BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS
AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

TRAFFIC ANALYSIS FROM VIDEO
ANALÝZA DOPRAVY Z VIDEA

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR
AUTOR PRÁCE
Bc. JAKUB SOCHOR

SUPERVISOR
VEDOUCÍ PRÁCE
Doc. Ing. ADAM HEROUT, Ph.D.

BRNO 2014

Abstrakt

V rámci této práce byl navržen a implementován systém pro analýzu dopravy z videa. Tento system umožňuje detekovat, sledovat a klasifikovat automobily. Systém je schopný detekovat pruhy z pohybu projíždějících automobilů a také je možné určit, zdali daný automobil jede v protisměru. Rychlost projíždějících automobilů je také měřena. Pro funkčnost systému není vyžadován žadný manuální vstup nebo kalibrace kamery, jelikož kamera je plně automacky zkalibrována pomocí úběžníků. Navržený systém pracuje s velkou přesností detekce, sledování a klasifikace automobilů a také rychlost automobilů je měřena s malou chybou. Systém je schopný pracovat v reálném čase a je aktuálně využíván pro nepřetržité online sledování dopravy. Největším přínosem této práce je plně automatické měření rychlostí projíždějích vozidel.

Abstract

A system for traffic analysis was designed and implemented during work on this thesis. The system is able to detect, track and classify vehicles. Also, the system is able to detect lanes or determine whether a vehicle is passing in wrong way. The speed of observed vehicles is also measured. The system does not require any manual input or calibration whatsoever as the video camera is fully automatically calibrated by detected vanishing points. The accuracy of the detection, tracking and classification is high and the speed of vehicles is measured with a low error. The system runs in real time and it is currently used for a continuous monitoring of traffic. The main contribution of the thesis is the fully automated speed measurement of passing vehicles.

Klíčová slova

analýza dopravy, detekce, sledování, klasifikace, měření rychlosti, detekce pruhu a směru jízdy

Keywords

traffic analysis, detection, tracking, classification, speed measurement, direction and lane detection

Citace

Jakub Sochor: Traffic Analysis from Video, diplomová práce, Brno, FIT VUT v Brně, 2014

Traffic Analysis from Video

Declaration

I hereby declare that this thesis is my own work that has been created under the supervision of Ing. Adam Herout, Ph.D. Where other sources of information have been used, they have been duly acknowledged.

. . . .
Jakub Sochor
May 19, 2014

Acknowledgment

I would like to thank to Adam Herout who suggested me to work on this thesis and provided a great support during my work on the thesis. Also, I would like to thank to Markéta Dubská and Roman Juránek for their advices and help with the thesis.

Contents

1 Introduction **2**

2 Existing Traffic Analysis Systems **4**
- 2.1 Traffic Parameters Extraction by Beymer et al., 1997 4
- 2.2 Classification of Vehicles in Traffic Video Streams by Morris et al., 2006 . . 6
- 2.3 Traffic Surveillance System by Hsieh et al., 2006 8

3 Related Computer Vision Algorithms for Traffic Analysis **13**
- 3.1 Detection 13
- 3.2 Tracking 17
- 3.3 Classification 19

4 Proposed Traffic Analysis System **25**
- 4.1 Initialization 26
- 4.2 Detection and Tracking 32
- 4.3 Three-Dimensional Bounding Boxes 35
- 4.4 Classification 39
- 4.5 Direction Estimation and Lane Detection 40
- 4.6 Speed Estimation 43

5 Implementation **46**
- 5.1 Traffic Analyser 46
- 5.2 Vehicles Annotator 47

6 Evaluation **48**
- 6.1 Detection and Tracking 48
- 6.2 Classification 49
- 6.3 Vehicle Speed Estimation 52
- 6.4 Direction Estimation and Lane Detection 55
- 6.5 Speed of Video Processing 58
- 6.6 Concluding Remarks 58

7 Conclusion **60**

Chapter 1

Introduction

There are many challenging tasks for the traffic surveillance applications from video. For example, systems for the traffic monitoring could determine age of vehicles, measure their speed, detect their type and the brand of the manufacturer. Also, complex crossroads and behaviour of drivers can be analyzed. However, all these tasks are rather advanced and complex if the system should work without any manual input from a single uncalibrated video camera. This thesis contributes to the state of the art mainly by the automatic speed measurement of vehicles.

The goal of the thesis is to design and implement a system for fully automated real time traffic analysis from a single uncalibrated video camera. The system should be able to count vehicles and classify them. Also, the speed of vehicles and lane which they are taking should be estimated. Vehicles passing in wrong way can also be detected. As the system should be usable for any traffic surveillance scene, it has to run in real time and be fully automated. Hence, the system does not use any manual input or calibration whatsoever. This constraint is challenging mainly for the speed measurement because it is necessary to automatically calibrate the video camera and to compute the correct scale. These types of

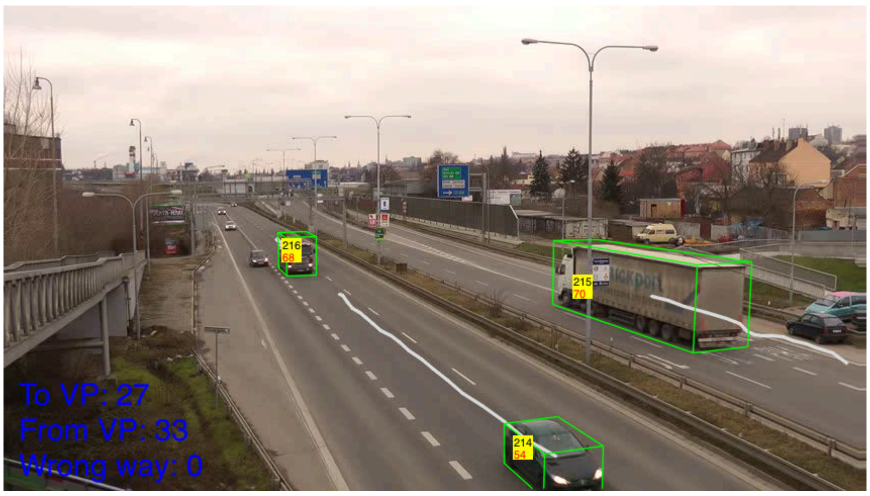


Figure 1.1: An example of a processed video by the proposed traffic analysis system. It shows the amount of vehicles passing in two directions and the number of vehicles which were passing in wrong way. Also, the identification number (black) and speed (red) is drawn in a yellow rectangle for each vehicle.

analysis systems have a wide spectrum of usage. For example, it is possible to monitor the traffic or try to predict characteristics of the future traffic flow.

The proposed system automatically calibrates the camera by detected vanishing points. Vehicles are detected by a background subtraction and tracked with Kalman filter. Histograms of Oriented Gradients and Support Vector Machines are used for the classification of vehicles. The lanes detection is based on the motion of observed vehicles. The automatic calibration and scale computation is used for the measurement of distances on the road plane in order to estimate speed of vehicles.

Other systems for the traffic analysis are studied in Chapter 2 and related computer vision algorithms which are used for the traffic analysis are presented in Chapter 3. The design of the proposed system is described in Chapter 4 and its implementation is discussed in Chapter 5. Also, the system is thoroughly evaluated in Chapter 6.

Chapter 2

Existing Traffic Analysis Systems

This chapter presents three different systems for traffic analysis. Each system is described into detail and the achieved results are discussed. Described systems represent widely used approaches for detection, tracking and classification of vehicles.

A popular approach for the detection and tracking of vehicles is to use some form of background subtraction and Kalman filter [20] to track the vehicles [16, 29, 18, 40, 1, 5, 9, 26, 23, 31]. Other approaches are based mainly on the detection of corner features, their tracking and grouping [2, 17, 6]. Also, part-based detections are used in several papers [5, 38] for the detection of vehicles.

Several types of features are used for the classification of vehicles. Forms of image features are used in some publications [29, 28]. Other works on the topic use shape of vehicles for the classification [27, 21, 14, 3, 28, 29] and some of these works [27, 21, 14, 3] use a three-dimensional model of vehicles for matching the shape. Also, some papers [16, 29] present artificial measurements in the image or contour of vehicles for obtaining the features for the classification.

2.1 Traffic Parameters Extraction by Beymer et al., 1997

The first studied system [2] for the traffic analysis is focused on measuring of a traffic parameters. The system is able to count cars, measure the velocity of the cars and other dimensions of the cars in the frame. The system is based on corner-feature detection. The detection step is followed by tracking and grouping of these features. Also, the user of the system has to manually define four points of correspondence for homography [37] a priori. The homography is used for parallelization of lane-dividing lines as it is shown in Figure 2.1. The user also has to specify a region of entrance and exit of the cars before running the system.

As it was mentioned, the image coordinates (x, y) are mapped into the transformed coordinates (X, Y) by the homography. The transformation matrix of the homography H is obtained in a manual way. The manually defined four points of correspondence unambiguously define the matrix H if the scaling of the transformation matrix is 1. If homogenous coordinates are used, the homography can be expressed as a linear transformation.

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

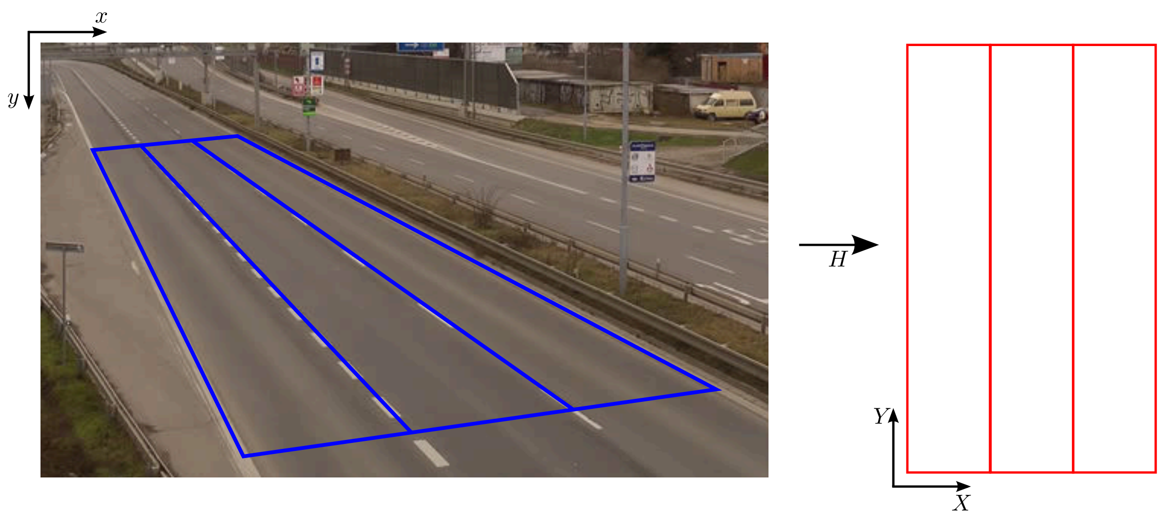


Figure 2.1: Mapping from image coordinates (x, y) into (X, Y) coordinates by homography H.

The corner feature detection is based on the gradient of pixel ∇I. The pixel is considered to be the corner feature if matrix $\nabla I \nabla I^T$ has the smallest eigenvalue bigger than a predefined threshold. If the corner feature is found, 9×9 neighborhood is extracted for matching of the features in the tracking module.

The Kalman filter [20] is used for the tracking of the features. The filter is explained in detail in Section 3.2.1. The state of the Kalman filter contains position and speed in both directions. The predicted position by the Kalman filter is used to find correlation peak with the extracted 9×9 template of the corner feature. The localized correlation peak is rejected if its distance from the predicted position is higher than a threshold.

The grouping of the corner features into one object is based on a common motion of these features. The grouping uses an unoriented graph $G = (V, E)$, while the corner features are vertices of the graph, and an edge between vertices denotes the grouping relationship. Last but not least, the connected components of the graph correspond to a vehicle hypothesis. A new feature is initially connected with surrounding features in some predefined radius. An edge (a, b) is kept in the graph until the difference of the motions of the features is above a threshold.

The history of positions of a feature a is expressed as a function $p_a(t)$ and the same way for the feature b. The grouping component computes relative displacement of the features over time $d(t) = p_a(t) - p_b(t)$. The edge (a, b) is removed if either (2.2) or (2.3) holds. When the last vertex of a connected component enters the exit region, a new passed car is detected.

$$\max_t d_x(t) - \min_t d_x(t) > t_x \tag{2.2}$$
$$\max_t d_y(t) - \min_t d_y(t) > t_y \tag{2.3}$$

The grouper can suffer from either oversegmentation or overgrouping. Oversegmentation denotes that the thresholds t_x and t_y are below optimum and a vehicle is divided into more than one object. On the other hand, if the grouping suffers from overgrouping, the thresholds t_x and t_y are above optimum and more vehicles are grouped into a one. The optimum thresholds were computed off-line by exhaustive search.

The true match rate of the system depends on the scene settings and it is between 73.9 % and 94.9 %. Unfortunately, the worst true match was measured for a highway. The error of measured velocity was below 10 % for all samples. The system runs in real-time on thirteen C40 DSPs and one 150 MHz Pentium joined into one computation unit.

2.2 Classification of Vehicles in Traffic Video Streams by Morris et al., 2006

The second studied system [29] about traffic analysis from a video stream focuses mainly on the classification. The cars are also tracked, however, it is only for better results of the classification.

The authors examined two different types of feature vectors, also as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) (both [37]) dimensionality reduction. The weighted K nearest neighbor algorithm is used for the classification.

However, the tracking algorithm will be described first. The authors used an adaptive background subtraction (Section 3.1.1) for the vehicle detection and the Kalman filter for tracking (Section 3.2.1).

The two types of used features for the classification are the following. First, the raw image pixels were used as the feature vector. However, the car was rescaled to size 64×32 pixels a priori. The second used feature vector is based on measurements done in the image of the vehicle. For example, the area, the width and height of the bounding box, the convex area and other measurements are these features used for the classification.

The main goal of the PCA and LDA is to reduce the dimensionality of a vector $\boldsymbol{x}$. Both PCA and LDA accomplish this reduction by projection of the vector $\boldsymbol{x}$ into M base vectors. PCA uses as the base vectors M eigenvectors $\boldsymbol{u}_i$ with the highest corresponding eigenvalues λ_i of the covariance matrix C of the training data with N samples. These eigenvectors are the directions with the highest variance. It is possible to use the following equations and eigenvalue decomposition for obtaining the eigenvalues and eigenvectors [37].

$$\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}_i \tag{2.4}$$

$$C = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T \tag{2.5}$$

$$C = U\Lambda U^T = \sum_{i=1}^{N}\lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \tag{2.6}$$

On the other hand, LDA selects the direction $\boldsymbol{u}$ that results in the largest ratio between the projected between-class and within-class variance. This is achieved by maximizing Equation (2.10). Matrices S_W and S_B are within-class and between-class scatter matrices. Vector $\boldsymbol{\mu}_k$ denotes the mean value within the class k and there is total K classes for the classification.

$$S_k = \sum_{i \in C_k} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \tag{2.7}$$

$$S_W = \sum_{k=1}^{K} S_k \tag{2.8}$$

$$S_B = \sum_{k=1}^{K} N_k (\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T \tag{2.9}$$

$$\boldsymbol{u}^* = \arg\max_{\boldsymbol{u}} \frac{\boldsymbol{u}^T S_B \boldsymbol{u}}{\boldsymbol{u}^T S_W \boldsymbol{u}} \tag{2.10}$$

Solving Equation (2.10) and selecting M best solutions is equivalent to searching the eigenvalues and eigenvectors (2.11) and taking the M eigenvectors $\boldsymbol{u}$ with the highest eigenvalues λ.

$$S_B \boldsymbol{u} = \lambda S_W \boldsymbol{u} \tag{2.11}$$

The authors of the publication used the weighted K nearest neighbor algorithm for the classification. Let χ_c be a set of vectors $\boldsymbol{x}$ which belongs to the same class c. The algorithm assigns a label c to a vector $\boldsymbol{x}$ with maximal W_c. The weight W_c of the assignment is calculated as the sum of multiplicative inverses of the distances to the nearest K vectors of the class c.

$$W_c = \sum_{i=1, \boldsymbol{x}_i \in \chi_c}^{K} \frac{1}{\|\boldsymbol{x}_i - \boldsymbol{x}\|} \tag{2.12}$$

$$L(\boldsymbol{x}) = \arg\max_{c} W_c \tag{2.13}$$

$$W(\boldsymbol{x}) = \max_{c} W_c \tag{2.14}$$

The possible combinations of the feature vectors and the dimensionality reductions were evaluated and compared. The possible combinations are PCA+image, LDA+image, PCA+measurements and LDA+measurements where the combinations differ in used feature vector and dimensionality reduction. The measurements feature vector is based on artificial measurements done in the image of a vehicle and the image feature vector contains raw pixels of the image of a vehicle. The results presented by authors show that the PCA+image slightly outperforms the other combinations for one image classification. The best achieved overall accuracy is 80.36 %.

The results can be improved by classifying the car multiple times in different frames of its track T. However, the weights W_c in frame t of the track has to be normalized (2.15). The final class is obtained by Equation (2.16):

$$W_c^t = \frac{W_c}{\sum_{c'} W_{c'}} \tag{2.15}$$

$$L_T = \arg\max_{c} \sum_{t \in T} W_c^t \tag{2.16}$$

The authors obtained overall accuracy 82.88 % of the classification with the track refinement of the classification. The classification accuracy was 87.43 % for the videos from the second day. The used dataset contained 1836 training samples and 611 evaluation samples.

2.3 Traffic Surveillance System by Hsieh et al., 2006

The last examined system [16] addresses both tracking and classification of the vehicles. The authors present a novel approach to the shadow elimination and a new feature for classification of the vehicles.

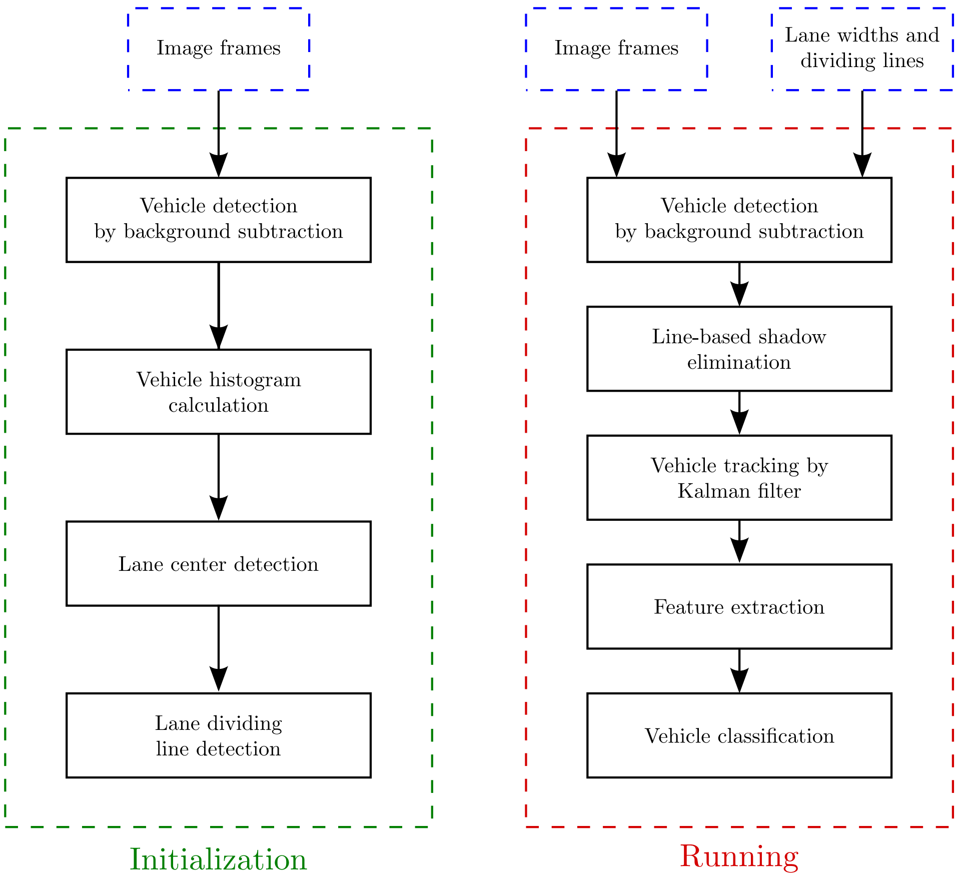


Image frames
Vehicle detection by background subtraction
Vehicle histogram calculation
Lane center detection
Lane dividing line detection
Initialization
Image frames
Lane widths and dividing lines
Vehicle detection by background subtraction
Line-based shadow elimination
Vehicle tracking by Kalman filter
Feature extraction
Vehicle classification
Running

Figure 2.2: The schema of the traffic analysis system [16]

The system has a nontrivial initialization part where the lanes are detected. The scheme of the whole system is in Figure 2.2. The authors used background subtraction for the motion detection which is done according to Equation (2.17), while I_k are intensities of the current frame and B_k is a model of the background in frame k. Kalman filter is used for tracking of the vehicles. The innovative parts of the paper will be presented in the following text.

$$D_k(x,y) = \begin{cases} 0 & \|I_k(x,y) - B_k(x,y)\| \leq T_d \\ 1 & \text{otherwise} \end{cases} \tag{2.17}$$

Lane Detection

Prior to running the system, lanes and lane-dividing lines have to be detected. Moving objects are detected in the initialization part and a two-dimensional histogram of the centers of the moving objects in the video stream is built.

The algorithm for detection of lane centers is executed after an appropriate amount of cars was accumulated into the histogram. Let us assume that the size of the image and the histogram is $w \times h$ pixels. The algorithm for the lane detection has the following steps. The output of the algorithm are the lane-dividing lines and the widths of the lanes.

1. Smooth the histogram H for all i and j. As the following equation shows, the smoothing is performed only in the horizontal direction.

$$\overline{H}(i,j) = \sum_{k=-2}^{2} H(i+k,j) \tag{2.18}$$

2. The mean value T_H^j is calculated for each row j.

$$T_H^j = \frac{1}{w} \sum_{i=1}^{w} \overline{H}(i,j) \tag{2.19}$$

3. Each pixel (i,j) is set to 1 if and only if the $\overline{H}(i,j)$ is a local maximum along the line j and $\overline{H}(i,j) > T_H^j$. The pixel is set to 0 otherwise.

4. All isolated connected segments are found, small segments are eliminated and adjacent segments are merged if they are close to each other. The detected segments are the centers of the lanes. Let us denote the center of the lane L_k in the row j as $C_{L_k}^j$.

5. The points of the lane-dividing lines DL_k^j and width of the lanes $w_{L_k}^j$ is calculated for each row j and lane L_k.

$$DL_k^j = \frac{1}{2}\left(C_{L_{k-1}}^j + C_{L_k}^j\right) \tag{2.20}$$

$$w_{L_k}^j = \left|C_{L_k}^j - C_{L_{k-1}}^j\right| \tag{2.21}$$

Shadow Elimination

The shadow detection is based on the fact that it is not possible that the car is in more lanes at the same time if the vehicle is not changing its lane. Let us assume that there is an occlusion of two cars caused by the shadow as it is shown in Figure 2.3.

Let us denote the points of the region with the detected motion as R_O and the lane-dividing line as L_k. The set of points U_k is the intersection of the set of points R_O and L_k ($U_k = R_O \cap L_k$). A new line L_k is fitted to the points U_k by least-squares approximation.

$$y = m_k x + b_k \tag{2.22}$$

It is possible to create new parallel lines to L_k by changing the parameter b_k. Let us denote as $b_k^{\min}$ and $b_k^{\max}$ the lowest and highest b_k for which line L_k^b has an intersection with region R_O. Line L_k is divided into two lines L_k^p and L_k^q which are moved to the left or right by decreasing or increasing the original b_k. The expansion is stopped when any of the pixels in the intersection $R_O \cap L_k^p$ ($R_O \cap L_k^q$ respectively) is not a shadow pixel. The expansion process of lines L_k^p and L_k^q is presented in Figure 2.4.

$$b_k^{\min} = \min_{p \in R_O} (y_p - m_k x_p) \tag{2.23}$$

$$b_k^{\max} = \max_{p \in R_O} (y_p - m_k x_p) \tag{2.24}$$

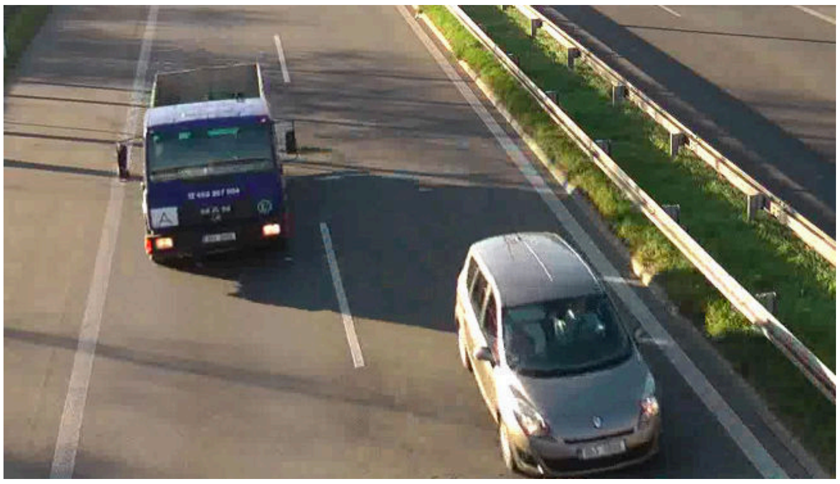


Figure 2.3: An occlusion caused by a shadow. As the figure shows, the shadow which is cast by the blue truck causes the occlusion with the other vehicle.

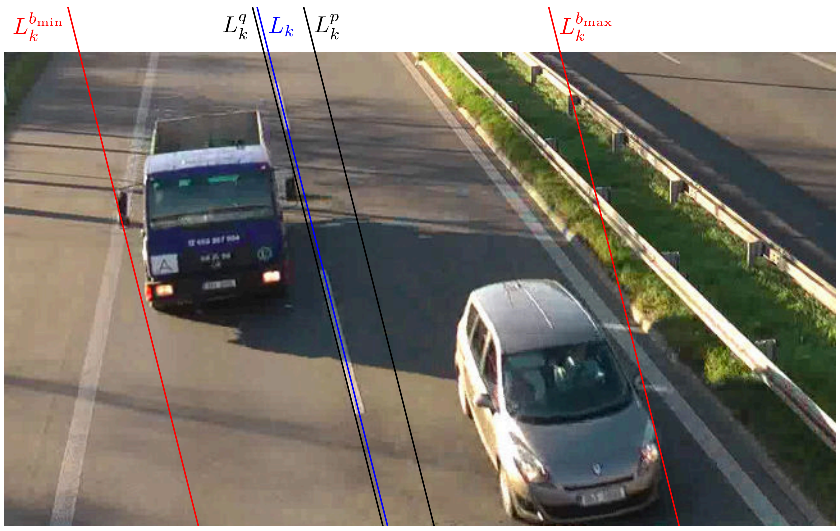


Figure 2.4: An example of the expansion of the lane-dividing line. The line L_k^q is moving to the left, L_k^p is moving to the right.

Pixel p is considered to be a shadow, if the probability $P(shadow|p) > 0.8$. The parameters μ_{shadow} and σ_{shadow} are obtained by a training process. The precise algorithm for the shadow elimination can be found in the original publication [16].

$$P(shadow|p) = \exp\left(\frac{(I(p) - \mu_{shadow})^2}{\sigma^2_{shadow}}\right) \tag{2.25}$$

Classification

Two features are used for the classification of the vehicles. The first one is normalized size of the detected vehicle $\overline{s}_v$. The normalization is based on the width of the lane which the vehicle is using in the given frame. The normalization is done according to the following equations, where c_v is the center of the detected vehicle and $X_{DL_i}(y_p)$ is the x coordinate

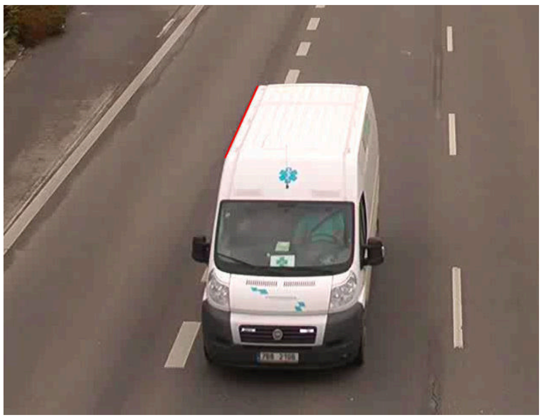


Figure 2.5: An example of the up-slanted edge for a van (red)

of the nearest lane-dividing line on the left from the point p and $X_{DL_{i+1}}(y_p)$ denotes the nearest line on the right.

$$\begin{aligned} W_i(p) &= \left|X_{DL_i}(y_p) - X_{DL_{i+1}}(y_p)\right| && (2.26)\\ \overline{s}_v &= \frac{s_v}{W_i^2(c_v)} && (2.27) \end{aligned}$$

The second feature used for the classification is linearity of the up-slanted edge of the vehicle. The linearity feature is based on error of the least-square error approximation of the up-slanted edge. The set of the up-slanted edge points H_i is detected, parametrized as a line by the least-square approximation and the linearity feature is expressed by the following equation. An example of the up-slanted edge is shown in Figure 2.5.

$$Lin(H) = \exp\left(-\sqrt{\frac{1}{|H|}\sum_{(x_i,y_i)\in H}(y_i - mx_i - b)^2}\right) \tag{2.28}$$

The classification itself is done by assigning the class l to the feature vector H_i according to equation (2.34). Let us assume that there is K classification classes, the class VC_k has n_k samples, V_j^k is a vehicle in the class VC_k and $f_r(V_j^k)$ is the r^{th} element of the feature vector V_j^k.

The μ_r^k denotes the mean value of the r^{th} feature in the class k and $\sigma_{r,k}$ stands for the standard deviation of the feature in the class. The similarity between two feature vectors is expressed as $S_k(H_i, V_j^k)$ and $S(H_i|VC_k)$ denotes the similarity between the feature vector H_i and the class VC_k. Finally, the probability that the feature vector H_i belongs to the class VC_k is expressed as $P(VC_k|H_i)$.

$$\mu_r^k = \frac{1}{n_k}\sum_{i=1}^{n_k} f_r(V_i^k) \tag{2.29}$$

$$\sigma_{r,k} = \sqrt{\frac{1}{n_k}\sum_{j=1}^{n_k}(f_r(V_j^k) - \mu_r^k)^2} \tag{2.30}$$

$$S_k(H_i, V_j^k) = \exp\left(-\sum_{r=1}^{2}\frac{(f_r(H_i) - f_r(V_j^k))^2}{\sigma_{r,k}^2}\right) \tag{2.31}$$

$$S(H_i|VC_k) = \frac{1}{n_k}\sum_{V_j^k \in VC_k} S_k(H_i, V_j^k) \tag{2.32}$$

$$P(VC_k|H_i) = \frac{S(H_i|VC_k)}{\sum_{k'=1}^{K} S(H_i|VC_{k'})} \tag{2.33}$$

$$l = \arg\max_k P(VC_k|H_i) \tag{2.34}$$

Three different videos were evaluated and the authors achieved 82.16 % vehicle counting accuracy with the shadow elimination enabled. The accuracy was lower more than 10 % without the shadow elimination. The accuracy of the classification was between 87.8 % and 91.1 % in different videos.

Chapter 3

Related Computer Vision Algorithms for Traffic Analysis

This chapter presents used algorithms in the proposed traffic analysis system. The algorithms are divided into three parts. The first part addresses the problem of detection of the car, the second one describes tracking and the last part of the chapter deals with classification of cars.

3.1 Detection

The algorithm dealing with the problem of detection of a moving car in a video is described in this part of the chapter. Description of an algorithm dealing with moving shadows is also included.

3.1.1 Motion Detection with Mixture of Gaussians

This section deals with the detection of moving objects based on adaptive background modeling of the scene with Mixture of Gaussians. The key publications for this section is an article written by Chriss Stauffer and W.E.L. Grimson [36] describing the idea of adaptive background modeling with Mixture of Gaussians. Some improvements to this approach are described in the Zoran Zivkovic's publication [46].

The idea behind every background subtraction is to model the background of a scene in some way and differentiate the foreground from the background with this model. The background subtraction can be used for pedestrian detection, vehicle detection or in general for any motion detection in a static scene. However, as the background subtraction models the background, it is essential so that the video camera does not move and there are no significant fast changes in the background so that the motion detection works properly.

The basic idea is to model the background color of each pixel in the scene with a Mixture of Gaussians. This mixture is used to describe and differentiate the background colors of the pixel from the foreground ones. For example, a semaphore, which periodically changes the lights from green to orange and red, can be in the traffic surveillance scenes. Hence, there will be pixels which will change periodically colors from black to green, and therefore only one Gaussian for the background of the pixel does not suffice. On the other hand, it is unlikely that some fast dynamic lightning change will occur in outdoor scenes. Such fast change could happen for example in indoor environment when someone in the room turns off the lights.

As it was mentioned above, the Mixture of Gaussians is used to model each pixel. It is necessary to describe three colors (red, green, blue). Hence, three-dimensional Gaussians are used. However, the Gaussians are not used directly in a form of three-dimensional Gaussian probability density function $\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma)$, Equation (3.1). The algorithm rather uses the covariance matrix Σ in the form of $\Sigma = \sigma^2 I$, where σ is the standard deviation in all directions and I a is 3×3 identity matrix.

$$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{\frac{3}{2}} |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu}) \right) \tag{3.1}$$

Only one pixel of the image will be taken into account as the others are processed in the same way. Let us denote the value of the pixel at time t as $\boldsymbol{x}^{(t)}$. Another important variable is the learning rate α, which specifies how fast a new cluster will be adopted. The following parameters will be held for each Gaussian. The weights, π_m, which are non-negative and sums up to one. The mean value $\boldsymbol{\mu}_m$ and the variance σ_m^2 is also kept for every Gaussian and maximally M Gaussians will be held for the pixel.

The pixel is processed in the following way. First, the Gaussian with the biggest π_m and the squared Mahalanobis distance $D_m^2(\boldsymbol{x}^{(t)})$ smaller than the threshold value T_g is found. The reasonable value for the threshold is 9, which implies that the Gaussian is considered to be *close* to the value if the value $\boldsymbol{x}^{(t)}$ is less than 3σ far from the mean. For the *close* Gaussian the ownership variable $o_m^{(t)}$ is set to 1. The other Gaussians have the ownership value equal to 0.

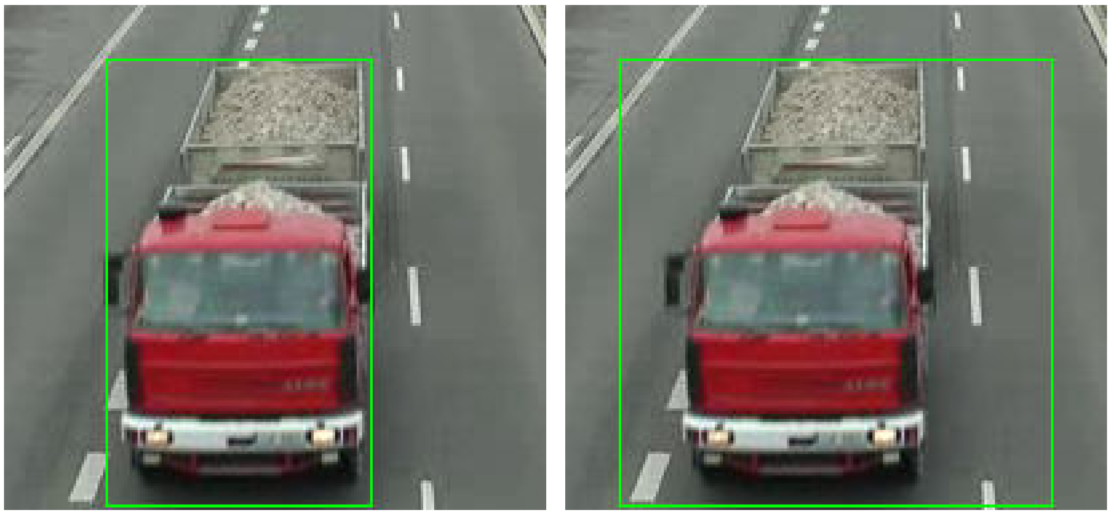
Second, parameters of the Gaussians are updated with respect to expressions (3.4), (3.5) and (3.6), while the value α is the learning rate and complexity reduction parameter c_T will be discussed later.

$$\begin{aligned}
\boldsymbol{\delta}_m &= \boldsymbol{x}^{(t)} - \boldsymbol{\mu}_m^{(t)} && (3.2)\\
D_m^2(\boldsymbol{x}^{(t)}) &= \boldsymbol{\delta}_m^T \boldsymbol{\delta}_m / \sigma_m^2 && (3.3)\\
\pi_m &\leftarrow \pi_m + \alpha(o_m^{(t)} - \pi_m) - \alpha c_T && (3.4)\\
\boldsymbol{\mu}_m &\leftarrow \boldsymbol{\mu}_m + o_m^{(t)}(\alpha/\pi_m)\boldsymbol{\delta}_m && (3.5)\\
\sigma_m^2 &\leftarrow \sigma_m^2 + o_m^{(t)}(\alpha/\pi_m)(\boldsymbol{\delta}_m^T \boldsymbol{\delta}_m - \sigma_m^2) && (3.6)
\end{aligned}$$

If no *close* Gaussian is found, it is necessary to create a new one for the current value of pixel $\boldsymbol{x}^{(t)}$. The parameters of the newly created Gaussian are $\pi_{m+1} = \alpha$, $\boldsymbol{\mu}_{m+1} = \boldsymbol{x}^{(t)}$, $\sigma_{m+1}^2 = \sigma_0^2$. If there would be more Gaussians than the maximal amount M, the weakest one is removed. The weights are normalized in order to sum up to one after the process.

For a given pixel, it is also determined if it should be considered as the background or the foreground. The squared Mahalanobis distance is used for this task. If a Gaussian which has the distance smaller than the predefined threshold c_{thr} is found, the pixel is treated as the background. If no such Gaussian is found, the pixel belongs to the foreground.

Zoran Zivkovic suggested an improvement [46] for the algorithm presented in the original article [36]. The improvement is in the term αc_T in Equation (3.4). The term is responsible for decreasing the weight of the Gaussian if there was no pixel with the color corresponding the Gaussian. If the weight of the Gaussian drops bellow zero after this subtraction, the Gaussian is removed from the model for the pixel. This approach brings more dynamics to the number of Gaussians used for modeling the background of the pixel and ability to select the proper number of Gaussians for the pixel with the upper limit M. This pruning slightly improves performance of the algorithm and more significantly it decreases the processing time of one image.



(a) Shadow detection enabled (b) Shadow detection disabled

Figure 3.1: Difference in precision of the motion detection

In this section the algorithm for the motion detection with the Mixture of Gaussians was presented. This detection can be successfully used for detection of people, cars and other objects in the scene captured by a stationary video camera.

3.1.2 Shadow Detection

In this section an algorithm for the detection of the shadows in the video is described. The algorithm was presented in the publication written by Thanarat Horprasert [15] and also in the survey of shadow detection algorithms [30]. The main reason for using some shadow detection is the precision of the motion detection. As Figure 3.1 shows, if the shadow detection is enabled, the motion is detected much more precisely; and therefore, the vehicle is detected with a higher accuracy.

The algorithm will be described with one pixel, as the other pixels are processed in the same way. The shadow detection algorithm works also with the background modeled by the Mixture of Gaussians. However, the algorithm will be explained with only one Gaussian without loss of generality. It is sufficient if the shadow is detected by one Gaussian from the Mixture of Gaussians in order to consider the pixel as the shadowed background.

The basic idea is to decompose the distortion of the mean value $\boldsymbol{\mu}$ of the Gaussian from the current pixel value $\boldsymbol{x}$. The decomposition is done into two components, brightness distortion β and chromaticity distortion CD. The decomposition is shown in Figure 3.2.

The brightness distortion is a scalar value β, which is computed by minimizing distance of the point $\boldsymbol{x}$ from line which is defined by points $(0, 0, 0)$ and $\boldsymbol{\mu}$, Equation (3.7). The point in 3D space $\beta\boldsymbol{\mu}$ is the orthogonal projection of $\boldsymbol{x}$ onto line defined by the coordinate origin and $\boldsymbol{\mu}$.

$$\beta = \arg\min_{\beta \in \mathbb{R}} \|\boldsymbol{x} - \beta\boldsymbol{\mu}\| \tag{3.7}$$

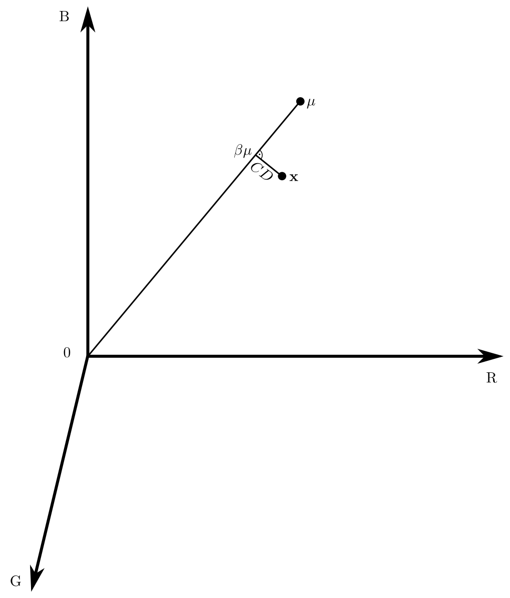


Figure 3.2: The decomposition of the distance between the mean value of the pixel color $\boldsymbol{\mu}$ and current pixel value $\boldsymbol{x}$ into the brightness distortion β and the chromaticity distortion CD

The chromaticity distortion CD is also a scalar value and can be calculated by equation (3.8). All the three points, ($\boldsymbol{x}$, $\boldsymbol{\mu}$ and $\beta\boldsymbol{\mu}$) form a right triangle and the coordinate origin is included in the line obtained by extending cathetus $\boldsymbol{\mu}$ and $\beta\boldsymbol{\mu}$.

$$CD = \|\boldsymbol{x} - \beta\boldsymbol{\mu}\| \tag{3.8}$$

However, also the standard deviation of the Gaussian is taken into account in the original publication [15] and the values β and CD are calculated by Equations (3.9) and (3.10). On the other hand, the Mixture of Gaussians presented in the previous section uses $\sigma = \sigma_R = \sigma_G = \sigma_B$, and therefore it is possible to simplify the equations.

$$\beta = \frac{\frac{x_R\mu_R}{\sigma_R^2} + \frac{x_G\mu_G}{\sigma_G^2} + \frac{x_B\mu_B}{\sigma_B^2}}{\left(\frac{\mu_R}{\sigma_R}\right)^2 + \left(\frac{\mu_G}{\sigma_G}\right)^2 + \left(\frac{\mu_B}{\sigma_B}\right)^2} \tag{3.9}$$

$$CD = \sqrt{\left(\frac{x_R - \beta\mu_R}{\sigma_R}\right)^2 \left(\frac{x_G - \beta\mu_G}{\sigma_G}\right)^2 \left(\frac{x_B - \beta\mu_B}{\sigma_B}\right)^2} \tag{3.10}$$

In order to normalize the brightness distortion β and the chromaticity distortion CD, the square root of mean is computed, Equations (3.11) and (3.12). These values are root mean square values for the lengths of catheti of the right triangle in Figure 3.2. Values a and b can be computed continuously and it is not necessary to keep all the history of the β and CD. After all these values are obtained, the normalized brightness distortion $\widehat{\beta}$ and chromaticity distortion $\widehat{CD}$ are calculated.

$$a = \sqrt{\frac{\sum_{i=0}^{N}(\beta - 1)^2}{N}} \tag{3.11}$$

$$b = \sqrt{\frac{\sum_{i=0}^{N} CD^2}{N}} \tag{3.12}$$

$$\widehat{\beta} = \frac{\beta - 1}{a} \tag{3.13}$$

$$\widehat{CD} = \frac{CD}{b} \tag{3.14}$$

Several thresholds are used for the final decision about the class of the pixel c. The first threshold τ_{CD} specifies the maximal chromaticity distortion which one pixel can have in order not to be considered as the foreground. If the normalized brightness distortion $\widehat{\beta}$ is smaller than $\tau_{\beta lo}$, the pixel is also considered to be the foreground. If the normalized brightness distortion and the chromaticity distortion is not within these limits, the pixel is treated as some particular kind of the background (normal, shadowed, lighted). At last, there is a pair of thresholds $\tau_{\beta 1}$ and $\tau_{\beta 2}$ specifying values of $\widehat{\beta}$ for which the pixel is considered to be background. The whole decision procedure is presented in Equation (3.15).

$$c = \begin{cases} \text{foreground} & \widehat{CD} > \tau_{CD} \ \vee \ \widehat{\beta} < \tau_{\beta lo} \ \ \text{else} \\ \text{background} & \widehat{\beta} < \tau_{\beta 1} \ \wedge \ \widehat{\beta} > \tau_{\beta 2} \ \ \text{else} \\ \text{shadowed} & \widehat{\beta} < 0 \ \ \text{else} \\ \text{lighted} & \text{otherwise} \end{cases} \tag{3.15}$$

However, if the procedure (3.15) processes only pixels of the image which were classified as the foreground by the Mixture of Gaussians, the rule for the background can be eliminated. Only two thresholds (τ_{CD} and $\tau_{\beta lo}$) are required after this reduction.

3.2 Tracking

This section describes the problem of tracking of a vehicle in a video stream. The goal of the tracking is to find correspondences of the cars in two or more consenquent video frames. An algorithm for this problem will be presented and described in detail.

3.2.1 Kalman Filter

This section addresses the Kalman filter [20] usable for tracking of objects in video stream. The algorithm was already successfully used for tracking of cars. An example of the usage of the algorithm was given by Jung and Ho [19] dealing with the traffic parameter extraction. The following description is based on technical paper written by Welch and Bishop [44].

However, Kalman filter can be used also for other applications, not just tracking (e.g filtering of a signal or an efficient computation of the least-squares problem). Generally, Kalman filter tries to approximate a discrete-time process defined by stochastic difference equation (3.16). Variable $\boldsymbol{x}_i \in \mathbb{R}^n$ is the state variable and $\boldsymbol{u}_k \in \mathbb{R}^l$ is the control unit for state x_k. Matrices A_k and B are used to relate the state and the control at time k into time $k + 1$.

$$\boldsymbol{x}_{k+1} = A_k \boldsymbol{x}_k + B \boldsymbol{u}_k + \boldsymbol{w}_k \tag{3.16}$$

The estimation of the mentioned linear process is done with measurement $\boldsymbol{z}_k \in \mathbb{R}^m$ and equation (3.17) holds for the variable z_k.

$$\boldsymbol{z}_k = H_k \boldsymbol{x}_k + \boldsymbol{v}_k \tag{3.17}$$

Random variables $\boldsymbol{v}$ and $\boldsymbol{w}$ add noise into the linear process defined by (3.16) and measurement (3.17). It is assumed that the variables are mutually independent and have normal probability distributions with zero mean value and covariance matrices Q and R.

$$p(\boldsymbol{w}) \sim \mathcal{N}(\mathbf{0}, Q) \tag{3.18}$$
$$p(\boldsymbol{v}) \sim \mathcal{N}(\mathbf{0}, R) \tag{3.19}$$

It is necessary to define an a priori state estimate $\widehat{\boldsymbol{x}}_k^-$ and an a posteriori estimate $\widehat{\boldsymbol{x}}_k$. The a priori estimate will be used before the measurement of the current state is done and the a posteriori will be used after the measurement is done. Let us also define a priori and a posteriori estimate errors as $\boldsymbol{e}_k^-$ and $\boldsymbol{e}_k$, Equations (3.20) and (3.21). Then, it is possible to define a priori (3.22) and a posteriori (3.23) error covariance matrices.

$$\boldsymbol{e}_k^- = \boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k^- \tag{3.20}$$
$$\boldsymbol{e}_k = \boldsymbol{x}_k - \widehat{\boldsymbol{x}}_k \tag{3.21}$$
$$P_k^- = E\left[\boldsymbol{e}_k^- (\boldsymbol{e}_k^-)^T\right] \tag{3.22}$$
$$P_k = E\left[\boldsymbol{e}_k \boldsymbol{e}_k^T\right] \tag{3.23}$$

The whole approximation process is divided into two steps, *prediction* and *correction*. First, the prediction step will be discussed. The purpose of the step is to estimate new value $\boldsymbol{x}_{k+1}$ from the previous state. The a priori state estimate $\widehat{\boldsymbol{x}}_{k+1}^-$, Equation (3.24), and the a priori error covariance matrix P_{k+1}, Equation (3.25), are calculated. The a posteriori estimates from the previous step k are used for the computation of the a priori estimates.

$$\widehat{\boldsymbol{x}}_{k+1}^- = A_k \widehat{\boldsymbol{x}}_k + B \boldsymbol{u}_k \tag{3.24}$$
$$P_{k+1}^- = A_k P_k A_k^T + Q_k \tag{3.25}$$

The current measurement $\boldsymbol{z}_k$ is performed after the prediction step. For the measurement, the a priori estimation of the state $\widehat{\boldsymbol{x}}_k^-$ can be used and for example, the search space can be reduced or objects which are too far from the a priori estimation can be eliminated.

It is necessary to perform the correction step after the measurement. First, the Kalman gain K_k is calculated by (3.26). The Kalman gain is used to compute the a posteriori estimate and the error covariance, Equations (3.27) and (3.28). The new a posteriori estimate and covariance matrix will be used again to calculate the a priori values for the estimate of the state and the error covariance matrix in the consequent time $k+1$.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{3.26}$$
$$\widehat{\boldsymbol{x}}_k = \widehat{\boldsymbol{x}}_k^- + K(z_k - H_k \widehat{x}_k^-) \tag{3.27}$$
$$P_k = (I - K_k H_k) P_k^- \tag{3.28}$$

It is also important to mention the impact of the covariance matrix of the normal probability distribution R and the a priori error covariance P_k^-. If the covariance matrix R of the measurement noise approaches zero, the measurement affects more the resulting a posteriori estimate as the Kalman gain K_k approaches to H_k^{-1}. On the other hand, if the a priori error covariance matrix approaches to zero, the Kalman gain K_k also approaches to zero and the resulting a posteriori estimate $\widehat{\boldsymbol{x}}_k$ will be equal to $\widehat{\boldsymbol{x}}_k^-$.

The Kalman filter was presented in a general form without any values for different matrices like A_k, H_k and others in this section. The used values for all these matrices will be described in Section 4.2 where the proposed detection and tracking of vehicles is presented.

3.3 Classification

The last part of this chapter focuses on algorithms for classification of vehicles. First, the used descriptor will be presented and the classification algorithm follows.

3.3.1 Histograms of Oriented Gradients

This section describes an image descriptor called Histograms of Oriented Gradients [8]. The descriptor was originally used for human detection. However, it is possible to use the descriptor also for other objects. The descriptor can be used both for the detection and the classification of objects.

The basic idea of the descriptor computation is to calculate histograms of gradients of the classified object. The histograms are then merged into one vector, which can have many dimensions, and the vector is classified by a classification algorithm. The basic principle is simple; however, there are several parameters of the computation which will be discussed. The most important parts of the Histograms of Oriented Gradients computation are in Figure 3.3. The figure presents the algorithm in a way which can be used for the classification of an object when the location of the object is already known. If the algorithm is used also for the localization of the object, the descriptor is computed over different positions of the sliding window. The algorithm has the following steps.

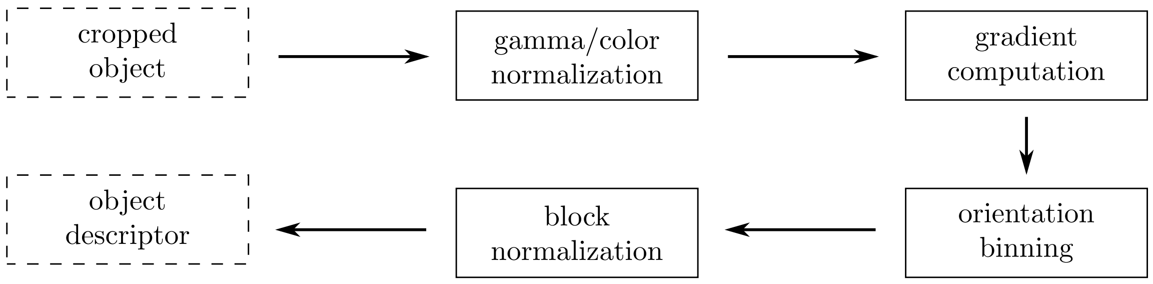


Figure 3.3: A basic pipeline for the Histograms of Oriented Gradients computation

Color Normalization The processed object is divided into multiple cells, which can have size 8×8 pixels. It is possible to normalize the colors in these cells; however, the authors of the descriptor claim that the normalization has only a modest effect on the performance [8]. On the other hand, the used color space is important and RGB color space outperforms greyscale.

Gradient Computation The best way of the gradient computation is to use simple kernels $[-1, 0, 1]$ to calculate derivates in x and y directions. The authors experimented with other kernels and the mentioned kernel provided the best results. For example uncentred kernel $[-1, 1]$, 3×3 Sobel kernel, 2×2 diagonal ones and 1D cubic corrected kernel $[1, -8, 0, 8, -1]$ were also evaluated. For color images, the gradient is calculated in all channels individually and the one with the largest norm is taken to the result.

Spatial/Orientation Binning The fundamental step is to create the histogram from gradient values in the cell. One histogram is created for each cell. The histogram is evenly spaced over $0 - \pi$ radians (*unsigned* gradient) or $0 - 2\pi$ radians (*signed* gradient). The contribution of the pixel to the histogram is weighted with respect to the magnitude of the gradient. The authors experimented with other weight functions, e.g. square or square root of the magnitude, however the magnitude itself gives the best results. In order to achieve the best results, the *unsigned* gradient with 9 bins should be used.

Descriptor Blocks The cells are grouped into the blocks for the normalization. The blocks can be either rectangular or circular. The authors suggest to use rectangular ones with size 2×2 cells. The final descriptor is the vector of all components from all the blocks. However, the blocks should have some overlapping, and therefore one cell contributes more times to the resulting descriptor. The overlapping can be for example one cell.

Normalization of the Descriptor Block The histograms are normalized in the blocks. The best results are provided by normalization with *L2-hys*, which is *L2-norm* (3.29) followed by limiting the maximum values of descriptor vector $\boldsymbol{v}$ to 0.2 and renormalizing after. The other tested normalization schemes were for example simple *L2-norm*, *L1-norm* and *L1-sqrt* (*L1-norm* followed by square root).

$$L2 - norm(\boldsymbol{v}) = \frac{\boldsymbol{v}}{\sqrt{\|\boldsymbol{v}\|_2^2 + \epsilon^2}} \tag{3.29}$$

The descriptor called Histograms of Oriented Gradients was presented in this section of the chapter and steps of the algorithm for its computation were also described. The descriptor can be used both for the detection and the classification of objects.

3.3.2 Support Vector Machine

The classifier Support Vector Machine will be described in this section. The classifier is widely used among many applications, not just Computer Vision problems. The description is based on book The Nature of Statistical Learning Theory [41]. Other notes about multi-class SVM are mentioned in technical report by Weston and Watkins [45] and other implementation details are described in paper written by Crammer and Singer [7].

The Support Vector Machine can be used for several applications, not only classification. For example, another possible is usage of the Support Vector Machine is regression. The key idea for the classification is to divide a feature space by a hyperplane which maximizes margin between the hyperplane and the closest vectors from all classes.

Let $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_l, y_l)\}$ be a finite set of pairs, where $\boldsymbol{x}_i \in \mathbb{R}^n$ and $y_i \in \mathbb{Y}$ for all $1 \leq i \leq l$. Vectors $\boldsymbol{x}_i$ are the classified objects and y_i is the class of the objects. A classifier is a function $f : \mathbb{R}^n \to \mathbb{Y}$ mapping an object to a class. Generally, the set $\mathbb{Y}$ can be any finite set. However, it is convenient to suppose that $\mathbb{Y}$ is a proper finite subset of integers. If it is not the case, it is always possible to create a mapping from $\mathbb{Y}$ to a subset of the integers. Let us also denote the number of elements of set S as l in the following text.

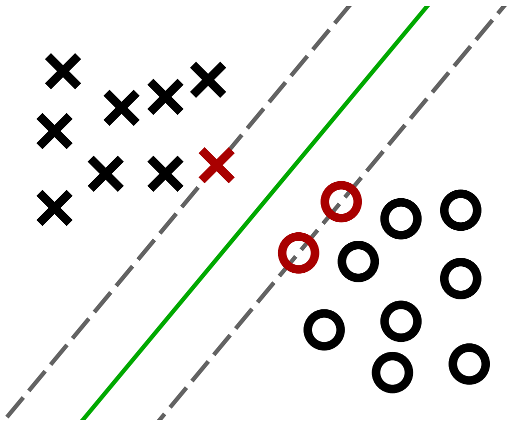


Figure 3.4: Maximal margin hyperplane (green), support vectors (red)

First, the linear Support Vector Machine will be described. Second, the linear SVM will be extended to the non-linear one. At last, multi-class SVM will be presented.

Linear SVM

Linear binary Support Vector Machines classify only objects from two classes. It is useful for the notation that the classes are -1 and 1, and therefore $\mathbb{Y} = \{-1, 1\}$. Let us suppose that the objects are linearly separable. The non-separable case will be discussed later.

The basic idea is to classify the objects by a hyperplane. The hyperplane can be used for the classification in the form of vector $\boldsymbol{w}$ and scalar threshold b. Vector $\boldsymbol{x}$ is then classified as the result of $\boldsymbol{w} \cdot \boldsymbol{x} - b$ where the dot product is defined by the following equation.

$$\boldsymbol{w} \cdot \boldsymbol{x} = \boldsymbol{w}^T \boldsymbol{x} \tag{3.30}$$

A hyperplane is called as the optimal hyperplane (the maximal margin hyperplane) if and only if, the vectors $\boldsymbol{x}_i$ are separated without errors, Equations (3.31) and (3.32) or (3.33), and the distance between the closest vector to the hyperplane is maximal. An example of the optimal hyperplane is in Figure 3.4. The closest vectors to the hyperplane are called support vectors (red in Figure 3.4). Let us denote the set of indices i of the support vectors in the set S as SV.

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{x}_i - b &\geq +1 \quad \text{if } y_i = +1 && (3.31)\\ \boldsymbol{w} \cdot \boldsymbol{x}_i - b &\leq -1 \quad \text{if } y_i = -1 && (3.32)\\ y_i \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_i - b) &\geq 1 && (3.33) \end{aligned}$$

In order to find the optimal hyperplane it is necessary to solve a quadratic programming problem. The function to minimize is $\Phi(\boldsymbol{w})$, Equation (3.34). The inequality constraints which a solution $\boldsymbol{w}$ has to satisfy are in Equation (3.35).

$$\begin{aligned} \Phi(\boldsymbol{w}) &= \frac{1}{2}(\boldsymbol{w} \cdot \boldsymbol{w}) && (3.34)\\ 1 &\leq y_i \cdot (\boldsymbol{w} \cdot \boldsymbol{x}_i - b) \qquad i = 1, \ldots, l && (3.35) \end{aligned}$$

The Lagrange functional can be used to solve the quadratic programming problem. Using this approach, the Lagrange multipliers α_i ($i = 1, \dots, l$) are searched instead of the vector $\boldsymbol{w}$. However, the vector of the optimal hyperplane can be obtained as a linear combination of the vectors $\boldsymbol{x}_i$, Equation (3.36). Moreover, only the Lagrange multipliers for the support vectors will be different from zero. Hence, it is possible to express the vector $\boldsymbol{w}$ as their combination (3.37). The scalar threshold b is then computed by equation (3.38), while $\boldsymbol{x}^*(-1)$ is a support vector for class -1 and $\boldsymbol{x}^*(1)$ is a support vector for class 1.

$$\boldsymbol{w} = \sum_{i=1}^{l} y_i \alpha_i \boldsymbol{x}_i \tag{3.36}$$

$$\boldsymbol{w} = \sum_{i \in SV} y_i \alpha_i \boldsymbol{x}_i \tag{3.37}$$

$$b = \frac{1}{2} \left(\boldsymbol{w} \cdot \boldsymbol{x}^*(1) + \boldsymbol{w} \cdot \boldsymbol{x}^*(-1) \right) \tag{3.38}$$

It is still necessary to solve the quadratic programming problem after the modifications. However, different equations and constraints are used. Therefore, it is required to minimize function $W(\alpha)$ (3.39) with non-negative α_i (3.40). The solution also has to satisfy the constraint (3.41).

$$W(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j (\boldsymbol{x}_i \cdot \boldsymbol{x}_j) \tag{3.39}$$

$$\alpha_i \geq 0 \qquad i = 1, \dots, l \tag{3.40}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{3.41}$$

The resulting classification function f with the optimal hyperplane is expressed by Equation (3.42). As one can notice, it is necessary to keep the scalar threshold b, support vectors and their Lagrange multipliers α and classes y.

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i \in SV} y_i \alpha_i (\boldsymbol{x}_i \cdot \boldsymbol{x}) - b \right) \tag{3.42}$$

If the vectors $\boldsymbol{x}$ are not separable by a hyperplane, the quadratic programming problem is slightly changed. The only modification is in constraint (3.40). The other constraint and the minimization function stays same also as the classification function (3.42).

Non-linear SVM

It is also possible to use the non-linear SVM for the classification. However, the non-linearity is not achieved by changing the separation hyperplane to a different separation element. Instead, the input vectors $\boldsymbol{x}$ are mapped into a higher dimensional space. On the other hand, the mapping has to be chosen a priori. The description of the mapping and the modifications in the training and classification follows.

As one can notice, input vectors $\boldsymbol{x}$ are used only in the form of dot product in the process of the training and classification. Hence, it is possible to replace the dot product by a kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ in every equation. This substitution also does not change the quadratic programming problem to a problem of a higher degree, as the variables of the problem are the Lagrangian multipliers.

As it was mentioned, the only difference to the linear SVM is the usage of the Kernel function instead of the dot product. Therefore, it is necessary to modify the equations which were using the dot product of the vectors and use the kernel functions. Hence, the minimized functions $W(\alpha)$ is changed, equation for computation the threshold b and the classification functions f itself is also modified. The resulting forms of these functions are the following:

$$\begin{aligned} W(\alpha) &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) && (3.43) \\ b &= \frac{1}{2} \big(K(\boldsymbol{w}, \boldsymbol{x}^*(1)) + K(\boldsymbol{w}, \boldsymbol{x}^*(-1)) \big) && (3.44) \\ f(x) &= \operatorname{sgn} \left(\sum_{i \in SV} y_i \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}) - b \right) && (3.45) \end{aligned}$$

There are several types of kernel functions which can be used. For example, the polynomial kernel function with degree d, Eq. (3.46). There is also the radial basis function, which is parametrized by a scalar γ. The RBF function is expressed in Equation (3.47).

$$\begin{aligned} K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^d && (3.46) \\ K_\gamma(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \exp\left(-\gamma |\boldsymbol{x}_i - \boldsymbol{x}_j|^2\right) && (3.47) \end{aligned}$$

Multi-class SVM

There are two ways how to classify an object into one of the multiple classes with SVM. The first one is to create a binary SVM for each of the classes, which will classify the object either as class m or not-class m. Hence, it is necessary to run the binary classifier up to k times, if the classification is done into k classes. The second approach uses only one SVM classifier which is able to classify the object into multiple classes. The first approach requires only a binary SVM, which was already presented. The SVM for multi-class classification will be described.

Let us suppose that there is l ($|S| = l$) training vectors and k classification classes ($|\mathbb{Y}| = k$). Let us also denote variables i, j as iteration variables over training pairs ($i, j \in \{1, \ldots, l\}$) and m as an iteration variable over classes ($m \in \{1, \ldots, k\}$). The sum of Lagrangian multipliers for a training pair i for all classes is labeled as A_i, Equation (3.49). The membership variable c_i^m is equal to 1 if and only if the training vector $\boldsymbol{x}_i$ belongs to class m.

$$\begin{aligned} c_i^m &= \begin{cases} 1 & y_i = m \\ 0 & y_i \neq m \end{cases} && (3.48) \\ A_i &= \sum_{m=1}^{k} \alpha_i^m && (3.49) \end{aligned}$$

The solved quadratic programming problem is more complex than the problem for binary SVM. For example, there is a Lagrangian multiplier α_i^m for each training vector $\boldsymbol{x}_i$ and class $m \in \mathbb{Y}$. The function to minimize is

$$W(\alpha) = 2 \sum_{i,m} \alpha_i^m + \sum_{i,j,m} \left(-\frac{1}{2} c_j^{y_i} A_i A_j + \alpha_i^m \alpha_j^{y_i} - \frac{1}{2} \alpha_i^m \alpha_i^j \right) K(\boldsymbol{x}_i, \boldsymbol{x}_j). \qquad (3.50)$$

And the constraints under which the function has to be minimized are the following.

$$\sum_{i=1}^{l} \alpha_i^m = \sum_{i=1}^{l} c_i^m A_i \qquad m \in \{1, \dots, k\} \tag{3.51}$$

$$\begin{gathered} 0 \leq \alpha_i^m \leq C, \quad \alpha_i^{y_i} = 0 \\ i \in \{1, \dots, l\}, \quad m \in \{1, \dots, k\} \setminus \{y_i\} \end{gathered} \tag{3.52}$$

The final classification function is stated as follows.

$$f(\boldsymbol{x}) = \arg\max_m \left(\sum_{i=1}^{l} (c_i^m A_i - \alpha_i^m) K(\boldsymbol{x}_i, \boldsymbol{x}) - b_m \right) \tag{3.53}$$

In this section the Support Vector Machine classifier was presented. The formulae for finding the optimal hyperplane and the classification itself were described in several versions of the classifier (namely linear, non-linear, multi-class).

Chapter 4

Proposed Traffic Analysis System

This chapter presents the proposed system for traffic analysis. The main goal of the system is to be able to generate statistics about traffic flow on a monitored road. The requirements on the proposed system are following.

- The system has to work fully automatically in a real time. No manual input can be used.
- The system should be able to detect, track and count vehicles.
- Determine types of vehicles (*personal*, *van*, *truck* or others).
- Detect lanes, lane-dividing lines and segment vehicles by their membership to the lanes. Also, the direction of vehicles should determined and vehicles passing in wrong way can be detected.
- Measure speed of passing vehicles. This is the most challenging task as it has to be done in a fully automated way without any manual calibration.

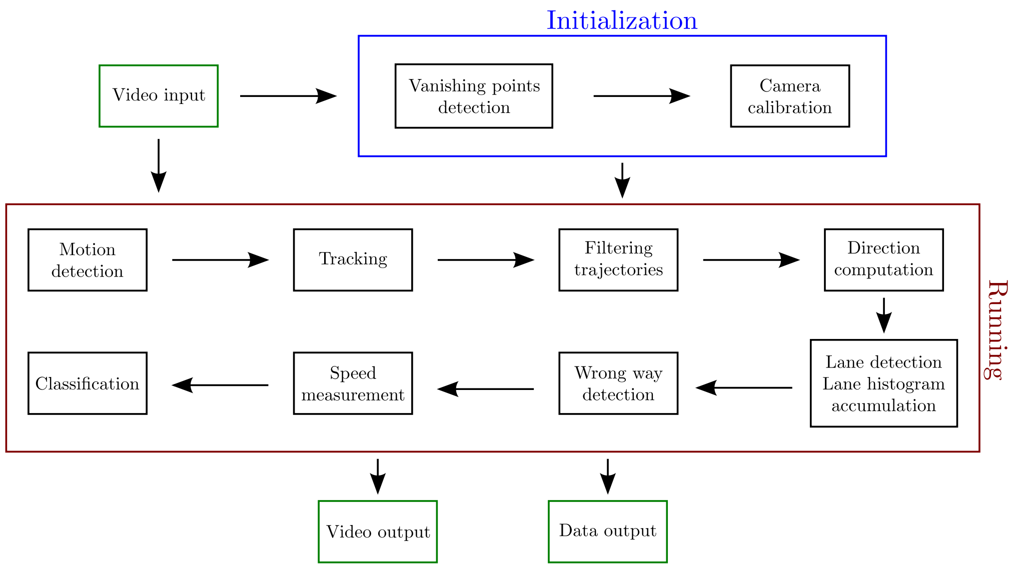


Figure 4.1: Pipeline of processing of the input video stream by the proposed system.

The overall design of the proposed system is shown in Figure 4.1. First, the initialization of the system is performed. The main purpose of the initialization is to calibrate the camera by detecting vanishing points of the scene. As the figure shows, vehicles are detected with motion detection and tracked. Observed passing objects which are detected and tracked are also filtered in order to detect vehicles more precisely. Then, the direction of each vehicle can be computed, it can be assigned to a lane, and the situation when the vehicle is passing in wrong way can be detected. Last but not least, the speed of the vehicle is measured and the class is assigned to the vehicle.

Two papers describing the proposed system were published so far. One paper appeared on the EEICT student conference [34] and the other one on CESCG seminar [35]. Also, I participated on papers which describe the calibration of the camera for traffic surveillance [11] and three-dimensional understanding of traffic scenes [12] which is used for the measurement of vehicles' speeds.

4.1 Initialization

The main goal of the initialization is to fully automatically calibrate the camera. The calibration is obtained by detected vanishing points. The algorithm for the calibration is based on paper by Dubská et al. [11], which I co-authored.

The vanishing point of the direction parallel to the vehicles' movement is denoted as the first vanishing point. The second vanishing point has the perpendicular direction to the movement of vehicles and the third vanishing point is perpendicular to the ground plane. Examples of detected vanishing points and their visualisation is shown in Figure 4.2. The vanishing points are visualised by arrows which are directed towards the vanishing points and are uniformly distributed in image as location of the arrows is irrelevant.

For visualisation of the first vanishing point, red arrows are used and green arrows are used for the second vanishing point. The third vanishing point is denoted by blue arrows. Horizon line, which connects the first and second vanishing point, is also drawn in Figure 4.2.

For the detection of the vanishing points, several assumptions about parameters of video camera are used. It is assumed that the camera has zero skew and square pixels. Also, it is assumed that the principal point is in the center of the image. From our experiences with video cameras, these assumptions are usually satisfied. For the detection of the first vanishing point, it is also assumed that vehicles follow more or less straight trajectories.

The main task of the calibration is to compute the focal length f from the detected vanishing points as other intrinsic parameters (skew, aspect ratio and position of principal point) of the camera are constrained. Description of the detection of the first and second vanishing point follows, together with the explanation of the computation of the third vanishing point and the focal length from the first two vanishing points. However, the accumulation of lines into so called *diamond space* will be described first, as it is used for the detection of the vanishing points.

It should be noted that the processed video is downsampled to $\sim 12.5\,\mathrm{FPS}$ in order for all movement and motion measurements to be stable and detectable. Therefore, only every s^{th} frame is processed. The amount of skipped frames s is computed according to (4.1) where fps is the real framerate of the video.

$$s = \left\lfloor \frac{fps}{12.5} \right\rfloor \tag{4.1}$$

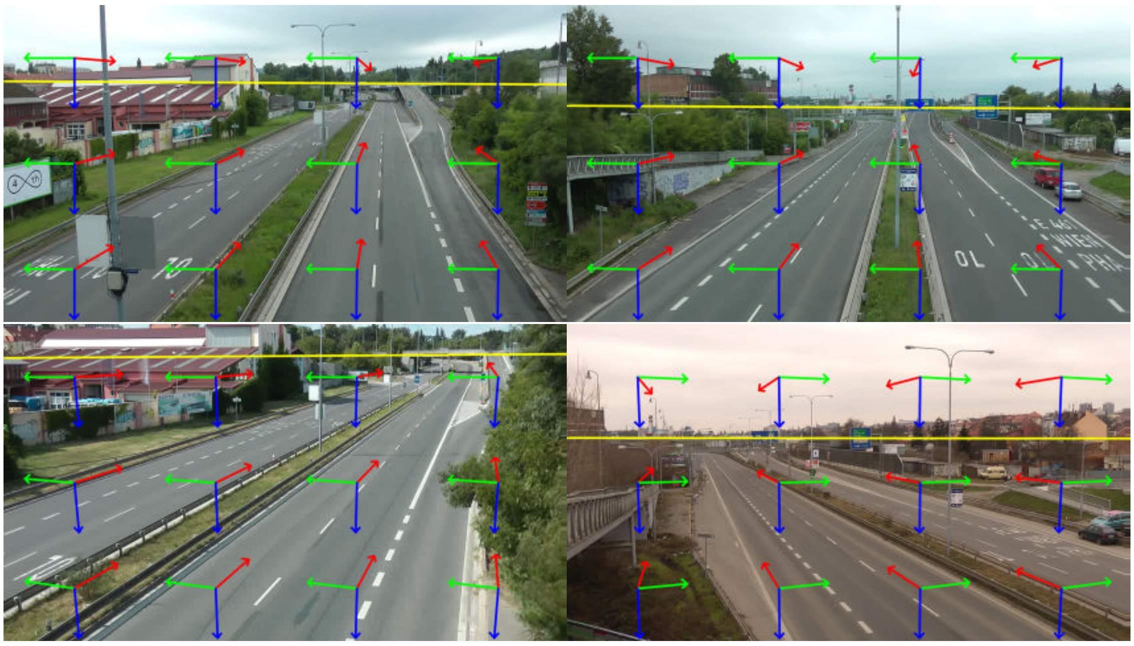


Figure 4.2: Detected vanishing points. Red arrows are directed toward the first vanishing point and green ones to the second vanishing point. The third vanishing point is denoted by blue arrows. Also, the horizon line which connects the first and second vanishing point is drawn by yellow color.

4.1.1 Accumulation of Lines into Diamond Space

Algorithms for the detection of the first two vanishing points have in common that it is necessary to localize an intersection of a high amount of lines. The diamond space described by Dubská and Herout [10] is used for this task.

The diamond space is used for representation of lines in a finite space. The lines from cartesian coordinates are mapped into the diamond space as a piece-wise linear polyline. The transformation into diamond space is based on a transformation from cartesian coordinates into parallel coordinates which is performed twice.

Each line (a, b, c) is transformed into a polyline in the diamond space. The polyline is defined by four points which are obtained by Equation (4.2) where sgn denotes non-zero signum. The parts of the polyline are defined by the consequent points in (4.2) and are one by one rasterized into the diamond space. Examples of the diamond spaces are shown in Figure 4.5.

$$\begin{aligned} &\alpha = \operatorname{sgn}(ab), \quad \beta = \operatorname{sgn}(bc), \quad \gamma = \operatorname{sgn}(ac) \\ &(a,b,c) \rightarrow \left[\frac{\alpha a}{c+\gamma a}, \frac{-\alpha c}{c+\gamma a}\right], \left[\frac{b}{c+\beta b}, 0\right], \left[0, \frac{b}{a+\alpha b}\right], \left[\frac{-\alpha a}{c+\gamma a}, \frac{\alpha c}{c+\gamma a}\right] \end{aligned} \tag{4.2}$$

Each point in the diamond space in homogenous coordinates can be transformed back to cartesian homogenous coordinates by (4.3). The distances of parallel axes which were used during the transformation from the cartesian coordinates to the parallel ones are denoted by d for the first transformation and D for the second one. Both theses distance can be equal to 1.

$$[x, y, w]_d \rightarrow [Dy, \operatorname{sgn}(x)dx + \operatorname{sgn}(y)Dy - dDw, x]_o \tag{4.3}$$

The point which belongs to the maximal amount of lines can be easily found in the diamond space by localization of the global maximum in the diamond space. It is also possible to detect the point with sub-pixel accuracy with usage of weighed mean and points which are around the global maximum.

4.1.2 First Vanishing Point Detection

Corner-features tracking is used for the detection of the first vanishing point. Hence, Good Features to Track [32] are detected in the video stream and KLT tracker [39] is used for the tracking of the corner features. Detected motion of the tracked features is extended into a line which is defined by image points (x_t, y_t) and (x_{t+1}, y_{t+1}) which are positions of the feature in frame t and $t+1$. Examples of the tracked points are shown in Figure 4.3

All these lines are accumulated into diamond space until the initialization is terminated. The initialization is terminated when the global maximum of the diamond space is bigger then a predefined threshold and therefore, a sufficient number of lines was accumulated. Afterwards, the coordinates of the global maximum in the diamond space are transformed into coordinates of the vanishing point in the image plane.

Examples of diamond spaces for the detected first vanishing points from Figure 4.2 are shown in the top row of Figure 4.5.

4.1.3 Second Vanishing Point Detection

The detection of the second vanishing point is performed after the first vanishing point was successfully detected because the first vanishing point is used for constraining the position of the second vanishing point.

The diamond space is also used for the detection of the second vanishing point. Background edge model is used to detect edges on moving objects – probable vehicles. The background model is updated with each frame in order to eliminate slow lighting changes. The edge background model stores for each pixel the confidence score of occurrence of an oriented edge. Eight bins are used for each pixel to store likelihoods for different orientations. Gradient $G(i,j)$ of edge in pixel (i,j) in image I is computed by convolution with Sobel kernels K_y and K_x. The edge background model for a pixel (i,j) is updated at correct bin for edge orientation $G(i,j)$.

$$\begin{aligned}
K_y &= \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 2 & 8 & 12 & 8 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -8 & -12 & -8 & -2 \\ -1 & -4 & -6 & -4 & -1 \end{bmatrix} && (4.4)\\
K_x &= K_y^T && (4.5)\\
G_x &= I * K_x && (4.6)\\
G_y &= I * K_y && (4.7)\\
G(i,j) &= \operatorname{atan2}(G_y(i,j), G_x(i,j)) && (4.8)
\end{aligned}$$

For each image point, several conditions are evaluated and if the conditions are satisfied, pixel (i,j) and its orientation $G(i,j)$ defines a line which contains the point (i,j) and is perpendicular to the orientation $G(i,j)$. The line is then accumulated into the diamond space. If a line should be accumulated to the diamond space, the point which defines the line has to meet the following conditions:

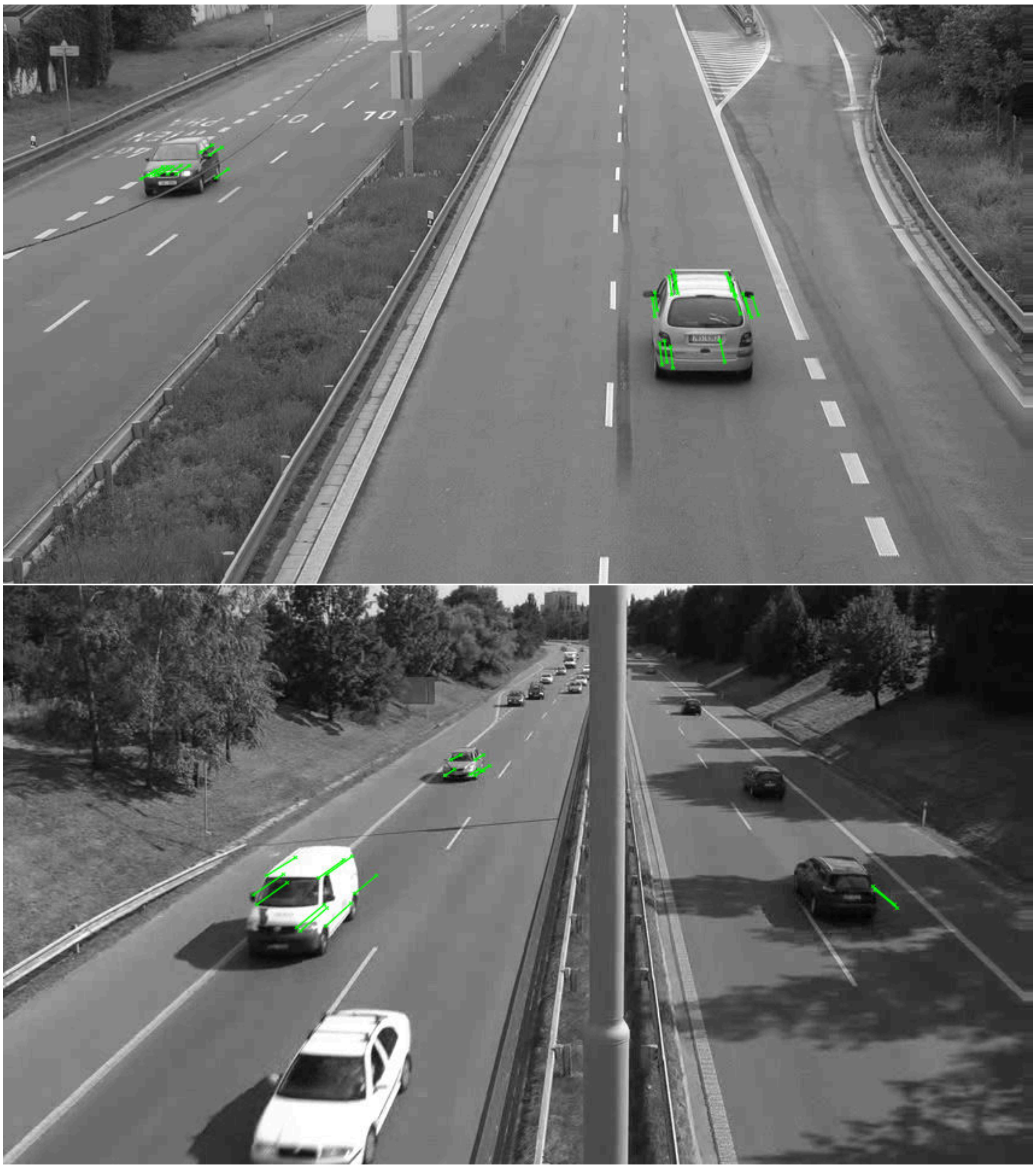


Figure 4.3: Detected and tracked corner features for the first vanishing point detection.

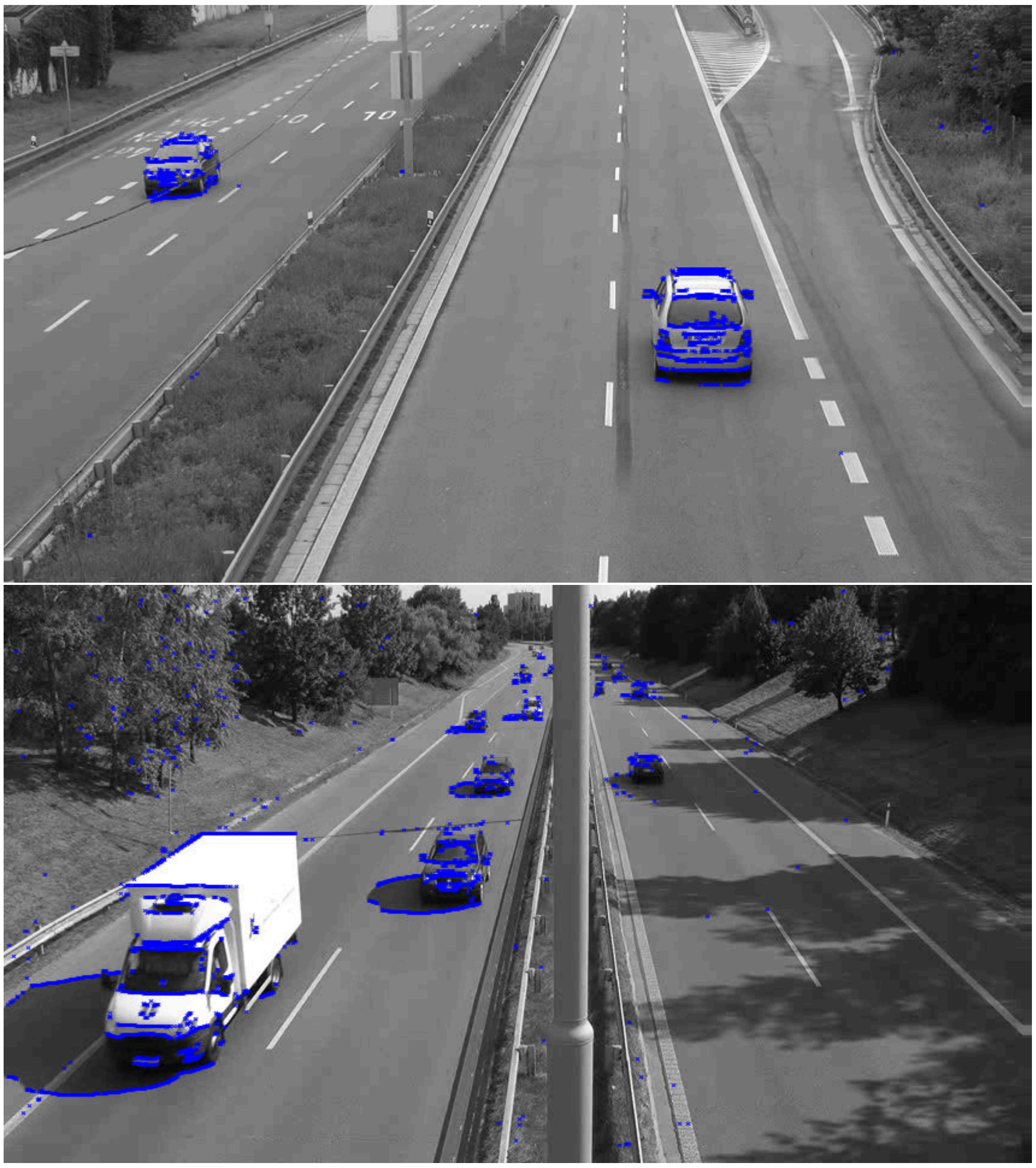


Figure 4.4: Examples of points which define lines which are accumulated into the diamond space for the second vanishing point detection.

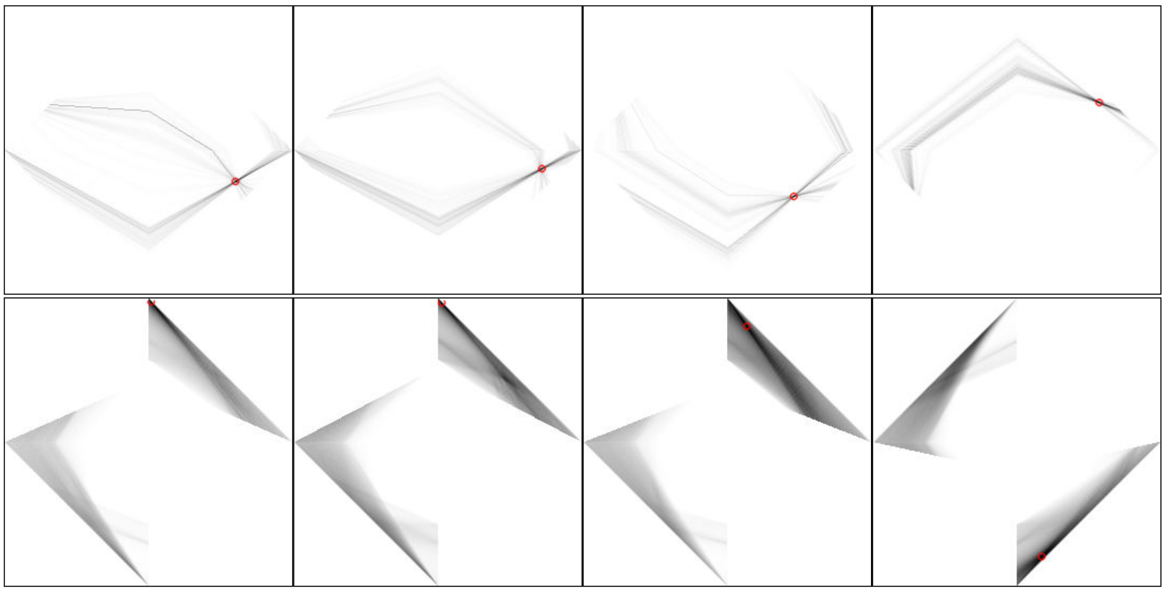


Figure 4.5: Top row presents diamond spaces for the first vanishing points in Figure 4.2. Images in the bottom row represent masked diamond spaces for the second vanishing points.

1. It has to be detected as an edge by Canny edge detector [4].
2. The confidence that the point belongs to the background has to be lower than a predefined threshold.
3. Magnitude of the gradient has to be higher than a predefined threshold.
4. The line which should be accumulated to the diamond space must not be directed towards the first vanishing point or must not be vertical. Both conditions have some level of tolerance.

The second vanishing point is detected if the global maximum in the diamond space is higher than a predefined threshold. However, the global maximum is searched in a masked diamond space as the location of the second vanishing point is constrained by the first vanishing point. Line which contains the principal point and is perpendicular to line which is defined by principal point and the first vanishing point has to separate the second vanishing point from the first one. If the second vanishing point would be on the same side of the line as the first one, the expression under square-root (4.9) would be negative.

4.1.4 Third Vanishing Point and Focal Length Computation

If the first two vanishing points are known and it is assumed that the principal point is in the center of the image, it is possible to compute the focal length and the third vanishing point. Also, the world coordinates of the vanishing points can be computed.

The focal length f is computed according to (4.9) where $U = (u_x, u_y)$ denotes the first vanishing point and $V = (v_x, v_y)$ defines the second one. Also, the principal point is denoted by $P = (p_x, p_y)$. World coordinates of vanishing points are denoted by U', V' and W' for the third vanishing point. The world coordinates of the third vanishing point are computed according to (4.13) and the coordinates can be transformed into the image coordinates by (4.14).

$$
\begin{aligned}
f &= \sqrt{-(U-P)\cdot(V-P)} && (4.9)\\
U' &= (u_x, u_y, f) && (4.10)\\
V' &= (v_x, v_y, f) && (4.11)\\
P' &= (p_x, p_y, 0) && (4.12)\\
W' &= (U'-P')\times(V'-P') && (4.13)\\
W &= \left(\frac{w'_x}{w'_z} f + p_x, \frac{w'_y}{w'_z} f + p_y\right) && (4.14)
\end{aligned}
$$

After the focal length and the third vanishing point are computed, the camera is calibrated up to scale as it is not possible to determine how far the road plane is from the camera. Examples of the detected vanishing points are shown in Figure 4.2.

4.2 Detection and Tracking

The vehicle detection is based on motion detection in the video scene. Mixture of Gaussians background subtraction [36, 46] is used for the motion detection. Also, shadow elimination [15] is used for higher accuracy of the motion detection. Noise in the detected motion is removed by morphological opening followed by morphological closing. Detected connected components are considered to be a potential vehicle. The process of the motion detection is shown in Figure 4.6. The motion detection approach was selected mainly for its speed as it can run without any problem in real time.

According to Newton's Laws of Motion [13] position of a rigid object at time t with acceleration a can be computed according to (4.15), where s_0 and v_0 is the initial position and speed. Therefore, if Kalman filter is used [20] with state vector $\boldsymbol{x}_k = (x, y, v_x, v_y)^T$

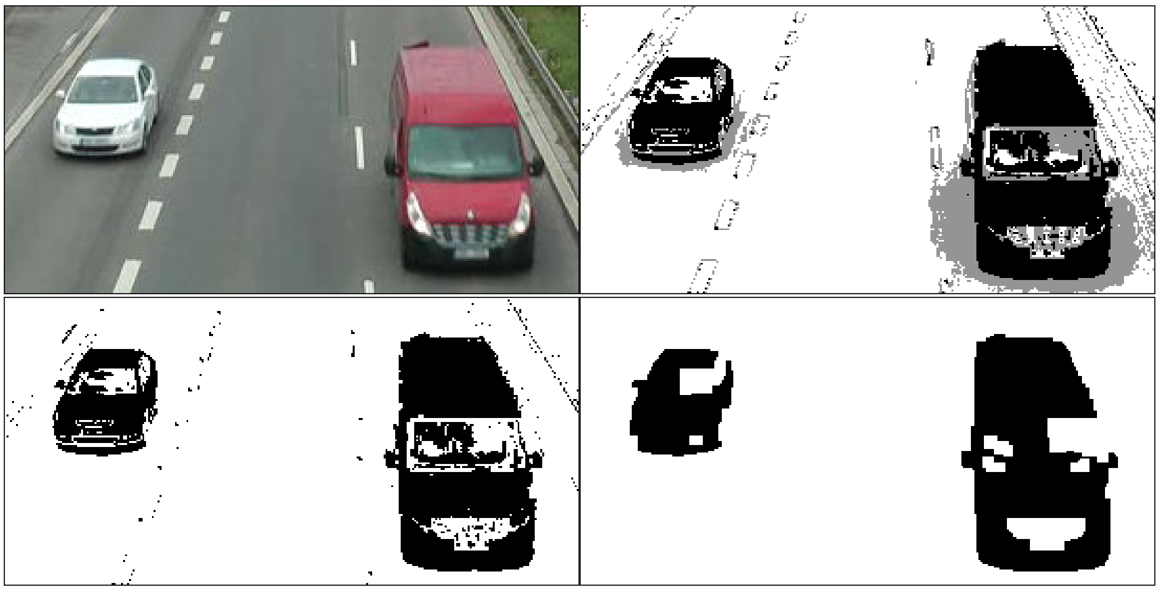


Figure 4.6: Process of motion detection. The top left image represents the original, the next one shows the detected motion with shadows which were removed in the third image. The result after the morphology opening and closing is shown in the bottom right image.

which contains the current position of a car and its velocity in image coordinates, it is possible to compute state vector in next step by (4.17).

$$s = s_0 + v_0 t + \frac{1}{2}at^2 \tag{4.15}$$

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.16}$$

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + a \begin{bmatrix} \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} \\ \Delta t \\ \Delta t \end{bmatrix} \tag{4.17}$$

However, it is useful to consider the acceleration of vehicles as the noise of the linear process. If it is assumed that the mean acceleration is equal to 0 and the acceleration has standard deviation σ_a, it is possible to compute the covariance matrix [42] by (4.18). Hence, the stochastic difference equation of Kalman filter is (4.19).

$$\begin{aligned} Q &= \left(\sigma_a \begin{bmatrix} \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} \\ \Delta t \\ \Delta t \end{bmatrix}\right) \cdot \left(\sigma_a \begin{bmatrix} \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} \\ \Delta t \\ \Delta t \end{bmatrix}\right)^T \\ &= \sigma_a^2 \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^4}{4} & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t^2 \\ \frac{\Delta t^3}{2} & \frac{\Delta t^3}{2} & \Delta t^2 & \Delta t^2 \end{bmatrix} \end{aligned} \tag{4.18}$$

$$\boldsymbol{x}_{k+1} = A\boldsymbol{x}_k + \mathcal{N}(\boldsymbol{0}, Q) \tag{4.19}$$

The measurement used for the tracking of vehicles by the Kalman filter uses the current positions of vehicles $(x, y)^T$ and the equation for the measurements is expressed by (4.22).

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.20}$$

$$R = \begin{bmatrix} \sigma_m^2 & \sigma_m^2 \\ \sigma_m^2 & \sigma_m^2 \end{bmatrix} \tag{4.21}$$

$$\boldsymbol{z}_k = H\boldsymbol{x}_k + \mathcal{N}(\boldsymbol{0}, R) \tag{4.22}$$

Several conditions are used for matching an object in the consequent frame to its predicted location. The first condition states that the matched object must have similar colors. This condition is enforced by correlating histograms of objects in HSV color space. The second and last condition is that the center of matched object must be inside of so called *matching rectangle*. The predicted location of a car is the center of this matching rectangle and the longer side of the rectangle is directed towards the first vanishing point, as it is shown in Figure 4.7, and the matching rectangle has size 30×15 pixels. This condition is built on the assumption that the vehicle is going either in direction towards the vanishing point or from the vanishing point, and therefore it is expected that in this direction can

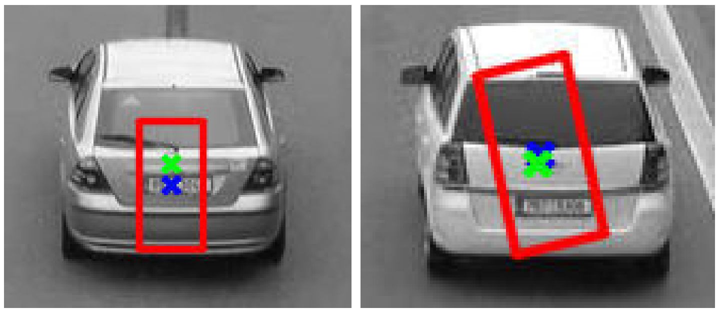


Figure 4.7: Examples of matching rectangles (red) for predicted object location (blue). The actual center of the detected connected component is drawn by green color. The figure shows that the bigger side of the rectangle is directed to the first vanishing point.

be higher displacement from the predicted location. Lastly, the closest connected component which meets the conditions presented above is found for each object and its predicted location in the consequent frame.

When a match is not found in several consequent frames, the tracked object is removed from the pool of tracked objects. Several filters are used for determining if the object should be accounted in the statistics of passed cars. The trajectory of the object is approximated by a line using least-squares approximation. After that, the distance of the first vanishing point from the line is measured. Let us denote this distance as d_{vp}. Also, the ratio r, Eq. (4.23),

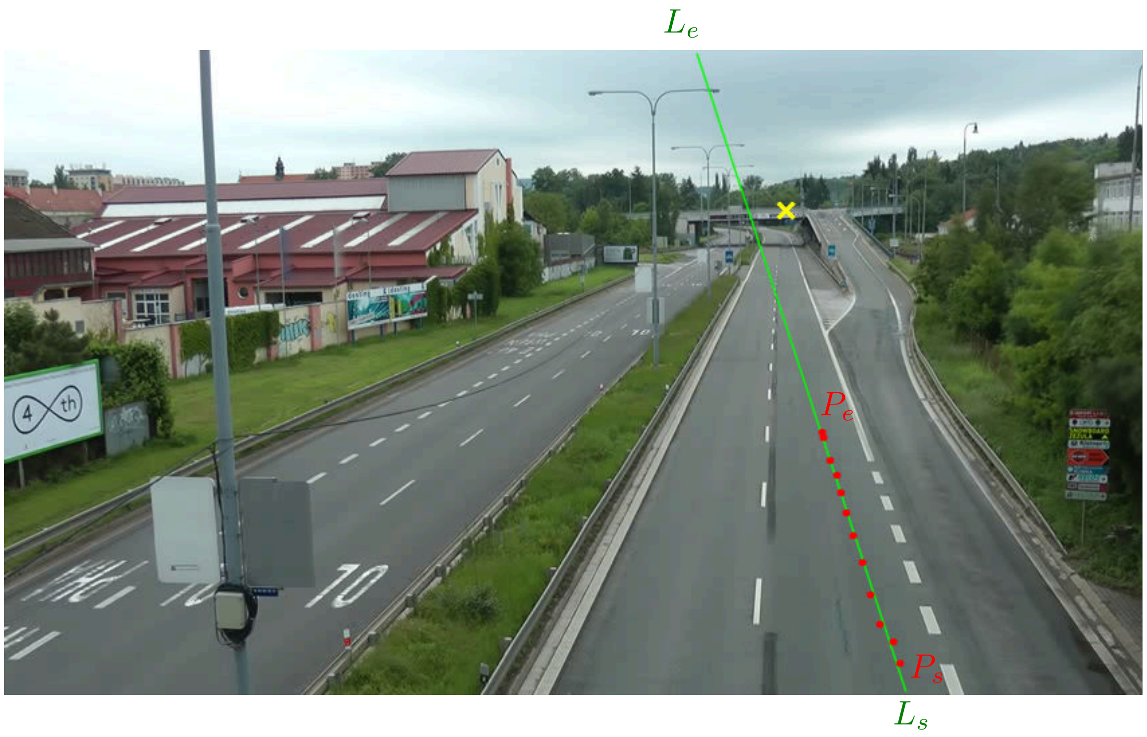


Figure 4.8: Measured distances for a passed object. The distance between approximated line (green) and the first vanishing point (yellow) is measured. Also, the distance between the first and last (P_s, P_e) point of the track of a vehicle is measured. The maximal distance which is possible to pass with a given trajectory is also measured (distance of L_s and L_e).

between passed distance and maximal possible distance which an object can pass in the given trajectory is measured, as it is shown in Figure 4.8. The object is accounted in the statistics as a vehicle if the *acc* variable is equal to 1, Equation (4.24), where t_{vp} and t_r are predefined thresholds.

$$r = \frac{\|P_e - P_s\|}{\|L_e - L_s\|} \tag{4.23}$$

$$acc = \begin{cases} 1 & d_{vp} \leq t_{vp} \text{ and } r \geq t_r \\ 0 & \text{otherwise} \end{cases} \tag{4.24}$$

4.3 Three-Dimensional Bounding Boxes

If all the three vanishing points are detected, the so called three-dimensional bounding boxes can be computed for a vehicle or any other object. The computation of the three-dimensional bounding box for a contour is far more advanced than two-dimensional. Figure 4.9 shows the difference between these two bounding boxes. As the figure presents, the two-dimensional one is just a tangent rectangle of the detected vehicle. On the other hand, the three-dimensional bounding box is a cuboid which also surrounds the detected vehicle; however, the cuboid defines the dimensions of the vehicle or the position of the base of the cuboid can be used further for more precise detection of lanes.

4.3.1 Computation of Tangent Lines to a Vehicle Blob

In order to be able to compute the three-dimensional bounding boxes, it is necessary to compute tangent lines to the contour of a vehicle from the vanishing points of the scene. Also, the detection of the tangents has to be effective as the lines are computed for each vehicle in every frame of the processed video. Furthermore, it is necessary to be able to distinguish the left tangent line and right tangent line. The left tangent line is a tangent line from a point P which is on the left side and the right tangent line is on the right side. See Figure 4.10 for examples of tangent lines from the detected vanishing points.

The proposed algorithm for the detection of the left and right tangent lines is written as Algorithm 4.1. The algorithm for the detection runs in $\mathcal{O}(n)$ with respect to the size of the contour C. The algorithm iterates over all points $I \in C$ and computes the angle

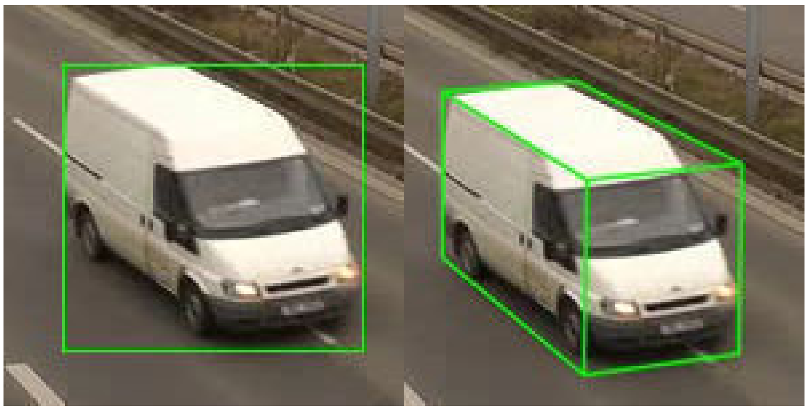


Figure 4.9: Difference between a two-dimensional bounding box (left) and a three-dimensional one (right).

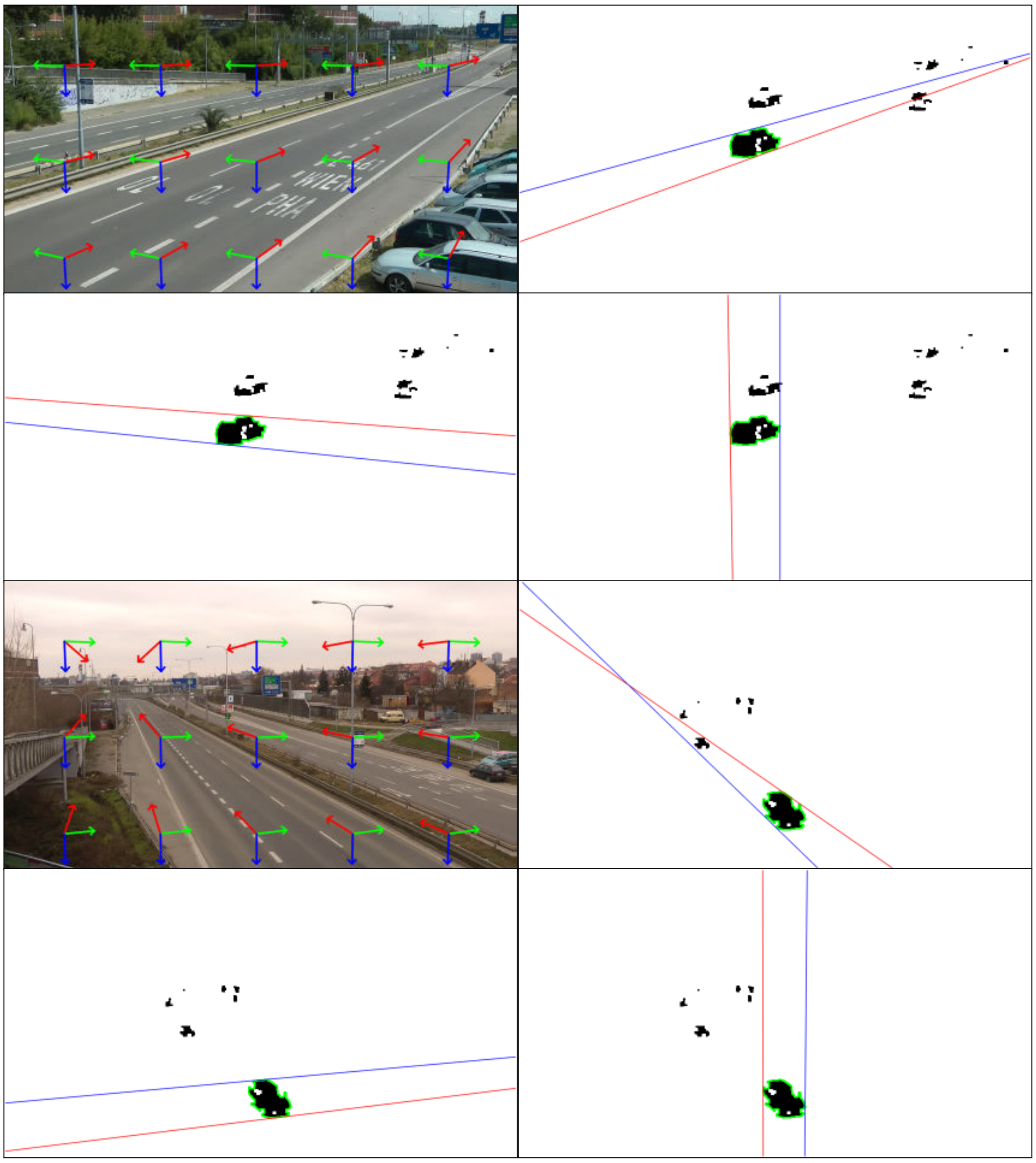


Figure 4.10: Examples of tangent lines for different scenes and vanishing points locations. The left tangent line is always drawn by red color and the right one has blue color. The images are in the following order: vanishing points location, tangent lines from the first vanishing point, from the second one and lastly from the third one.

Algorithm 4.1 Computation of the left and right tangent line.

```
Input: Contour $C = \{I_1, I_2, \dots, I_N\}$, point $P = (p_x, p_y)$
Output: $leftTangetLine, rightTangentLine$ for contour $C$ from point $P$
 1: $minAngle \leftarrow \infty$
 2: $maxAngle \leftarrow -\infty$
 3: $leftTangentLine, rightTangentLine \leftarrow \mathbf{0}$
 4: $rightMost \leftarrow \mathrm{findRightMostPoint}(C)$
 5: for all $I \in C$ do
 6:     $currentLine \leftarrow \overleftrightarrow{PI}$
 7:     $angle \leftarrow \mathrm{atan2}(I_y - P_y, I_x - P_x)$
 8:     if $angle < 0 \wedge rightMost_x < p_x$ then
 9:         $angle \leftarrow angle + 2\pi$
10:     if $angle < minAngle$ then
11:         $minAngle \leftarrow angle$
12:         $leftTangentLine \leftarrow currentLine$
13:     if $angle > maxAngle$ then
14:         $maxAngle \leftarrow angle$
15:         $rightTangentLine \leftarrow currentLine$
```

between half-line $\overrightarrow{PI}$ and positive half of the axis x of the image. Afterwards, the point I_l with the minimal angle is the left tangent point and the right tangent point I_r has the maximal angle.

However, as the function `atan2` returns values in interval $\langle -\pi, \pi \rangle$, it is necessary to perform some transformations of the computed angles if the point P from which the tangent lines are casted, is on the right side of the contour C. As it is written on lines 8 and 9 of the algorithm, if the point P has a higher x coordinate and the angle is below zero, value 2π is added to the angle. Therefore, the angles are transformed into interval $\langle 0, 2\pi \rangle$. Examples of the detected lines for all three vanishing points are shown in Figure 4.10.

4.3.2 Computation of Three-Dimensional Bounding Boxes

The three-dimensional bounding box can be computed after the tangent lines are detected. However some notation has to be introduced in order to be able to describe the process of computation of the bounding box. Hence, let us denote the left and right tangent line from vanishing point X as t_l^X and t_r^X. Also, points U, V and W denote the first, second and the third vanishing point.

The process of computation of the bounding box follows and it is also shown in Figure 4.11. It should be noted that when the configuration of the vanishing points with respect to the center of the vehicle blob is different from the one in Figure 4.11, the computation of the three-dimensional bounding box slightly differs from the presented one.

The first step of the computation is to determine location of points A, B, C by following equations. Points A, B and C are points of the base of the computed cuboid which are not hidden behind the vehicle.

$$A = t_r^U \cap t_l^V \tag{4.25}$$
$$B = t_l^V \cap t_r^W \tag{4.26}$$
$$C = t_r^U \cap t_l^W \tag{4.27}$$

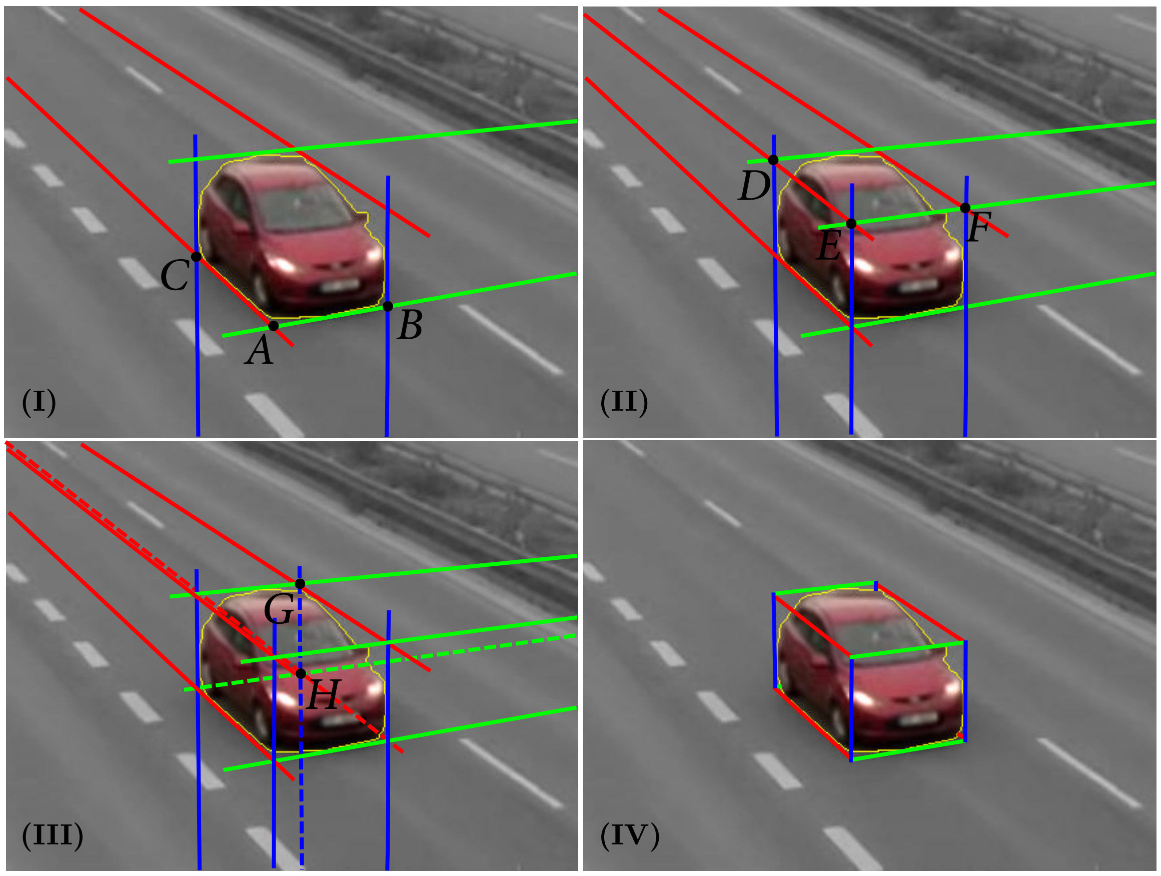


Figure 4.11: Construction of vehicle's 3D bounding box. **(I)** Tangent lines and their relevant intersections A, B, C. **(II)** Derived lines and their intersections E, D, F. **(III)** Derived lines and intersection H. **(IV)** Constructed bounding box. The image is adopted from paper [12] which I co-authored.

Afterwards, the points D, E, F are about to be located. Points D and F can be computed unambiguously by (4.28) and (4.29). However, the position of point E can be derived either from point D or F. Therefore, point E is selected by (4.32) so that the distance $|AE|$ is larger than the other one. This selection ensures that the whole contour of a vehicle will be enclosed by the bounding box.

$$D = t_r^V \cap t_l^W \tag{4.28}$$

$$F = t_l^U \cap t_r^W \tag{4.29}$$

$$E_D = \overleftrightarrow{DU} \cap \overleftrightarrow{AW} \tag{4.30}$$

$$E_F = \overleftrightarrow{FV} \cap \overleftrightarrow{AW} \tag{4.31}$$

$$E = \begin{cases} E_D & |AE_D| \geq |AE_F| \\ E_F & |AE_D| < |AE_F| \end{cases} \tag{4.32}$$

At the last step, the two remaining points G, H of the cuboid are computed by following equations. When all these eight vertices of the bounding cuboid are computed, the cuboid

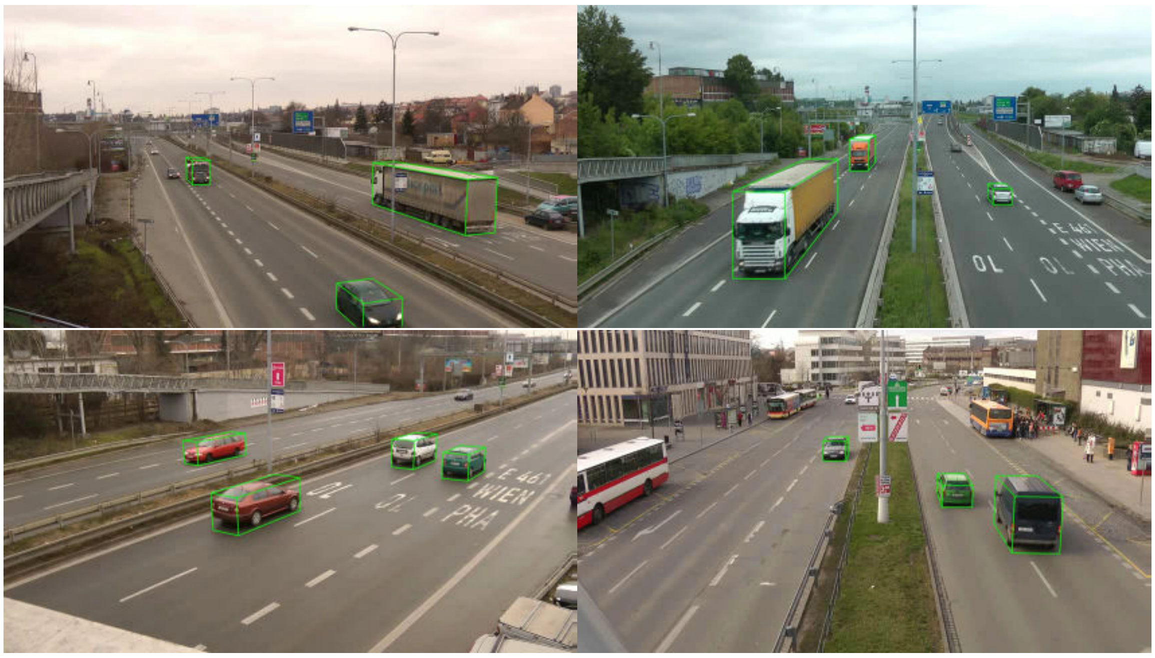


Figure 4.12: Examples of detected three-dimensional bounding boxes for different scenes.

can be drawn to an image, as it is shown in Figure 4.11.

$$G = \overleftrightarrow{DV} \cap \overleftrightarrow{FU} \tag{4.33}$$
$$H = \overleftrightarrow{CV} \cap \overleftrightarrow{BU} \tag{4.34}$$

Figure 4.12 shows examples of detected bounding boxes for several different scenes. For further processing, it is important that the bounding boxes surround the vehicle very tightly.

4.4 Classification

The proposed system for the classification of vehicles uses the described Histograms of Oriented Gradients (chapter 3.3.1) and the Support Vector Machine (chapter 3.3.2) with the RBF kernel. The image of a vehicle is rescaled to size 64×64 pixels prior to the computation of the Histograms of Oriented Gradients.

A vehicle can be classified in two following ways. The first one uses only one image of the vehicle and the second one uses a whole track of the vehicle. Hence, if the whole track is used, the final class c for a vehicle and its track T is computed according to Equation (4.35), while c_i^m is equal to 1 if the image $i \in T$ was classified by the SVM classifier to class m.

$$c = \arg\max_{m \in \mathbb{Y}} \left(\sum_{i \in T} c_i^m \right) \tag{4.35}$$

The description of used dataset and evaluation of the accuracy of both these methods can be found in chapter 6.2.

4.4.1 Fusion with Vehicle Dimensions

It is also possible to add to the classification of vehicles the dimensions of the vehicles as features for the classification if the dimensions are known. The computation of the dimensions in the image and the scaling of the dimension to the real world dimensions is discussed in chapter 4.6.

There are several ways how to fuse these features together. One of them is so called *early fusion* [33] and the second one is *late fusion*. The early fusion approach joins the two different feature vectors into a longer one and classifies the longer feature vector. On the other hand, the late fusion approach classifies the different features vectors separately and the final decision is based on the output of the two classifiers.

The late fusion approach was selected for the classification mainly because the classification of the vehicles with HOG features works without any error on the training dataset. SVM with RBF kernel was also used for the classification of vehicles which is based just on the dimensions of vehicles. The final class for the whole track T is computed according to (4.36) where h_i^m is equal to 1 if the HOG feature vector of the image $i \in T$ was classified as class m. On the other hand, value d_i^m is equal to 1 if the feature vector which contains the dimensions of a vehicle $i \in T$ was classified as class m. The weight of the classifier based on the HOG feature vector is denoted as $w \in \langle 0, 1 \rangle$.

$$c = \arg\max_{m \in \mathbb{Y}} \left(w \sum_{i \in T} h_i^m + (w-1) \sum_{i \in T} d_i^m \right) \tag{4.36}$$

Evaluation for the fusion of the classifiers can be also found in chapter 6.2 where is the best value for weight w discussed and evaluated.

4.5 Direction Estimation and Lane Detection

When an object left the scene and it was successfully evaluated as a vehicle, it is possible to determine the direction of the vehicle and to detect the lane which the vehicle is passing. The detection of lanes and lane-dividing lines is based on an actual motion of vehicles. There are other approaches [17, 24] which are using lane-dividing lines drawn on the road. However, using the motion of vehicles is more robust for roads where the lines are not drawn properly or they are not drawn at all.

4.5.1 Direction Estimation

It is possible to determine the direction of a vehicle with respect to the first vanishing point U. The direction can be either *To VP* or *From VP* and it is computed according to (4.37) where P_e, respectively P_s, is the last, respectively the first, point of the track of the vehicle.

$$dir = \begin{cases} \text{To VP} & \|U - P_e\| < \|U - P_s\| \\ \text{From VP} & \text{otherwise} \end{cases} \tag{4.37}$$

4.5.2 Lanes and Lane-Dividing Lines Detection

The detection of lanes and lane-dividing lines uses the points of trajectories of vehicles. First the algorithm for the detection of lanes will be described and after that the process of assigning a lane to a vehicle will be discussed.

Prior to the running the algorithm for the detection of lanes or lane-dividing lines, the three-dimensional bounding boxes has to be computed. Therefore, consider a set S_i of computed bounding boxes of a vehicle i. Each element of S_i is an eight-tuple bb_j of the image coordinates of the three-dimensional bounding boxes' vertices $A_j, B_j, C_j, \ldots, H_j$.

The algorithm for the lanes detection uses the centers of gravity X_j of the bases of the bounding boxes bb_j for the accumulation. The center of gravity of the base can be computed according to (4.38). On the other hand, the detection of lane-dividing lines uses points on line segment A_jB_j. Hence, set $\mathcal{X}_j$ contains points coordinates of pixels on line segment A_jB_j. It should be noted that it is possible to use also centers of gravity of two-dimensional bounding boxes for the detection of lanes. However, the accuracy of the detection is much lower and it will be discussed in chapter 6.4.

$$X_j = \overleftrightarrow{A_jH_j} \cap \overleftrightarrow{B_jC_j} \tag{4.38}$$

Both detections of lanes and lane-dividing lines uses function τ to project points to line $y = 0$. The projection is expressed by Equation (4.39) where U denotes the first vanishing point. All of the centers of gravity of bases X_j and points of the front lower edge $\mathcal{X}_j$ are projected by this function.

$$\tau(X) = b \cap \overleftrightarrow{UX} \qquad b : y = 0 \tag{4.39}$$

For each passed vehicle and its projected points, histograms of x coordinates of the projected points are built. The histograms are created separately for lanes and lane-dividing lines. Examples of the histograms are shown in Figure 4.13. As the figure shows, the values of the histogram for the lanes detection are much lower then the values of the histogram for the lane-dividing lines detection. This is caused by the fact that set $\mathcal{X}_j$ contains much more points than just one.

The centers of lanes c_m can be then detected as local maxima in histogram H in a predefined surroundings. Also, the value of the local maximum $H(c_m)$ has to be higher than

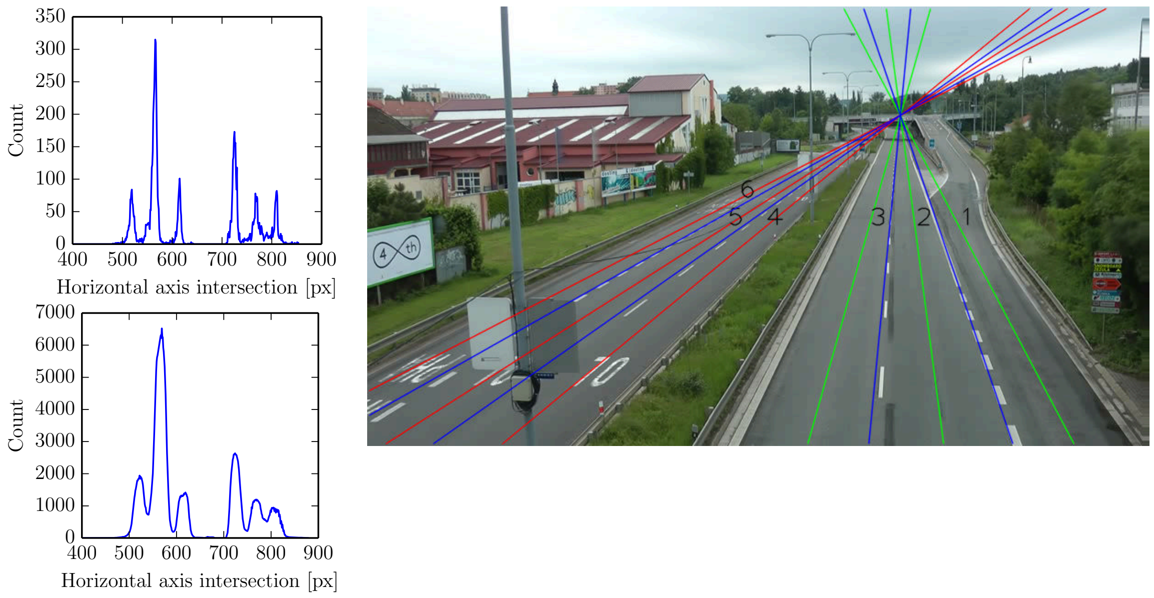


Figure 4.13: *Top left:* Histogram of projected centers of gravity of the base of vehicles. *Bottom left:* Histogram of projected points of front edge of the base of vehicles. *Right:* Detected lanes (green, red) and lane dividing lines (blue).

Algorithm 4.2 Merging of old and new lanes identification numbers.

```
Input: Previous mapping of ids and centers of lanes $M_{N-200} = \{(id_1, c_1), \dots, (id_m, c_m)\}$
Input: Currently detected centers of lanes $C_N = \{c_1, \dots, c_o\}$
Output: Current mapping of ids and centers of lanes $M_N = \{(id_1, c_1), \dots, (id_o, c_o)\}$
 1: $M_N \leftarrow \emptyset$
 2: for all $c \in C_N$ do
 3:     $bestMatchId \leftarrow -1$
 4:     $bestMatchDiff \leftarrow \infty$
 5:     for all $(id, c') \in M_{N-200}$ do
 6:         $distance \leftarrow \|c - c'\|$
 7:         if $distance \leq maxDistance \wedge distance < bestMatchDiff$ then
 8:             $bestMatchId \leftarrow id$
 9:             $bestMatchDiff \leftarrow distance$
10:     if $bestMatchDiff \neq \infty$ then
11:         $M_N \leftarrow M_N \cup \{(bestMatchId, c)\}$
12:         $M_{N-200} \leftarrow M_{N-200} \setminus \{(bestMatchId, M_{N-200}(bestMatchId))\}$
13:     else
14:         $M_N \leftarrow M_N \cup \{(nextFreeId, c)\}$
15:         $nextFreeId \leftarrow nextFreeId + 1$
```

a predefined threshold which is relative the to global maximum of the histogram. On the other hand, the centers of lane-dividing lines are detected as local minima in a predefined surroundings higher than a predefined absolute threshold. All these lanes and lines are drawn as lines defined by its center and the first vanishing point, as it is shown in Figure 4.13.

A unique identification number is assigned to each detected lane as it is also shown in the figure. However, as the detection algorithm runs after every 200 vehicles are observed, it is necessary to search correspondences between lanes detected after $N - 200$ observed vehicles and N observed vehicles in order to obtain temporal consistency of the detected lanes and their identification numbers. The lanes' identification numbers would change after every 200 accumulated lines if the correspondences were not found; and therefore, it would be impossible to create long-term statistics for vehicles passing in the detected lanes.

Therefore, Algorithm 4.2 is used for obtaining the identification numbers of lanes detected after N vehicles were observed. The algorithm uses previous mapping of the identification numbers and detected lanes and tries to find for each previously detected lane the closest newly detected lane with a predefined maximal displacement of the lanes' centers.

When the lanes are successfully detected, it is possible to assign a line to each newly passing vehicle. The assignment is not done for vehicles which were observed before the lanes were detected because the system targets mainly on online processing. For each newly observed vehicle i, the lane l_i is assigned according to following equation where C_N is a set of lanes' centers, M_N is the mapping of the identification numbers and the lanes' centers. Also value μ_x^i denotes the mean of x coordinates of projected bases' centers of gravity X_j^i.

$$l_i = M_N^{-1}\left(\arg\min_{c \in C_N} \|c - \mu_x^i\|\right) \tag{4.40}$$

The dominant direction of lanes is also computed for each lane l. The dominant direction dir_l is computed according to (4.41), where l_{VP} is the amount of points in the local maximum of the cluster which have direction towards the first vanishing point and l_{tot} is the number of all points in the cluster. A reasonable value for threshold t_{dom} is 0.1.

$$dir_l = \begin{cases} \text{To VP} & \dfrac{l_{VP}}{l_{tot}} \geq 1 - t_{dom} \\ \text{From VP} & \dfrac{l_{VP}}{l_{tot}} \leq t_{dom} \\ \text{Mixed} & \text{otherwise} \end{cases} \tag{4.41}$$

When the dominant direction for a lane is known, it is possible to detect vehicles which are traveling in wrong way. The detection is based on the detected direction dir of the vehicle and the dominant direction dir_l of the lane which the vehicle belongs to. The wrong way variable ww is determined by (4.42).

$$ww = \begin{cases} \text{True} & dir = \text{To VP} \wedge dir_l = \text{From VP} \\ \text{True} & dir = \text{From VP} \wedge dir_l = \text{To VP} \\ \text{False} & \text{otherwise} \end{cases} \tag{4.42}$$

4.6 Speed Estimation

The speed measurement relies on two crucial things. The first one is the measurement of distances on the plane of the monitored road and the other one is time measurement. The time measurement can be accomplished without any problem using frame numbers and the framerate of the processed video. On the other hand, the distance measurement is far more advanced. In order to be able to measure the distance, it is necessary to compute real world positions of points on the ground plane. Therefore, the first step is to compute the position for a given point in image on the ground plane.

However, as the camera is calibrated by the vanishing points only up to scale, these distance measurements are done only in so called relative units which does not corresponds to meters. Hence, the scale factor has to be also determined so that the relative units of distances of points on the ground plane can be transformed to meters.

4.6.1 Distance Measurement on Ground Plane

As it was stated above, in order to be able to measure real distance $|XY|$ where X, Y are points in image coordinates, it is necessary to compute world coordinates of points X and Y on the ground plane. Therefore, the points have to be projected on the ground plane.

However, parameters of ground plane $\wp$ have to be computed first. Three dimensional system, which is shown in Figure 4.14, is used with camera location $O = [p_x, p_y, 0]$ where $[p_x, p_y]$ is position of the principal point in the image. Also, $P = [p_x, p_y, f]$ are the world coordinates of the principal point and f is the focal length. Normal vector $n_\wp$ of ground plane $\wp$ can be computed as $W' - P$ where W' are the world coordinate of the third vanishing point. Nevertheless, the last parameter d of the ground plane is unknown as the distance of the ground plane from the camera is not known. Therefore, an arbitrary value is chosen to be this last parameter and the scale of the objects on the ground plane will be addressed later.

When the parameters of the ground plane are known it is possible to project a point $X = [x, y]$ on the ground plane. The projection is done according to function $\rho(X)$ which

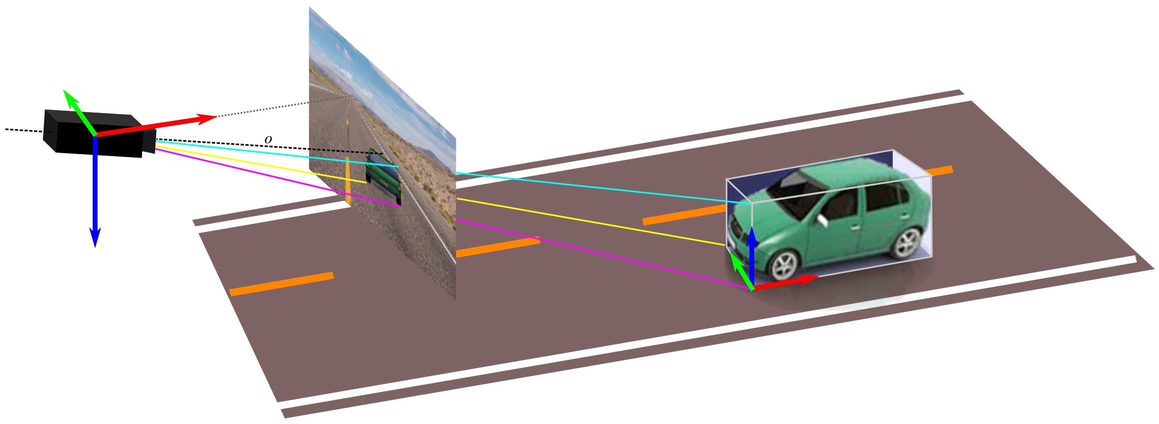


Figure 4.14: Example of a scene with a camera which is observing a vehicle and its coordintate system. Line o connects points $O = [p_x, p_y, 0]$ and $P = [p_x, p_y, f]$ where $[p_x, p_y]$ are coordinates of the principal point. The image is adopted from paper [12] which I coauthored.

is defined by Equation (4.43), where $X' = [x, y, f]$. Hence, the relative distance of image points X, Y can be computed by (4.44).

$$\begin{aligned} \rho(X) &= \wp \cap \overleftrightarrow{OX'} && (4.43) \\ d_r(X, Y) &= |\rho(X)\rho(Y)| && (4.44) \end{aligned}$$

It should be noted that in this way only world coordinates of points on the ground plane can be computed. One exception to this rule are points for which the position of the orthogonal projection to the ground plane is known, as it will be shown.

4.6.2 Adaptation of Relative Distances to Real World

The distances estimated by the equation presented above are only relative and do not have a direct correspondence to some units of measurement, for example meters. On the other hand, it is sufficient to scale all of the distances measured anywhere on the ground plane to meters by one scale factor λ which does not change with location of points in the image.

The computation of scale λ is based on that the vehicles should have some reasonable sizes. Therefore, median values for length, width and height were obtained and it is assumed that the sizes of observed vehicles by the traffic analysis system will correspond to the median values. So, relative length l_r, width w_r and height h_r is computed for each observed vehicle at each frame by following equations.

$$\begin{aligned} A_W &= \wp \cap \overleftrightarrow{OA} && (4.45) \\ B_W &= \wp \cap \overleftrightarrow{OB} && (4.46) \\ C_W &= \wp \cap \overleftrightarrow{OC} && (4.47) \\ E_W &= p_E \cap \overleftrightarrow{OE};\ p_E \perp \wp \wedge A_w \in p_E && (4.48) \\ l_r &= |A_W C_W| && (4.49) \\ w_r &= |A_W B_W| && (4.50) \\ h_r &= |A_W E_W| && (4.51) \end{aligned}$$

For each dimension a histogram of values is accumulated. Hence, histograms of vehicles' lengths, widths and heights are created. Global maxima l_m, w_m and h_m of the histograms are detected and anti-aliased as values l_M, w_M and h_M

$$\begin{aligned} & \gamma \in \{l, w, h\} \\ \gamma_m &= \arg\max_i H_\gamma(i) \qquad (4.52) \\ \gamma_M &= \frac{\sum\limits_{i=\gamma_m-5}^{\gamma_m+5} (i \cdot H_\gamma(i))}{\sum\limits_{i=\gamma_m-5}^{\gamma_m+5} H_\gamma(i)} \qquad (4.53) \end{aligned}$$

When these values are known, it is assumed that they correspond to the median size of vehicles and therefore the scales λ_l, λ_w and λ_h can be computed. In an ideal case all these scales would be equal. However, the scales are usually slightly different because of different influence of perspective and rounded corners; hence, the minimum is selected as the final scale λ.

$$\begin{aligned} &\lambda_l = \frac{4.27}{l_M} \qquad \lambda_w = \frac{1.74}{w_M} \qquad \lambda_h = \frac{1.51}{h_M} \\ &\lambda = \min\{\lambda_l, \lambda_w, \lambda_h\} \end{aligned} \qquad (4.54)$$

Finally, the real distance of two image points X and Y in meters can be computed by Equation (4.55) which scales the relative distance to meters.

$$d(X,Y) = \lambda \cdot d_r(X_w, Y_w) = \lambda \cdot |\rho(X)\rho(Y)| \qquad (4.55)$$

Therefore, the speed of a vehicle i can be computed at each frame t if the distance of the centers of gravity of the vehicle base L_t^i and L_{t-1}^i are computed and time difference Δt is measured. Final speed $\hat{v}^i$ of vehicle i is computed as median of vehicle's speeds at each frame.

$$v_t^i = \frac{d(L_t^i, L_{t-1}^i)}{\Delta t} \qquad (4.56)$$

Chapter 5

Implementation

This chapter describes some implementation details of the proposed system and deployment of the system for real online traffic analysis. The system is being used for the surveillance of traffic flow already.

5.1 Traffic Analyser

For the sake of processing speed, the proposed traffic analysis system is implemented in C++. The program is implemented as a command line utility and uses OpenCV[1] and Boost[2] libraries. The architecture of the system is modular and some different data output or a video output can be easily added to the system. Also, some parts of the system are covered with unit tests implemented with library CppUnit[3].

The system can be used either for an offline traffic analysis from a recorded video on a disk or an online traffic analysis which cooperates with Click2Stream[4]. The offline analysis processes a video and generates several files which include information about the processed video with a traffic surveillance scene. For example, an image with detected lanes is generated or a video file with observed vehicles can be created. Also, a file with data about the observed vehicles is generated. The file contains speeds of vehicles, their class, time of entrance and leaving of the scene. Lane identification number, direction and wrong way flag is also generated for each observed vehicle.

On the other hand, the online traffic analysis processes a video stream which is generated by an IP camera. The data about the observed traffic scene can be sent to a server which processes them and generates aggregated statistics about the traffic flow. Also, the processed video can be sent to the server together with additional information. The same types of information about the observed vehicles, as are written to the output file in the offline analysis, are sent to the server.

One program is used for all these types of traffic analysis and the type of input and other configuration data are provided by a configuration file which controls how the system will work. The type of video and data output is also controlled by the the configuration file.

[1] http://opencv.org

[2] http://www.boost.org

[3] http://cppunit.sourceforge.net

[4] http://www.click2stream.com

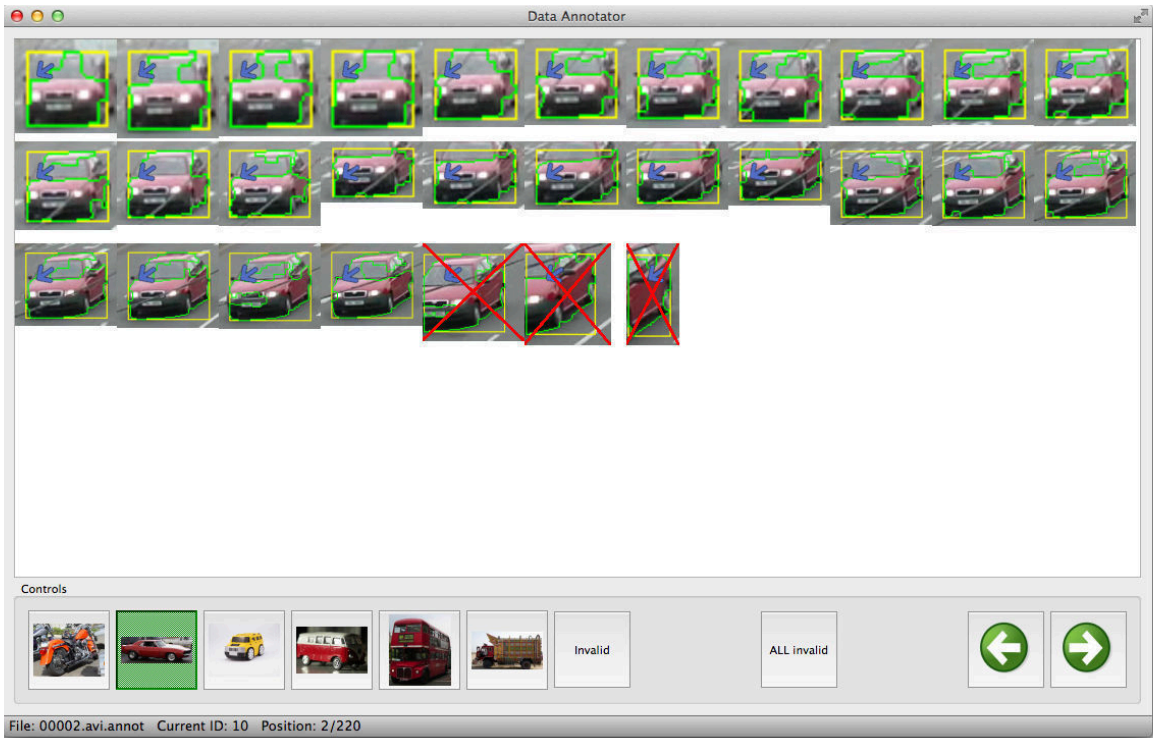


Figure 5.1: Screenshot of the application for vehicles annotation.

5.2 Vehicles Annotator

A tool for annotation of vehicles was also designed and implemented for creating a dataset for the vehicles classification. The tool uses a file which was generated by the traffic analyser and therefore, it provides benefits such, as a class can be assigned to a whole track of vehicles all at once, as it is shown in Figure 5.1.

The program for annotation of vehicles is based on QT[5] GUI framework. When a video and annotation is opened, the program loads all images which should be annotated into memory. Hence, the annotation of vehicles can be faster and it does not have to load any more information from the video. Therefore, the transition to a next track of a vehicle to annotate is faster.

[5]http://qt-project.org

Chapter 6

Evaluation

This chapter presents an evaluation of the proposed traffic analysis system. The accuracy of detection and tracking is discussed together with the evaluation of the classification accuracy. Examples of detected lanes and lane dividing lines are also shown. The evaluation of vehicles' speed measurement is also presented. Last but not least, the speed of video processing is discussed.

6.1 Detection and Tracking

A manually annotated dataset was created for the evaluation of accuracy of the detection and tracking of vehicles. Imaginary line, see Figure 6.1, which is crossing the center of image and dividing frames into two equal parts was displayed and for each car the location and time of crossing the line was annotated. Almost 30 minutes of video was annotated in this way resulting in almost 800 vehicles in the dataset.

The comparison with the ground truth annotation was performed in the following way. For each vehicle which was detected by the traffic analysis system, the trajectory is approximated by a line and the intersection of the approximation with the imaginary line is computed. A match with the ground truth is a vehicle which has trajectory with close intersection to the ground truth intersection and projected time of passing this intersection does not differ too much. If there are more vehicles which satisfy this condition, the vehicle with the smallest time and intersection difference is selected as the match with the ground truth. This way of evaluation was selected because the system targets mainly on an overall statistics of passed vehicles.

Nine various configurations which have different maximal distance to the first vanishing point and minimal passed distance of a vehicle were created and evaluated. The ROC and Precision-Recall curves are in Figure 6.1. Configuration providing the best results has F-Measure [25] equal to 0.915 (Precision is 0.905 and Recall 0.925). The False Negative cases are caused mainly by vehicle occlusions. The occlusions are caused either by a shadow which connects vehicles into one connected component or by a situation when a vehicle partially covers some other vehicle. The False Positives are caused primarily by the motion detection incorrectly dividing a vehicle into two objects and both these objects are tracked and treated as vehicles.

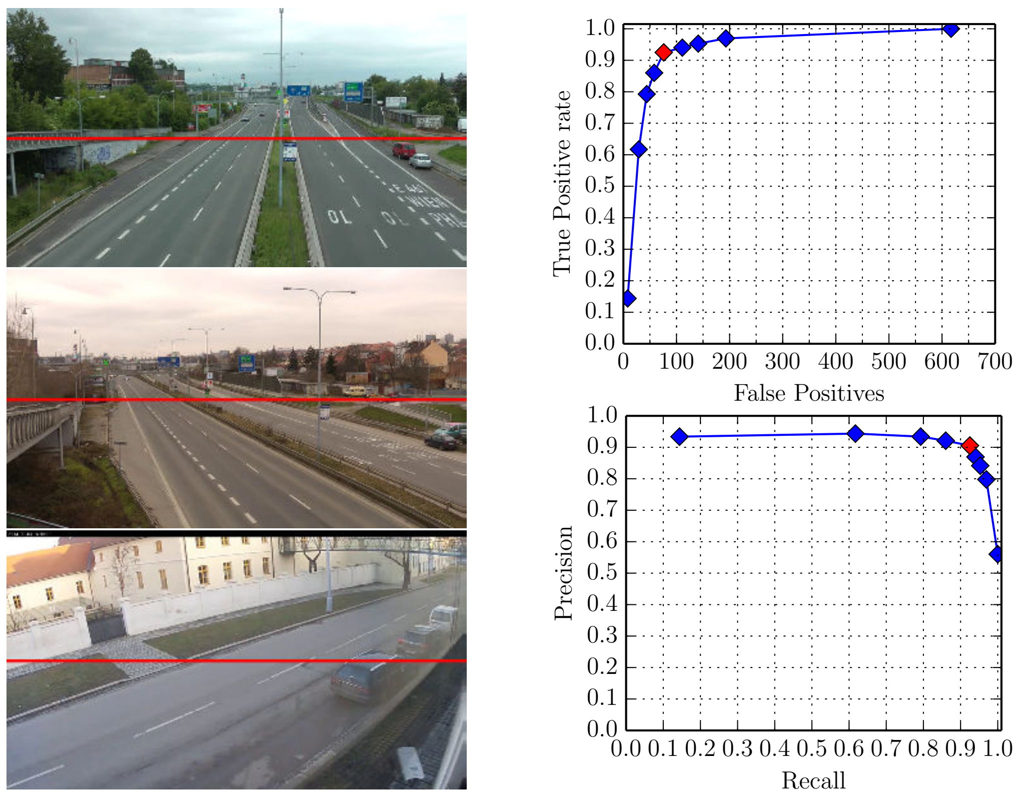


Figure 6.1: *Left:* Examples of videos used for the evaluation of detection and tracking. The virtual red line in the middle of the images was used for the ground truth annotation. *Right top:* ROC curve *Right bottom:* Precision-Recall curve. The configuration providing the best results has F-Measure equal to 0.915 (Precision is 0.905 and Recall 0.925) and is marked by red color.

6.2 Classification

A manually annotated dataset was obtained for the training and evaluation of classification of vehicles. Program presented in Section 5.2 was used for this annotation. The program provides significantly higher efficiency for annotation as a whole track of a vehicle is annotated at the same time. Four classes were selected for the classification, namely *personal*, *van*, *bus* and *truck*. Other classes like *motorcycle* or *SUV* are not so frequent on roads in Czech Republic which were used for the obtaining of dataset. Also, as Table 6.2 shows, vehicles of the *bus* class are also rare on roads in Czech Republic. Therefore, this class is not included in the evaluation as it is impossible to properly train and evaluate the classification on so few samples of different vehicles from the *bus* class. Examples of vehicles from the used classification classes are shown in Figure 6.2.

The whole manually annotated dataset was divided into training and evaluation dataset in the following way. For evaluation was selected random 15 % of vehicles from the whole dataset which were removed from the training dataset. It is important to mention that 15 % of unique vehicles was selected for the evaluation and not 15 % of images of vehicles. This approach was selected because it is important to evaluate how the classifier will work with images of vehicles which were not used for training.

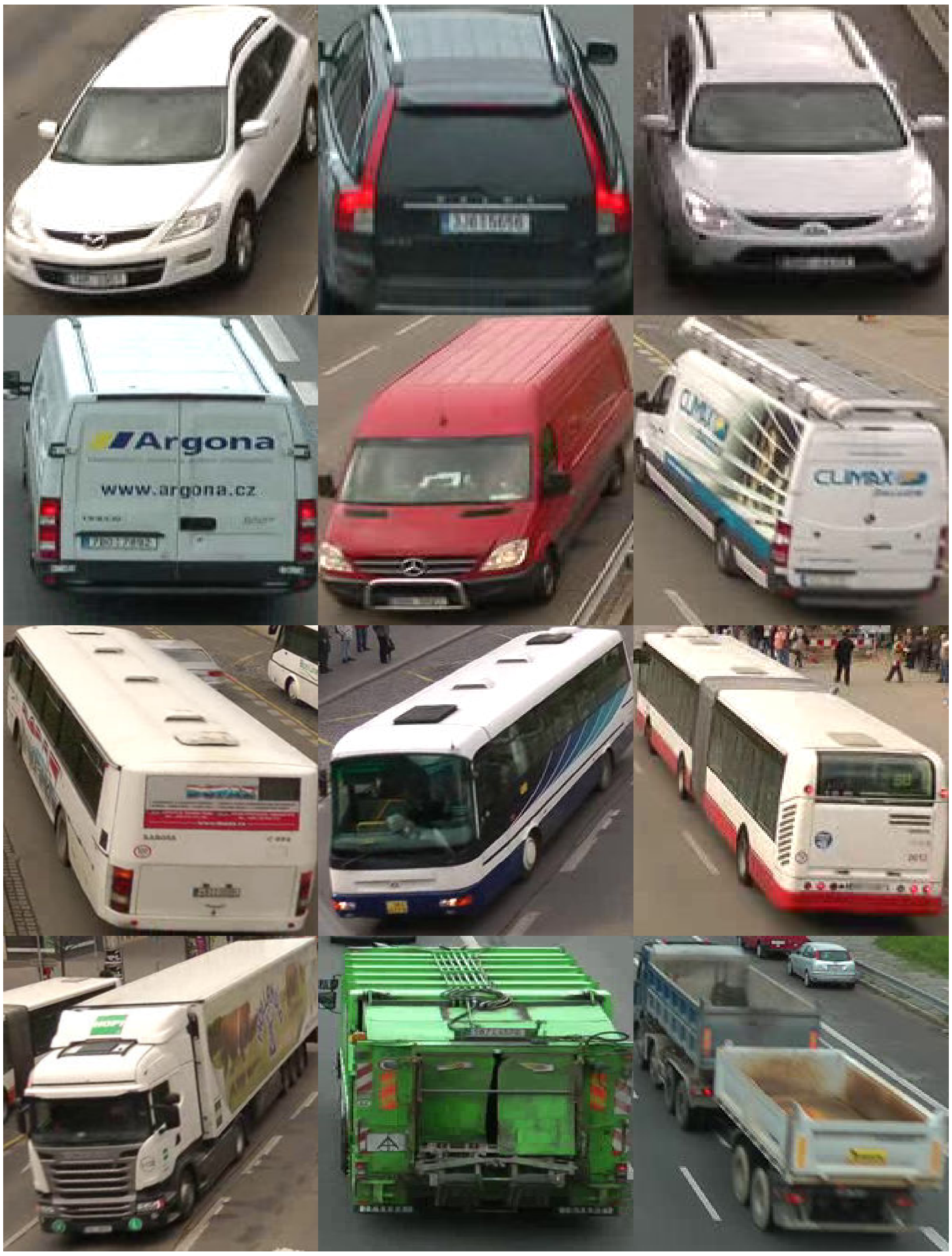


Figure 6.2: Examples of vehicles in the classification dataset. From top to bottom: personal, van, bus, truck.

	train		evaluation	
personal	1851	(23227)	324	(4166)
van	574	(9386)	72	(1333)
bus	43	(1879)	8	(329)
truck	369	(8433)	70	(1717)

Table 6.1: Dataset size. The total amount of all images from one class is in parentheses.

	one image	whole track
personal	98.1 %	99.4 %
van	92.2 %	93.1 %
bus	49.5 %	50.0 %
truck	93.4 %	95.7 %
total with buses	83.3 %	84.6 %
total without buses	**94.6** %	**96.1** %

Table 6.2: Accuracy of the classification of vehicles. As the table shows, if a whole track is used for the classification, then the results are slightly higher. Buses are not included in the total classification accuracy results because there is too few different instances of the bus class in the dataset.

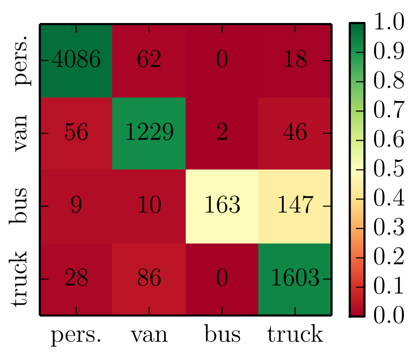
Cross-validation [43] technique was used for the training of the classifiers. The training dataset was randomly divided into ten groups and one of these groups was used for the cross-validation and the other groups were used for the training. The training process was performed ten times and every time the trained classifier was evaluated with the cross-validation dataset. After all, the classifier with the best cross-validation results was selected as the result of the training process.

The accuracy results of the classification are shown in Table 6.2 and confusion matrices for the classification are in Figure 6.3. As the table and figure show, if the whole track is used for the classification, the accuracy is slightly higher. Therefore, it makes sense to use the whole track of a vehicle for its classification. As it was mentioned, the *bus* class is not included in the overall evaluation results because there is too few buses in the training and evaluation dataset.

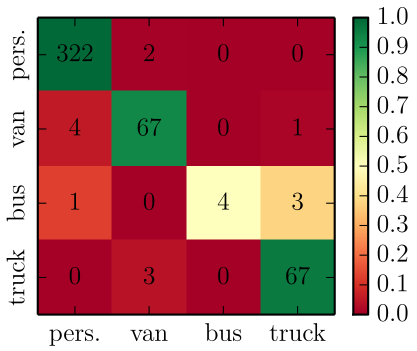
The table and figure with results also show that the highest accuracy was achieved for the *personal* classification class. This situation is caused by the size of the training dataset as there is much higher amount of personal vehicles in the dataset. It corresponds with that there is much more personal vehicles on roads than vehicles of other classes.

Also, almost a half of buses are classified as truck as Figure 6.3 shows. These incorrect classifications are caused by the small amount of unique buses in the training dataset and the diversity of trucks on roads as there are many totally different trucks, for examples see Figure 6.2. Therefore, there can be buses which are more similar to trucks in the training dataset rather than buses.

The dependence of accuracy was also evaluated if only a portion of the training dataset was used for the training process. This reduction of the training dataset was achieved by

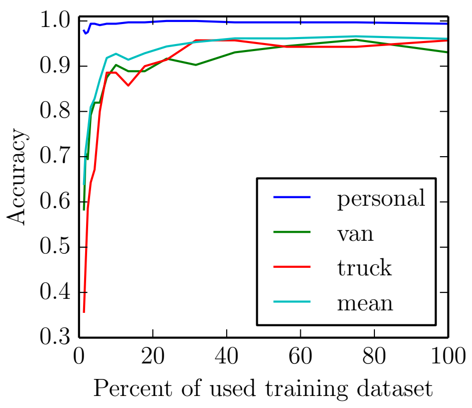


(a) one image



(b) whole track

Figure 6.3: Classification confusion matrices.



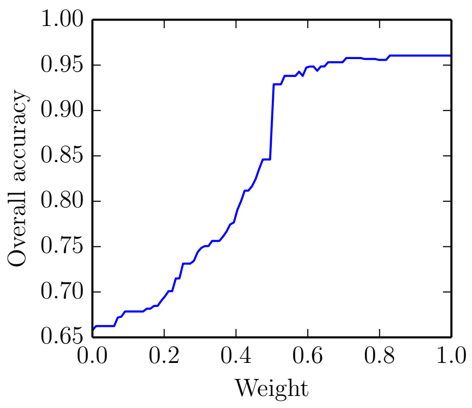


Figure 6.4: Left: Dependence of classification accuracy on the training dataset size. Right: Dependence of classification accuracy on weight of the classifier based on images.

removing every fourth image. This process generated dataset with 75 % of images of the original full dataset and then the process was performed again on the reduced dataset. Therefore, sixteen datasets which have size 0.75^i of the original full dataset, for $0 \leq i \leq 15$, were generated. For each of the dataset, a classifier was trained and the classifiers were evaluated on the same evaluation dataset. The results of the evaluation are shown in Figure 6.4. As the figure presents, the accuracy is almost totally stable for larger datasets. Hence, increasing size of the dataset, with exception of the *bus* class, would probably not increase the accuracy of the classification.

The fusion of classification by the HOG features and real dimensions of vehicles was also evaluated. The evaluation is focused mainly on computing the optimal weight for the final result of the fusion. The dependency of the final accuracy of the fusion on the weight is in Figure 6.4. As the graph shows, the best results are for weight equal to 1, which implies using only the classifier based on the HOG features. Therefore, the fusion with the dimensions of vehicles does not improve the classification results.

6.3 Vehicle Speed Estimation

This section presents an evaluation of the accuracy of the speed measurement. First, the accuracy of length measurements on a road are presented. Also, ground truth speed

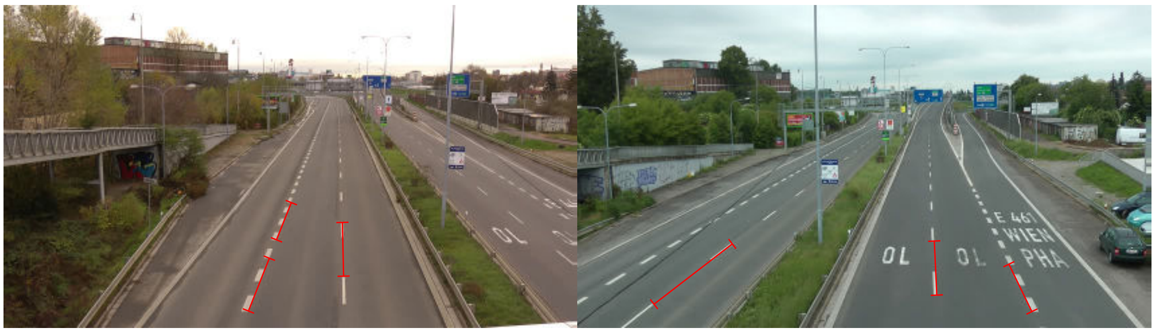


Figure 6.5: Examples of measured distances used for the evaluation of the accuracy of the distance measurement.

of a vehicle which was obtained with GPS sensor is compared with the speed estimated by the proposed traffic analysis system.

6.3.1 Accuracy of Length Measurement

In order to be able to measure the speed of vehicles it is necessary to measure the distance which vehicles passed and time. For the measurement of time, the number of frame and framerate can be used without any problem for the time measurement. However, much more complicated task is to measure distances of two points on the ground plane. If the camera is calibrated by the vanishing points, see Section 4.1, it is possible measure distances in relative units as the camera is calibrated up to scale. Hence, it is necessary to compute the scale factor as it is discussed in Section 4.6.

The accuracy of the length measurement was evaluated on seven different videos with 854×480 resolution. For each video, three different distances of easily recognizable points on the ground plane were measured manually, as it is shown in Figure 6.5. A laser distance meter was used for the ground truth measurements. Each of the distances was also measured in the video and measurement error e was computed according to (6.1), where g is the ground truth distance and m denotes the distance computed by the proposed traffic analysis system.

$$e = \frac{|g - m|}{g} \tag{6.1}$$

vehicles in video	best	worst	mean
438	5.73 %	10.74 %	8.13 %
1 274	5.36 %	6.62 %	5.81 %
1 194	3.71 %	4.71 %	4.05 %
1 865	0.51 %	2.69 %	1.39 %
430	14.66 %	15.62 %	15.08 %
397	2.14 %	5.65 %	4.27 %
655	1.27 %	8.99 %	4.70 %

Table 6.3: Errors for distance measurements in seven different video streams. The ground truth was obtained by a manual way on the road and three different distances were measured for each video. The errors are computed according to (6.1) and are in percents.

For each video was computed the best, mean and the worst error of the distance measurement as Table 6.3 shows. As it is shown in the table, the error is for almost all videos bellow 10 %. Two crucial things influences the error of the measurement. The first one is the accuracy of the camera calibration. Also, the amount of passed vehicles has effect on the error because if a higher amount of vehicles is observed, the statistics about vehicles' lengths, widths and heights are more representative. Therefore, the accuracy for the video with the highest amount of passed vehicles is the best one and videos with just $\sim$ 400 observed vehicles is much worse.

6.3.2 Accuracy of Speed Estimation

In order to evaluate not only the distance measurement, but also the final accuracy of the speed measurements, ground truth of the speed of a vehicle was obtained. The vehicle with known ground truth speed was passing with cruise control enabled so that the speed is stable and the ground truth speed was measured by a GPS sensor.

Table 6.4 presents the evaluation of the speed measurement with computed error for each measurement and mean error for whole video. As the table shows, the error for the video with the higher amount of passed vehicles is significantly lower. Again, the error depends on the camera calibration and the amount of passed vehicles which were used for the scale computation.

As the evaluation of the speed measurement shows, the errors of the speed measurements are reasonably low. Therefore, it can be used for generation of statistics of passing vehicles' speed.

vehicles	ground truth [km/h]	measured speed [km/h]	error
397	51	58	13.73 %
	83	79	4.82 %
	65	69	6.15 %
	78	85	8.97 %
	73	75	2.74 %
		Mean:	6.73 %
655	60	61	1.67 %
	51	50	1.96 %
	83	84	1.20 %
	65	69	6.15 %
	78	80	2.56 %
	73	76	4.11 %
		Mean:	2.94 %

Table 6.4: Evaluation of speed measurement compared to ground truth. The presented errors are reasonably low and usable for statistics of vehicles' speed.

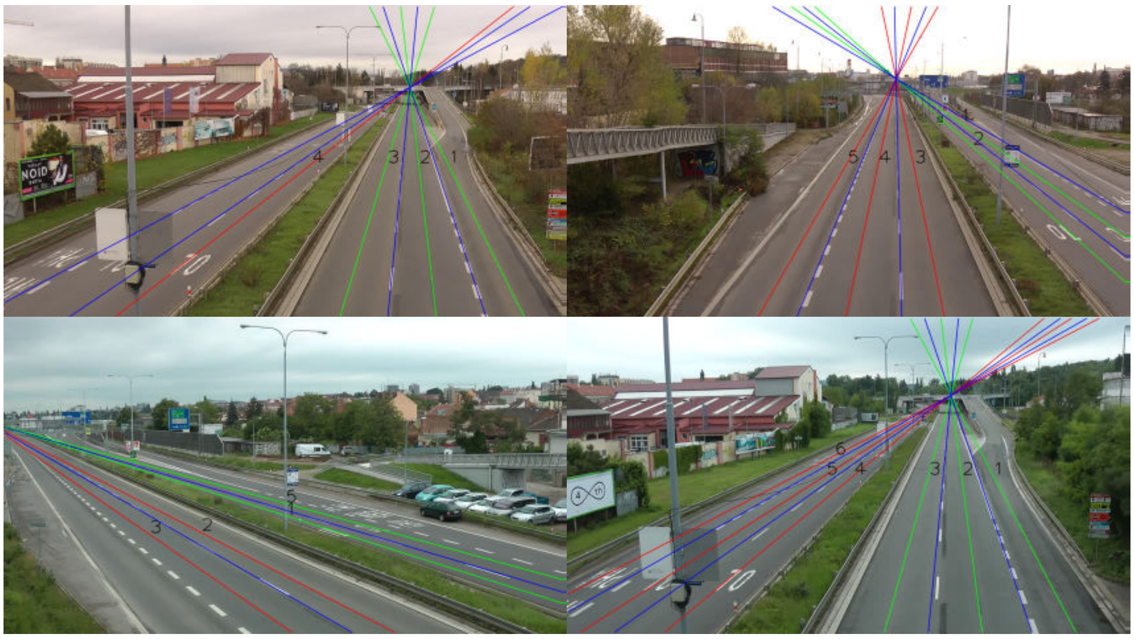


Figure 6.6: Detected lanes (green, red) and lane-dividing lines (blue). As it is shown, the lanes and lane-dividing lines are detected with a high accuracy even for distant lanes and lane-dividing lines.

6.4 Direction Estimation and Lane Detection

Several videos were processed in order to evaluate the lanes and lane-dividing lines detection. The results of the processed videos are shown in Figure 6.6. The lane-dividing lines are drawn by blue color and are detected with a high accuracy. Also as the figure shows, the direction of the lanes was correctly detected. The lanes with direction toward the first vanishing point are drawn by green color, and red color is used for the lanes which have direction from the first vanishing point. As the figure shows, also the centers of the lanes were precisely detected and they are almost exactly in the center of the lanes.

The difference between lanes detection with centers of two-dimensional and three-dimensional bounding boxes was evaluated. Figure 6.7 shows the results. As one can notice, the difference between left column (two-dimensional centers) and right column (three-dimensional centers) is significant. The lanes in the right column are detected much more precisely for more distant lanes from the video camera. Also, some lanes were not detected at all with the two-dimensional centers approach as the left column shows.

Several video sequences with a sufficient number of cars were obtained and stability of detected lanes was evaluated for these videos. The results of the evaluation are in Figure 6.8 and as the graphs show, the detected lanes are almost totally stable and do not change with passing cars. It should be noted that the detected lanes are recomputed always after next 200 cars were observed.

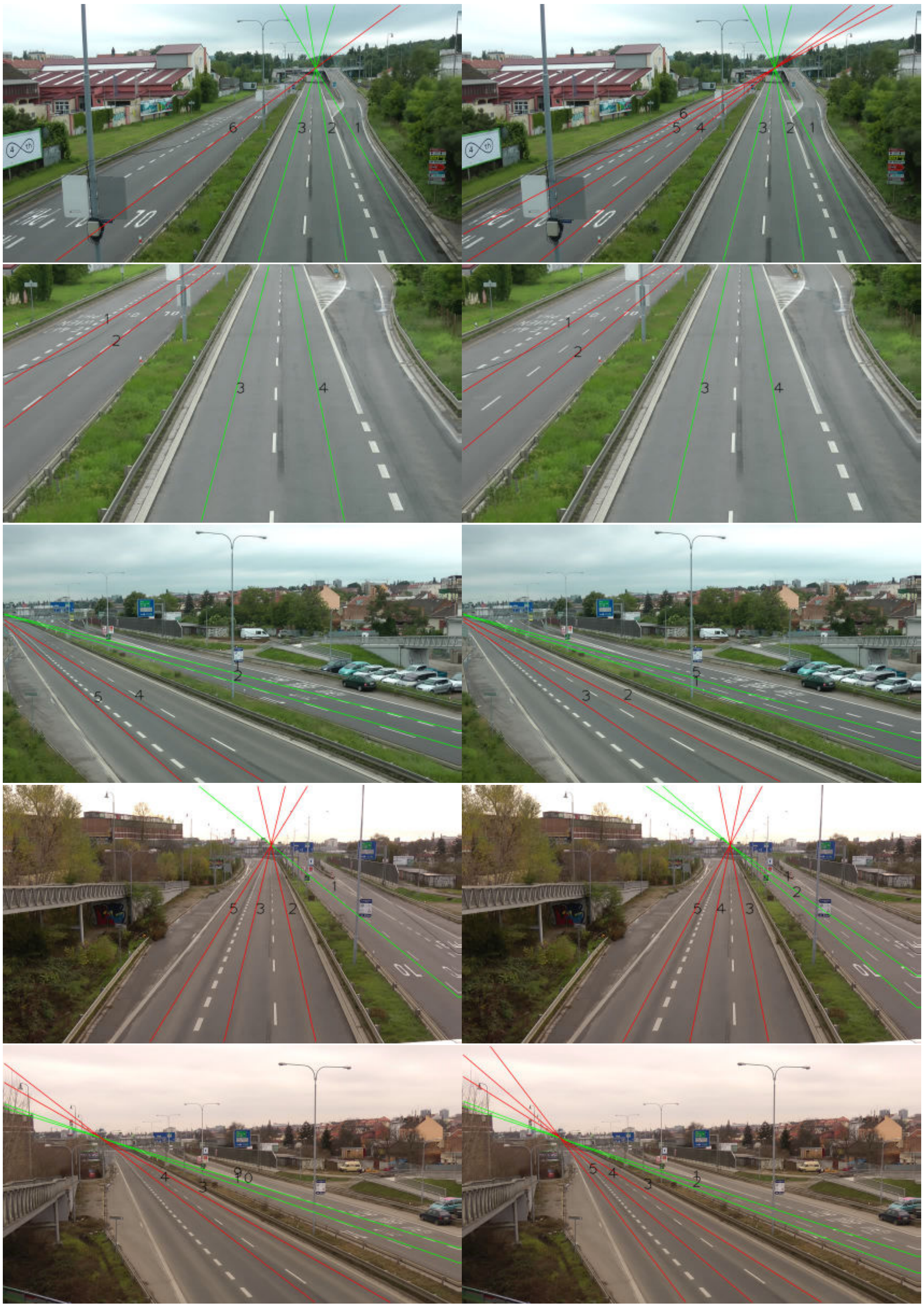


Figure 6.7: Comparison of detection with three-dimensional and two-dimensional vehicles centers. The left column uses two-dimensional centers and the right one three-dimensional centers.

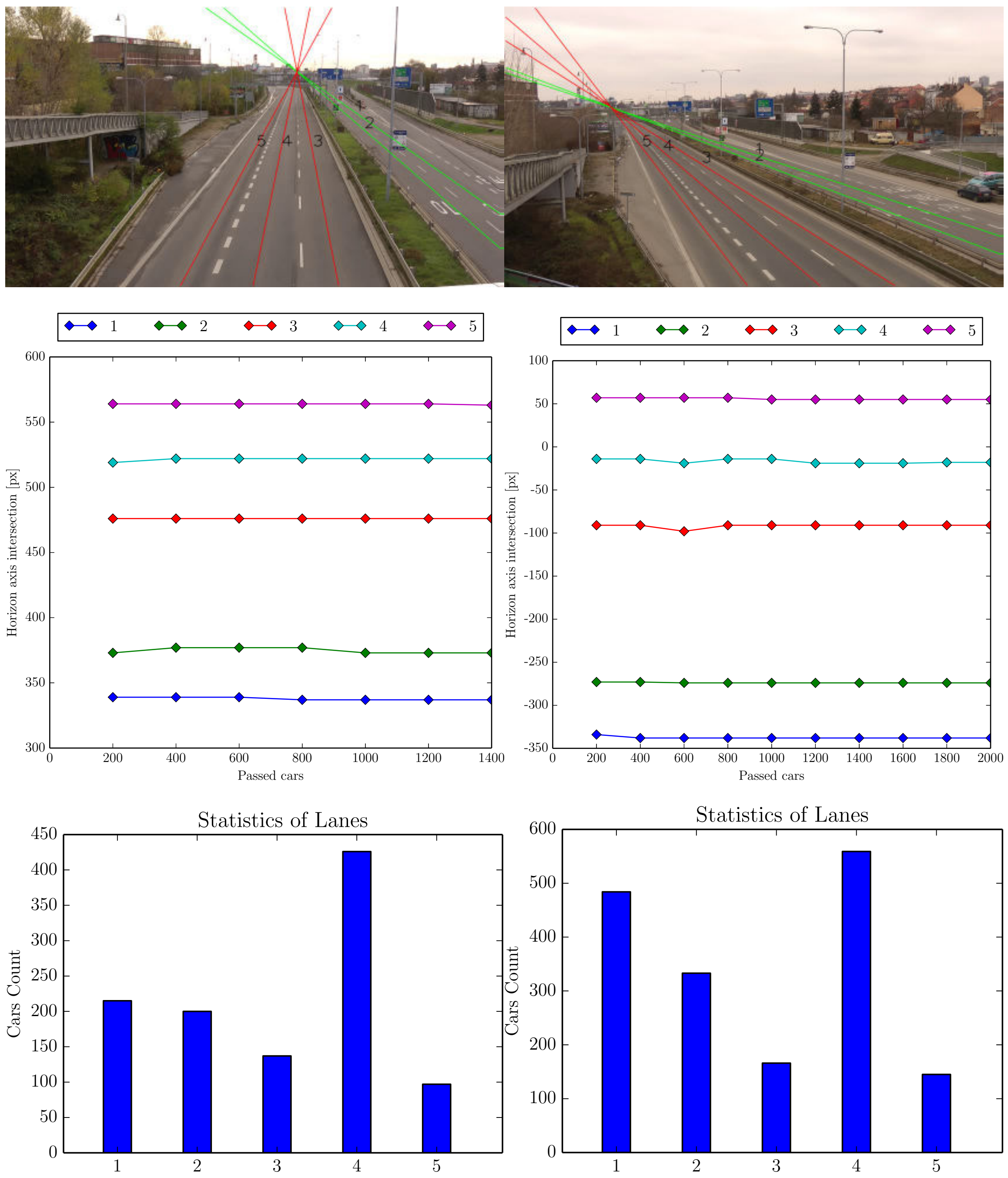


Figure 6.8: Stability of lanes detection for long video sequences. The top line of images presents the detected lanes. Only lanes which were valid for the last frame of video are drawn. The middle images show changes in detected lanes over time as new cars were observed in video. Finally, the bottom line shows the statistics of observed cars in the detected lanes.

resolution	traffic intensity	FPS
854×480	high	70.40
	low	118.09
1920×1080	high	19.06
	low	26.86

Table 6.5: Result of processing speed measurement. Videos with high traffic intensity contain ~ 40 vehicles per minute and video streams with low traffic intensity ~ 3.5 vehicles per minute. It should be noted, that the system uses video streams with ~ 12.5 FPS; and therefore, it can run without any problem in real time even for full-HD video with the high traffic intensity. The system was evaluated on 195 minutes of video.

6.5 Speed of Video Processing

If the traffic analysis system should be usable for real traffic surveillance applications, it has to run in real time. Therefore, the processing speed of the system was also evaluated. As Table 6.5 shows, the framerates are sufficient to run in real time. It should be noted that the system usually works with video streams which are smaller than full-HD; however, the system runs in real time even for full-HD videos as the system uses only ~ 12.5 FPS. The downsampling of the framerate of video streams is used so that the motion and passed distances of vehicles are stable and measurable.

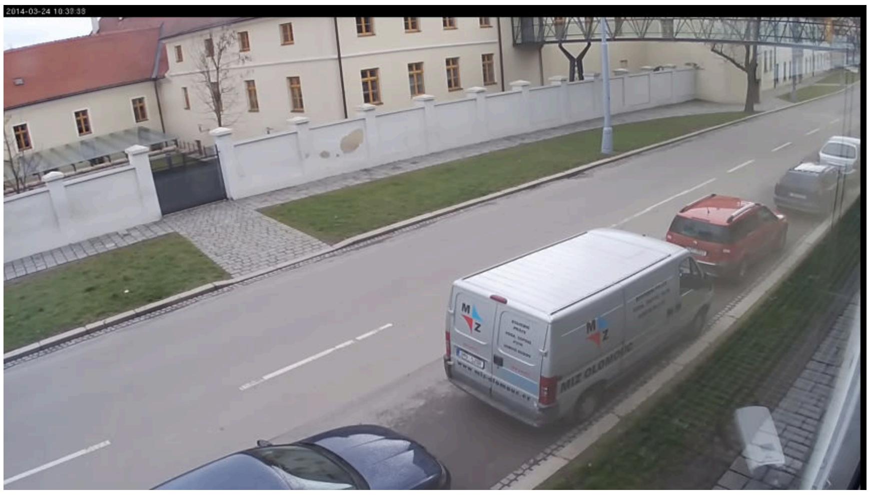
The processing speed measurement was done on a computer with i3-4330 3.50 GHz processor and 8 GB RAM. The measured framerates also include reading and decompression of videos and therefore, it influences significantly the framerates for videos with full-HD resolution.

6.6 Concluding Remarks

The proposed traffic analysis system was evaluated and the results are satisfactory for a traffic surveillance application which targets on overall statistics about the traffic flow on the monitored road. The detection and tracking of vehicles provides high accuracy and the results of the classification of vehicles is also satisfactory. The speed of vehicles can be measured with a high precision and the lanes can be detected even accurately as it is not influenced by the video camera's angle of view. Also, vehicles passing in a wrong way can be detected.

The system is being used for an online traffic surveillance. The system can run continuously and generate statistics for a monitored traffic road. The system has been already deployed and monitors a traffic scene which is shown in Figure 6.9. Also, the figure shows mean week traffic flow density on the monitored road. The data about the traffic flow were obtained by the proposed traffic analysis system. The system will be used for monitoring of a larger amount of traffic surveillance scenes in the future.

The future development of the system can focus on several tasks. For example, two video cameras monitoring a highway from different sides can be used for handling of occlusions. Also, a more robust shadow detection technique could be used for the shadow elimination of sharp shadows. Another interesting topic would be using a structure from motion [22] for the shadow elimination or the classification of vehicles.



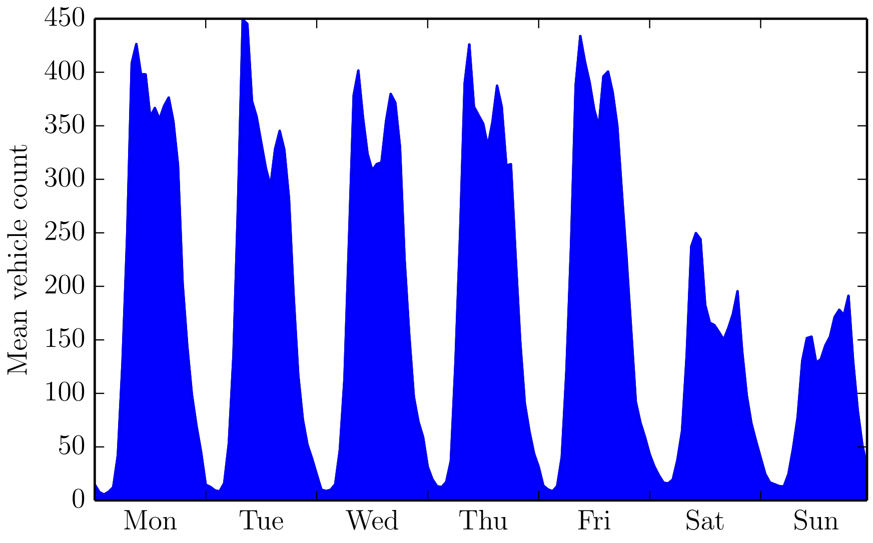


Figure 6.9: Statistics from the monitored traffic scene which was observed over 90 days. The top image shows the scene and the bottom image presents mean week traffic flow.

Chapter 7

Conclusion

The goal of the thesis was to design and implement a system for traffic analysis. The system is able to detect, track and classify vehicles. Also, the system detects lanes and vehicles passing in wrong way. The speed of vehicles is measured and the system works in real time and in a fully automated way without any manual input whatsoever.

I studied existing system and algorithms for the traffic analysis and described them in chapters 2 and 3. I proposed a system for the traffic analysis and the system was evaluated thoroughly. Also, I created a functional program implementing the system which runs in real time. The possibilities for future work were discussed and a poster and a video were created for presentation of the thesis.

Two papers describing the proposed system were published so far. The first one was published on EEICT conference [34] and I won the 1st place on the conference in category *Processing of signals, image and electric power systems*. Also, the other paper was published and accepted to oral presentation on CESCG seminar [35]. The system was already successfully deployed and it is used for a monitoring of a traffic scene. Also, I cooperated on papers [11, 12] which are focused on automatic calibration and speed measurement of vehicles. My contribution to the research was implementation of the system which was used for evaluation and testing of the proposed methods.

The system was thoroughly evaluated and the achieved results are satisfactory for traffic surveillance systems which targets on overall statistics about the traffic flow. The vehicles are counted and classified with a high accuracy. The lanes can be detected and the position is not influenced by the video camera's angle of view. Also, the speeds of vehicles are estimated with a low error. The main contribution of the thesis is the full automation of the system and the measurement of vehicles' speed.

Bibliography

[1] Aydos, C., Hengst, B., and Uther, W. Kalman filter process models for urban vehicle tracking. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on* (Oct 2009), pp. 1–8.

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

[3] Buch, N., Orwell, J., and Velastin, S. Detection and classification of vehicles for urban traffic scenes. In *Visual Information Engineering, 2008. VIE 2008. 5th International Conference on* (July 2008), pp. 182–187.

[4] Canny, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-8*, 6 (Nov 1986), 679–698.

[5] Cheng, H.-Y., and Hsu, S.-H. Intelligent highway traffic surveillance with self-diagnosis abilities. *Intelligent Transportation Systems, IEEE Transactions on 12*, 4 (Dec 2011), 1462–1472.

[6] Coifman, B., Beymer, D., McLauchlan, P., and Malik, J. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies 6*, 4 (1998), 271 – 288.

[7] Crammer, K., and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res. 2* (Mar. 2002), 265–292.

[8] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 1, pp. 886–893 vol. 1.

[9] Du, Y., and Yuan, F. Real-time vehicle tracking by kalman filtering and gabor decomposition. In *Information Science and Engineering (ICISE), 2009 1st International Conference on* (Dec 2009), pp. 1386–1390.

[10] Dubská, M., and Herout, A. Real projective plane mapping for detection of orthogonal vanishing points. In *British Machine Vision Conference, BMVC* (2013).

[11] Dubská, M., Herout, A., Juránek, R., and Sochor, J. Fully automatic roadside camera calibration for traffic surveillance. Submitted to: *IEEE Transactions on ITS* (2014).

[12] Dubská, M., Sochor, J., and Herout, A. Automatic camera calibration for traffic understanding. Submitted to: *British Machine Vision Conference, BMVC* (2014).

[13] Fowles, G. R., and Cassiday, G. L. *Analytical mechanics*, 7th ed. ed. Thomson Brooks/Cole, Belmont, CA, 2005.

[14] Hodlmoser, M., Micusik, B., Liu, M.-Y., Pollefeys, M., and Kampel, M. Classification and pose estimation of vehicles in videos by 3d modeling within discrete-continuous optimization. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on* (Oct 2012), pp. 198–205.

[15] Horprasert, T., Harwood, D., and Davis, L. S. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE ICCV* (1999), vol. 99, pp. 1–19.

[16] Hsieh, J.-W., Yu, S.-H., Chen, Y.-S., and Hu, W.-F. Automatic traffic surveillance system for vehicle tracking and classification. *Intelligent Transportation Systems, IEEE Transactions on 7*, 2 (2006), 175–187.

[17] Huang, L. Real-time multi-vehicle detection and sub-feature based tracking for traffic surveillance systems. In *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on* (March 2010), vol. 2, pp. 324–328.

[18] Jung, Y.-K., and Ho, Y.-S. Traffic parameter extraction using video-based vehicle tracking. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on* (1999), pp. 764–769.

[19] Jung, Y.-K., and Ho, Y.-S. Traffic parameter extraction using video-based vehicle tracking. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on* (1999), pp. 764–769.

[20] Kalman, R. E. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, 82 (Series D) (1960), 35–45.

[21] Khan, S., Cheng, H., Matthies, D., and Sawhney, H. 3d model based vehicle classification in aerial imagery. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (June 2010), pp. 1681–1687.

[22] Koenderink, J. J., and van Doorn, A. J. Affine structure from motion. *J. Opt. Soc. Am. A 8*, 2 (Feb 1991), 377–385.

[23] Koller, D., Weber, J., and Malik, J. Robust multiple car tracking with occlusion reasoning. Springer-Verlag, pp. 189–196.

[24] Lai, A. H. S., and Yung, N. H. C. Lane detection by orientation and length discrimination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 30*, 4 (Aug 2000), 539–548.

[25] MANNING, C. D., RAGHAVAN, P., AND SCHÜTZE, H. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

[26] MELO, J., NAFTEL, A., BERNARDINO, A., AND SANTOS-VICTOR, J. Detection and classification of highway lanes using vehicle motion trajectories. *Intelligent Transportation Systems, IEEE Transactions on 7*, 2 (June 2006), 188–200.

[27] MESSELODI, S., MODENA, C., AND ZANIN, M. A computer vision system for the detection and classification of vehicles at urban road intersections. *Pattern Analysis and Applications 8*, 1-2 (2005), 17–31.

[28] MITHUN, N., RASHID, N., AND RAHMAN, S. Detection and classification of vehicles from video using multiple time-spatial images. *Intelligent Transportation Systems, IEEE Transactions on 13*, 3 (Sept 2012), 1215–1225.

[29] MORRIS, B., AND TRIVEDI, M. Robust classification and tracking of vehicles in traffic video streams. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE* (2006), pp. 1078–1083.

[30] PRATI, A., MIKIC, I., TRIVEDI, M. M., AND CUCCHIARA, R. Detecting moving shadows: Algorithms and evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 25*, 7 (2003), 918–923.

[31] RAD, R., AND JAMZAD, M. Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters 26*, 10 (2005), 1597 – 1607.

[32] SHI, J., AND TOMASI, C. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on* (Jun 1994), pp. 593–600.

[33] SNOEK, C. G. M., WORRING, M., VAN GEMERT, J. C., GEUSEBROEK, J.-M., AND SMEULDERS, A. W. M. The challenge problem for automated detection of 101 semantic concepts in multimedia. In *Proceedings of the 14th Annual ACM International Conference on Multimedia* (New York, NY, USA, 2006), MULTIMEDIA '06, ACM, pp. 421–430.

[34] SOCHOR, J. Fully automated real-time traffic analysis from video. In *Proceedings of the 20th Conference STUDENT EEICT 2014* (2014), vol. 2, Brno University of Technology, pp. 54–56.

[35] SOCHOR, J. Fully automated real-time vehicles detection and tracking with lanes analysis *(to appear)*. In *Proceedings of The 18th Central European Seminar on Computer Graphics* (2014), Technical University Wien.

[36] STAUFFER, C., AND GRIMSON, W. E. L. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition* (1999), vol. 2, pp. 246–252.

[37] SZELISKI, R. *Computer Vision: Algorithms and Applications*, 1st ed. Springer-Verlag New York, Inc., New York, NY, USA, 2010.

[38] Tian, B., Li, Y., Li, B., and Wen, D. Rear-view vehicle detection and tracking by combining multiple parts for complex urban surveillance. *Intelligent Transportation Systems, IEEE Transactions on 15*, 2 (April 2014), 597–606.

[39] Tomasi, C., and Kanade, T. *Detection and tracking of point features.* School of Computer Science, CMU, 1991.

[40] Tseng, B., Lin, C.-Y., and Smith, J. Real-time video surveillance for traffic monitoring using virtual line analysis. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on* (2002), vol. 2, pp. 541–544 vol.2.

[41] Vapnik, V. N. *The Nature of Statistical Learning Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[42] Wasserman, L. *All of statistics: a concise course in statistical inference.* Springer, New York, 2004.

[43] Webb, A. *Statistical Pattern Recognition*, 2nd ed. ed. Wiley, New Jersey, 2002.

[44] Welch, G., and Bishop, G. An introduction to the kalman filter. Tech. rep., Chapel Hill, NC, USA, 1995.

[45] Weston, J., and Watkins, C. Multi-class support vector machines, 1998.

[46] Zivkovic, Z. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on* (2004), vol. 2, pp. 28–31 Vol.2.

List of Appendices

A Papers Written with the Thesis **66**

Appendix A

Papers Written with the Thesis

I wrote and cooperated on several papers during work on the thesis. Some of them were already accepted and other are in review process. The papers are appended to this thesis.

- **Fully Automated Real-Time Traffic Analysis from Video** published on EEICT student coference 2014. Also, I won the 1st place on the conference in category *Processing of signals, image and electric power systems* with this paper.
- **Fully Automated Real-Time Vehicles Detection and Tracking with Lanes Analysis** published on CESCG seminar 2014. The paper was accepted for oral presentation.
- **Fully Automatic Roadside Camera Calibration for Traffic Surveillance** submitted to IEEE Transactions on ITS. My contribution to the paper is implementation of the system which was used for the evaluation of computational speed of vanishing points accumulation.
- **Automatic Camera Calibration for Traffic Understanding** submitted to BMVC 2014. My contribution to the paper is implementation of the system which was used for the evaluation of computational speed of the proposed method for speed measurement of vehicles.

FULLY AUTOMATED REAL-TIME TRAFFIC ANALYSIS FROM VIDEO

Jakub Sochor
Master Degree Programme (2), FIT BUT
E-mail: xsocho06@stud.fit.vutbr.cz

Supervised by: Adam Herout
E-mail: herout@fit.vutbr.cz

Abstract: This paper describes briefly a fully automated system for traffic surveillance which is able to count passing cars, determine their direction and the lane which they are taking. The system works without any manual input whatsoever and it is able to automatically calibrate the camera by detection of vanishing points in the video sequence. The proposed system is able to work in real time and therefore it is ready for deployment in real traffic surveillance applications. The system uses motion detection and tracking with the Kalman filter. The lane detection is based on clustering of trajectories of vehicles.

Keywords: motion detection, tracking, vehicles, traffic surveillance camera, direction detection, lanes detection, real-time

1 INTRODUCTION

This paper presents a system for fully automated traffic analysis from a single uncalibrated camera. The camera is initially automatically calibrated and then statistics for the monitored traffic are generated. It is possible to use these statistics for many applications, for example simple monitoring of the traffic or more advanced predictions of the traffic flow. The system is currently able to count passing cars, determine their direction and lane which they are using. The classification of vehicles and speed measurement are going to be added in the near future.

2 PROPOSED METHODS FOR TRAFFIC ANALYSIS

This section describes the proposed methods for the traffic analysis which are used in order to achieve the goals which were presented above.

2.1 INITIALIZATION

Prior to running the algorithm for traffic analysis, it is necessary to initialize the system. The initialization focuses on the calibration of the camera by detecting vanishing points in the scene. It is assumed that the principal point is in the center of the image and the vehicles have approximately straight trajectories. Two vanishing points are detected and the third vanishing point and the focal length is computed during the initialization. The detection of the vanishing points is described in detail in paper written by Dubská et al. [1], which is currently submitted to IEEE Transactions on Intelligent Transportation Systems.

2.2 DETECTION AND TRACKING

The vehicle detection is based on motion detection in the video scene. Mixture of Gaussians background subtraction [4] is used for the motion detection. Also, shadow detection [2] is used for higher

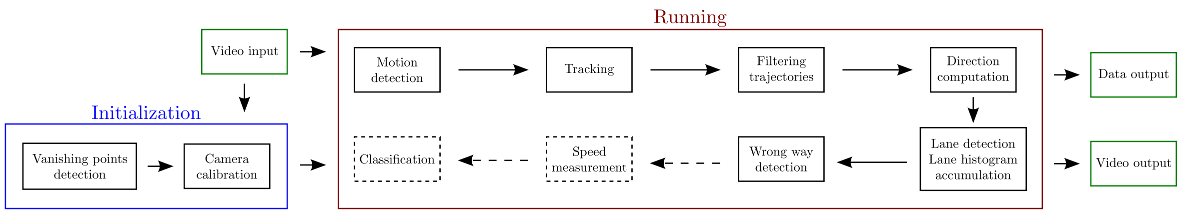


Figure 1: Overall pipeline of processing of the input video stream. The parts of pipeline which will be implemented in the future, namely Classification and Speed measurement, are shown in dashed boxes.

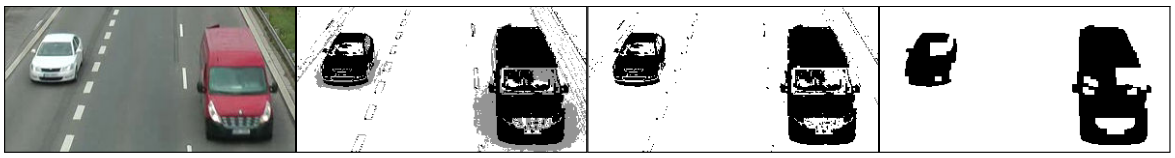


Figure 2: Process of motion detection. The leftmost image represents the original, the next one shows the detected motion with shadows which were removed in the third image. The result after the morphology opening and closing is shown in the last image.

accuracy of the motion detection. Noise in the detected motion is removed by morphological opening followed by morphological closing. Detected connected components are considered to be a potential vehicle. The Kalman filter [3] is used for prediction of the new position of the potential car and associating cars in consequent frames. The state variable of the Kalman filter $(x, y, v_x, v_y)^T$ contains the current position of the car and its velocity in image coordinates. Several conditions are evaluated for a tracked object and if the conditions are satisfied, the object is added to the statistics as a vehicle.

2.3 Direction Estimation and Lane Detection

The estimation of the direction of a vehicle is based on distance of the first and last point of the vehicle's trajectory. If the last point of the trajectory is closer to the first vanishing point, the vehicle is treated as it is going to the first vanishing point. On the other hand, if the first point is closer, the car is going from the vanishing point. The detection of lanes is based on accumulating the trajectories to one-dimensional histogram and searching for local maxima. The accumulation is based on the angle of the trajectories with the horizontal axis of the image. Statistics of the directions of vehicles are created for each lane and these statistics are used for the detection of vehicles which are travelling in the wrong way.

3 ACHIEVED RESULTS

The following section presents the achieved results. The processing speed is 57.97 FPS for a video in resolution 854×480 which contains traffic with high intensity. The processing speed is 28.59 FPS for a Full-HD video sequence and therefore, the system works in real time. The system was evaluated on a machine with Intel Dual-Core i5 1.8 GHz and 8GB DDR3 RAM.

3.1 Detection and Tracking

A manually annotated dataset was created for the evaluation of the accuracy of the detection and tracking of vehicles. For each vehicle in the video sequence, the dataset contains time and position of crossing the vehicle's trajectory with a virtual line. The time and position of the crossings are used for searching correspondences between the annotated vehicles and actually detected objects evaluated as vehicles. The accuracy computation of the detection and tracking is based on these correspondences.

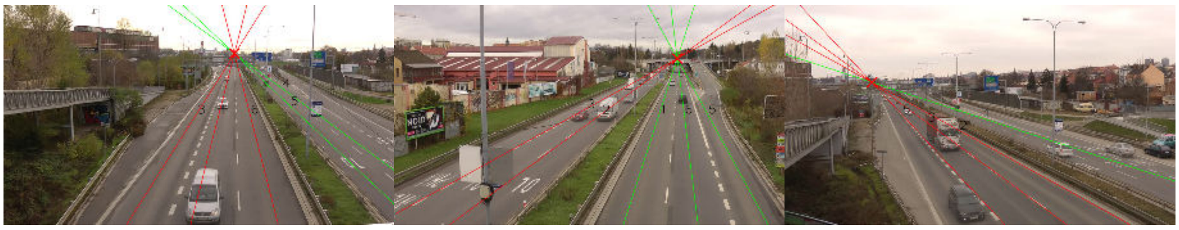


Figure 4: Detected lanes for long video sequences. Green color means that the majority of cars is heading towards the first vanishing point and red color denotes the opposite case. It should be noted that the centers of cars, which are used for lanes detection, are not in the middle of the lanes because of the angle of view.

Configuration providing the best results has F-Measure equal to 0.915 (Precision is 0.905 and Recall 0.925). The False Negative cases are caused mainly by vehicle occlusions. The occlusions are caused either by a shadow which connects vehicles into one connected component or by a situation when a vehicle partially covers some other vehicle. The False Positives are caused primarily by the motion detection incorrectly dividing a vehicle into two objects and both these objects are tracked and treated as vehicles.

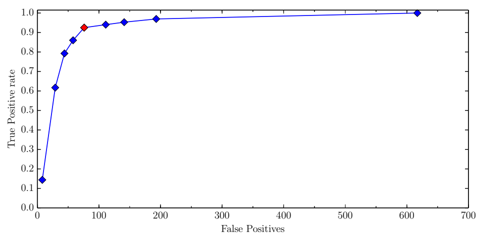


Figure 3: ROC curve of detection and tracking. Configuration with the best F-Measure is marked by red color.

3.2 Direction Estimation and Lane Detection

Several video sequences with a sufficient number of cars were obtained and stability of detected lanes was evaluated for these videos. The detected lanes in the video sequences are shown in Figure 4. The position of the lanes is almost totally stable and does not change with a higher amount of passed vehicles. Also the directions of the lanes were correctly estimated as shown in Figure 4.

4 CONCLUSION

This paper presents a system for fully automated traffic analysis from a single uncalibrated camera. The camera is automatically calibrated, vehicles are detected, tracked and their direction is computed. Also, the lanes are detected and therefore cars travelling in the wrong way can be detected. The system is ready for deployment and it is currently used for online traffic analysis. Future development will focus mainly on classification of the vehicles and speed measurement.

REFERENCES

[1] M. Dubská, A. Herout, R. Juránek, and J. Sochor. Fully automatic roadside camera calibration for traffic surveillance. Submitted to: *IEEE Transactions on ITS*, 2014.

[2] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE ICCV*, volume 99, pages 1–19, 1999.

[3] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[4] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999.

Fully Automated Real-Time Vehicles Detection and Tracking with Lanes Analysis

Jakub Sochor[*]
Supervised by: Adam Herout[†]

Faculty of Information Technology
Brno University of Technology
Brno / Czech Republic

Abstract

This paper presents a fully automated system for traffic surveillance which is able to count passing cars, determine their direction, and the lane which they are taking. The system works without any manual input whatsoever and it is able to automatically calibrate the camera by detecting vanishing points in the video sequence. The proposed system is able to work in real time and therefore it is ready for deployment in real traffic surveillance applications. The system uses motion detection and tracking with the Kalman filter. The lane detection is based on clustering of trajectories of vehicles. The main contribution is a set of filters which a track has to pass in order to be treated as a vehicle and the full automation of the system.

Keywords: motion detection, tracking, vehicles, traffic surveillance camera, direction detection, lanes detection, real-time

1 Introduction

This paper presents a fully automated system for traffic analysis. These types of analysis systems have a wide spectrum of usage. For example, it is possible to monitor the traffic or try to predict characteristics of the future traffic flow. The presented system is able to count passing cars, determine their direction and lane which they are taking. The goal is to run the system without any manual calibration or input whatsoever. The full automatism of the system is required if the system should be usable with already mounted uncalibrated cameras which are spread over highways. Therefore, the camera is automatically calibrated prior to running the traffic surveillance system. Real time processing is another requirement which needs to be satisfied for usage in real traffic surveillance applications.

Some methods for calibration of the camera require user input [29, 3] and therefore they can not be used in fully automated systems. Approaches for the calibration are usually focused on detection of vanishing point of the direction parallel to moving vehicles [6, 10, 23, 25]. There are several ways how to detect the vanishing point. Detected lines [25, 6] or lanes [25, 10] can be used for obtaining this vanishing point. On the other hand, Schoepflin and Dailey [23] use motion of vehicles and assume that they have straight parallel trajectories. Kanhere et al. [16] detect vehicles by a boosted detector and observe their movement, and Zhang et al. [30] analyze edges present on the vehicles.

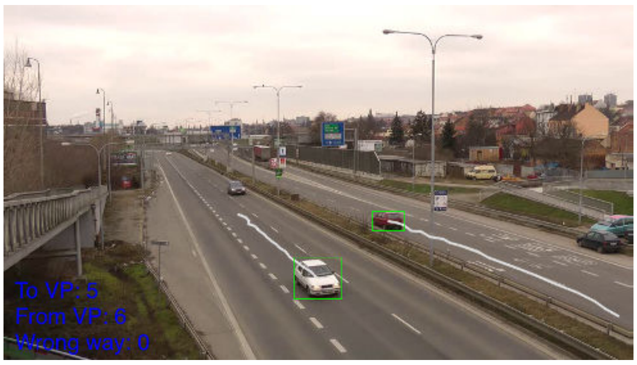


Figure 1: Example of video scene processed by the proposed traffic analysis system. Information about passing cars and their directions are displayed in output video.

A popular approach to detection and tracking of vehicles is to use some form of background subtraction and Kalman filter [15] to track the vehicles [12, 21, 14, 28, 1, 4, 7, 20, 17, 22]. Other approaches are based mainly on detection of corner features, their tracking and grouping [2, 13, 5]. Also, Cheng and Hsu [4] use pairing of headlights for the detection of vehicles at night.

Two main approaches are used for the detection of lanes. The first one is based on detection of the lane dividing lines [13, 18]. The other approach is based on motion of vehicles and their trajectories. Tseng et al. [28] use a virtual line perpendicular to vehicles' motion and compute intersections of the line with trajectories of vehicles. Hsieh et al. [12] use a two-dimensional histogram of accumulated centers of vehicles and Melo et al. [20] approximate the trajectories with low-degree polynomials

[*]xsocho06@stud.fit.vutbr.cz
[†]herout@fit.vutbr.cz

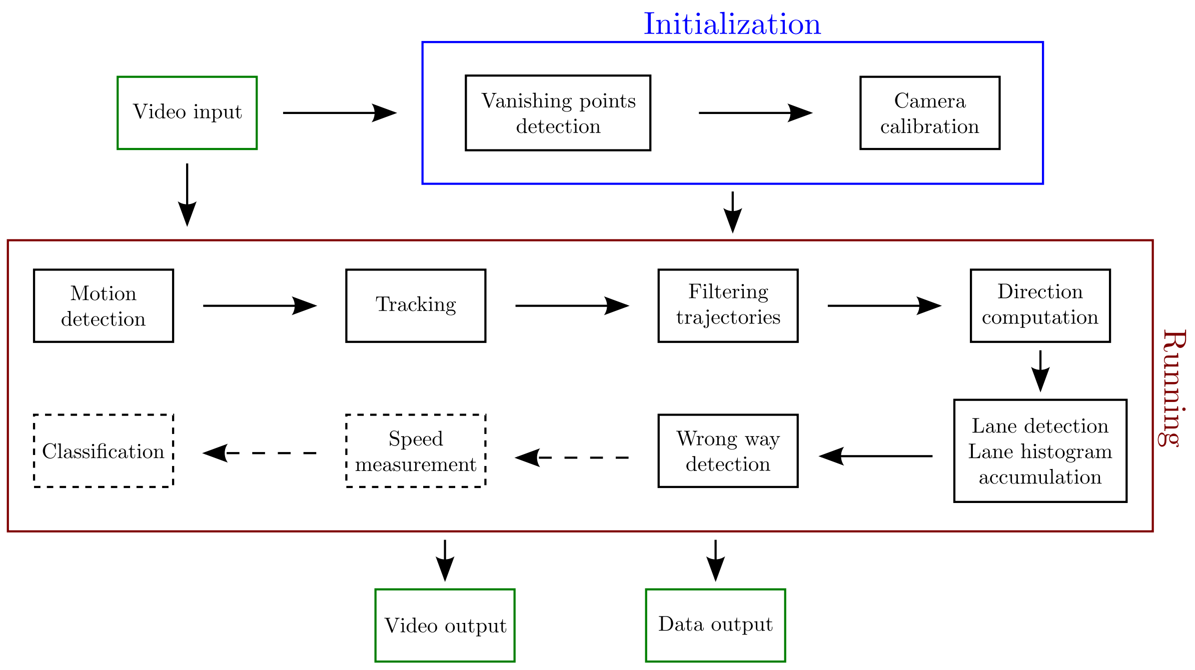


Figure 2: Pipeline of processing of the input video stream. Parts of the pipeline which will be implemented in the future, namely Classification and Speed measurement, are shown in dashed boxes.

and cluster these approximations.

The system proposed in this paper uses detection of vehicles by background subtraction [26, 31] and Kalman filter [15] for tracking. Prior to running the algorithm, the camera is calibrated by the detected vanishing points and the vanishing point of direction parallel to the motion of vehicles is used for higher accuracy of tracking. The detection of lanes is based on trajectories of vehicles and their approximation by a line.

2 Proposed Method for Traffic Surveillance

This section of the paper presents methods used in the system for detection and tracking of cars. The direction and lane detection is also discussed in detail. The overall processing pipeline is shown in Figure 2.

The main goal of the system is to create statistics of traffic on a road which is monitored by a camera. These statistics include the number of passed cars, their direction and lane.

2.1 Initialization

It is required to initialize the system prior to processing a video stream. The main purpose of the initialization is to find vanishing points of the scene and use the vanishing points to calibrate the camera. This is performed in a fully automated way and no user input is used. Arrows directed to the vanishing points are used for visualisation of the vanishing points. An example of the visualisation of the vanishing points is in Figure 3.

The vanishing point of the direction parallel to the vehicle movement is denoted as the first vanishing point. The second vanishing point has perpendicular direction to the movement of vehicles and the third vanishing point is perpendicular to the ground plane. However, only the first vanishing point is required for the tasks described in this paper; therefore, only detection of this vanishing point will be described. The detection of the other vanishing points is described in a paper written by Dubská et al. [8], currently submitted to IEEE Transactions on Intelligent Transportation Systems.

First Vanishing Point Detection

Corner feature tracking is used for the detection of the first vanishing point. Hence, Good Features to Track [24] are detected in the video stream and KLT tracker [27] is used for the tracking of the corner features. Detected motion of the tracked features is extended into a line which is defined by image points (x_t, y_t) and (x_{t+1}, y_{t+1}) which are positions of the feature in frame t and $t+1$.

All these lines are accumulated into the *diamond space* [9] until the initialization is terminated. The initialization is terminated when the global maximum of the diamond space is bigger then a predefined threshold and therefore a sufficient number of lines was accumulated. Afterwards, the coordinates of the global maximum in the diamond space are transformed into coordinates of the vanishing point in the image plane.

The diamond space is based on Cascaded Hough trans-

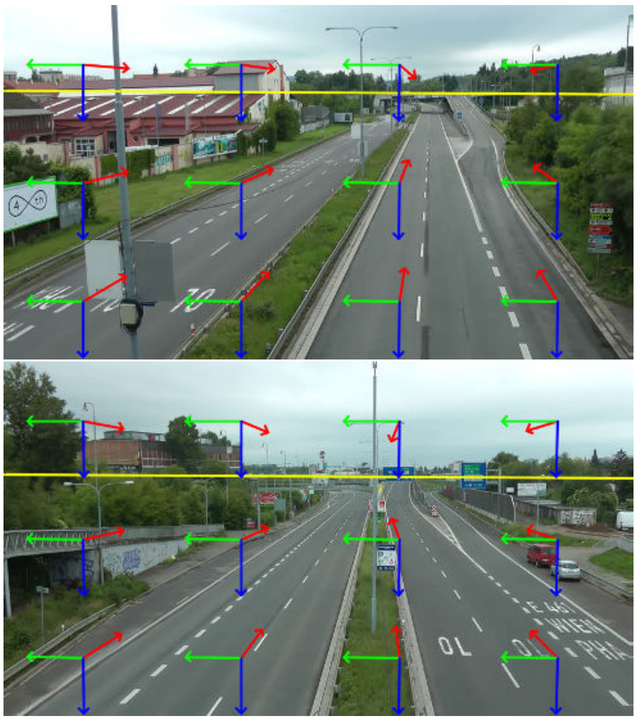


Figure 3: Detected vanishing points. Red arrows are pointing to the first vanishing point, green to the second one, and the third vanishing point is defined by the blue arrows. Yellow horizon line connects the first and second vanishing point.

form and parallel coordinates. Each line which is accumulated into the diamond space has to be transformed into coordinates in this space. The transformation divides the line into three line segments which are accumulated into the diamond space. Examples of the diamond space are in Figure 4.

It should be noted that the system uses a video downsampled to a framerate close to 10 FPS, so that the movement of corner features is detectable and measurable.

2.2 Vehicle Detection and Tracking

The vehicle detection is based on motion detection in the video scene. Mixture of Gaussians background subtraction [26, 31] is used for the motion detection. Also, shadow removal [11] is used for higher accuracy of the motion detection. Noise in the detected motion is removed by morphological opening followed by morphological closing. Detected connected components are considered to be a potential vehicle. The motion detection approach was selected mainly for its speed.

Kalman filter [15] is used for prediction of the new position of a car and for associating cars in consequent frames. The state variable of the Kalman filter $(x, y, v_x, v_y)^T$ contains the current position of the car and its velocity in image coordinates.

Several conditions are used for matching an object in the

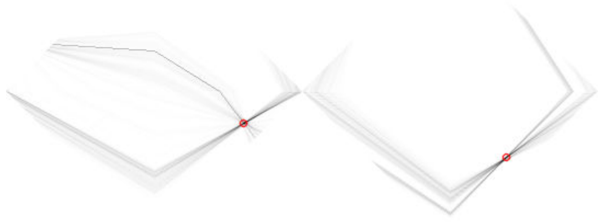


Figure 4: Examples of diamond spaces for detection of the first vanishing point with located global maximum

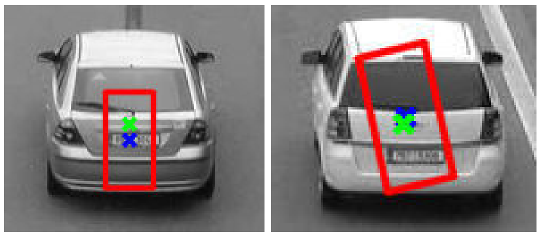


Figure 5: Examples of matching rectangles (red) for predicted object location (blue). The actual center of the detected connected component is drawn by green color. The figure shows that the longer side of the rectangle is directed to the first vanishing point.

consequent frame to its predicted location. The first condition states that the matched object must have similar colors. This condition is enforced by correlating histograms of objects in HSV color space. The second and last condition is that the center of matched object must be inside of so called *matching rectangle*. The predicted location of a car is the center of this matching rectangle and the longer side of the rectangle is directed towards the first vanishing point, as it is shown in Figure 5, and the matching rectangle has size 30×15 pixels. This condition is built on the assumption that the vehicle is going either in the direction towards the vanishing point or from the vanishing point, and therefore it is expected that in this direction can be higher displacement from the predicted location. Lastly, the closest connected component which meets the conditions presented above is found for each object and its predicted location in the consequent frame.

When a match is not found in several consequent frames, the tracked object is removed from the pool of tracked objects. Several filters are used for determining if the object should be accounted in the statistics of passed cars. The trajectory of the object is approximated by a line using least squares approximation. After that, the distance of the first vanishing point from the line is measured. Let us denote this distance as d_{vp}. Also, the ratio r, Eq. (1), between passed distance and maximal possible distance which an object can pass in the given trajectory is measured, Figure 6 shows the positions of P_e, P_s, L_e and L_s. The object is accounted in the statistics as a vehicle if the *acc* variable is equal to 1, Equation (2), where t_{vp} and t_r are predefined thresholds.

$$r = \frac{||P_e - P_s||}{||L_e - L_s||} \quad (1)$$

$$acc = \begin{cases} 1 & d_{vp} \leq t_{vp} \text{ and } r \geq t_r \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

2.3 Direction Estimation and Lane Detection

For a new vehicle which is about to be added to the statistics, the direction of the vehicle and its lane is calculated. Rule (3), which compares the relative positions of the first vanishing point and the last and first position of the vehicle, is used for computing the direction.

$$dir = \begin{cases} \text{To VP} & ||VP_1 - P_e|| < ||VP_1 - P_s|| \\ \text{From VP} & \text{otherwise} \end{cases} \quad (3)$$

The detection of lanes is based on clustering of the trajectories of cars. Therefore, the trajectory is also approximated by a line with least squares approximation, see green line in Figure 6. Each cluster of the lines corresponds to a lane in the monitored traffic surveillance scene and the clustering is performed in a one-dimensional space, where the values of the trajectory lines are their angles with axis x in the image. The clusters itself are searched as local maxima in the histogram of the angles. Hence, the clusters have to be a local maximum in the histogram in a predefined surroundings and also the maximum has to have at least a predefined amount of accumulated lines. The closest lane is assigned to a new passing vehicle as the lane which the vehicle is using. The closest lane computation is also based on the angles of the trajectory line and the lane with axis x.

This clustering is always performed after every 200 trajectory lines are accumulated and a unique identification number is assigned to each cluster. Let us denote the set of clusters as $C_N = \{(c_1, a_1), \ldots, (c_n, a_n)\}$ where N is the number of accumulated lines, and pair (c_i, a_i) denotes one cluster, where c_i is its identification number and a_i the angle corresponding to the found local maximum. Correspondences for clusters C_N and C_{N-200} are searched in order to obtain the temporal consistency of detected lanes in the scene. The clusters' identification numbers would change after every 200 accumulated lines if the correspondences were not found; and therefore, it would be impossible to create long-term statistics for cars passing in the detected lanes.

The identification number of the found correspondence is assigned to a cluster if the correspondence is found. A new unique identification number is assigned to the cluster otherwise. The correspondence for a cluster $(c_i, a_i) \in C_N$ is a cluster $(c_j, a_j) \in C_{N-200}$ for which (4) and (5) hold. The distance function is computed according to Equation (6) which compensates that the angles 0 and 2π have distance from each other 0.

$$a_j = C_{N-200}\left(\arg\min_c |dist(C_{N-200}(c), a_i)|\right) \quad (4)$$

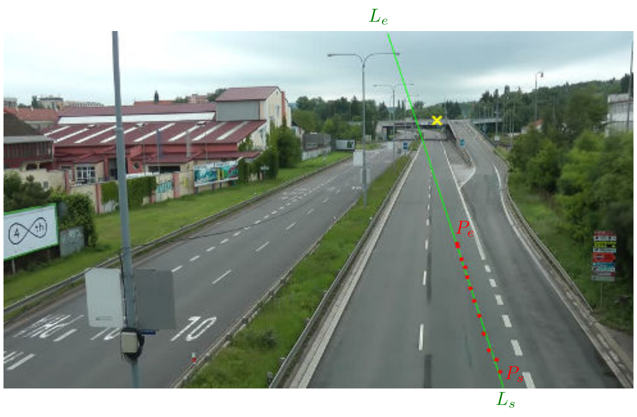


Figure 6: Measured distances for a passed object. The distance between approximated line (green) and the first vanishing point (yellow) is measured. Also, the distance between the first and last (P_s, P_e) point of the track of a vehicle is measured. The maximal distance which is possible to pass with a given trajectory is also measured (distance of L_s and L_e).

$$dist(a_j, a_i) \leq t_d \quad (5)$$

$$dist(x, y) = \min(2\pi - |x - y|, |x - y|) \quad (6)$$

The dominant direction is also computed for each cluster c of the trajectory lines. The dominant direction dir_c is computed according to (7), where l_{VP} is the amount of the trajectories in the cluster which have direction towards the first vanishing point and l is the number of all trajectories in the cluster. Reasonable value for threshold t_{dom} is 0.1.

$$dir_c = \begin{cases} \text{To VP} & \frac{l_{VP}}{l} \geq 1 - t_{dom} \\ \text{From VP} & \frac{l_{VP}}{l} \leq t_{dom} \\ \text{Mixed} & \text{otherwise} \end{cases} \quad (7)$$

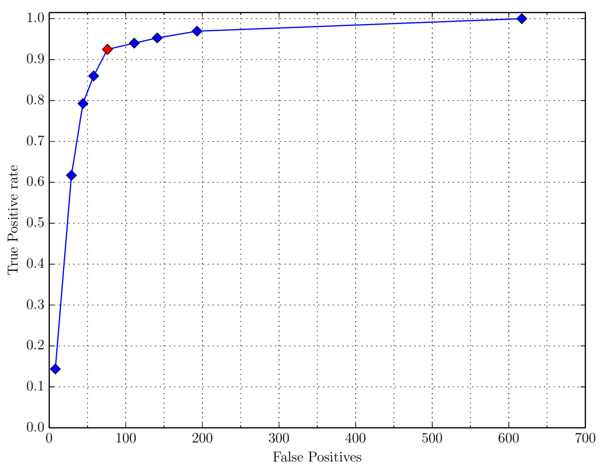
When the dominant direction for a lane is known, it is possible to detect vehicles which are traveling in wrong way. The detection is based on the detected direction *dir* of the vehicle and the dominant direction dir_c of the lane which the vehicle belongs to. The wrong way variable *ww* is determined by (8).

$$ww = \begin{cases} \text{True} & dir = \text{To VP} \land dir_c = \text{From VP} \\ \text{True} & dir = \text{From VP} \land dir_c = \text{To VP} \\ \text{False} & \text{otherwise} \end{cases} \quad (8)$$

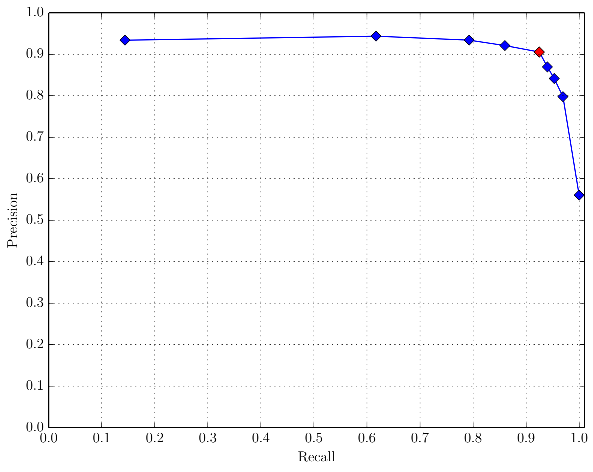
3 Results

This section presents the achieved results and methods of evaluation of the algorithms, which were presented above. The speed of video processing is also discussed.

The presented traffic analysis system was evaluated on several video streams. The processed video streams have resolution 854×480 and the video camera was located several meters above the road. The angle of the video camera varies as Figure 8 shows.



(a) ROC curve



(b) Precision-Recall curve

Figure 7: ROC and Precision-Recall curves for detection and tracking of vehicles in video. Configuration providing the best results has F-Measure equal to 0.915 and is marked by red color.

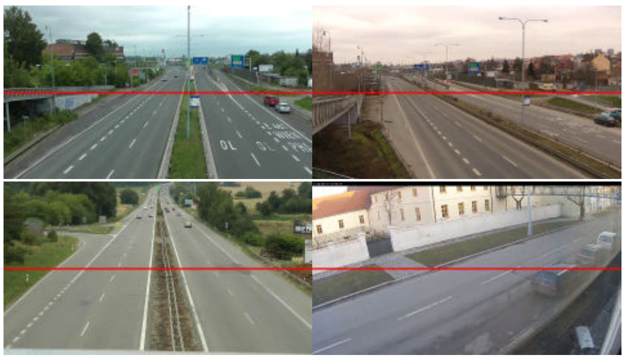


Figure 8: Examples of videos for detection and tracking evaluation. Virtual line which was used for manual ground truth annotation is drawn by red color.

3.1 Detection and Tracking

A manually annotated dataset was created for the evaluation of accuracy of the detection and tracking of vehicles. Imaginary line, see Figure 8, which is crossing the center of image and dividing frames into two equal parts was displayed and for each car, the location and time of crossing the line was annotated. Almost 30 minutes of video was annotated in this way resulting in almost 800 vehicles in the dataset.

The comparison with the ground truth annotation was performed in the following way. For each vehicle which was detected by the traffic analysis system, the trajectory is approximated by a line and the intersection of the approximation with the imaginary line is computed. A match with the ground truth is a vehicle which has trajectory with close intersection to the ground truth intersection and projected time of passing this intersection does not differ too much. If there are more vehicles which satisfy this condition, the vehicle with the smallest time and intersection difference is selected as the match with the ground truth. This way of evaluation was selected because the system targets mainly on overall statistics of passed vehicles.

Nine various configurations which have different maximal distance to the first vanishing point and minimal passed distance of a vehicle were created and evaluated. The ROC and Precision-Recall curves are in Figure 7. Configuration providing the best results has F-Measure [19] equal to 0.915 (Precision is 0.905 and Recall 0.925). The False Negative cases are caused mainly by vehicle occlusions. The occlusions are caused either by a shadow which connects vehicles into one connected component or by a situation when a vehicle partially covers some other vehicle. The False Positives are caused primarily by the motion detection incorrectly dividing a vehicle into two objects and both these objects are tracked and treated as vehicles.

3.2 Direction Estimation and Lane Detection

Several video sequences with a sufficient number of cars were obtained and stability of detected lanes was evaluated for these videos. The results of the evaluation are in Figure 9 and as the graphs show, the detected lanes are almost totally stable and do not change with passing cars. It should be noted that the detected lanes are recomputed always after next 200 cars were observed. Also the directions of the lanes were correctly detected as shown in Figures 9 and 10.

3.3 Evaluation of Speed

Processing speed of the system was also evaluated and the results are in Table 1. The measured framerates include also reading and decoding the video. The system was evaluated on a machine with Intel Dual-Core i5 1.8 GHz and

resolution	traffic intensity	FPS
854×480	high	57.97
	low	82.43
1920×1080	high	28.59
	low	47.88

Table 1: Processing speed evaluation. Approximately 110 minutes of video were used for the evaluation. The videos were divided into groups with respect to the traffic intensity. It should be also noted that the system uses video stream downsampled to ~ 10 FPS, so that the movement is detectable and measurable.

8GB DDR3 RAM. As the table shows, the system can be used for real-time analysis of Full-HD traffic surveillance video. The framerates are higher in videos with lower traffic intensity. The video sequences with higher traffic intensity contain more motion and vehicles which need to be tracked; therefore, more computational resources are used.

4 Conclusions

This paper presents a system for fully automated traffic analysis from a single uncalibrated camera. The camera is automatically calibrated, vehicles are detected, tracked and their direction is computed. Also, the lanes are detected and therefore cars travelling in the wrong way can be detected. The system works in real time and in a fully automated way and therefore it can be used for online traffic analysis with any camera which is monitoring a highway or a street. The system is ready for deployment and it is currently used for online traffic analysis.

The system is able to work under bad lightning and weather conditions. However, for example at night or during rainy weather, the accuracy of detection and tracking decreases slightly because of light reflections from the road. On the other hand, the initialization process can be performed at night without any problem, it will just take longer time because there is a lower amount of vehicles on streets at night.

The main contribution and advantage of the proposed traffic analysis system is that the system works without any manual input whatsoever and the set of conditions which a trajectory of a moving object in video is considered to be a vehicle. Future development of the system will focus mainly on complex crossroads and shadow elimination. Also, elimination of pedestrians from statistics should be addressed.

References

[1] C. Aydos, B. Hengst, and W. Uther. Kalman filter process models for urban vehicle tracking. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–8, Oct 2009.

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

[3] F.W. Cathey and D.J. Dailey. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *Intelligent Vehicles Symposium*, pages 777–782, 2005.

[4] Hsu-Yung Cheng and Shih-Han Hsu. Intelligent highway traffic surveillance with self-diagnosis abilities. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1462–1472, Dec 2011.

[5] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271 – 288, 1998.

[6] Rong Dong, Bo Li, and Qi-mei Chen. An automatic calibration method for PTZ camera in expressway monitoring system. In *World Congress on Computer Science and Information Engineering*, pages 636–640, 2009.

[7] Yuren Du and Feng Yuan. Real-time vehicle tracking by kalman filtering and gabor decomposition. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 1386–1390, Dec 2009.

[8] M. Dubská, A. Herout, R. Juránek, and J. Sochor. Fully automatic roadside camera calibration for traffic surveillance. Submitted to: *IEEE Transactions on ITS*, 2014.

[9] Markéta Dubská and Adam Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *British Machine Vision Conference, BMVC*, 2013.

[10] George S. K. Fung, Nelson H. C. Yung, and Grantham K. H. Pang. Camera calibration from road lane markings. *Optical Engineering*, 42(10):2967–2977, 2003.

[11] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE ICCV*, volume 99, pages 1–19, 1999.

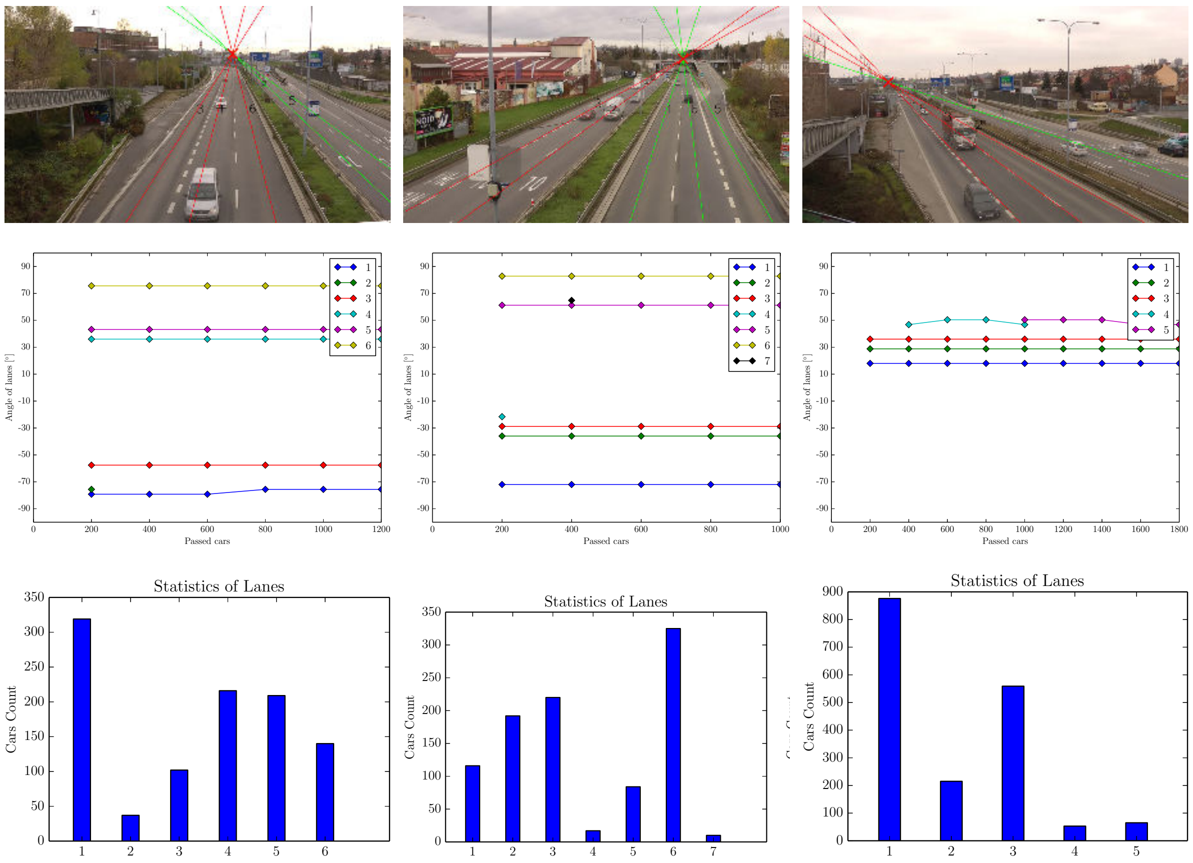


Figure 9: Stability of lanes detection for long video sequences. The top line of images presents the detected lanes. Only lanes which were valid for the last frame of video are drawn. The middle images show changes in detected lanes over time as new cars were observed in video. Finally, the bottom line shows the statistics of observed cars in the detected lanes.

[12] Jun-Wei Hsieh, Shih-Hao Yu, Yung-Sheng Chen, and Wen-Fong Hu. Automatic traffic surveillance system for vehicle tracking and classification. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):175–187, 2006.

[13] Lili Huang. Real-time multi-vehicle detection and sub-feature based tracking for traffic surveillance systems. In *Informatics in Control, Automation and Robotics (CAR), 2010 2nd International Asia Conference on*, volume 2, pages 324–328, March 2010.

[14] Young-Kee Jung and Yo-Sung Ho. Traffic parameter extraction using video-based vehicle tracking. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pages 764–769, 1999.

[15] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME – Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[16] Neeraj K Kanhere, Stanley T Birchfield, and Wayne A Sarasua. Automatic camera calibration using pattern detection for vision-based speed sensing. *Journal of the Transportation Research Board*, 2086(1):30–39, 2008.

[17] Dieter Koller, Joseph Weber, and Jitendra Malik. Robust multiple car tracking with occlusion reasoning. pages 189–196. Springer-Verlag, 1993.

[18] A. H S Lai and N. H C Yung. Lane detection by orientation and length discrimination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(4):539–548, Aug 2000.

[19] Christopher D MANNING, Prabhakar RAGHAVAN, and Hinrich SCHÜTZE. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[20] J. Melo, A. Naftel, A. Bernardino, and J. Santos-Victor. Detection and classification of highway lanes using vehicle motion trajectories. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):188–200, June 2006.

[21] B. Morris and M. Trivedi. Robust classification and tracking of vehicles in traffic video streams. In *In-*

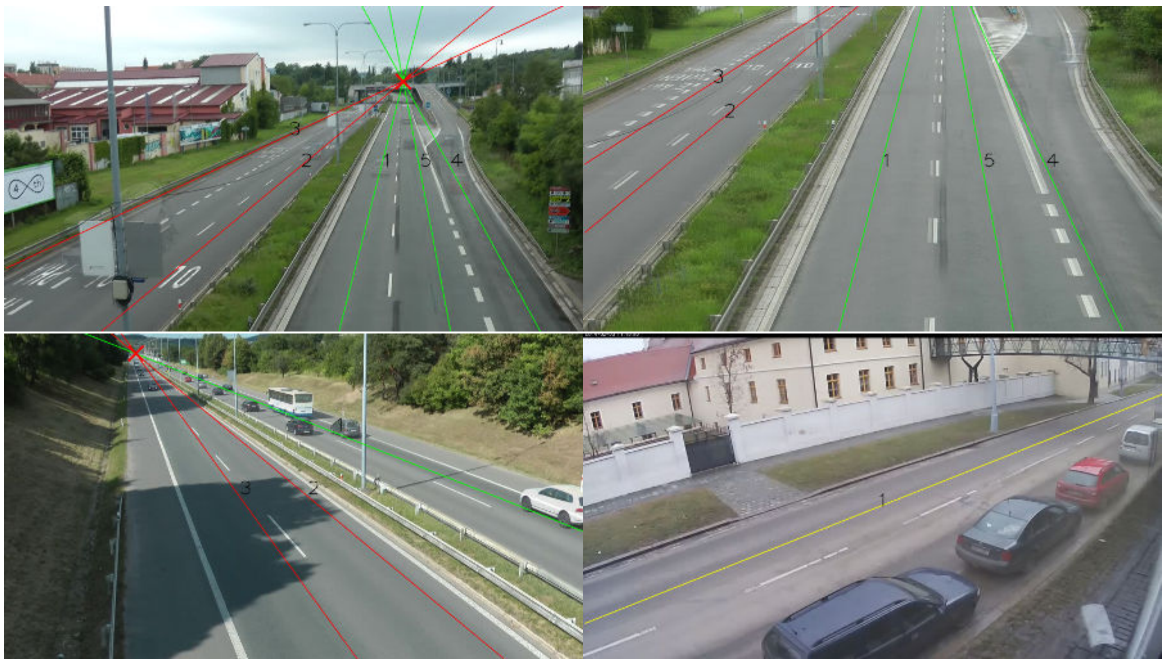


Figure 10: Example of detected lanes and dominant direction of cars in the lanes. Green color means that the majority of cars is heading towards the first vanishing point and red the opposite. Yellow color means that there is no dominant direction for the given lane. Example of this situation is shown in bottom right image. It should be noted, that the centers of cars, which are used for lanes detection, are not in the middle of the lanes because of the angle of view.

telligent Transportation Systems Conference, 2006. ITSC '06. IEEE, pages 1078–1083, 2006.

[22] Roya Rad and Mansour Jamzad. Real time classification and tracking of multiple vehicles in highways. *Pattern Recognition Letters*, 26(10):1597 – 1607, 2005.

[23] T.N. Schoepflin and D.J. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, 2003.

[24] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, Jun 1994.

[25] Kai-Tai Song and Jen-Chao Tai. Dynamic calibration of Pan–Tilt–Zoom cameras for traffic monitoring. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(5):1091–1103, 2006.

[26] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999.

[27] Carlo Tomasi and Takeo Kanade. *Detection and tracking of point features*. School of Computer Science, CMU, 1991.

[28] B.L. Tseng, Ching-Yung Lin, and J.R. Smith. Real-time video surveillance for traffic monitoring using virtual line analysis. In *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 541–544 vol.2, 2002.

[29] Kunfeng Wang, Hua Huang, Yuantao Li, and Fei-Yue Wang. Research on lane-marking line based camera calibration. In *International Conference on Vehicular Electronics and Safety, ICVES*, 2007.

[30] Zhaoxiang Zhang, Tieniu Tan, Kaiqi Huang, and Yunhong Wang. Practical camera calibration from moving objects for traffic scene surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(3):518–533, 2013.

[31] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31 Vol.2, 2004.

Fully Automatic Roadside Camera Calibration for Traffic Surveillance

Markéta Dubská[1], Adam Herout[1,2], Roman Juránek[1], Jakub Sochor[1]

[1]Graph@FIT, Brno University of Technology, CZ
[2]click2stream, Inc.

This paper deals with automatic calibration of roadside surveillance cameras. We focus on parameters necessary for measurements in traffic surveillance applications. Contrary to the existing solutions, our approach requires no a priori knowledge and it works with a very wide variety of road settings (number of lanes, occlusion, quality of ground marking), and with practically unlimited viewing angles. The main contribution is that our solution works fully automatically – without any per-camera or per-video manual settings or input whatsoever – and it is computationally cheap.

Our approach uses tracking of local feature points and analyzes the trajectories in a manner based on Cascaded Hough Transform and parallel coordinates. An important assumption for the vehicle movement is that at least a part of the vehicle motion is approximately straight – we discuss the impact of this assumption on the applicability of our approach and show experimentally, that this assumption does not limit the usability of our approach severely.

We efficiently and robustly detect vanishing points which define the ground plane and vehicle movement. Our algorithm also computes parameters for radial distortion compensation. Experiments show that the obtained camera parameters allow for measurements of length (and potentially speed) with $\sim 2\,\%$ mean accuracy. The processing is performed easily in real time and typically, a two minutes long video is sufficient for stable calibration.

***Index Terms*—Camera Calibration, Vanishing Points, Hough Transform, Diamond Space, PClines, Speed Estimation, Orthogonal Vanishing Points, Ground Plane Estimation, Camera Surveillance, Camera Distortion Correction**

I. Introduction

THE number of internet-connected cameras is quickly increasing and a notable amount of them are used in traffic monitoring. At the moment, many of these are used primarily for simply transferring the image to a human operator and they lack automatic processing. This is mainly because machine vision-based data mining algorithms require manual configuration and maintenance, involving a considerable effort of skilled personnel and in many cases also measurements and actions taken in the actual real life scene [1]–[5]. The goal of our research is to provide fully automatic (no manual input whatsoever) traffic processing algorithms – leading towards vehicle classification and counting, speed measurement, congestion detection, etc. Different applications can be fostered by compensation of local projection [6].

In this paper, we are dealing with the problem of fully automatic camera calibration in a common traffic monitoring scenario. We automatically determine radial distortion compensation parameters and solve the calibration of internal camera parameters as well as external parameters (camera orientation and position up to scale with respect to the dominant motion of the vehicles). The solution is fully automatic in the sense that there are no inputs or configuration parameters specific to a particular traffic setting, camera type, etc. Also, we are not assuming any appearance model of the vehicles which would differ for different countries, time periods and the kind of traffic on the particular road. Moreover, we assume no a priori knowledge about the road itself (number of lanes, appearance of the marking, presence of specific features, etc.). The only assumption is approximately straight movement of the vehicles. Experiments show that normal curves on highways are still “straight-enough” to meet our assumption. Sharp corners cannot be directly handled by our approach and will be subject of further study.

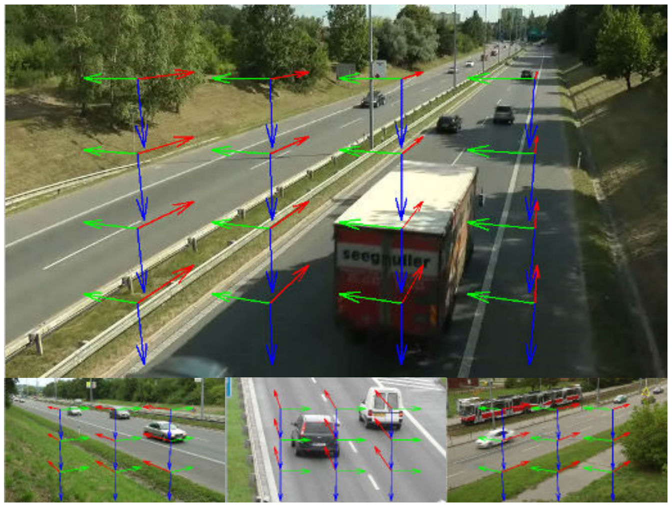


Fig. 1. We propose a fully automatic approach to camera calibration by recognizing the three dominant vanishing points which characterize the vehicles and their motion. **top:** Three recognized vanishing points. **bottom:** Various scenes where the algorithm automatically calibrates the camera.

Manuscript received ??, 2014; revised ??, 2014. Corresponding author: Adam Herout (http://www.fit.vutbr.cz/∼herout)

Because of the lack of information which can be obtained from road data, the majority of methods assume the principal point to be in the center of the image [7], [8]. A popular assumption also is a horizontal horizon line, i.e. zero roll of the camera [7], [9], [10] – which turns out to be severely limiting – because it is difficult to find an actual roadside camera meeting this assumption accurately enough.

Some works require user input in the form of annotation of the lane marking with known lane width [2] or marking dimensions [3], camera position [2], [11], average vehicle size [12] or average vehicle speed [7]. A common feature of virtually all methods is detection of the vanishing point corresponding to the direction of moving vehicles. A popular approach to obtaining this VP is to use road lines [5], [8] or lanes [8], [10], more or less automatically extracted from the image. These approaches typically require a high number of traffic lanes and a consistent and well visible lane marking.

Another class of methods disregard the line marking on the road (because of its instability and impossible automatic detection) and observe the motion of the vehicles, assuming straight and parallel trajectories in a dominant part of the view. Schoepflin and Dailey [7] construct an activity map of the road with multiple lanes and segment out individual lanes. Again, this approach relies on observing a high number of traffic lanes – high-capacity motorways and similar settings. Other researchers detect vehicles by a boosted detector and observe their movement [13], or analyze edges present on the vehicles [14]. Beymer et al. [4] accumulate tracked feature points, but also require the user to provide defined lines by manual input. Kanhere and Birchfield [15] took a specific approach for cameras placed low above the street level. Once the calibration and scene scale is available, the road plane can be rectified and various applications such as speed measurement can be done in a straightforward manner [3], [9], [11], [16].

We assume that the majority of vehicles move in straight, mutually parallel trajectories. Our experiments verify that our approach is tolerant to a high rate of outliers from this assumption and it is easily and reliably applicable on a vast majority of real traffic surveillance videos. The calibration of internal and external parameters of the camera is achieved by first computing three orthogonal vanishing points [17]–[19] which define the vehicle motion.

The calibration is provided up to scale which is generally impossible to determine (from just an image, one can never tell a matchbox models from real vehicles). The scale can be provided manually [2], [11] or recognized by detecting known objects [12].

We harness a finite and linear parameterization of the real projective plane recently published by Dubská and Herout [20]. They streamlined the Cascaded Hough Transform by stacking two stages and skipping one intermediate accumulation step. This approach allows for detection of vanishing points (and triplets of orthogonal vanishing points) from noisy linelet data. This input data can be coming online and be accumulated to a fixed-size accumulation space – which is suitable in online video processing. Instead of using road marking or lane border edges [3], [5], [7] we accumulate fractions of trajectories of detected feature points on moving objects and relevant edges.

Contrary to previous works, our approach provides the camera calibration fully automatically for very versatile camera views and road contexts (coming and going vehicles, from the side, from the top, small roads and highways, close-up and fisheye lenses, ...). The method can be applied on virtually any roadside camera without any user input whatsoever – the experiments presented in the paper were done without any per-camera or per-video settings.

We collected a set of video recordings of traffic on roads of different scales, with various cameras, in different weather conditions. The MATLAB source codes and the video dataset are publicly available for comparison and future work[1].

II. Automatic Roadside Camera Calibration

Let us consider a road scene with a single principal direction of the vehicle movement. The position of the ground plane and the direction of the vehicle movement relative to the camera can be defined and described by three vanishing points [17], [18].

Figure 1 shows the vanishing points and the color notation and VP visualization to be used throughout this paper: *red:* **First** vanishing point – in the direction of the car motion; *green:* **Second** vanishing point – in the ground plane, perpendicular to the vehicle motion; *blue:* **Third** vanishing point – perpendicular to the ground plane. We find it intuitive to use dense markers pointing towards the three vanishing points. The positions of the markers are irrelevant – they are distributed in a regular grid.

We assume cameras without skew and with equal scale in horizontal and vertical directions (aspect ratio = 1, square pixels). From our experience, these assumptions are perfectly met for all practically used surveillance cameras. Also, following most previous works in this field [7]–[10], we assume the camera's principal point to be at the image center. This assumption is met approximately, not exactly (contrary to the previous ones). However, for the target applications of our algorithms – distance/speed measurement, re-projection of observed objects into the 3D space, etc. – the error caused by this assumption is tolerable (see measurements in Section III).

Although we can afford to consider this much simplified camera model, radial distortion – usually perceived as a more advanced camera parameter – cannot be omitted for some cameras. Practically used cameras, even the expensive and sophisticated ones, tend to exhibit a high degree of radial distortion – see Figure 9 for sample images from state-of-the-art GoPro mobile and Axis surveillance cameras. Radial distortion compensation is therefore dealt with in Section II-D.

A. First Vanishing Point Extraction

The vanishing point of direction parallel to the movement of the vehicles is considered to be the **first** vanishing point. For its detection, a Hough transform based on the parallel coordinates is used [20]. This method maps the whole 2D projective plane into a finite space referred to as the *diamond space* by a piecewise linear mapping of lines.

[1]http://medusa.fit.vutbr.cz/pclines

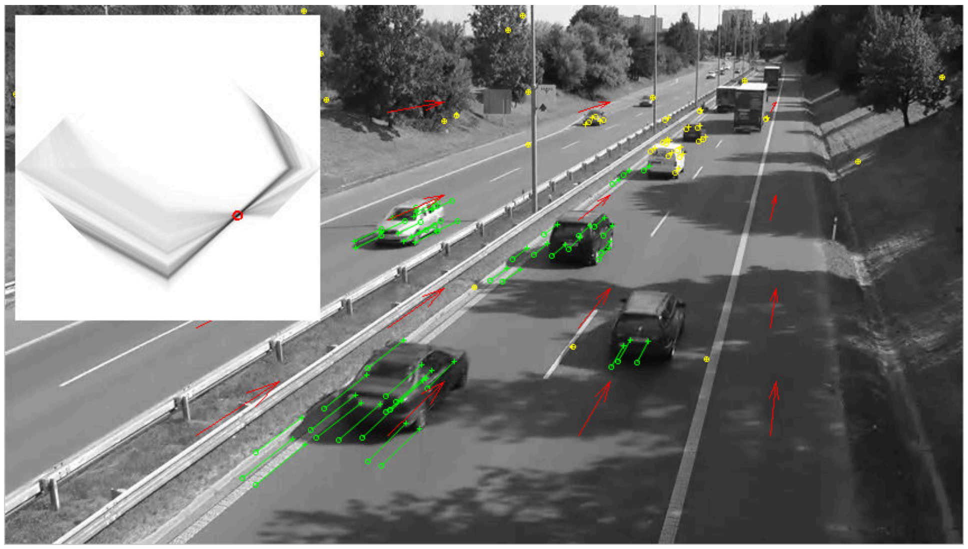


Fig. 2. Illustration of the tracked points. Points marked by green exhibit a significant movement and they are accumulated. Points marked by yellow are stable points and do not vote. The accumulated diamond space is in the top left corner.

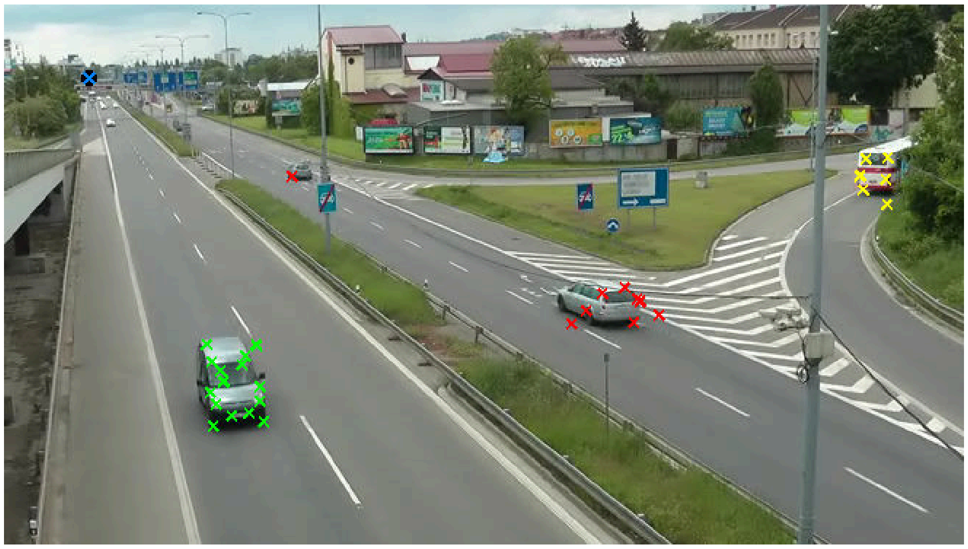


Fig. 3. Tracked points and their classification based on the first VP position. Points marked by red/green move towards/from the first VP. Points marked by yellow are moving elsewhere.

In each video frame, feature points are detected (minimum eigenvalue algorithm [21] is used in the experiments) and tracked by KLT tracker [22] in the subsequent frame. Successfully detected and tracked points exhibiting a significant movement are treated as fragments of vehicle trajectories. These fragments of trajectories are extended to infinite lines, assuming that they pass through the first vanishing point. All these lines vote in the *diamond space* accumulator. The most voted point is considered to be the first vanishing point.

The diamond space turns out to be robust to noise and it provides reliable estimates of the most distinctive vanishing point in the frame. Experiments show that in most cases, observing only two or three vehicles suffices to find a good estimate of the first VP. Later, with more observation and accumulation, the first vanishing point is very stable and accurate – we have not seen a single video where the first vanishing point was not established correctly or was unstable.

Figure 2 shows the tracked points accumulated to the diamond space. Once the first VP is determined, moving points can be discerned whether they move towards the VP or from it, or whether they are moving in a completely different direction, see Fig. 3.

B. Second Vanishing Point

The second vanishing point is the direction parallel to the road (or the ground plane) and perpendicular to the first direction. Again, the diamond space [20] is used for its detection. Many edges on the vehicles coincide with the second vanishing point and thus we let them vote in the accumulation space.

We use an edge background model in order to select only edges on moving objects – probable vehicles. The model is updated by each frame to deal with shadows and other slow

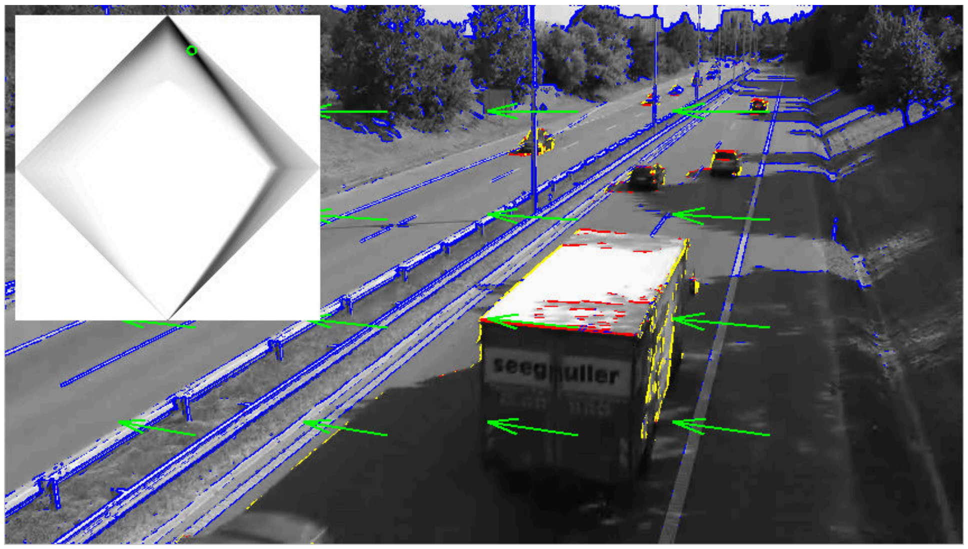


Fig. 4. Accumulation of the 2nd vanishing point. Blue edges belong to the background (Fig. 5). Yellow edges are omitted from voting because of their vertical direction or direction towards the first VP. Red edges are accumulated to the diamond space (in the corner; green circle marks the maximum).

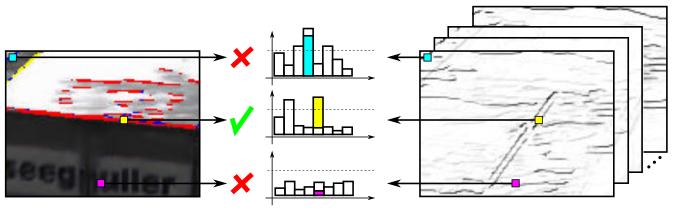


Fig. 5. Model of background edges. Significant edges are further processed (Fig. 4) only if recognized as foreground (*yellow*). Background (*cyan*) and insignificant (*magenta*) edges are omitted.

changes. The edge background model stores for each pixel the confidence score of occurrence of an oriented edge. We use eight bins to store likelihoods for different orientations, Fig. 5. Note, that the background model is computationally inexpensive because only strong edges are used, Fig. 4.

The edges passing the background test are further processed and filtered. The first vanishing point is known from previous processing and edges supporting this VP are excluded from accumulation. Also the edges with approximately vertical direction are omitted from voting, based on the assumption of scene horizon being approximately horizontal (with a high tolerance $\pm 45°$). This condition can be disregarded when the first VP is detected to be close to infinity. In such a case, the edges supporting the second VP are allowed to have vertical direction. However, these cases are not frequent, because traffic surveillance cameras are rarely observing the road exactly from profile. Figure 4 shows the edge background model, omitted and accumulated edges together with the diamond space.

C. Third Vanishing Point, Principal Point and Focal Length

The third vanishing point corresponds to the direction perpendicular to the ground plane. Unfortunately, in majority of the roadside views, there seems to be minimal amount of edges supporting the third VP. Instead of finding the third VP, we calculate its position by using the first two VPs (U and V) and the assumption that the principal point P is in the middle of the image. Two VPs and position of the principal point are sufficient for computing focal length f:

$$\begin{aligned} U = (u_x, u_y) \quad V = (v_x, v_y) \quad P = (p_x, p_y) \\ f = \sqrt{-(U-P)\cdot(V-P)}. \end{aligned} \tag{1}$$

With a known f, the third VP, denoted as W, can be calculated as

$$\begin{aligned} U' &= (u_x, u_y, f) \quad V' = (u_x, u_y, f) \\ P' &= (p_x, p_y, 0) \\ W' &= (U' - P') \times (V' - P'), \end{aligned} \tag{2}$$

where U', V', W', P' stand for the world coordinates of the points and U, V, P for the coordinates in the image plane.

When one of the first two vanishing points is in infinity, the focal length f and also the third vanishing point cannot be calculated. However, for some applications, e.g. the distance/speed measurement, knowing just first two VPs is enough. In these cases, the vanishing points in infinity are handled gracefully thanks to use of homogeneous coordinates and because the diamond space used for their detection represents the whole projective plane, including the ideal points (points in infinity).

D. Radial Distortion

The previous sections discussed the core contribution of our article – calibration of the road scene. This section extends the contribution by one more step, which is optional. In practice, some real-life cameras exhibit a large degree of radial distortion; however, some cameras are practically free from it – depending on the used optics. Depending on the particular camera, application of radial distortion compensation might not be necessary. That is why we have not discussed this issue until now; although this phase of radial distortion compensation precedes the calibration in a practical

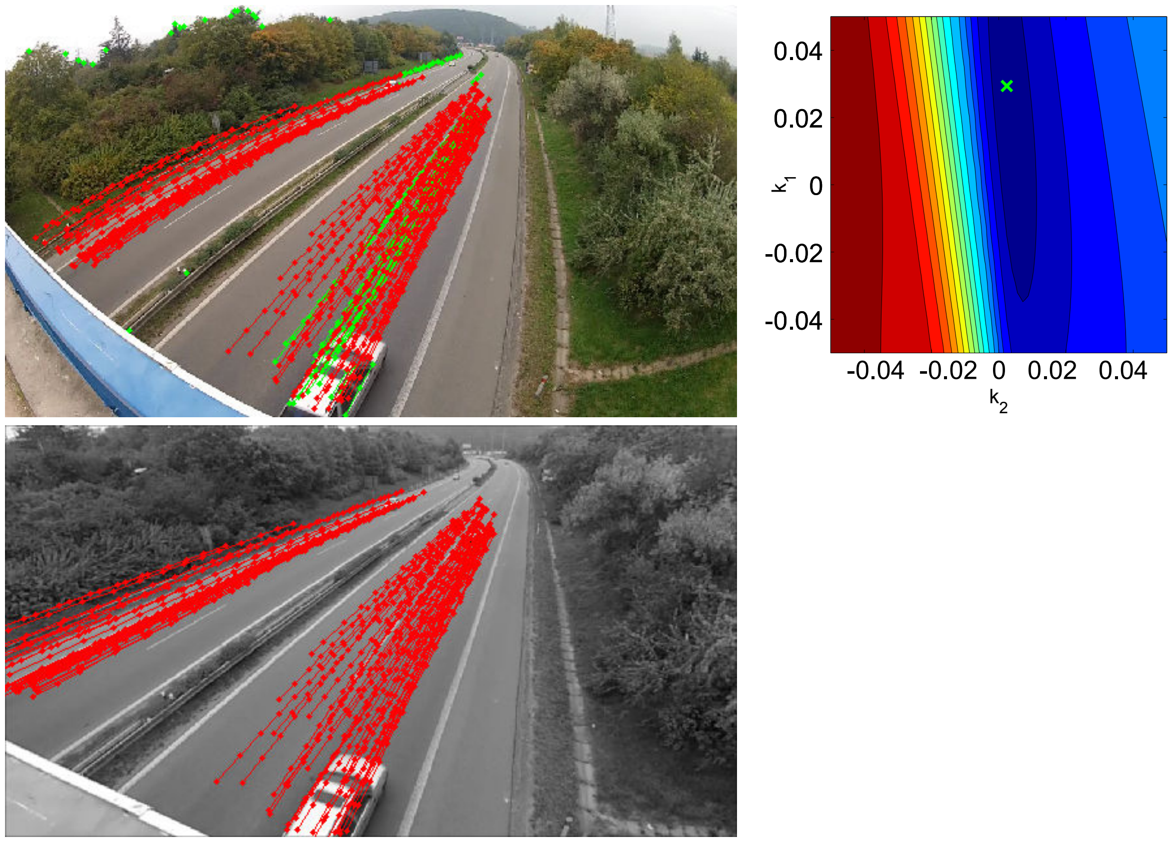


Fig. 6. Radial distortion compensation. Even though for the calibration itself just a part of the road has to be approximately straight, for radial distortion compensation, the road has to be straight along a major part of the vehicle movement. In this example, trajectories end before the turn because motion of the cars is small there and the tracker got naturally lost. **left top:** Original image with trajectories. **right:** Parameter space with value calculated from trajectories using (4). Green cross stands for the optimal parameter combination found by the evolution algorithm. The color gradient shows the error for each combination of the parameters k_1, k_2. **left bottom:** Undistorted image using the optimal coefficients.

implementation. It should be noted that apart from radial distortion, practically observed cameras are equipped with reasonable optics. Therefore, the assumption of principal point being in the image center (or close to it), absence of skew and equal scale in x and y directions ("square pixels") are kept and the presented algorithm is general.

Provided the assumption of the road being straight, the tracked trajectories can be used to compensate for the camera's radial distortion [23]. It should be noted that the requirement for straight road is much less strict in the case of the calibration itself (refer to Figure 8). The corrected position $(\overline{x}, \overline{y})$ of input point (x, y) can be modeled by the polynomial radial distortion model [24] defined by:

$$\begin{aligned} \overline{x} &= (x - x_c)(1 + k_1 r^2 + k_2 r^4 + \dots) \\ \overline{y} &= (y - y_c)(1 + k_1 r^2 + k_2 r^4 + \dots), \\ r &= \sqrt{(x - x_c)^2 + (y - y_c)^2} \end{aligned} \qquad (3)$$

where (x_c, y_c) define the position of the distortion center and coefficients k_i are unknown distortion parameters.

In order to find $K = \{k_1, k_2, \dots..\}$, the extracted trajectories T are used. Each point is tracked until the tracker is lost. Each trajectory represented by a sequence of points $\tau = \{a_1, a_2, \dots\} \in T$, are projected by (3) to their transformed versions $\overline{\tau}_K$. Optimal parameters K^* are found by minimization of the sum of square differences of all points in all trajectories to their best fitting lines $\ell_{\overline{\tau}_K}$:

$$K^* = \arg\min_K \sum_{\overline{\tau} \in T} \sum_{\overline{a} \in \overline{\tau}} (\ell_{\overline{\tau}_K} \cdot \overline{a})^2. \qquad (4)$$

We use $(1 + \lambda)$-ES evolutionary strategy (with $\lambda = 8$) [25] to search for the first two coefficients. The optimization was done on-line. When new trajectories were tracked, one iteration of the optimization was executed. The whole radial distortion compensation process is shown in Figure 6.

In practice, the radial distortion compensation is computed first. Then, accumulation of the vanishing points is performed on the undistorted tracked features, i.e. the tracked features and edges are transformed by eq. (3) and algorithms in Sections II-A and II-B are working on $(\bar{x}, \bar{y})$ pairs. Once the radial distortion is compensated for, accumulation of the two VPs can happen simultaneously.

E. Camera Calibration from the Vanishing Points

The problem of obtaining camera projection matrix $\mathbf{P}$ from detected vanishing points has already been discussed several time. Therefore, in this section we provide only a brief overview; more information can be found elsewhere [10], [14]. Projection matrix $\mathbf{P}$ transforms a world point $[x_w, y_w, z_w, 1]'$ into point $[x_p, y_p, 1]'$ in the image plane:

$$\lambda_p [x_p, y_p, 1]' = \mathbf{P} [x_w, y_w, z_w, 1]'. \qquad (5)$$

Projection matrix $\mathbf{P}$ can be decomposed into three matrices – one with intrinsic parameters of the camera $\mathbf{K}$ and two with extrinsic parameters: rotation matrix $\mathbf{R}$ and translation vector $\mathbf{T}$:

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}\ \mathbf{T}]. \tag{6}$$

With the assumption of zero skew, location of the principal point in the center of image plane and unit aspect ratio, the only parameter to be found in matrix $\mathbf{K}$ is the focal length f derived before (1). The rotation matrix $\mathbf{R}$ is fully defined by the positions of the three orthogonal VPs. For the translation vector $\mathbf{T}$, additional information have to be known; it can be derived from the height of the camera from the ground – as discussed by Zhang et al. [14] – or a known distance of two projected points.

III. Experimental Results

We evaluated our approach on 5 groups of videos, each containing 5–8 videos. Videos in a common group share the same camera intrinsic parameters (no changes of camera settings were done between shots) but have different extrinsic parameters and capture different scenes or scenes from a different view.

First, we logically intended to evaluate the calibration accuracy by comparing the obtained camera parameters with calibration parameters obtained by checkerboard-based calibration [26]. In many cases, the focal length of the cameras was high (cameras zoomed in) and the checkerboard calibration was inaccurate itself (different individual calibrations ended in dramatically different results). In cases of some cameras, it could have been caused by the camera automatically refocusing to the calibration pattern (close to the cameras) and back to the traffic scene. That is why we had to select a different approach to evaluation – based on the intended application tasks: distance/speed measurement. The same approach has already been used in literature [14], which allows for fair comparison.

In order to evaluate the accuracy of the vanishing points detection, we compute the precision of length measurements in videos similarly to Zhang et al. [14]. From each video, 15–25 pairs of feature points are tracked in 21 subsequent frames. These points are projected with the matrix obtained from the vanishing points and we evaluate the stability of their distance d. Error of i^{th} pair in j^{th} sequence is calculated as

$$e_{ji} = \left| 1 - \frac{d_{ij}}{\overline{d}_j} \right|, \tag{7}$$

where $\overline{d}_j$ is the mean distance in the j^{th} sequence. For each video, two errors are computed from e_{ji} – the worst error (e^v_w) and the mean error (e^v_m). The same is computed for each group of videos (e^g_w, e^g_m).

Table I shows the worst and mean error for the groups and the computed focal lengths. The focal length f is taken from the video with the lowest e^v_m in the group. We mention it here in order to illustrate the differences in the camera settings. Larger f leads to smaller length-measurement error due to smaller perspective distortion and consequent smaller dependence on the point tracker accuracy.

group	g1	g2	g3	g4	g5
e^g_w (%)	6.5	1.8	10.1	5.3	4.0
e^g_m (%)	1.2	0.2	1.3	0.8	0.7
f	705.7	7163.7	674.6	769.6	2465.1

TABLE I
Mean and worst length-measurement error for groups of videos in % and the computed focal lengths.

video	v1	v2	v3	v4	v5	v6
e^v_w (%)	5.5	6.0	5.2	4.7	5.3	6.4
e^v_m (%)	1.0	1.3	1.1	0.8	1.3	1.6
f	742.0	688.2	685.0	705.7	803.8	830.0

TABLE II
Mean and worst length-measurement error for individual videos within group g1 and computed focal lengths.

As a particular example, Table II shows the mean and worst errors and the computed focal lengths for all videos from group g1 (one video from the group is shown in Fig. 2).

The extracted focal length in Table II differs. Its error can be caused by some videos having the second VP near infinity and by camera refocusing automatically (because the viewpoint changed). However, an error in estimation of camera f does not prevent the measurement (the desired traffic surveillance application) to be precise enough in all cases. This is because the VP near infinity does not increase the measurement error (although it spoils f). Zhang et al. [14] report similar measurements (single scene, 28 point pairs, 6 sequences), their mean error appears to be 6 %, the worst 19 %, the second worst 13 %.

A. Evaluation of Speed

Our algorithm is fairly efficient – capable of processing the video stream in real time. In order to demonstrate and evaluate this, we created an efficient implementation in C++ and processed the input videos used in the evaluation above. Table III shows the speed measurement results. The measurements were done on a machine with Intel Dual-Core i5 1.8 GHz and 8GB DDR3 RAM and the framerates are reported for pure computation (video decompression etc. are omitted).

resolution	traffic intensity	VP1	VP2
		(ms)	
854×480	high	19	14
	medium	19	11
1920×1080	high	98	69
	medium	97	57

TABLE III
Speed evaluation. *VP1* – first VP accumulation (Sec. II-A), *VP2* – second VP accumulation (Sec. II-B). The times are averaged from all measured videos (approximately 3 minutes each) in two groups (*traffic intensity*) which differ slightly in their quantity of observed cars. It should be noted that our algorithm processes only ~ 5 frames per second so that the movement of tracked points is measurable and stable. Therefore, the measured times in all cases (including the full-HD video processing) allow for comfortable real-time processing.

Our C++ implementation of first VP detection uses only a limited number of tracked feature points therefore the framerates are invariant to the traffic intensity. However, the

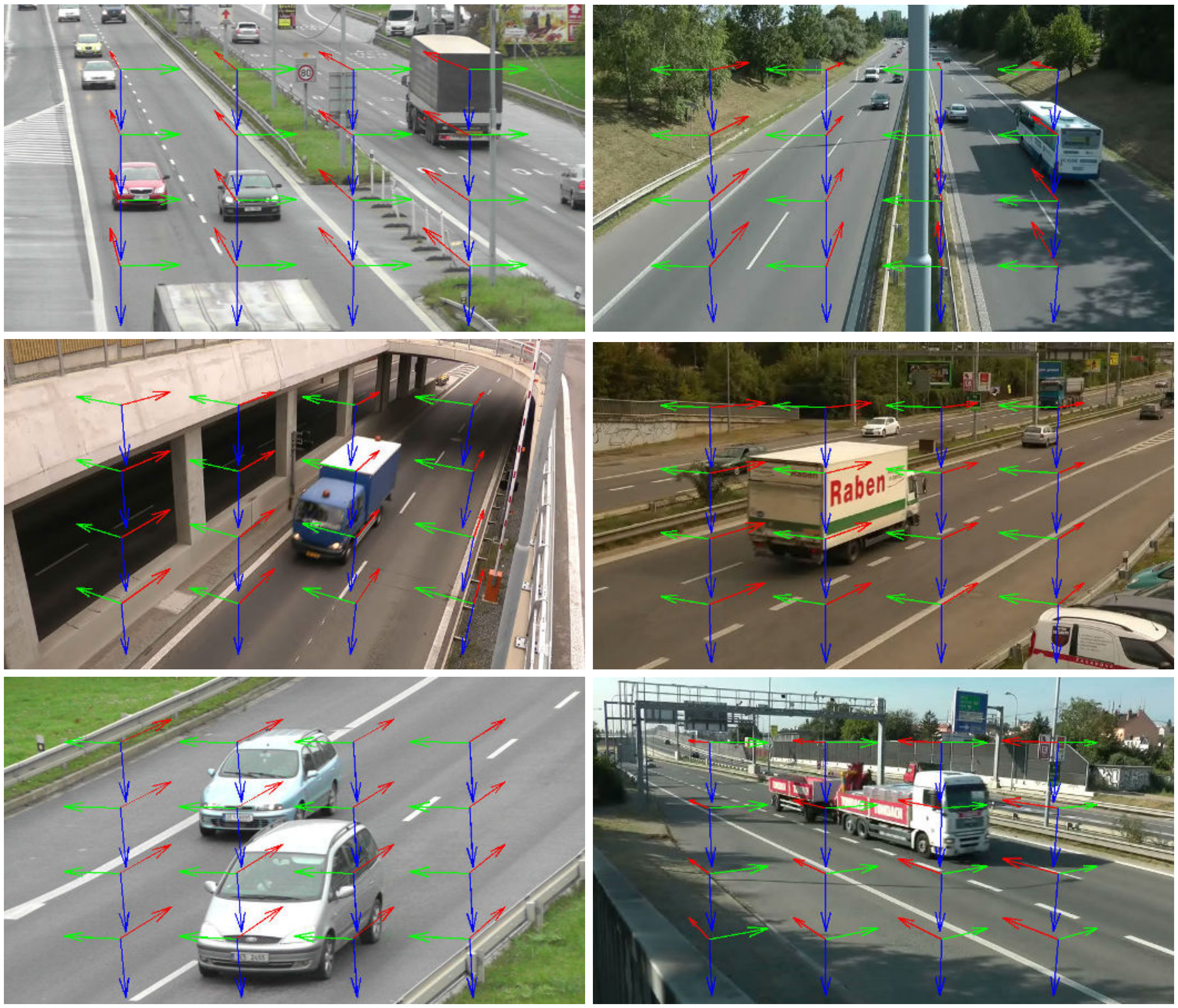


Fig. 7. Examples of real-life videos: Automatic detection of three vanishing points. Here is a small selection of traffic scenes, more samples can be found in the supplementary video. Three vanishing points are robustly found regardless of camera f (zoom), shadows, lighting conditions, camera position with respect to the road (for reasonable traffic surveillance scenarios).

framerate of the second VP detection differs with respect to the traffic intensity. The variance is caused especially by the necessity to handle edge points of passing cars and therefore accumulate more lines corresponding to the edge points into the diamond space [20]. The second VP detection framerate also depends on motion of other objects (people, moving trees etc.) in the video stream.

B. Accumulation Convergence

The convergence of the accumulation of first and second VP is shown in Figure 11. For the vast majority of the videos, the first vanishing point remains totally stable since 160 seconds of 50 FPS video and the second VP is stable after processing 250 seconds of video (Zhang et al. [14] mention processing of 2 hours of recording).

IV. Conclusions

We present a method for fully automatic calibration of roadside cameras. It requires no user input and assumes very little about the scene itself. We experimented with videos taken at roads of different classes (from small streets with little traffic to busy highways) by various cameras (handycam, GoPro, IP cameras, high-end and cheap ones) and lenses (from close-up views to nearly fisheye). The results are stable, reliable and usable by various applications without any per-video or per-camera settings. The efficient implementation is fast and the concept is thus ready for real-time implementation on low-end hardware, even on full-HD video sources.

The solution consists of a method for compensation of radial distortion and a way of obtaining three orthogonal vanishing points related to the motion of the vehicles. These three orthogonal VPs define intrinsic and extrinsic camera

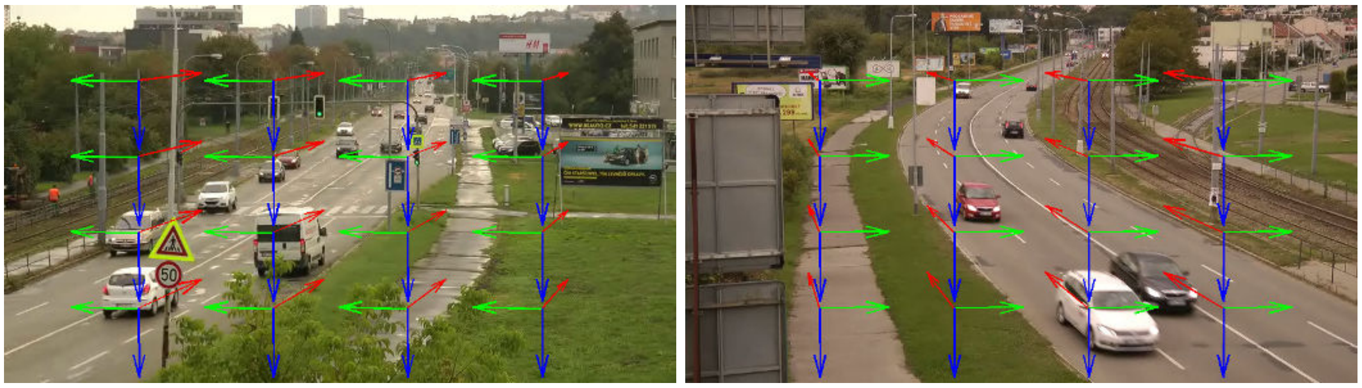


Fig. 8. Examples of real-life videos: Successful camera calibration/orientation in scenes with bent roads. These are to illustrate that the assumption of *straight* vehicle motion is not very strict. However, radial distortion would be more difficult to compensate in these scenes.

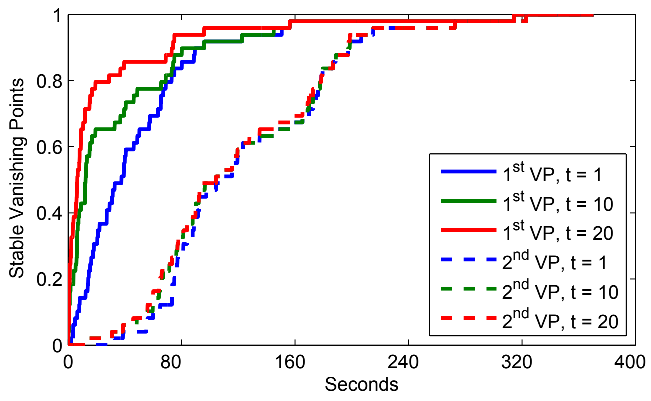


Fig. 11. Convergence of VP1 and VP2 computation. We computed the pixel distance between the vanishing point detected at the given time and the final detection. For a threshold t, the time after which the distance is lower then t is evaluated. The graph shows the fraction of videos reaching the threshold condition withing the given number of seconds. It should be noted that only every 4^{th} frame in a 50fps video was processed for the same reason as in Table III.

parameters. Virtually any roadside scene captured by a static camera can be fully automatically calibrated up to scale. Our method assumes approximately straight motion of the vehicles at least along a large portion of their motion. Bent roads and sharp corners followed/preceded by a straight stretch of the road are easily dealt with.

The main contribution and advantage is that we strictly avoid any real-life measurement in the scene and/or any manual input. Our algorithms open space for fully automatic processing of almost any traffic surveillance video footage. We collected a set of evaluation videos (see supplementary video for examples) accompanied by ground truth calibration parameters. This dataset, together with the MATLAB sources is made available for comparison and future work[2].

References

[1] N. Kanhere and S. Birchfield, "A taxonomy and analysis of camera calibration methods for traffic monitoring applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 441–452, 2010.

[2]http://medusa.fit.vutbr.cz/pclines

[2] K. Wang, H. Huang, Y. Li, and F.-Y. Wang, "Research on lane-marking line based camera calibration," in *International Conference on Vehicular Electronics and Safety, ICVES*, 2007.
[3] F. Cathey and D. Dailey, "A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras," in *Intelligent Vehicles Symposium*, 2005, pp. 777–782.
[4] 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, CVPR*, 1997.
[5] R. Dong, B. Li, and Q.-m. Chen, "An automatic calibration method for PTZ camera in expressway monitoring system," in *World Congress on Computer Science and Information Engineering*, 2009, pp. 636–640. [Online]. Available: http://dx.doi.org/10.1109/CSIE.2009.763
[6] L. Liu, J. Xing, G. Duan, and H. Ai, "Scene transformation for detector adaptation," *Pattern Recognition Letters*, 2013.
[7] T. Schoepflin and D. Dailey, "Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, no. 2, pp. 90–98, 2003.
[8] K.-T. Song and J.-C. Tai, "Dynamic calibration of Pan–Tilt–Zoom cameras for traffic monitoring," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 36, no. 5, pp. 1091–1103, 2006. [Online]. Available: http://dx.doi.org/10.1109/TSMCB.2006.872271
[9] X. C. He and N. H. C. Yung, "A novel algorithm for estimating vehicle speed from two consecutive images," in *IEEE Workshop on Applications of Computer Vision, WACV*, 2007.
[10] G. S. K. Fung, N. H. C. Yung, and G. K. H. Pang, "Camera calibration from road lane markings," *Optical Engineering*, vol. 42, no. 10, pp. 2967–2977, 2003. [Online]. Available: http://dx.doi.org/10.1117/1.1606458
[11] T.-W. Pai, W.-J. Juang, and L.-J. Wang, "An adaptive windowing prediction algorithm for vehicle speed estimation," in *IEEE Intelligent Transportation Systems*, 2001.
[12] D. Dailey, F. Cathey, and S. Pumrin, "An algorithm to estimate mean traffic speed using uncalibrated cameras," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 2, pp. 98–107, 2000.
[13] N. K. Kanhere, S. T. Birchfield, and W. A. Sarasua, "Automatic camera calibration using pattern detection for vision-based speed sensing," *Journal of the Transportation Research Board*, vol. 2086, no. 1, pp. 30–39, 2008.
[14] Z. Zhang, T. Tan, K. Huang, and Y. Wang, "Practical camera calibration from moving objects for traffic scene surveillance," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 3, pp. 518–533, 2013.
[15] N. K. Kanhere and S. T. Birchfield, "Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 148–160, 2008. [Online]. Available: http://dx.doi.org/10.1109/TITS.2007.911357
[16] F. Cathey and D. Dailey, "Mathematical theory of image straightening with applications to camera calibration," in *Intelligent Transportation Systems Conference*, 2006.
[17] R. Cipolla, T. Drummond, and D. P. Robertson, "Camera calibration

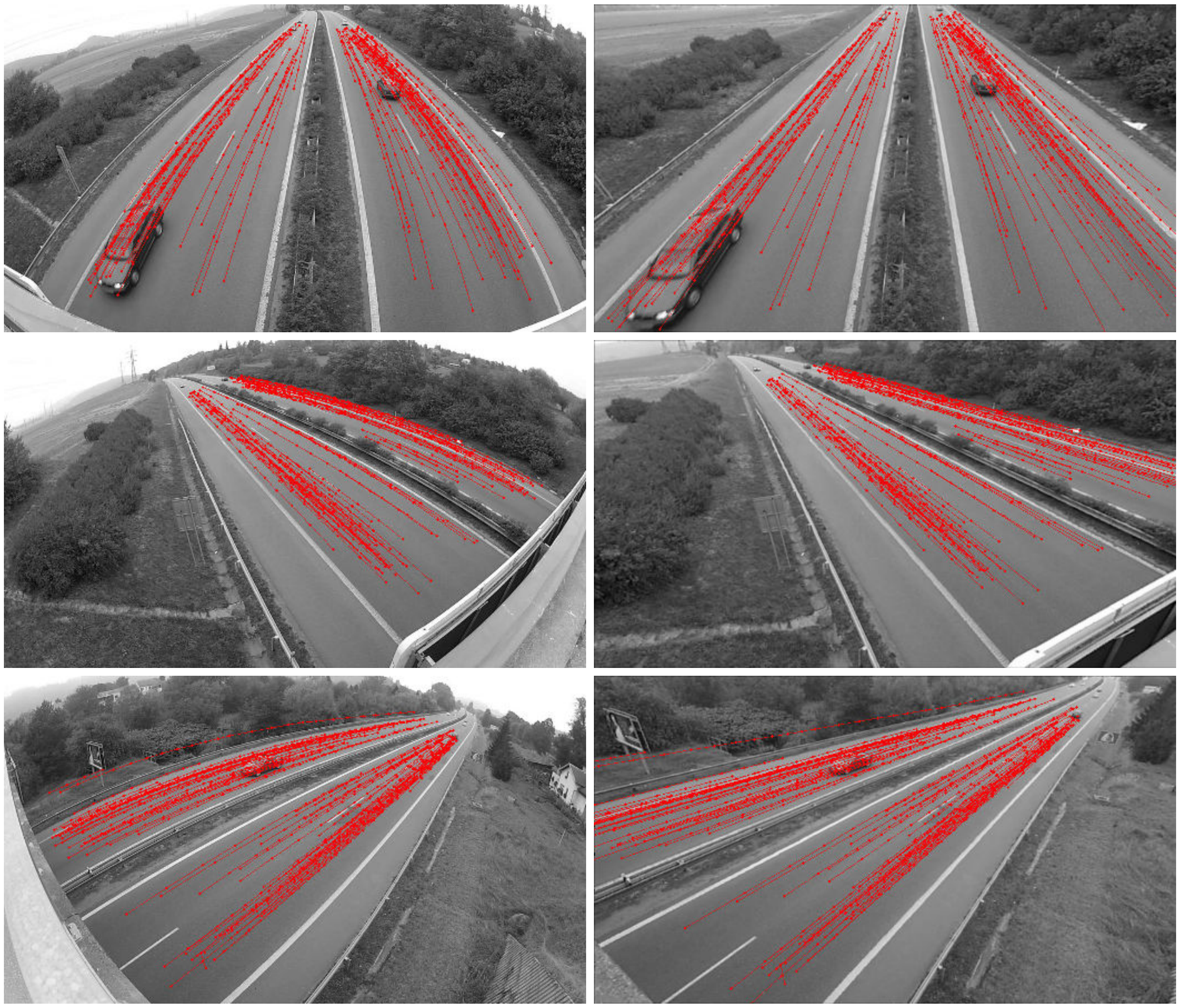


Fig. 9. Examples of real-life videos: Radial distortion estimation and compensation. **left:** Original video (converted to grayscale). **right:** Processed video with radial distortion compensated. **red polylines:** Tracks of features used in the computation.

from vanishing points in image of architectural scenes." in *British Machine Vision Conference, BMVC*, 1999.

[18] B. Caprile and V. Torre, "Using vanishing points for camera calibration," *International Journal of Computer Vision*, vol. 4, no. 2, pp. 127–139, 1990.

[19] J. Deutscher, M. Isard, and J. MacCormick, "Automatic camera calibration from a single manhattan image," in *European Conference on Computer Vision, ECCV*, 2002, pp. 175–188. [Online]. Available: http://dx.doi.org/10.1007/3-540-47979-1-12

[20] M. Dubská and A. Herout, "Real projective plane mapping for detection of orthogonal vanishing points," in *British Machine Vision Conference, BMVC*, 2013.

[21] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 1994, pp. 593–600.

[22] C. Tomasi and T. Kanade, *Detection and tracking of point features*. School of Computer Science, CMU, 1991.

[23] F. Devernay and O. Faugeras, "Straight lines have to be straight," *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 2001.

[24] D. C. Brown, "Close-range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[25] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies &ndash;a comprehensive introduction," vol. 1, no. 1, pp. 3–52, May 2002.

[26] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI*, vol. 22, no. 11, pp. 1330–1334, 2000.

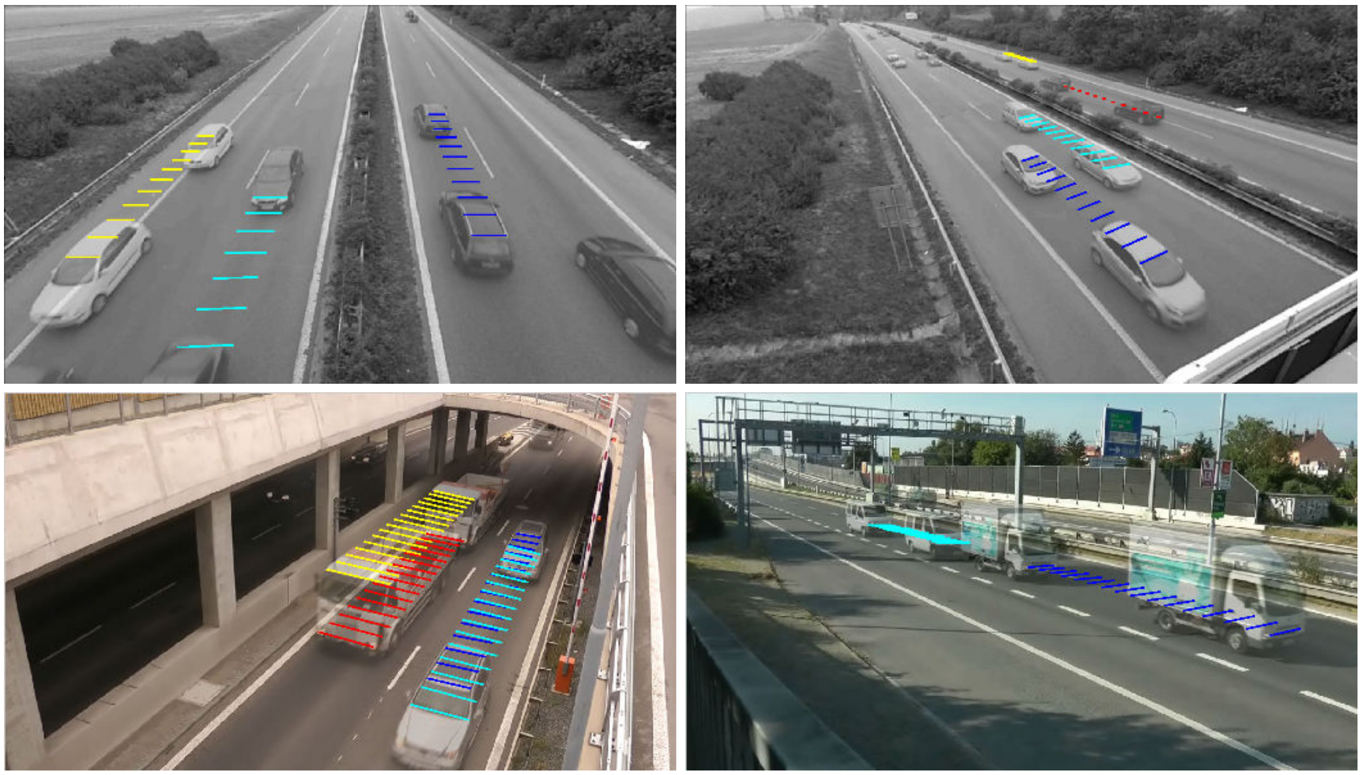


Fig. 10. Examples of real-life videos: Ghost images (composition of two images of different moments in time by blending) and measured equal distances. These lines are observations of identical object measures at different times (consecutive frames). These distances are used in the evaluation (Tables I, II).

Markéta Dubská received the MS degree at Faculty of Information Technology, Brno University of Technology, Czech Republic. She is currently a PhD student at Department of Computer Graphics and Multimedia at FIT Brno University of Technology. Her research interests include computer vision, geometry and computation using parallel coordinates.

Jakub Sochor received the Bachelor degree at Faculty of Information Technology, Brno University of Technology, Czech Republic. He is currently in the last year of the master's degree at FIT BUT. Jakub Sochor focuses on research in computer vision, especially in traffic surveillance.

Adam Herout received his PhD from Faculty of Information Technology, Brno University of Technology, Czech Republic, where he works as associate professor and leads the Graph@FIT research group. His research interests include fast algorithms and hardware acceleration in computer vision. Adam Herout is a co-founder of click2stream.com, which provides web streaming from network cameras and real-time computer vision in the cloud.

Roman Juránek received MS degree from the Brno University of Technology, CZ, in 2007. In 2012 he defended his PhD at the Faculty of Information Technology, Brno University of Technology, CZ. He is member of Graph@FIT research group at Department of Computer Graphics and Multimedia on FIT BUT. His professional interests include Computer Vision, Machine Learnin and Pattern Recognition.

Automatic Camera Calibration for Traffic Understanding

Markéta Dubská[1]
idubska@fit.vutbr.cz
Jakub Sochor[1]
xsocho06@stud.fit.vutbr.cz
Adam Herout[12]
herout@fit.vutbr.cz

[1] Graph@FIT
Brno University of Technology, CZ
[2] click2stream, Inc.

Abstract

We propose a method for fully automatic calibration of traffic surveillance cameras. This method allows for calibration of the camera – including scale – without any user input, only from several minutes of input surveillance video. The targeted applications include speed measurement, measurement of vehicle dimensions, vehicle classification, etc. The first step of our approach is camera calibration by determining three vanishing points defining the stream of vehicles. The second step is construction of 3D bounding boxes of individual vehicles and their measurement up to scale. We propose to first construct the projection of the bounding boxes and then, by using the camera calibration obtained earlier, create their 3D representation. In the third step, we propose a method to using the dimensions of the 3D bounding boxes for calibration of the scene scale. This facilitates new automatic applications based on measurement of speed and real-world dimensions. We collected a dataset with ground truth speed and distance measurements and evaluate our approach on it. The achieved mean accuracy of speed and distance measurement is below 2 %. Our efficient C++ implementation runs in real time on a low-end processor (Core i3) with a safe margin even for full-HD videos.

1 Introduction

Automatic visual surveillance is useful in organization of traffic – for collecting statistical data [22], for immediate controlling of traffic signals [21], for law enforcement [17, 30], etc. Existing systems typically require manual setup, often involving physical measurements in the scene of interest [13]. Our goal is to process traffic data fully automatically, without any user input. This includes assessment of camera intrinsic parameters, extrinsic parameters in relation to the stream of traffic, and scale of the ground plane which allows for measurement in the real world units – Fig. 1.

Some existing works in automatic traffic surveillance require user input in the form of annotation of the lane marking with known lane width [32] or marking dimensions [4], camera position [24, 32], average vehicle size [7, 29] or average vehicle speed [25]. A common

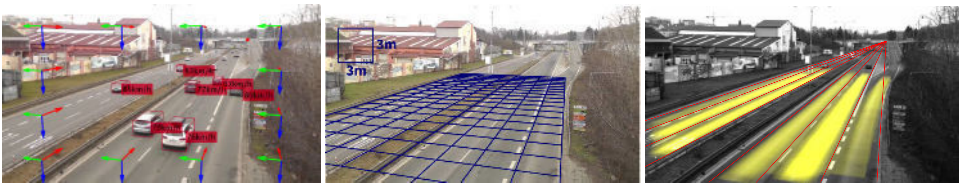


Figure 1: We automatically determine 3 orthogonal vanishing points, construct vehicle bounding boxes *(left)*, and automatically determine the camera scale by knowing the statistics of vehicle dimensions. This allows us to measure dimensions and speed *(middle)* and analyze the traffic scene *(right)*.

feature of virtually all methods is detection of the vanishing point corresponding to the direction of moving vehicles (full camera calibration requires three orthogonal vanishing points [3, 6, 9]). A popular approach to obtaining this VP is to use road lines [10, 26, 35] or lanes [8, 12, 26], more or less automatically extracted from the image. These approaches typically require a high number of traffic lanes and a consistent and well visible lane marking. Another class of methods disregard the line marking on the road (because of its instability and impossible automatic detection) and observe the motion of the vehicles, assuming straight and parallel trajectories in a dominant part of the view. Schoepflin and Dailey [25] construct an activity map of the road with multiple lanes and segment out individual lanes. Again, this approach relies on observing a high number of traffic lanes – high-capacity motorways and similar settings. Other researchers detect vehicles by a boosted detector and observe their movement [19], or analyze edges present on the vehicles [34]. Beymer et al. [2] accumulate tracked feature points, but also require the user to provide defined lines by manual input. Kanhere and Birchfield [18] took a specific approach for cameras placed low above the street level. Once the calibration and scene scale is available, the road plane can be rectified and various applications such as speed measurement can be done [4, 5, 14, 24, 36].

In our approach, we assume that the majority of vehicles move in approximately straight, mutually parallel trajectories.[1] Also, the trajectories do not have to be approximately straight across their whole span – only a significant straight part is sufficient. This makes our approach easily and reliably applicable on a vast majority of real traffic surveillance videos. The calibration of internal and external parameters of the camera is achieved by first computing three orthogonal vanishing points which define the vehicle motion [1].

Similarly to others [25, 26], we assume a pinhole camera with principal point in the image center. The principal point would be difficult (or impossible) to obtain otherwise, because the camera cannot move and no calibration pattern can be used. At the same time, this assumption does not harm the targeted applications (speed/distance measurement, traffic lane identification, ...). Unlike previous works, we do not assume exactly horizontal scene horizon [12, 14, 25]. We find this assumption too limiting and we deal with it by properly finding the second vanishing point defining the scene (Sec. 2.1). We assume zero radial distortion of the camera, but our previous work [1] offers a solution for automatic radial distortion compensation.

Once the camera intrinsic and extrinsic calibration (up to scale) defined by three orthogonal vanishing points is determined, we propose to construct 3D bounding boxes of the vehicles based on the assumption of flat ground plane. The dimensions of the 3D bounding boxes of a number of observed cars (experiments show that after processing approximately

[1]Experiments verify that our method is tolerant to a high number of outliers from this assumption.

50 cars, the scale is within 2 % from the final value) can be used for adaptation of the scale to a known distribution of car dimensions. The proposed 3D bounding boxes are easily constructed and their construction is computationally cheap. At the same time, they provide some 3D insight into the scene observed by a stationary camera, unavailable to existing approaches mentioned earlier. We are showing that once the camera calibration including scale is computed, our method allows for reasonably accurate measurement of vehicle speed and various dimensions in the scene, including 3D dimensions of passing vehicles. The bounding boxes can be used for other tasks as well – we are showing improved analysis of traffic lanes directly obtained from the geometry of the bounding boxes.

2 Traffic Analysis from Uncalibrated Cameras

Section 2.1 reviews our camera calibration algorithm [1]. Based on it, we propose to construct 3D bounding boxes of observed vehicles (Sec. 2.2). The dimensions of bounding boxes are statistically domain-adapted to known distribution of vehicle dimensions (Sec. 2.3) in order to obtain the scene-specific scale.

2.1 Camera Calibration from Vehicle Motion

In order to make this paper self-contained, we briefly summarize our calibration method [1] (currently in the publication process). This method enables recovering of the focal length of the camera and its orientation with respect to the stream of traffic. It detects two originally orthogonal directions – 1st in the direction of the traffic and 2nd which is perpendicular to the 1st direction and parallel to the road. Assuming that the camera's principal point is in the center of the projection plane, the 3rd orthogonal direction and the focal length can be calculated. The first two directions are detected using their vanishing points on the projection plane. The detection method uses Hough transform based on the parallel coordinates [11]. This method maps the whole 2D projective plane into a finite space referred to as the *diamond space* by a piecewise linear mapping of lines.

For the detection of the 1st vanishing point, feature points are detected and tracked by KLT tracker in the subsequent frame. Successfully detected and tracked points exhibiting a significant movement are treated as fragments of vehicle trajectories. These fragments of trajectories are extended to infinite lines, assuming that they pass through the first vanishing point. All these lines vote in the *diamond space* accumulator. The most voted point is considered to be the first vanishing point. Figure 2 (*left*) shows the tracked points accumulated to the diamond space.

The second vanishing point corresponds to the direction parallel to the road (or the ground plane) and is perpendicular to the first direction. Again, the diamond space [11] is used for its detection. Many edges on the vehicles coincide with the second vanishing point and thus we let them vote in the accumulation space. An edge background model is used in order to select only edges on moving objects – probable vehicles. The model is updated by each frame to deal with shadows and other slow changes. The edge background model stores for each pixel the confidence score of occurrence of an oriented edge (eight bins are used to store likelihoods for different orientations). The edges passing the background test are further processed and filtered. The first vanishing point is known from the previous processing and edges supporting this VP are excluded from accumulation. Also the edges with approximately vertical direction are omitted from voting, based on the assumption of scene

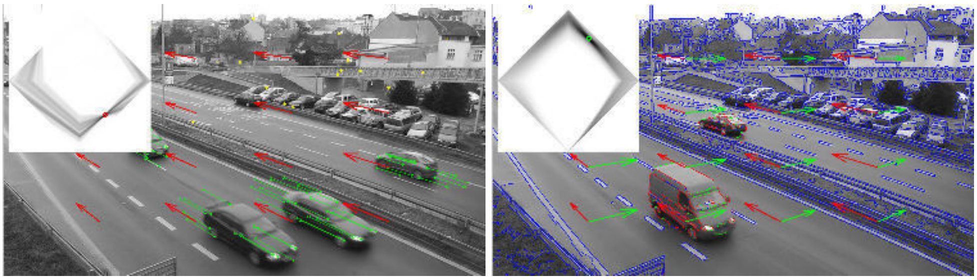


Figure 2: (*left*) Illustration of the tracked points used for estimation of the 1st VP. Points marked by green exhibit a significant movement and they are accumulated. Points marked by yellow are stable points and do not vote. The accumulated diamond space is in the top left corner. (*right*) Accumulation of the 2nd vanishing point. Blue edges belong to the background. Red edges are omitted from voting because of their vertical direction or direction towards the first VP. Green edges are accumulated to the diamond space (in the top left corner; green circle marks the maximum).

horizon being approximately horizontal (with a high tolerance, e.g. $\pm 45°$). This condition can be disregarded when the first VP is detected to be close to infinity. In such a case, edges supporting the second VP are allowed to have vertical direction. Figure 2 (*right*) shows the edge background model, omitted and accumulated edges together with the diamond space.

2.2 Construction of 3D Bounding Boxes

The next step of our approach is construction of 3D bounding boxes of the observed vehicles (see Fig. 3 (IV) for an example). We assume that vehicle silhouettes can be extracted by background modeling and foreground detection [27, 37]. Detection of foreground blobs for vehicles can be done reliably, including removal of shadows [15]. Further we assume that the vehicles of interest are moving from/towards the first vanishing point (Sec. 2.1). In fact, all detected foreground blobs in the input video are filtered by this criterion, which leads to disposal of invalid blobs.

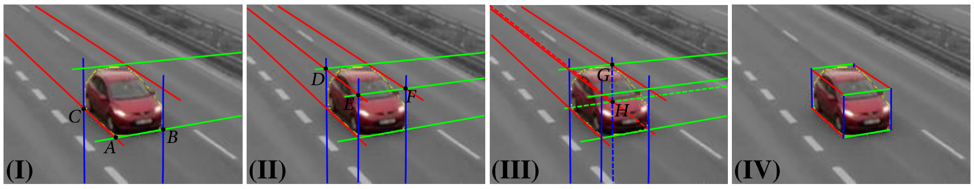


Figure 3: Construction of vehicle's 3D bounding box. **(I)** Tangent lines and their relevant intersections A,B,C. **(II)** Derived lines and their intersections E,D,F. **(III)** Derived lines and intersection H. **(IV)** Constructed bounding box.

Our approach is based on an observation, that vehicle blobs tend to have some edges very stable and reliable. Refer to Fig. 3 for an illustration where the detected blob of the car is colored and rest of the image is desaturated. In the given situation, red lines pass through the 1st VP and they are tangent to the vehicle's blob. Green lines are blob's tangents coinciding with the 2nd VP; blue tangents pass through the 3rd VP. The two tangents corresponding to the VP are lines with minimal and maximal orientation passing thought the VP and the points from convex hull of the blob.

Because the blobs are not accurate and the cars are not exactly boxes, the fitting of the bounding box is ambiguous, i.e. the order in which the tangents and their intersections are extracted matters. We propose the following order, which appears to be the most stable one. Firstly, point A is constructed as the intersection of the lower red and green tangent. Then, points B and C are defined by intersections of the lower green tangent with right blue and the lower red with left blue, respectively, Fig. 3 (I). Constructed line segments AB and AC define the shorter and the longer side of the box base. Point D lies on the intersection of the upper green tangent and the left blue tangent. Together with the line passing through point A and the 3$^{\text{rd}}$ VP it uniquely defines point E, Fig. 3 (II). Point E can be also constructed using point F – leading to an alternative position of point E. We choose point E with the larger distance $|AE|$, which ensures that the whole blob will be enclosed in the bounding box. With known F and D, the point G is the intersection of the line through D and 2$^{\text{nd}}$ VP with line through F and 1$^{\text{st}}$ VP, Fig. 3 (III).

When the configuration of the vanishing points with respect to the center of the foreground blob is different from the one discussed in the previous paragraphs, the set and order of used tangent lines and points slightly changes. The change is self-evident and follows the principles sketched above. Figure 4 shows other possible orientations of the bounding box with respect to different configurations of VPs.

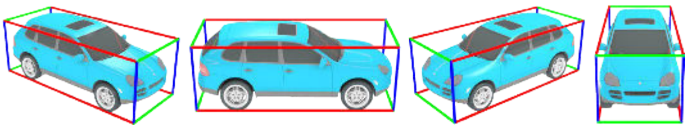


Figure 4: Different bounding boxes depending on positions of the vanishing points with respect to the camera. Because of rounded corners of the car, the edges of the bounding box would not fit tight to the car. However, in most cases, at least one dimension fits tight and this is enough to find the scale.

Because the roof and sides of the car are typically somewhat bent, the detected bounding box can be slightly smaller that in reality. However, we count with this inaccuracy in the domain adaptation procedure and prefer the best matching pair of bounding box sides for further computation. The experiments show that the final accuracy is not harmed by the slightly diminished detected bounding boxes (Sec. 3).

In order to be able to determine the vehicles dimensions accurately, shadows need to be removed from the detected foreground objects. Elaborate shadow removal exceeds the scope of our work, but it has been addressed by other researchers [31, 33]. In our work, we assume only the presence of soft shadows and we use the method of Horprasert et al. [15] for their removal.

2.3 Statistical Domain Adaptation of Vehicle Dimensions

Having the bounding box projection, it is directly possible to calculate the 3D bounding box dimensions (and position in the scene) up to precise scale. The construction is shown in Figure 5. We consider a three-dimensional coordinate system with camera coordinates $O = [p_x, p_y, 0]$, center of the projection plane $P = [p_x, p_y, f]$ (where $[p_x, p_y]$ is the principal point) and three orthogonal directions derived from the detected vanishing points in the image. Firstly, plane $\wp$ parallel to the road ground plane is constructed – its orientation is known

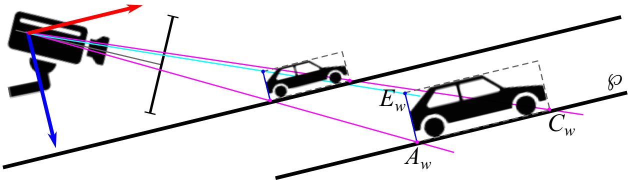


Figure 5: Calculation of the world coordinates. Plane $\wp$ is parallel to the road and it is derived from the detected VPs. Its distance is selected arbitrarily and precise scale is found later, Fig. 6. The camera is placed in $O = [p_x, p_y, 0]$ and world points of the base of the bounding box are intersections of plane $\wp$ with rays from O through points A, C (constructed earlier in the projection plane). Other points are intersections of rays from O through projected points and rays perpendicular to $\wp$ passing through points A_w, B_w, C_w, H_w.

since the direction of the 3rd VP is perpendicular to this plane; its distance from the camera is chosen arbitrarily. Figure 5 shows two possible placements of the plane and the influence of such placement – the closer the plane is to the camera, the smaller the objects appear. The detected corners of the bounding box (points A, B, C, E) are projected to the plane:

$$\begin{aligned} &A_w = \wp \cap \overleftrightarrow{OA}, \quad B_w = \wp \cap \overleftrightarrow{OB}, \quad C_w = \wp \cap \overleftrightarrow{OC}, \\ &E_w = p_E \cap \overleftrightarrow{OE}; \ p_E \perp \wp \wedge A_w \in p_E. \end{aligned} \tag{1}$$

When the world coordinates of the bounding box corners are known, it is possible to determine the (somehow scaled) dimensions of the box: $(l, w, h) = (|A_w C_w|, |A_w B_w|, |A_w E_w|)$. Scale factor λ must be found so that the actual metric dimensions are defined as $(\mathbf{l}, \mathbf{w}, \mathbf{h}) = \lambda(l, w, h)$. For this purpose, we collect statistical data about sold cars and their dimensions and form a histogram of their bounding box dimensions. Relative sizes of the cars (l, w, h) are accumulated into a histogram as well. Histograms confirm the intuitive assumption that vehicles have very similar width and height (peaks in histograms are more significant) but they differ in their length. By fitting the statistics of known dimensions and the measured data from the traffic, for each dimension we obtain a scale (Fig. 6). In an ideal case, all these scales are equal. However, because different influences of perspective and rounded corners of the cars (Fig. 4), they are not absolutely the same. For the final scale λ, we choose the smallest of the scales. The motivation here is that the detected bounding boxes tend to be smaller (and therefore the scale λ is greater) because cars are not perfectly boxed and from specific views, some edges of the bounding box did not fit tightly to the car (see Fig. 4).

3 Experimental Evaluation

Our method presented here allows for automatic obtaining camera intrinsic and extrinsic parameters, including the scene scale on the ground plane. This allows for multiple applications, previously unavailable without entering human calibration input. This section evaluates the accuracy relevant to the most straightforward applications: Distance measurements, speed measurements (Sec. 3.1), and analysis of traffic lanes (Sec. 3.2). Section 3.3 shows that the algorithm is capable of running in real time on a low-end processor. Figure 9 shows example images of achievable results.

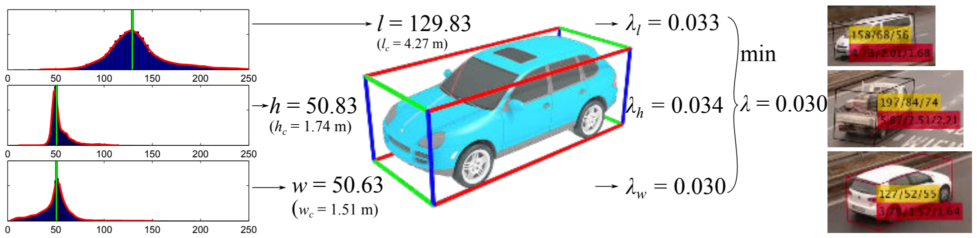


Figure 6: Calculation of scene scale λ. For simplicity, we use only the median of each dimension. *(left)* Median (green bar) for each dimension is found (l, w, h) in the measured data. *(middle)* Scales are derived separately based on known median car size (l_c, w_c, h_c) as $\lambda_l = l_c/l; \lambda_w = w_c/w; \lambda_h = h_c/h$. The final scale is the minimum from these three scales. *(right)* Examples of relative size of the vehicles (yellow) and real dimensions in meters after scaling by factor λ (red).

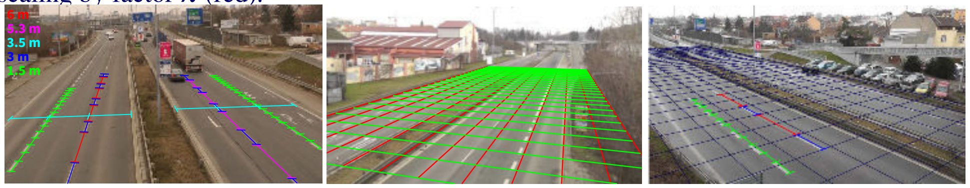


Figure 7: *(left)* Scene with measured ground truth distances used for accuracy evaluation. *(middle)* Grid projected to the road (i.e. ground plane). The size of the squares is $3m \times 3m$. *(right)* Different view of a scene with detected ground plane with $3m \times 3m$ squares and some of the measured ground truth distances.

3.1 Distance & Speed Measurement

When the scene scale λ is known, measurements can be carried out in the image. Figure 7 *(middle)* shows a uniform grid with square $3m \times 3m$ placed over the ground plane. We measured several distances on the road plane, Fig. 7 *(left)*, and evaluated error in distance measurements by our approach. This evaluation is similar to the work of Zhang et al. [34]; however, we evaluate the absolute dimension in meters, while Zhang et al. evaluate relative distances supposed to be equal. They report average error of measurement "less then 10%". Our average error is 1.9% with worst case 5.6%. Table 1 shows results on five videos observing the same scene.

	1.5 m	3 m	3.5 m	5.3 m	6 m	all
v1	2.0/3.3 (29)	2.1/3.9 (7)	4.5/5.5 (3)	3.1/5.6 (5)	2.1/2.4 (3)	2.3 /5.6 (47)
v2	1.6/2.3 (15)	1.3/2.4 (7)	1.3/2.3 (3)	3.3/3.3 (2)	0.7/.17 (3)	1.5/ 3.3 (30)
v3	1.9/3.5 (13)	2.5/3.2 (6)	1.0/1.6 (3)	2.7/3.0 (3)	2.7/3.3 (3)	2.1/ 3.3 (28)
v4	1.0/1.9 (13)	1.8/3.5 (6)	2.3/3.1 (3)	3.7/5.3 (3)	0.9/2.0 (3)	1.6/ 5.3 (28)
v5	2.4/3.6 (15)	1.0/2.5 (6)	0.9/1.7 (3)	1.5/2.5 (3)	1.1/1.7 (3)	1.7/ 3.6 (30)
all	1.8/3.6 (85)	1.7/3.9 (32)	2.0/5.5 (15)	2.8/5.6 (16)	1.5/3.3 (15)	**1.9**/**5.6**(163)

Table 1: Percentage error of absolute distance measurements (5 videos). The error is evaluated as $|l_m - l_{gt}|/l_{gt} * 100\%$, where l_{gt} is ground truth value and l_m is distance measured by presented algorithm. For each video and each distance we evaluate the average and worst error. The number in parentheses stands for the number of measurements of the given length. The bold numbers are average and worst error over all videos and all measurements.

When measuring the vehicle speed (Tab. 2), we take into account corner A of the bounding box, which lies directly on the road (this is an arbitrary choice – any point from the box base can be used). Vehicles in the video are tracked and their velocity is evaluated over the whole straight part of the track. It is also possible to calculate instant speed of the vehicle as the distance vehicle passes between subsequent video frames, but it is not perfectly stable because of inaccuracies in detection of the bounding box and image discretization. It should be noted that once the camera is calibrated including the scale, for computing the average speed of a vehicle, its blob segmentation does not need to be very precise, because even though a part of the vehicle is missing, the speed measurements are still accurate.

	v1 (5)	v2 (3)	v3 (5)	v4 (5)	v5 (4)	v6 (5)	all (23)
mean	2.39	2.90	1.49	1.65	1.31	2.58	**1.99**
worst	3.47	3.63	3.18	3.77	2.40	4.26	**4.26**

Table 2: Percentage error in speed measurement (6 videos). For obtaining the ground truth values, we drove cars with cruise control and get the speed from GPS. The error is evaluated as $|s_m - s_{gt}|/s_{gt} * 100\%$, where s_{gt} is speed from GPS and s_m is speed calculated by presented algorithm. The number in parentheses stands for the number of evaluated measurements.

The average speed of the vehicle was $75\frac{km}{h}$ and therefore 2% error causes $\pm 1.5\frac{km}{h}$ deviation. A similar evaluation was provided by Dailey [7] who used distribution of cars length for scale calculation and reached average deviation $6.4\frac{km}{h}$ or by Grammatikopoulos [13] whose algorithm has accuracy $\pm 3\frac{km}{h}$ but requires manual distance measurements to determine the scale.

3.2 Detection of Traffic Lanes

Having the 3D vehicle bounding boxes, it is also possible to obtain accurate segmentation of traffic lanes, even from views where cars from one lane overlap ones from another. Existing methods accumulate trajectories of the blobs [16], the whole blobs, pixels different to background model [25, 28] or lines on the road [20]. All these methods tend to fail when the camera views the road from side. In our approach, for each vehicle's trajectory we accumulate a filled quad strip with quad vertices $A_i, B_i, A_{i+1}, B_{i+1}$, where i denotes points in i-th video frame. After accumulation, minima are found on the line perpendicular to road direction (i.e. line passing through the 2nd VP) and these are set to be lanes' borders. Accumulation of the above mentioned quad is suitable for finding the borders between the lanes. In some cases, centers of lanes (locations with dominant vehicle movement) are of interest – in that case, only trajectories of a center point in the vehicle base (e.g. $(A_i + B_i)/2$) are accumulated. Figure 8 shows a comparison of different lane segmentation methods with our approach based on projection of “correct” bounding boxes.

3.3 Computational Speed

We created an efficient C++ implementation of the proposed algorithm and evaluated the computational speed on 195 minutes of video. This measurement was done on a computer with an i3-4330 3.50 GHz processor and 8 GB DDR3 RAM. The measured framerates also include reading and decompression of videos (considerable load for full-HD videos). It should be noted that optimal framerate for running the detection/tracking algorithm is around 12.5 FPS, because the cars must move measurably from one frame to the next one. Therefore,

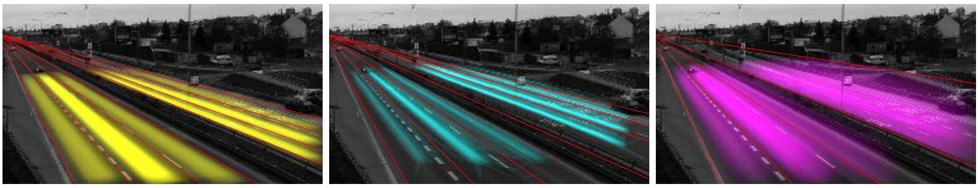


Figure 8: Traffic lane segmentation. *(left)* Our approach based on 3D bounding boxes. Lanes are correctly segmented even for side views. *(middle)* Method using trajectories of the centers of blobs [16]. *(right)* Method based on activity map [28].

resolution	low traffic intensity	high traffic intensity
854×480	116.93 FPS	93.79 FPS
1920×1080	24.98 FPS	19.64 FPS

Table 3: Results of processing speed measurement. High traffic: ~ 40 vehicles per minute; low traffic: ~ 3.5 vehicles per minute. It should be noted that the system uses video streams with ~ 12.5 FPS; and therefore, it can run safely in real time even for full-HD video with the high traffic intensity.

"real-time processing" in this case means running faster than 12.5 FPS. The results in Tab. 3 show that the system can work in real time with a safe margin.

4 Conclusions and Future Work

We presented a method for understanding traffic scenes observed by stable roadside cameras. The calibration is done by first computing three orthogonal vanishing points and thus calibrating the camera. Then, we propose to extract 3D bounding boxes of passing vehicles by first constructing their 2D projections. We propose methodology for using the statistics of these 3D bounding boxes, so that scene scale can be automatically determined.

Our method is fully automatic – no user input is required during the whole process. Experimental results show that the mean error of speed and distance measurement is below 2 % (worst 5.6 % for distance and 4.3 % for speed). This outperforms existing approaches and provides sufficient accuracy for statistical traffic analysis. Besides measurement, our approach can facilitate other traffic analysis task, as shown on the case of traffic lane segmentation. The algorithm works in real time with a safe margin. Our measurements show that the system is able to process 93 FPS of normal video input. The extracted bounding boxes can be used for various traffic analyses – on the example of traffic lane segmentation we are showing its benefits for traffic scene understanding.

We are exploring ways how to use the bounding boxes for facilitating various computer vision tasks. Their knowledge can improve and/or speed up scanning window-based detection and recognition algorithms. Our bounding boxes can serve as a starting point for fitting of detailed 3D models to the visual data [23]. We are also working on a multi-camera system resistant to mutual occlusions of vehicles – the bounding boxes constructed by multiple cameras can be cheaply fused into one stream of results.

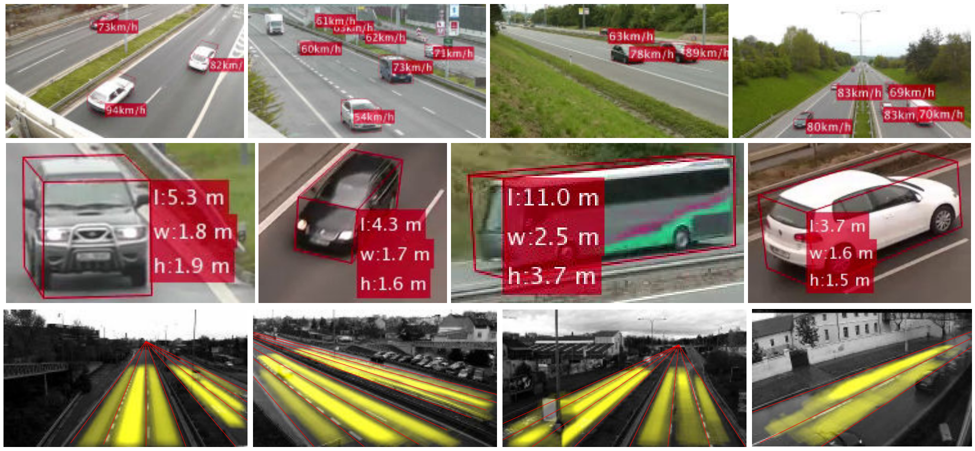


Figure 9: Examples of achieved results (see supplementary video for further illustration). (1st *row*) Different scenes with measured vehicle speed. (2nd *row*) Cropped out vehicles with estimated dimensions. (3rd *row*) Road lanes detected using 3D bounding boxes.

References

[1] authors. Fully automatic roadside camera calibration for traffic surveillance. *Submitted to IEEE Transactions on Itelligent Transportation Systems*.

[2] 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, CVPR*, 1997. doi: 10.1109/CVPR.1997.609371.

[3] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *International Journal of Computer Vision*, 4(2):127–139, 1990.

[4] F.W. Cathey and D.J. Dailey. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *Intelligent Vehicles Symposium*, pages 777–782, 2005. doi: 10.1109/IVS.2005.1505199.

[5] F.W. Cathey and D.J. Dailey. Mathematical theory of image straightening with applications to camera calibration. In *Intelligent Transportation Systems Conference*, 2006. doi: 10.1109/ITSC.2006.1707413.

[6] Roberto Cipolla, Tom Drummond, and Duncan P Robertson. Camera calibration from vanishing points in image of architectural scenes. In *British Machine Vision Conference, BMVC*, 1999.

[7] D.J. Dailey, F.W. Cathey, and S. Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):98–107, 2000. ISSN 1524-9050. doi: 10.1109/6979.880967.

[8] Bart De Schutter and Bart De Moor. Optimal traffic light control for a single intersection. *European Journal of Control*, 4(3):260–276, 1998.

[9] J. Deutscher, M. Isard, and J. MacCormick. Automatic camera calibration from a single manhattan image. In *European Conference on Computer Vision, ECCV*, pages 175–188. 2002. ISBN 978-3-540-43748-2. doi: 10.1007/3-540-47979-1-12. URL http://dx.doi.org/10.1007/3-540-47979-1-12.

[10] Rong Dong, Bo Li, and Qi-mei Chen. An automatic calibration method for PTZ camera in expressway monitoring system. In *World Congress on Computer Science and Information Engineering*, pages 636–640, 2009. ISBN 978-0-7695-3507-4. doi: 10.1109/CSIE.2009.763. URL http://dx.doi.org/10.1109/CSIE.2009.763.

[11] Markéta Dubská and Adam Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *British Machine Vision Conference, BMVC*, 2013.

[12] George S. K. Fung, Nelson H. C. Yung, and Grantham K. H. Pang. Camera calibration from road lane markings. *Optical Engineering*, 42(10):2967–2977, 2003. doi: 10.1117/1.1606458. URL http://dx.doi.org/10.1117/1.1606458.

[13] Lazaros Grammatikopoulos, George Karras, and Elli Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pages 332–338, 2005.

[14] Xiao Chen He and N. H C Yung. A novel algorithm for estimating vehicle speed from two consecutive images. In *IEEE Workshop on Applications of Computer Vision, WACV*, 2007. doi: 10.1109/WACV.2007.7.

[15] T. Horprasert, D. Harwood, and L. S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *Proc. IEEE ICCV*, volume 99, pages 1–19, 1999.

[16] Jun-Wei Hsieh, Shih-Hao Yu, Yung-Sheng Chen, and Wen-Fong Hu. Automatic traffic surveillance system for vehicle tracking and classification. *Intelligent Transportation Systems, IEEE Transactions on*, 7(2):175–187, 2006.

[17] Shunsuke Kamijo, Yasuyuki Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *Intelligent Transportation Systems, IEEE Transactions on*, 1(2):108–118, 2000.

[18] N. K. Kanhere and S. T. Birchfield. Real-time incremental segmentation and tracking of vehicles at low camera angles using stable features. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):148–160, 2008. ISSN 1524-9050. doi: 10.1109/TITS.2007.911357. URL http://dx.doi.org/10.1109/TITS.2007.911357.

[19] Neeraj K Kanhere, Stanley T Birchfield, and Wayne A Sarasua. Automatic camera calibration using pattern detection for vision-based speed sensing. *Journal of the Transportation Research Board*, 2086(1):30–39, 2008.

[20] Andrew HS Lai and Nelson HC Yung. Lane detection by orientation and length discrimination. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(4):539–548, 2000.

[21] Stefan Lämmer and Dirk Helbing. Self-control of traffic lights and vehicle flows in urban road networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008 (04), 2008. doi: 10.1088/1742-5468/2008/04/P04019.

[22] J de Ortuzar and Luis G Willumsen. *Modelling transport*. 2011.

[23] A. Ottlik and H.-H. Nagel. Initialization of model-based vehicle tracking in video sequences of inner-city intersections. *International Journal of Computer Vision*, 80(2): 211–225, 2008. ISSN 0920-5691. doi: 10.1007/s11263-007-0112-6. URL http://dx.doi.org/10.1007/s11263-007-0112-6.

[24] Tun-Wen Pai, Wen-Jung Juang, and Lee-Jyi Wang. An adaptive windowing prediction algorithm for vehicle speed estimation. In *IEEE Intelligent Transportation Systems*, 2001. doi: 10.1109/ITSC.2001.948780.

[25] T.N. Schoepflin and D.J. Dailey. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Transactions on Intelligent Transportation Systems*, 4(2):90–98, 2003. ISSN 1524-9050. doi: 10.1109/TITS.2003.821213.

[26] Kai-Tai Song and Jen-Chao Tai. Dynamic calibration of Pan–Tilt–Zoom cameras for traffic monitoring. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(5):1091–1103, 2006. ISSN 1083-4419. doi: 10.1109/TSMCB.2006.872271. URL http://dx.doi.org/10.1109/TSMCB.2006.872271.

[27] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, volume 2, pages 246–252, 1999.

[28] BD Stewart, I Reading, MS Thomson, TD Binnie, KW Dickinson, and CL Wan. Adaptive lane finding in road traffic image analysis. 1994.

[29] Tuan Hue Thi, Sijun Lu, and Jian Zhang. Self-calibration of traffic surveillance camera using motion tracking. In *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008.

[30] David Vallejo, Javier Albusac, Luis Jimenez, Carlos Gonzalez, and Juan Moreno. A cognitive surveillance system for detecting incorrect traffic behaviors. *Expert Systems with Applications*, 36(7):10503–10511, 2009.

[31] J. M. Wang, Y. C. Chung, C. L. Chang, and S.W. Chen. Shadow detection and removal for traffic images. In *Networking, Sensing and Control, 2004 IEEE International Conference on*, volume 1, pages 649–654 Vol.1, March 2004. doi: 10.1109/ICNSC.2004.1297516.

[32] Kunfeng Wang, Hua Huang, Yuantao Li, and Fei-Yue Wang. Research on lane-marking line based camera calibration. In *International Conference on Vehicular Electronics and Safety, ICVES*, 2007. doi: 10.1109/ICVES.2007.4456361.

[33] Mei Xiao, Chong-Zhao Han, and Lei Zhang. Moving shadow detection and removal for traffic sequences. *International Journal of Automation and Computing*, 4(1):38–46, 2007. ISSN 1476-8186. doi: 10.1007/s11633-007-0038-z. URL http://dx.doi.org/10.1007/s11633-007-0038-z.

[34] Zhaoxiang Zhang, Tieniu Tan, Kaiqi Huang, and Yunhong Wang. Practical camera calibration from moving objects for traffic scene surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(3):518–533, 2013. ISSN 1051-8215. doi: 10.1109/TCSVT.2012.2210670.

[35] Yuan Zheng and Silong Peng. A practical roadside camera calibration method based on least squares optimization. *Intelligent Transportation Systems, IEEE Transactions on*, 15(2):831–843, April 2014. ISSN 1524-9050. doi: 10.1109/TITS.2013.2288353.

[36] Zhigang Zhu, Bo Yang, Guangyou Xu, and Dingji Shi. A real-time vision system for automatic traffic monitoring based on 2D spatio-temporal images. In *Proceedings of WACV*, 1996.

[37] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31 Vol.2, 2004. doi: 10.1109/ICPR.2004.1333992.