen. Code of practice, Bundesanstalt für Straßenwesen (BASt), Bergisch Gladbach, Germany, 1999.

[5] J. Chen, T. Evan, and L. Zhidong. A machine learning framework for real-time traffic density detection. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 23(07):1265–1284, 2009.

[6] S.-C. S. Cheung and C. Kamath. Robust background subtraction with foreground validation for urban traffic video. *EURASIP Journal on Applied Signal Processing*, 14:2330–2340, 2005.

[7] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*, volume 2. Cambridge Univ Press, 2000.

[8] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th Intl. Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[9] F. Pletzer, R. Tusch, L. Böszörmenyi, B. Rinner, O. Sidla, M. Harrer, and T. Mariacher. Feature-based level of service classification for traffic surveillance. In *14th Intl. IEEE Conf. on Intel. Transport. Systems (ITSC)*, pages 1015–1020, 2011.

[10] M. Pucher, D. Schabus, P. Schallauer, Y. Lypetskyy, F. Graf, H. Rainer, M. Stadtschnitzer, S. Sternig, J. Birchbauer, W. Schneider, et al. Multimodal highway monitoring for robust incident detection. In *13th Intl. IEEE Conf. on Intelligent Transportation Systems (ITSC)*, pages 837–842, 2010.

[11] P. M. Roth, S. Sternig, H. Grabner, and H. Bischof. Classifier grids for robust adaptive object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2727–2734, 2009.

[12] M. Schneider, M. Linauer, N. Hainitz, and H. Koller. Traveller information service based on real-time toll data in Austria. *Intelligent Transport Systems, IET*, 3(2):124–137, 2009.

[13] O. Sidla, E. Wildling, and Y. Lypetskyy. Vehicle detection methods for surveillance applications. In *Proceedings of SPIE*, volume 6384, 2006.

[14] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[15] Xtralis Pty Ltd, Melbourne, Australia. *Data sheet of ASIM by Xtralis Triple-Tech Combination Detectors (TT 290 Series)*.

[16] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *17th International Conference on Pattern Recognition*, volume 2, pages 28–31, 2004.