AUTOMATIC TRAFFIC VIDEO SURVEILLANCE: FINE-GRAINED RECOGNITION OF VEHICLES AND AUTOMATIC SPEED MEASUREMENT

AUTOMATICKÁ ANALÝZA DOPRAVY Z VIDEA:
ROZPOZNÁNÍ TYPŮ VOZIDEL A AUTOMATICKÉ MĚŘENÍ RYCHLOSTI

PH.D. THESIS
DISERTAČNÍ PRÁCE

AUTHOR ING. JAKUB SOCHOR
AUTOR PRÁCE

SUPERVISOR PROF. ADAM HEROUT, PH.D.
VEDOUCÍ PRÁCE

BRNO 2018

CONTENTS

1 INTRODUCTION

United Nations Economic Commission for Europe in document *Intelligent Transport Systems (ITS) for sustainable mobility* [6, page 18] claims that:

> *Intelligent Transport Systems play an important role in shaping the future ways of mobility and the transport sector. We expect that through the use of ITS applications, transport will become more efficient, safer and greener. The huge potentials and benefits, however, can only be reaped if ITS solutions are put in place – internationally harmonized as much as possible.*

Also, in my opinion, it will be possible to use Intelligent Transport Systems for tasks which will increase comfort and safety of drivers and pedestrians. For example, it will be possible to navigate vehicles and control lights in a way that will improve permeability of the traffic network. Another improvement for the drivers could be automatic warning about collisions on the road ahead of the drivers. Or in an ideal case, it would be possible to predict precisely where vehicles are heading and prevent congestion before it will even happen. Another task where the traffic surveillance system can be beneficial is estimation of demographic statistics. There is for example a recent paper by Gebru et al. [9] tackling the demographic data acquisition using fine-grained recognition of vehicles.

However, for all these tasks and Intelligent Transportation Systems in general, it is necessary to have a high amount of statistical data about the traffic flow on roads. Also, in order to be the system deployable on a large scale, it is useful that the system is cheap and it does not require any sensor settings on a per-sensor basis.

Such cheap sensor can be a camera. Therefore, my focus in this Ph.D. thesis is on enabling acquisition of complex statistical data from traffic surveillance cameras in a **fully automatic** manner. In particular, I address the problems of **automatic speed measurement** of vehicles from camera and **fine-grained recognition of vehicles**. Methods for acquisition of other traffic flow statistics are addressed in paper [Soc14], which is based on my Master's thesis.

1.1 PROBLEM DEFINITION

The primary goal of this Ph.D. thesis is to push the state of the art in two areas of research: fine-grained recognition of vehicles and automatic speed measurement of vehicles. Both these algorithms are focused mainly on traffic surveillance cameras.

The **fine-grained recognition of vehicles** is a task where the method is expected to determine the exact model of a given vehicle in an image. The differentiation should be done up to model years of the vehicles as they may differ in vehicle geometry. The goal regarding the fine-grained recognition of vehicles in this thesis is to develop a method which will be able to recognize vehicles on images taken by a surveillance

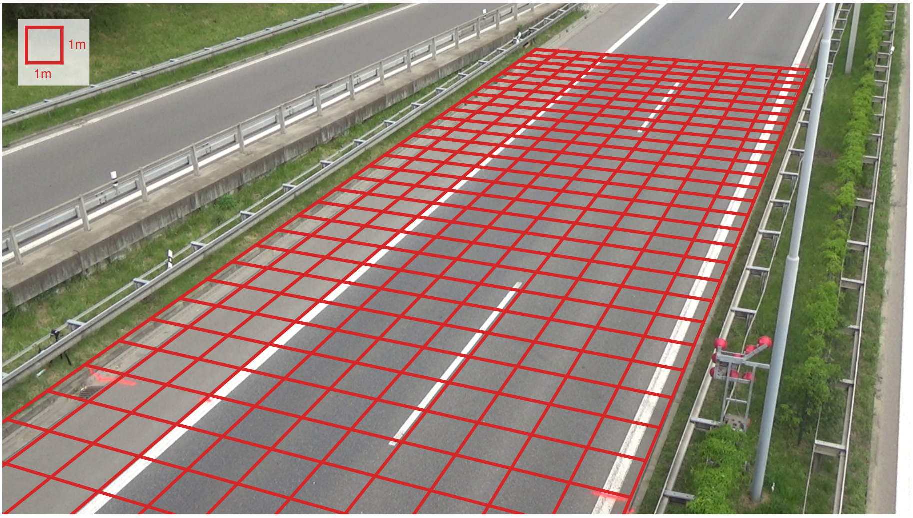

Figure 1.1: An example of calibration for speed measurement obtained by a fully automatic algorithm proposed in the thesis. The calibration is represented by an orthogonal regular grid with 1 m sides.

camera. The requirement of applicability with surveillance cameras has several implications. First, it is necessary to handle low resolution images with significant video compression. Also, the method should be able to recognize images of vehicles taken from an **arbitrary viewpoint**.

The other area of addressed research is **automatic speed measurement** from a single monocular surveillance camera. For the speed measurement, it is necessary to be able to measure time and distances on the road. The time measurement in video sequences with known framerate is relatively direct. However, the measurement of real world distances on the road plane is more challenging, considering the fact that it should be done in a **fully automatic** manner. For the distance measurement, it is necessary to calibrate the camera (i.e. estimate **intrinsic** and **extrinsic** camera parameters) and also estimate the **scale** of the scene (or distance from camera to the road plane). With all these information available, it possible to measure distances on the road plane. See Figure 1.1 for an example of the full (including scale) calibration.

1.2 CORE CONTRIBUTIONS

My contributions to fine-grained recognition of vehicles include improving classification accuracy of Convolutional Neural Networks [16] using automatically constructed 3D bounding boxes constructed around vehicles [DSH14] using traffic surveillance data. The results show that the proposed method consistently improves classification accuracy **by up to 12 percentage points** with different CNNs [15, 25, 11, 8]. The classification error was also **reduced by up to 50 %**. The contributions were presented in the following papers:

- *BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition* – **CVPR[1], 2016** [SHH16]. The first paper dealing with the fine-grained recognition using the 3D bounding boxes. The method was applied both on fine-grained classification and verification with consistent improvement in both tasks.

- *BoxCars: Improving Vehicle Fine-Grained Recognition using 3D Bounding Boxes in Traffic Surveillance* – **IEEE T-ITS[2], 2018** [SŠH18]. Extended journal version of the previous paper. The classification results were further improved and complex and in-depth analysis of the method is presented. Also, we propose a method for 3D bounding box estimation in situations where it is not possible to construct the precise 3D bounding box from the surveillance data.

My contributions to the speed measurement are based on the proposed algorithm for precise traffic surveillance camera calibration. The experimental results show that our method achieves **1.10 km/h** speed measurement mean error while outperforming both state-of-the-art method and manual calibration in the speed measurement task. The contributions are described in these papers:

- *Comprehensive Dataset for Automatic Single Camera Visual Speed Measurement* – **IEEE T-ITS[3], 2018, under review** [SJŠ$^+$18]. Survey and dataset paper for speed measurement. The dataset BrnoCompSpeed is by far the largest dataset for speed measurement from video with precise ground truth. The dataset contains more than 18 hours of videos from various viewpoints and varying traffic intensity. The dataset also contains more than 20,000 of vehicles with precise ground truth speed. The paper is currently under review with last status "Accept as Regular Paper after Minor Revision" while the reviewers requested only very subtle changes in the text of the paper.

- *Traffic Surveillance Camera Calibration by 3D Model Bounding Box Alignment for Accurate Vehicle Speed Measurement* – **CVIU[4], 2017** [SJH17]. Paper with proposed method for traffic camera calibration for speed measurement. The method is based on vanishing point detection and alignment of 3D models of several common vehicle types to estimate the scene scale. The method was evaluated on the BrnoCompSpeed dataset and the final mean speed measurement error is 1.10 km/h.

All these papers present my contribution to the state of the art in Intelligent Transportation Systems and Computer Vision in two important areas (automatic speed measurement and fine-grained recognition of vehicles). Furthermore, results or our ongoing research showed that the 3D bounding boxes improve performance even for vehicle re-identification task [SŠJH18].

[1] IEEE Conference on Computer Vision and Pattern Recognition
[2] IEEE Transactions on Intelligent Transportation Systems – IF: 3.724
[3] IEEE Transactions on Intelligent Transportation Systems – IF: 3.724
[4] Computer Vision and Image Understanding – IF: 2.498

2 BOXCARS: IMPROVING FINE-GRAINED RECOGNITION OF VEHICLES USING 3D BOUNDING BOXES IN TRAFFIC SURVEILLANCE

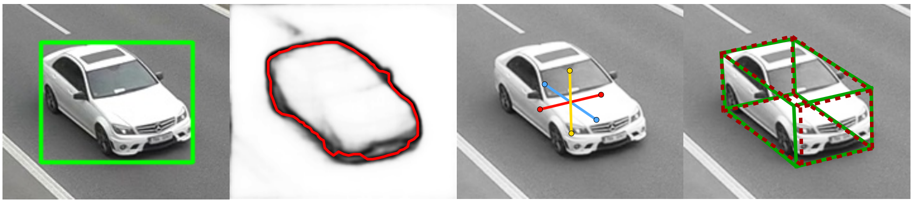

Figure 2.1: Example of automatically obtained 3D bounding box used for fine-grained vehicle classification. **left:** vehicle with 2D bounding box annotation, **center left:** estimated contour, **center right:** estimated directions to vanishing points, **right:** 3D bounding box automatically obtained from surveillance video (green) and our estimated 3D bounding box (red).

2.1 INTRODUCTION

Fine-grained recognition of vehicles is interesting, both from the application point of view (surveillance, data retrieval, etc.) and from the point of view of general fine-grained recognition research applicable in other fields. For example, Gebru et al. [9] proposed an estimation of demographic statistics based on fine-grained recognition of vehicles. In this article, we are presenting methodology which considerably increases the performance of multiple state-of-the-art CNN architectures in the task of fine-grained vehicle recognition. We target the traffic surveillance context, namely images of vehicles taken from an **arbitrary viewpoint** – we do not limit ourselves to frontal/rear viewpoints. As the images are obtained from surveillance cameras, they have challenging properties – they are often small and taken from very general viewpoints (high elevation). We also construct the training and testing sets from images from different cameras as it is common for surveillance applications that it is not known a priori under which viewpoint the camera will be observing the road.

We propose an orthogonal approach to these methods and use CNNs with a modified input to achieve better image normalization and data augmentation (therefore, our approach can be combined with other methods). We use 3D bounding boxes around vehicles to normalize vehicle image (see Figure 2.3 for examples). This work is based on our previous conference paper [SHH16]; it pushes the performance further and we mainly propose a new method on how to build the 3D bounding box without any prior knowledge (see Figure 2.1). Our input modifications are able to significantly

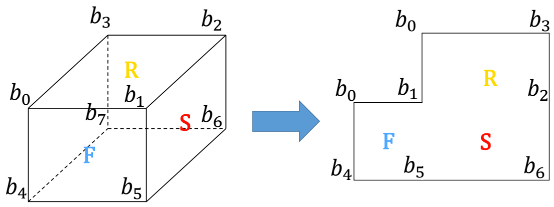

Figure 2.2: 3D bounding box and its unpacked version.

increase the classification accuracy (up to **12 percentage points**, classification error is reduced by up to **50 %**).

2.2 PROPOSED METHODOLOGY FOR FINE-GRAINED RECOGNITION OF VEHICLES

In agreement with recent progress in the Convolutional Neural Networks [28, 15, 2], we use CNN for both classification and verification (determining whether a pair of vehicles has the same type). However, we propose to use several data normalization and augmentation techniques to significantly boost the classification performance (up to 50 % error reduction compared to base net). We utilize information about 3D bounding boxes obtained from traffic surveillance camera [DSH14]. Finally, in order to increase the applicability of our method to scenarios where the 3D bounding box is not known, we propose an algorithm for bounding box estimation both at training and test time.

2.2.1 *Image Normalization by Unpacking the 3D Bounding Box*

We based our work on 3D bounding boxes proposed by [DSH14] (Fig. 2.3) which can be automatically obtained for each vehicle seen by a surveillance camera. These boxes allow us to identify the side, roof, and front (or rear) side of vehicles in addition to other information about the vehicles. We use these localized segments to normalize the image of the observed vehicles (considerably boosting the recognition performance).

The normalization is done by unpacking the image into a plane. The plane contains rectified versions of the front/rear (**F**), side (**S**), and roof (**R**). These parts are adjacent to each other (Fig. 2.2) and they are organized into the final matrix **U**:

$$\mathbf{U} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{F} & \mathbf{S} \end{pmatrix} \tag{2.1}$$

The unpacking itself is done by obtaining homography between points b_i (Fig. 2.2) and perspective warping parts of the original image. The left top submatrix is filled with zeros. This unpacked version of the vehicle is used instead of the original image to feed the net. The unpacking is beneficial as it localizes parts of the vehicles, normalizes their position in the image and it does all that without the necessity of using

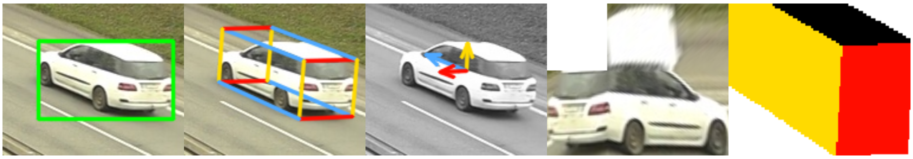

Figure 2.3: Examples of data normalization and auxiliary data fed to nets. **Left to right:** vehicle with 2D bounding box, computed 3D bounding box, vectors encoding viewpoints on the vehicle (**View**), unpacked image of the vehicle (**Unpack**), and rasterized 3D bounding box fed to the net (**Rast**).

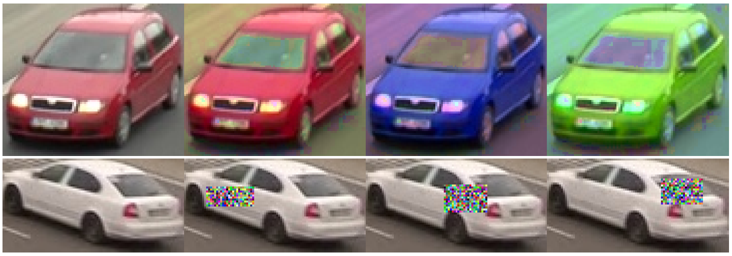

Figure 2.4: Examples of proposed data augmentation techniques. Left most image contains the original cropped image of the vehicle and other images contains augmented versions of the image (**Top – Color**, **Bottom – ImageDrop**).

DPM or other algorithms for part localization. Later in the text, we will refer to this normalization method as **Unpack**.

2.2.2 *Extended Input to the Neural Nets*

It it possible to infer additional information about the vehicle from the 3D bounding box and we found out that these data slightly improve the classification and verification performance. One piece of this auxiliary information is the encoded viewpoint (direction from which the vehicle is observed). We also add a rasterized 3D bounding box as an additional input to the CNNs. Compared to our previously proposed auxiliary data fed to the net [SHH16], we handle frontal and rear vehicle sides differently.

View. The viewpoint is extracted from the orientation of the 3D bounding box – Fig. 2.3. We encode the viewpoint as three 2D vectors v_i, where $i \in \{f, s, r\}$ (*front/rear*, *side*, *roof*) and pass them to the net. Vectors v_i are connecting the center of the bounding box with the centers of the box's faces. Therefore, it can be computed as $v_i = \overrightarrow{C_c C_i}$. Point C_c is the center of the bounding box and it can be obtained as the intersection of diagonals $\overleftrightarrow{b_2 b_4}$ and $\overleftrightarrow{b_5 b_3}$. Points C_i for $i \in \{f, s, r\}$ denote the centers of each face, again computed as intersections of face diagonals. In contrast to our previous approach [SHH16], which did not take the direction of the vehicle into account; instead, we encode the information about the vehicle direction ($d = 1$ for vehicles going to camera, $d = 0$ for vehicles going from the camera), in order to determine which side of the bounding box is the frontal one. The vectors are normalized to have a unit size; storing them with a different normalization (e.g. the front one normalized, the other in the proper ratio) did not improve the results.

Rast. Another way of encoding the viewpoint and also the relative dimensions of vehicles is to rasterize the 3D bounding box and use it as an additional input to the net. The rasterization is done separately for all sides, each filled by one color. The final rasterized bounding box is then a four-channel image containing each visible face rasterized in a different channel. Formally, point p of the rasterized bounding box **T** is obtained as

$$\mathbf{T}_p = \begin{cases} (1,0,0,0) & p \in \varoslash b_0 b_1 b_4 b_5 \text{ and } d = 1 \\ (0,1,0,0) & p \in \varoslash b_0 b_1 b_4 b_5 \text{ and } d = 0 \\ (0,0,1,0) & p \in \varoslash b_1 b_2 b_5 b_6 \\ (0,0,0,1) & p \in \varoslash b_0 b_1 b_2 b_3 \\ (0,0,0,0) & \text{otherwise} \end{cases} \tag{2.2}$$

where $\varoslash b_0 b_1 b_4 b_5$ denotes the quadrilateral defined by points b_0, b_1, b_4 and b_5 in Figure 2.2.

Finally, the 3D rasterized bounding box is cropped by the 2D bounding box of the vehicle. For an example, see Figure 2.3, showing rasterized bounding boxes for different vehicles taken from different viewpoints.

2.2.3 *Additional Training Data Augmentation*

In order to increase the diversity of the training data, we propose additional data augmentation techniques. The first one (denoted as **Color**) deals with the fact that for fine-grained recognition of vehicles (and some other objects), their color is irrelevant. The other method (**ImageDrop**) deals with some potentially missing parts of the vehicle. Examples of the data augmentation are shown in Figure 2.4. Both these augmentation techniques are done only with predefined probability during training, otherwise they are not modified. During testing, we do not modify the images at all.

Color. In order to increase training samples color variability, we propose to randomly alternate the color of the image. The alternation is done in the HSV color space by adding the same random values to each pixel in the image (each HSV channel is processed separately).

ImageDrop. Inspired by Zeiler et al. [32], who evaluated the influence of covering a part of the input image on the probability of the ground truth class, we take this a step further and in order to deal with missing parts on the vehicles, we take a random rectangle in the image and fill it with random noise, effectively dropping any information contained in that part of the image.

2.2.4 *Estimation of 3D Bounding Box from a Single Image*

As the results (Section 2.4) show, the most important part of the proposed algorithm is **Unpack** followed by **Color** and **ImageDrop**. However, the 3D bounding box is required for unpacking the vehicles and we acknowledge that there may be scenarios

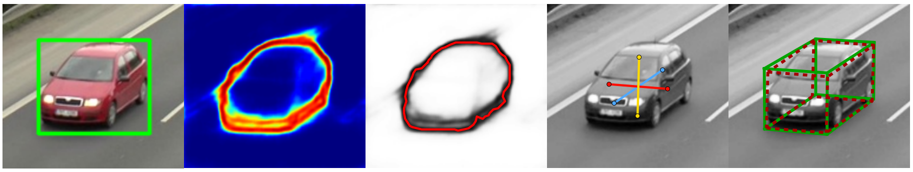

Figure 2.5: Estimation of 3D bounding box. **Left to right:** image with vehicle 2D bounding box, output of contour object detector [30], our constructed contour, estimated directions towards vanishing points, ground truth (**green**) and estimated (**red**) 3D bounding box.

when such information is not available. For these cases, we propose a method on how to estimate the 3D bounding box for both training and test time when only limited information is available.

As proposed by [DSH14], the vehicle's contour and vanishing points are required for the bounding box construction. Therefore, it is necessary to estimate the contour and vanishing points for the vehicle. For estimating the vehicle contour, we use Fully Convolutional Encoder-Decoder network designed by Yang et al. [30] for general object contour detection and masks with probabilities of vehicles contours for each image pixel. To obtain the final contour, we search for global maxima along line segments from 2D bounding box centers to edge points of the 2D bounding box (see Figure 2.5 for examples).

We found out that the exact position of the vanishing point is not required for 3D bounding box construction, but the directions to the vanishing points are much more important. Therefore, we use regression to obtain the directions towards the vanishing points and then assume that the vanishing points are in infinity.

Following the work by Rothe et al. [21], we formulated the regression of the direction towards the vanishing points as a classification task into bins corresponding to angles and we used ResNet50 [11] with three classification outputs. We found this approach more robust than a direct regression. We added three separate fully connected layers with softmax activation (one for each vanishing point) after the last average pooling in the ResNet50. Each of these layers generates probabilities for each vanishing point belonging to the specific direction bin (represented as angles). We quantized the angle space by bins of $3°$ from $-90°$ to $90°$ (60 bins per vanishing point in total).

As the training data for the regression we used BoxCars116k dataset (Section 2.3) with the test samples omitted. The direction to vanishing points were obtained by method [DSH14, DHJS15]; however, the quality of the ground truth bounding boxes was manually verified during annotation of the dataset and imprecise samples were removed by the annotators. To construct the lines on which the vanishing points are, we use the center of the 2D bounding box. Even though there is bias in the direction of the training data (some bins have very low number of samples), it is highly unlikely that for example, the first vanishing point direction will be close to horizontal.

With all this estimated information it is then possible to construct the 3D bounding box in both training and test time. It is important to note that by using this 3D bounding box estimation, it is possible to use this method outside the scope of traffic surveillance. It is only necessary to train the regressor of vanishing points directions.

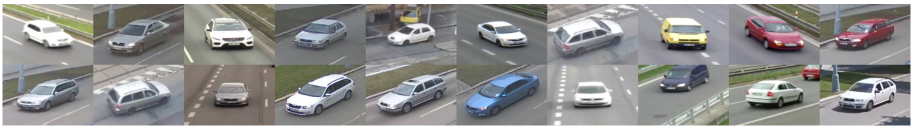

Figure 2.6: Collate of random samples from the BoxCars116k dataset.

For the training of such a regressor, it is possible to use either the directions themselves or viewpoints on the vehicle and focal lengths of the images.

Using this estimated bounding box, it is possible to unpack the vehicle image in test time without any additional information required. This enables the usage of the method when the traffic surveillance data are not available. The results in Section 2.4.3 show that by using this estimated 3D bounding boxes, our method still significantly outperforms other convolutional neural networks without input modification.

2.3 BOXCARS116K DATASET

We collected and annotated a new dataset *BoxCars116k*. The dataset is focused on images taken from surveillance cameras as it is meant to be useful for traffic surveillance applications. We do not restrict that the vehicles are taken from the frontal side (Fig. 2.6). We used surveillance cameras mounted near streets and tracked passing vehicles. The cameras were placed on various locations around Brno, Czech Republic and recorded the passing traffic from an arbitrary (reasonable) surveillance viewpoint. Each correctly detected vehicle (by Faster-RCNN [20] trained on COD20k dataset [13]) is captured in multiple images, as it passes by the camera; therefore, we have more visual information about each vehicle.

2.3.1 *Dataset Acquisition*

The dataset is formed by two parts. The first part consists of data from *BoxCars21k* dataset [SHH16] which were cleaned up and some imprecise annotations were then corrected (e.g. missing model years for some uncommon vehicle types).

We also collected other data from videos relevant to our previous work [DSH14, DHJS15, SJŠ+18]. We detected all vehicles, tracked them and for each track collected images of the respective vehicle. We downsampled the framerate to ~ 12.5 FPS to avoid collecting multiple and almost identical images of the same vehicle.

The new dataset was annotated by multiple human annotators with an interest in vehicles and sufficient knowledge about vehicle types and models. The annotators were assigned to clean up the processed data from invalid detections and assign exact vehicle type (make, model, submodel, year) for each obtained track. While preparing the dataset for annotation, 3D bounding boxes were constructed for each detected vehicle using the method proposed by [DSH14]. Invalid detections were then distinguished by the annotators based on these constructed 3D bounding boxes. In the cases when all 3D bounding boxes were not constructed precisely, the whole track was invalidated.

Vehicle type annotation reliability is guaranteed by providing multiple annotations for each valid track (~ 4 annotations per vehicle). The annotation of a vehicle type is considered as correct in the case of at least three identical annotations. Uncertain cases were authoritatively annotated by the authors.

The tracks in *BoxCars21k* dataset consist of exactly 3 images per track. In the new part of the dataset, we collect an arbitrary number of images per track (usually more than 3).

2.3.2 *Dataset Statistics*

The dataset contains 27 496 vehicles (116 286 images) of 45 different makes with 693 fine-grained classes (make & model & submodel & model year) collected from 137 different cameras with a large variation of viewpoints. The dataset also includes information about the 3D bounding box [DSH14] for each vehicle and an image with a foreground mask extracted by background subtraction [27, 36]. The dataset has been made publicly available[1] for future reference and evaluation.

Compared to "web-based" datasets, the new *BoxCars116k* dataset contains images of vehicles relevant to traffic surveillance which have specific viewpoints (high elevation), usually small images, etc. Compared to other fine-grained surveillance datasets, our dataset provides data with a high variation of viewpoints.

2.3.3 *Training & Test Splits*

Our task is to provide a dataset for fine-grained recognition in traffic surveillance without any viewpoint constraint. Therefore, we have constructed the splits for training and evaluation in a way which reflects the fact that it is not usually known beforehand from which viewpoints the vehicles will be seen by the surveillance camera.

Thus, for the construction of the splits, we randomly selected cameras and used all tracks from these cameras for training and vehicles from the rest of the cameras for testing. In this way, we are testing the classification algorithms on images of vehicles from previously unseen cameras (viewpoints). This splits selection process implies that some of the vehicles from the test set may be taken under slightly different viewpoints from the ones that are in the training set.

We constructed two splits. In the first one (**hard**), we are interested in recognizing the precise type, including the model year. In the other one (**medium**), we omit the difference in model years and all vehicles of the same subtype (and potentially different model years) are present in the same class. We selected only types which have at least 15 tracks in the training set and at least one track in the testing set. The hard split contains 107 fine-grained classes with 11 653 tracks (51 691 images) for training and 11 125 tracks (39 149 images) for testing.

[1] https://medusa.fit.vutbr.cz/traffic

2.4 EXPERIMENTS

We thoroughly evaluated our proposed algorithm on the BoxCars116k dataset. First, we evaluated how these methods improved classification accuracy with different nets, compared them to the state of the art, and analyzed how using approximate 3D bounding boxes influence the achieved accuracy. Then, we searched for the main source of improvements, analyzed improvements of different modifications separately, and also evaluated the usability of features from the trained nets for the task of vehicle type identity verification.

In order to show that our modifications improve the accuracy independently on the used nets, we use several of them:

- **AlexNet** [15]
- **VGG16**, **VGG19** [25]
- **ResNet50**, **ResNet101**, **ResNet152** [11]
- CNNs with Compact Bilinear Pooling layer [8] in combination with VGG nets denoted as **VGG16+CBL** and **VGG19+CBL**.

As there are several options how to use the proposed modifications of input data and add additional auxiliary data, we define several labels which we will use:

- **ALL** – All five proposed modifications (Unpack, Color, ImageDrop, View, Rast).
- **IMAGE** – Modifications working only on the image level (Unpack, Color, ImageDrop).
- **CVPR16** – Modifications as proposed in our previous CVPR paper [SHH16] (Unpack, View, Rast – however, the View and Rast modifications differ from those ones used in this paper as the original modifications do not distinguish between the frontal and rear side of vehicles).

2.4.1 *Improvements for Different CNNs*

The first experiment which was done was evaluation how our modifications have improved classification accuracy for different CNNs.

All the nets were fine-tuned from models pre-trained on ImageNet [22] for approximately 15 epochs which was sufficient for the nets to converge. We used the same batch size (except for ResNet151, where we had to use a smaller batch size because of GPU memory limitations), the same initial learning rate and learning rate decay and the same hyperparameters for every net (initial learning rate $2.5 \cdot 10^{-3}$, weight decay $5 \cdot 10^{-4}$, quadratic learning rate decay, loss is averaged over 100 iterations). We also used standard data augmentation techniques as a horizontal flip and randomly moving bounding box [25]. As ResNets do not use fully connected layers, we only use **IMAGE** modifications for them.

For each net and modification we evaluate the accuracy improvement of the modification in percentage points and also evaluate the classification error reduction.

Table 2.1: Summary statistics of improvements by our proposed modifications for different CNNs. The improvements over baseline CNNs are reported as single sample accuracy/track accuracy in percentage points. We also present classification error reduction in the same format.

	modif.	**improvement [pp]**		**error reduction [%]**	
		mean	best	mean	best
medium	ALL	7.49/6.29	11.84/10.99	26.83/34.50	36.71/50.32
	IMAGE	7.19/6.15	12.09/11.63	27.38/36.21	35.23/49.55
	CVPR16	2.99/3.18	5.22/5.65	10.86/17.71	19.76/32.25
hard	ALL	7.00/5.83	11.14/10.85	25.59/33.52	33.40/48.76
	IMAGE	6.74/5.81	11.02/10.53	26.12/35.95	33.04/47.33
	CVPR16	2.12/2.44	3.56/3.92	7.93/14.57	12.68/24.10

The summary results for both medium and hard splits are shown in Table 2.1. As we have correspondences between the samples in the dataset and know which samples are from the same track, we are able to use mean probability across track samples and merge the classification for the whole track. Therefore, we always report the results in the form of *single sample accuracy/whole track accuracy*. As expected, the results for whole tracks are much better than for single samples. For the traffic surveillance scenario, we consider to be more important the whole track accuracy as it is rather common to have a full track of observations of the same vehicle.

There are several things which should be noted about the results. The most important one is that our modifications significantly improve classification accuracy (up to **+12 percentage points**) and reduce classification error (up to **50 % error reduction**). Another important fact is that our new modifications push the accuracy much further compared to the original method [SHH16].

The table also shows that the difference between **ALL** modifications and **IMAGE** modifications is negligible and therefore it is reasonable to only use the **IMAGE** modifications. This also results in CNNs which just use the **Unpack** modification during test time as the other image modifications (Color, ImageDrop) are used only during fine-tuning of CNNs.

Moreover, the evaluation shows that the results are almost identical for the hard and medium split; therefore, we will only report additional results on the hard split, as it is the main goal to distinguish also the model years. The names for the splits were chosen to be consistent with the original version of dataset [SHH16] and the small difference between medium and hard split accuracies is caused mainly by the size of the new dataset.

2.4.2 *Comparison with the State of the Art*

In order to examine the performance of our method, we also evaluated other state-of-the-art methods for fine-grained recognition. We used three different algorithms for general fine-grained recognition with a published code. We always first used the code

Table 2.2: Comparison of different vehicle fine-grained recognition methods. Accuracy is reported as single image accuracy/whole track accuracy. Processing speed was measured on a machine with GTX1080 and CUDNN. * FPS reported by authors.

method	**accuracy [%]**	**speed [FPS]**
AlexNet [15]	66.65/77.75	963
VGG16 [25]	77.26/86.71	173
VGG19 [25]	76.74/86.06	146
Resnet50 [11]	75.48/84.61	155
Resnet101 [11]	76.46/85.31	95
Resnet152 [11]	77.68/86.20	66
BCNN (VGG-M) [17]	64.83/72.22	87*
BCNN (VGG16) [17]	69.64/78.56	10*
CBL (VGG16) [8]	70.38/80.11	165
CBL (VGG19) [8]	70.69/80.26	141
PCM (AlexNet) [24]	63.24/73.94	15
PCM (VGG19) [24]	75.99/85.24	4
AlexNet + ALL (ours)	77.79/88.60	580
VGG16 + ALL (ours)	**84.13**/**92.27**	154
VGG19 + ALL (ours)	**84.12**/92.00	133
VGG16+CBL + ALL (ours)	75.06/83.42	146
VGG19+CBL + ALL (ours)	75.62/83.76	126
Resnet50 + IMAGE (ours)	82.27/90.79	151
Resnet101 + IMAGE (ours)	83.41/91.59	93
Resnet152 + IMAGE (ours)	83.74/91.71	65

to reproduce the results in respective papers to ensure that we are using the published work correctly. All of the methods use CNNs and the used net influences the accuracy; therefore, the results should be compared with respective base CNNs.

It was impossible to evaluate methods focused only on fine-grained recognition of vehicles as they are usually limited to frontal/rear viewpoint or require 3D models of vehicles for all the types. In the following text we define labels for each evaluated state-of-the-art method and describe details for the method separately.

BCNN. Lin et al. [17] proposed to use **B**ilinear **CNN**. We used VGG-M and VGG16 networks in a symmetric setup (details in the original paper), and trained the nets for 30 epochs (the nets converged around the 20th epoch). We also used image flipping to augment the training set.

CBL. We modified compatible nets with **C**ompact **B**i**L**inear Pooling proposed by [8] which followed the work of [17] and reduced the number of output features of the bilinear layers. We used the Caffe implementation of the layer provided by the authors and used 8 192 features. We trained the net using the same hyper-parameters, protocol, and data augmentation as described in Section 2.4.1.

Table 2.3: Comparison of classification accuracy (percent) on the hard split with standard nets without any modifications, IMAGE modifications using 3D bounding box from surveillance data, and IMAGE modifications using estimated 3D BB (Section 2.2.4).

net	no modification	GT 3D BB	estimated 3D BB
AlexNet	66.65/77.75	77.67/88.28	74.81/87.30
VGG16	77.26/86.71	83.79/92.23	80.60/90.59
VGG19	76.74/86.06	83.91/92.17	81.43/91.57
VGG16+CBL	70.38/80.11	75.04/83.16	72.83/82.92
VGG19+CBL	70.69/80.26	75.47/83.56	73.09/83.09
ResNet50	75.48/84.61	82.27/90.79	79.60/90.40
ResNet101	76.46/85.31	83.41/91.59	80.20/90.42
ResNet152	77.68/86.20	83.74/91.71	80.87/90.93

PCM. Simon et al. [24] propose **P**art **C**onstellation **M**odels and use neural activations (see the paper for additional details) to get the parts of the model. We used AlexNet (BVLC Caffe reference version) and VGG19 as base nets for the method. We used the same hyper-parameters as the authors with the exception of fine-tuning number of iterations which was increased, and the C parameter of used linear SVM was cross-validated on the training data.

The results of all comparisons can be found in Table 2.2. As the table shows, our method significantly outperforms both standard CNNs [15, 25, 11] and methods for fine-grained recognition [17, 24, 8]. The results for fine-grained recognition methods should be compared with the same used base network as for different networks, they provide different results. Our best accuracy (84 %) is better by a large margin compared to all other variants (both standard CNN and fine-grained methods).

In order to provide approximate information about the processing efficiency, we measured how many images different methods are able to process per second (referenced as FPS). The measurement was done with GTX1080 and CUDNN whenever possible. In the case of BCNN we reported the numbers as reported by the authors, as we were forced to save some intermediate data to disk because we were not able to fit all the data to memory (~200 GB). The results are also shown in Table 2.2; they show that our input modification decreased the processing speed; however, the speed penalty is small and the method is still usable for real-time processing.

2.4.3 *Influence of Using Estimated 3D Bounding Boxes instead of the Surveillance Ones*

We also evaluated how the results will be influenced when, instead of using the 3D bounding boxes obtained from the surveillance data (long-time observation of video [DSH14, DHJS15]), the estimated 3D bounding boxes (Section 2.2.4) would be used instead.

The classification results are shown in Table 2.3; they show that the proposed modifications still significantly improve the accuracy even if only the estimated 3D bounding box – the less accurate one – is used. This result is fairly important as it enables to transfer this method to different (non-surveillance) scenarios. The only additional data which is then required is a reliable training set of directions towards the vanishing points (or viewpoints and focal length) from the vehicles (or other rigid objects).

2.5 CONCLUSION

This article presents and sums up multiple algorithmic modifications suitable for CNN-based fine-grained recognition of vehicles. Some of the modifications were originally proposed in a conference paper [SHH16], while others are results of the ongoing research. We also propose a method for obtaining the 3D bounding boxes necessary for the image unpacking (which has the largest impact on performance improvement) without observing a surveillance video, but only working with the individual input image. This considerably increases the application potential of the proposed methodology (and the performance for such estimated 3D boxes is only somewhat lower than when "proper" bounding boxes are used). We focused on a thorough evaluation of the methods: we coupled them with multiple state-of-the-art CNN architectures [25, 11], and measured the contribution/influence of individual modifications.

Our method significantly improves the classification accuracy (up to **+12 percentage points**) and reduces the classification error (up to **50 % error reduction**) compared to the base CNNs. Also, our method outperforms other state-of-the-art methods [17, 24, 8] by **9 percentage points** in single image accuracy and by **7 percentage points** in whole track accuracy.

We collected, processed, and annotated a dataset *BoxCars116k* targeted to fine-grained recognition of vehicles in the surveillance domain. Contrary to a majority of existing vehicle recognition datasets, the viewpoints are greatly varying and correspond to surveillance scenarios; the existing datasets are mostly collected from web images and the vehicles are typically captured from eye-level positions. This dataset has been made publicly available for future research and evaluation.

3 TRAFFIC SURVEILLANCE CAMERA CALIBRATION BY 3D MODEL BOUNDING BOX ALIGNMENT FOR ACCURATE VEHICLE SPEED MEASUREMENT

3.1 INTRODUCTION

Surveillance systems pose specific requirements on camera calibration. Their cameras are typically placed in hardly accessible locations and the optics is focused to larger distances, making the common pattern-based calibration approaches (such as classical [33]) unusable. That is why many solutions place markers to the observed scene and/or measure existing geometric features [26, 3, 31, 19]. These approaches are laborious and inconvenient both in terms of camera setup (manually clicking on the measured features in the image) and in terms of physically visiting the scene and measuring the distances.

In our paper, we focus on *precise* and at the same time *fully automatic* traffic surveillance camera calibration including scene scale for speed measurement. The proposed speed measurement method needs to be able to deal with significant viewpoint variation, different zoom factors, various roads and densities of traffic. If the method should be applicable for large-scale deployment, it needs to run fully automatically without the necessity to stop the traffic on the road for its installation or for performing calibration measurements.

Our solution uses camera calibration obtained from two detected vanishing points and it is built on our previous work [DSH14, DHJS15]. However, this calibration procedure only allows to reconstruct the rotation matrix and intrinsic parameters from the vanishing points, and it is still necessary to obtain the scene scale. We propose to detect vehicles on the road by Faster-RCNN [20], classify them into a few common fine-grained types by a CNN [15] and use bounding boxes of 3D models for the known classes to align the detected vehicles. The vanishing point-based calibration allows for full reconstruction of the viewpoint on the vehicle and the only free parameter in the alignment is therefore the scene scale. Figure 3.1 shows an example of the 3D model and the aligned images. Our experiments show that our method (mean speed measurement error 1.10 km/h) significantly outperforms existing automatic camera calibration method by Dubská et al. [DSH14] (error reduction by 86 % – mean error 7.98 km/h) and also calibration obtained from manual measurements on the road (error reduction by 19 % – mean error 1.35 km/h). This is important because in the previous approaches, the automation always compromised the accuracy, forcing the system developer to trade off between them. Our work shows that manual calibration (though laborious, thorough, and carried out according to state-of-the-art approaches) is inferior to the fully automatic approach based on computer vision methods.

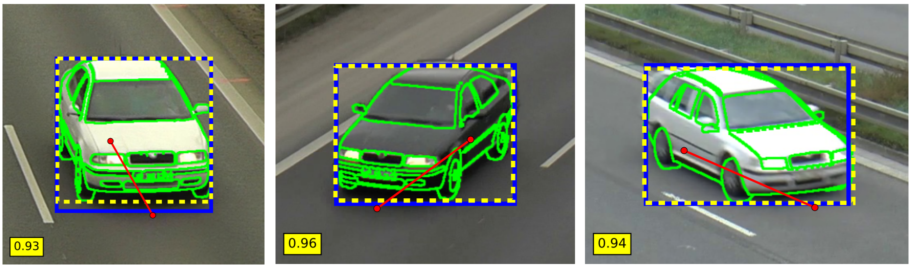

Figure 3.1: Example of detected vehicles and 3D model bounding box aligned to the vehicle detection bounding box.

3.2 TRAFFIC CAMERA MODEL

The main goal of camera calibration in the application of speed measurement is to be able to measure distances on the road plane between two arbitrary points in meters (or different length units), therefore we only focus on a camera model which enables to measure distance between two points on the road plane.

For convenience and better comparison of the methods, we adopt the traffic camera model and notation proposed in previous papers [DSH14, DHJS15]; however, to make the paper self-contained, we briefly describe the model and notation. For intrinsic parameters of our camera model, we assume to have zero pixel skew and principal point $\mathbf{c}$ in the center of the image. The method also assumes the road section to be flat and straight; the experiments reported in the previous work and our experiments as well show that this requirement is not very strict, because most roads that are not sharply curved locally meet this assumption for practical purposes.

Homogeneous 2D image coordinates are referenced by bold small letters $\mathbf{p} = [p_x, p_y, 1]^T$, points on the image plane $\overline{\mathbf{p}} = [p_x, p_y, f]^T$ in 3D, where f is the focal length, are denoted by small bold letters with overline. Finally, other 3D points (on the road plane) are denoted by bold capital letters $\mathbf{P} = [P_x, P_y, P_z]^T$.

Figure 3.2 shows the camera model and its notation. For convenience, we assume that the origin of the image coordinate system is at the center of the image; therefore, the principal point $\mathbf{c}$ has 2D homogeneous coordinates $[0, 0, 1]^T$ (3D coordinates of the center of camera projection are $[0, 0, 0]^T$). As it is shown, the road plane is denoted by $\boldsymbol{\rho}$. We encode vanishing points in the following way. The first one (in the direction of vehicles' flow) is referenced as $\mathbf{u}$; the second vanishing point (whose direction is perpendicular to the first one and which is parallel to the road plane) is denoted by $\mathbf{v}$; and the third one (direction perpendicular to the road plane) is $\mathbf{w}$.

Using the first two vanishing points $\mathbf{u}$, $\mathbf{v}$ and the principal point $\mathbf{c}$, it is possible to compute focal length f, the third vanishing point $\mathbf{w}$, the road plane normalized normal vector $\mathbf{n}$, and the road plane $\boldsymbol{\rho}$. However, the road plane is computed only up to scale (as it is not possible to recover the distance to the road plane only from the

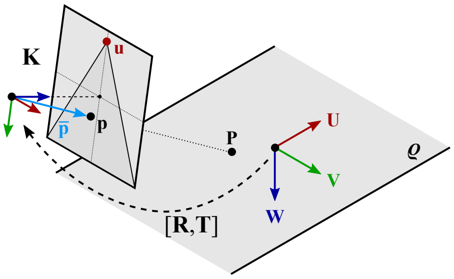

Figure 3.2: Camera model and coordinates. Points denoted by small letters represent points in image space while points in the world space on the road plane $\boldsymbol{\rho}$ are represented by capital letters. The representation stays the same for both finite and ideal points.

vanishing points) and therefore, we add arbitrary value $\delta = 1$ as the constant term in Equation (3.6).

$$
\begin{aligned}
f &= \sqrt{-\mathbf{u}^{T} \cdot \mathbf{v}} && (3.1)\\
\overline{\mathbf{u}} &= [u_x, u_y, f]^{T} && (3.2)\\
\overline{\mathbf{v}} &= [v_x, v_y, f]^{T} && (3.3)\\
\overline{\mathbf{w}} &= \overline{\mathbf{u}} \times \overline{\mathbf{v}} && (3.4)\\
\mathbf{n} &= \frac{\overline{\mathbf{w}}}{\|\overline{\mathbf{w}}\|} && (3.5)\\
\boldsymbol{\rho} &= \left[\mathbf{n}^{T}, \delta\right]^{T} && (3.6)
\end{aligned}
$$

With known road plane $\boldsymbol{\rho}$, it is possible to compute 3D coordinates $\mathbf{P} = [P_x, P_y, P_z]^T$ of an arbitrary point $\mathbf{p} = [p_x, p_y, 1]^T$ by projecting it to the road plane using the following equations:

$$
\begin{aligned}
\overline{\mathbf{p}} &= [p_x, p_y, f]^{T} && (3.7)\\
\mathbf{P} &= -\frac{\delta}{\left[\overline{\mathbf{p}}^{T}, 0\right] \cdot \boldsymbol{\rho}} \overline{\mathbf{p}} && (3.8)
\end{aligned}
$$

It is possible to measure distances on the road plane directly with 3D coordinates $\mathbf{P}$; however, as the road plane is shifted to a predefined distance by the constant term, the distance $\|\mathbf{P}_1 - \mathbf{P}_2\|$ between points $\mathbf{P}_1$ and $\mathbf{P}_2$ is not directly expressed in meters (or other real-world units of distance). Therefore, it is necessary to introduce another calibration parameter referenced as the scene scale λ, which converts the distance $\|\mathbf{P}_1 - \mathbf{P}_2\|$ from pseudo-units on the road plane to meters by scaling the distance to $\lambda\|\mathbf{P}_1 - \mathbf{P}_2\|$.

Using the assumption of the principal point in the center of the image and zero pixel skew, it is necessary for the calibration method to compute two vanishing points ($\mathbf{u}$ and $\mathbf{v}$ in our case) together with the scene scale λ, yielding 5 degrees of freedom. Methods to convert these camera parameters to the standard intrinsic and extrinsic camera model $\mathbf{K}\ [\mathbf{R}\ \mathbf{T}]$ were discussed before in several papers [34, 7, 35], therefore we refer to them.

3.3 CAMERA CALIBRATION AND VEHICLE TRACKING

We adopted the calibration method by [DSH14], which gives the image coordinates of the vanishing points and scene scale information. We improved the method with a more precise detection of the vanishing points, and we infer the scene scale by using 3D models of frequently passing cars.

Our method measures the speed of passing cars detected by Faster-RCNN [20] and tracked by a combination of background subtraction and Kalman filter [14] assisted by the detector. This method, more sophisticated than the previous method [DSH14], gives less false positives and a comparable recall rate. In the case of very dense flow when vehicles overlap each other in the camera image (which does occur rarely even in real conditions), our method would miss some of the cars as we target free-flow conditions. In the following text, we describe in detail the components of the method and evaluate it in Section 3.4.

3.3.1 *Vanishing Point Estimation from Edgelets*

We adopted the algorithm proposed by [DHJS15] (based on detection of two orthogonal vanishing points) for the detection of the first vanishing point and propose to use a similar algorithm for detecting the second vanishing point. However, we improved the detection of the second vanishing point by using edgelets instead of image gradients used in the previous paper [DHJS15]. This change, although subtle, improves the calibration and speed measurement considerably, as the results in Section 3.4.3 show.

We start with the detection of vanishing points from which the camera rotation with respect to the road can be estimated. The first vanishing point $\mathbf{u}$ is estimated from the movement of the vehicles by a form of cascaded Hough Transform [DHJS15] of lines formed by tracking points of interest on the moving vehicles. This is a more stable approach than finding closest point to the lines in an algebraic way, because it is more robust to tracking noise and it is not influenced by vehicles that change lane (and therefore vanishing point of their movement is different from the rest of the vehicles). Similarly to [DHJS15], we use the Min-eigenvalue point detector [23] and the KLT tracker [29].

For detecting the second vanishing point $\mathbf{v}$, we use edges on passing vehicles as many lines formed by the edges coincide with $\mathbf{v}$. This step heavily relies on correct estimation of the orientation of the edges. The angle can be easily computed from gradients, but angles close to $k\pi/2$ are almost impossible to accurately recover on small neighborhoods. We estimate edge orientation from a larger neighborhood by analysis of the shape of image gradient magnitude (edgelets). The detection process is shown in Figure 3.3.

Edgelets are detected by the following algorithm. Given an image $\mathbf{I}$, first, we find seed points $\mathbf{s}_i$ as local maxima of gradient magnitude of the image $\mathbf{E} = \|\nabla \mathbf{I}\|$, keeping

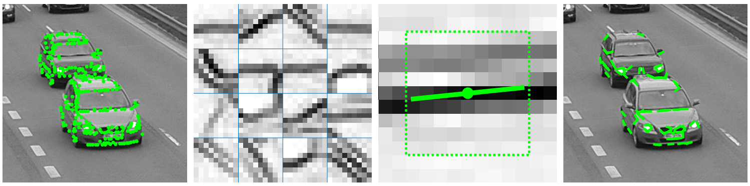

Figure 3.3: Visualization of edgelet detection. From left to right – Seed points $\mathbf{s}_i$ as local maxima of image gradient (foreground mask was used to filter interesting areas); Patches gatherded around the seed points from which is computed the edge orientation; Detail of an edgelet and its orientation superimposed on the gradient image; Top 25 % of edgelets detected in the image.

only the strong ones with magnitudes above a threshold. From 9×9 neighborhood of each seed point $\mathbf{s}_i = [x_i, y_i, 1]^T$, matrix $\mathbf{X}_i$ is formed:

$$\mathbf{X}_i = \begin{bmatrix} w_1(m_1 - x_i) & w_1(n_1 - y_i) \\ w_2(m_2 - x_i) & w_2(n_2 - y_i) \\ \vdots & \vdots \\ w_k(m_k - x_i) & w_k(n_k - y_i) \end{bmatrix} \tag{3.9}$$

where $[m_k, n_k, 1]^T$ are coordinates of the neighboring pixels ($k = 1 \dots 81$) and w_k is their gradient magnitude from $\mathbf{E}$, i.e. for 9×9 neighborhood, the size of $\mathbf{X}_i$ is 81×2. Then, from (3.10), singular vectors and values of $\mathbf{X}_i$ can be computed as:

$$\mathbf{W}_i \mathbf{\Sigma}_i^2 \mathbf{W}_i^T = \mathrm{SVD}\left(\mathbf{X}_i^T \mathbf{X}_i\right), \tag{3.10}$$

where

$$\mathbf{W}_i = [\mathbf{a}_1, \mathbf{a}_2] \tag{3.11}$$

$$\mathbf{\Sigma}_i = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}. \tag{3.12}$$

Vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ represent the eigenvectors of $\mathbf{X}_i$, while λ_1 and λ_2 denote the corresponding eigenvalues. Edge orientation is then the first singular column vector $\mathbf{d}_i = \mathbf{a}_1$ from (3.11) and the edge quality is the ratio of singular values $q_i = \frac{\lambda_1}{\lambda_2}$ from (3.12). Each edgelet is then represented as a triplet $\mathcal{E}_i = (\mathbf{s}_i, \mathbf{d}_i, q_i)$.

We gather the edgelets from the input video, keeping only the strong ones which do not coincide with already estimated $\mathbf{u}$, and accumulate them to the Diamond Space accumulator [5]. The position of the global maximum in the accumulator is taken as the second vanishing point $\mathbf{v}$. It should be noted that in this step, additional filtering can be applied – e.g. mask the Diamond Space to find only plausible solutions (i.e. avoid imaginary focal length from Equation (3.1)), or to find solutions within a certain range of focal lengths or horizon inclinations (when known in advance). This may improve the robustness of the second vanishing point estimation.

3.3.2 *Vehicle Detection and Tracking*

During the speed measurement, passing cars are detected in each frame by the Faster-RCNN (FRCN) detector [20] but any detector can be used as well (e.g. ACF, LDCF [4]). We trained the detector on COD20K dataset [13] containing approximately 20 k car instances for training from views of surveillance nature. The detection rate of the detector is 96 % with 0.02 false positive detections per image on the test part of COD20K dataset. The detector yields a coarse information about locations of cars in the image (bounding boxes are not precisely aligned). We use a simple heuristic to remove detections that would lead to imprecise tracking and ultimately to wrong speed estimation – those that are slightly occluded by other detections and that are farther from the camera. Therefore we track only cars that are fully visible.

For the tracking, we use a simple background model that builds a background reference image by moving average. In the foreground image, compact blobs are detected and the FRCN detections are used to group those blobs that correspond to one car. From each group of blobs, the convex hull and its 2D bounding box are extracted. Finally, we track the 2D bounding box of the convex hull using Kalman filter to get the movement of the car.

For each tracked car, we extract a reference point for speed measurement. The convex hull is used to construct the 3D bounding box [DSH14] and we take the center of the bottom-front edge – the reference point located in the ground/road plane. Each track is represented by a sequence of bounding boxes and reference points both constructed from the convex hull. Our method inherits all the advantages and limitations of the similar approaches based on extraction of the vehicle's foreground mask. We rely on the extractor to do its job properly, and we can take advantage of works dealing with different issues related to for example lighting and weather (for example contour extractors such as [30], or semantic segmentation methods such as [18]).

3.3.3 *Scale Inference using 3D Model Bounding Box Alignment*

The previous state-of-the-art automatic method for scale inference in traffic surveillance by [DSH14] used three-dimensional bounding boxes built around the vehicle and mean dimensions of vehicles to compute the scale. However, this approach has two main drawbacks. The obvious one is in the usage of mean dimensions of vehicles. However, the more important one is not that much obvious: the constructed bounding box is too tight around the vehicle and the tightness is largely influenced by the particular viewpoint direction. This causes systematic errors in the calibration depending on the camera location with respect to the road, leading to high sensitivity to viewpoint change.

We propose to use a different approach to the scale inference, overcoming the mentioned imprecisions. We use fine-grained types of the vehicles (i.e. make, model, variant, model year) and for a few (two in our experiments) common types we obtained 3D models which are rendered to the image and we align them to the real observed vehicles in order to obtain the proper scale.

As it is necessary to know the precise vehicle classes (up to model year) for our scale inference method, we used BoxCars dataset with such images [SHH16] and we also

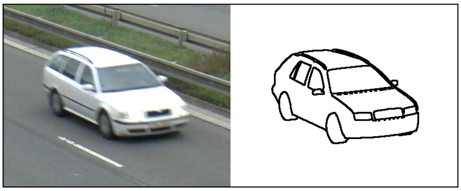

Figure 3.4: Examples of used 3D models (showing only edges) render under the same viewpoint as the corresponding real vehicle on the road. The left image show model which we will refer as Combi and the other two images show 3D model Sedan. Both the models are for Skoda Octavia mk1 which is common on the observed streets.

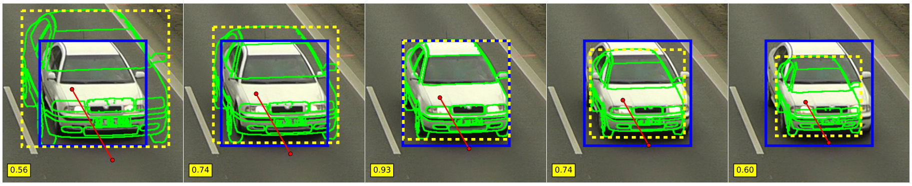

Figure 3.5: Development of IoU (yellow boxes) metric for different scales (**left to right**), vehicle types and viewpoints (**top to bottom**). The left two images show larger rendered vehicle, the middle one show the best match, and the right two images show smaller rendered vehicle. The rendered vehicle is shown only in a form of edges with yellow rectangle as bounding box of the rendered model and blue rectangle denotes the detected vehicle bounding box.

collected some other training data from videos related to papers by [DSH14, DHJS15]. The classification of vehicles is done only into a few most common fine-grained vehicle types on roads in the area plus one class for all the others vehicles. The full training dataset contained ~23 k tracks and ~92 k images of vehicles. We used a CNN [15] for the classification itself. The classification accuracy on the validation set (~7 k of images) was 0.97. As only single instances of vehicles are classified by the CNN, we use mean probability over all of the detections belonging to one vehicle track to improve the recognition rates.

For each vehicle, we also build a 3D bounding box around it [DSH14] to obtain the center $\mathbf{b}$ of the vehicle's base in image coordinates. To obtain the viewpoint vector $\boldsymbol{\phi}$, we first compute the rotation matrix $\mathbf{R}$ which has columns equal to normalized $\overline{\mathbf{u}}$, $\overline{\mathbf{v}}$, and $\overline{\mathbf{w}}$ and then it is possible to compute the 3D viewpoint vector as $\boldsymbol{\phi} = -\mathbf{R}^\top \overline{\mathbf{b}}$. The minus sign is necessary as we need the viewpoint vector going from the vehicle to the camera, not the opposite one.

Once the viewpoint vector to the vehicle, the vehicle's class, and its position on the screen are determined, we render the appropriate 3D model given the parameters. The only open variable is the scale of the vehicle to be rendered (i.e. the distance between the vehicle and the camera). Examples of the two used 3D models are shown in Figure 3.4. Therefore, we render images of the vehicle in multiple different scales and match the bounding boxes of the rendered vehicles with the bounding box detected in the video by using the Intersection-over-Union (IoU) metric. Examples of such matches can be found in Figure 3.5. The figure also shows in red two interesting points related to the vehicle: points on the base of the 3D models representing front $\mathbf{f}$ and rear $\mathbf{r}$ of the vehicle. Finally, for all vehicle instances i and scales j, these points are projected on the road plane, yielding $\mathbf{F}_{ij}$ and $\mathbf{R}_{ij}$ and they are used to compute the scale λ_{ij} (Eq. (3.13), where l_{t_i} is the real world length of the type t_i). For all considered combinations of i and j, the IoU matching metric m_{ij} is computed.

$$\lambda_{ij} = \frac{l_{t_i}}{\|\mathbf{F}_{ij} - \mathbf{R}_{ij}\|} \tag{3.13}$$

To obtain the final camera's scale λ^*, all the scales λ_{ij} are taken into account together with metrics m_{ij}. We consider only cases with m_{ij} larger then a predefined threshold (we used 0.85 in our experiments) to eliminate poor matches. Finally, we compute λ^* according to Equation (3.14). The probability $p\left(\lambda \,|\, (\lambda_{ij}, m_{ij})\right)$ is computed by kernel density estimation with a discretized space.

$$\lambda^* = \arg\max_\lambda \; p\left(\lambda \,|\, (\lambda_{ij}, m_{ij})\right) \tag{3.14}$$

In order to further improve the scale inference, we use several training videos from BrnoCompSpeed dataset [SJŠ+18]. We train the scale-correcting linear regression $\lambda^*_{reg} = \alpha\lambda^* + \beta$, using the manually obtained scales as the ground truth. Even though this step is not necessary, it improves the scale acquisition furthermore by correcting the imprecise geometry of the obtained 3D models.

We also experimented with an alignment metric based on matching of edges on the rendered and detected vehicles (based on distance transform). However, the speed measurement did not improve further. The biggest problem with this method is that most of the edges on the vehicles are blurry and therefore not detected at all. However, the vehicle detector [20] is able to detect the vehicles properly and in most cases accurately. Also, the proposed algorithm using just the bounding boxes is much more efficient in terms of storage (it is possible to store just the bounding boxes, not the images) and computation.

3.3.4 *Speed Measurement of Tracked Cars*

The speed measurement itself is done by following the methodology proposed by [SJŠ+18]. Given a tracked car with reference points $\mathbf{p}_i$ and timestamps t_i for each of the reference point, where $i = 1 \ldots N$, the speed v is calculated from Equation (3.15) by projecting the reference points $\mathbf{p}_i$ to the ground plane $\mathbf{P}_i$ (see Equation (3.8)).

$$v = \operatorname*{median}_{i=1\ldots N-\tau} \left(\frac{\lambda^*_{reg} \|\mathbf{P}_{i+\tau} - \mathbf{P}_i\|}{t_{i+\tau} - t_i} \right) \tag{3.15}$$

The speed is computed as the median value of speeds between consecutive time positions. However, for stability of the measurement, it is better not to use the next frame, but the time position several video frames apart. This is controlled by constant τ and for all our experiments, we use $\tau = 5$ (the time difference is usually 0.2 s).

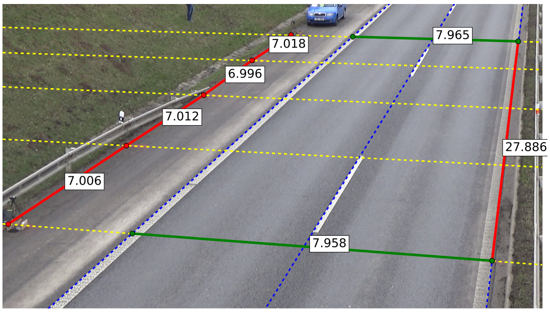

Figure 3.6: An example of manually measured distances between markers on the road plane. Other examples can be found in the original BrnoCompSpeed publication [SJŠ+18]. Blue lines denote the lane dividing lines, lines perpendicular to the vehicles direction are shown in yellow. Finally, measured distances between two points towards the first (second) vanishing point are shown by red (green) color.

3.4 EXPERIMENTS AND RESULTS

To evaluate our proposed methods for camera calibration and scene scale inference, we use a very recent dataset BrnoCompSpeed [SJŠ+18] which contains over 20 k vehicles with precise ground truth speed from multiple locations. The dataset also contains markers on the road with known dimensions between them. For an example of such road markers, see Figure 3.6. The ground truth distances can be used for either calibration or evaluation of distance measurement on the road plane. It is also possible to evaluate the accuracy of vanishing points estimation by using the markings [SJŠ+18]. In the following text we will refer to various methods for camera calibration which are defined as:

- **ITS15** – Automatic camera calibration method as described by Dubská et al. [DHJS15].
- **Edgelets** – Camera calibration method proposed in this paper, Section 3.3.1.
- **ManualCalib** – We use known distances (Figure 3.6) on the road for manual calibration of the camera. In agreement with the previous papers [1, 10, 12] we use intersection lanes dividing lines (blue dashed lines in Figure 3.6) for estimation of the first vanishing point **u**. As there are usually more than just two lane dividing lines, we use least squares minimization to obtain the intersection of multiple lines. Formally, given lines $\mathbf{l}_i$ with normalized normal vectors, we compute vanishing point **u** by solving $\mathbf{Au} = -\mathbf{b}$ in a least squares manner, where rows of **A** contain transposed normal vectors of the lines and rows of **b** contain constant terms of the lines.

  The second vanishing point **v** can be obtained in the same manner (as the intersection of yellow dashed lines in Figure 3.6, since they are perpendicular to the vehicle flow on the road). However, we found out that it is more accurate and robust to use the intersection only as a first guess and then use measured distances on the road to optimize the vanishing point position using Equation (3.16).

$$\mathbf{v}^* = \arg\min_{\mathbf{v}} \left(\sum_{(\mathbf{p}_1,\mathbf{p}_2,d)\in\mathcal{D}_2} |\lambda\|\mathbf{P}_1 - \mathbf{P}_2\| - d| \right), \tag{3.16}$$

where set $\mathcal{D}_2$ contains image endpoints and distances measured on the road towards the second vanishing point (green line segments in Figure 3.6) and scale λ is computed for the given vanishing points $\mathbf{u}, \mathbf{v}$ by Equation (3.17). It should be noted that the computation of 3D coordinates $\mathbf{P}_i$ of image point $\mathbf{p}_i$ depends on the vanishing points (see Equation (3.8) for details). The optimization itself is done by grid search (we loop over discretized feasible positions of $\mathbf{v}$ corresponding to reasonable focal lengths and evaluate the optimization objective (3.16)).

The usage of standard manual methods based on calibration patterns (e.g checkerboards) proposed by [33] is impractical as it would require a large checkerboard (more than $10\,\mathrm{m}^2$) placed on the road.

We also define method names for different approaches for scale inference:

- **BMVC14** – Scale inference method proposed by [DSH14].
- **BBScale + reg** – Our method for scale calibration using bounding box matching (Section 3.3.3) with scale correction regression.
- **ManualScale** – Scale computed from manually measured distances between markers towards the first vanishing point on the road. The scale is computed as the mean value of Equation (3.17) from a set of endpoints and distances $(\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, d_i)$ towards the first vanishing point (red line segments in Figure 3.6).

$$\lambda = \mathbb{E}\left[\frac{d_i}{\|\mathbf{P}_{i,1} - \mathbf{P}_{i,2}\|}\right] \tag{3.17}$$

- **SpeedScale** – Scale is computed from the ground truth speed measurements and it minimizes the speed measurement error for given camera calibration. It can be understood as the lower error bound for the given camera calibration method. The scale is computed as the mean value of Equation (3.18) where set $\mathcal{M}$ contains pairs of ground truth speed $\hat{v}_i$ and measured speed v_i. It is assumed that scale $\lambda = 1$ was used for computation of speeds v_i.

$$\lambda = \mathbb{E}\left[\frac{\hat{v}_i}{v_i}\right] \tag{3.18}$$

If not stated otherwise, the evaluation was done on BrnoCompSpeed – Split C (contains more than 10 k of vehicle tracks for evaluation), because our method requires parameter tuning for the scale correction regression and split C provides sufficient amount of data for training and testing. For each metric, we report mean, median, and 99 percentile error for both absolute units ($err = |\hat{r} - r|$) and relative units ($err = |\hat{r} - r|/\hat{r} \cdot 100\%$), where $\hat{r}$ denotes the ground truth measurement, and r represents the measured value.

Table 3.1: Errors of distance measurement ratios (see Section 3.4.1 for details). The first row for each calibration method contains absolute errors; the relative errors in percents are in the second row.

system	mean	median	99 %
Edgelets (ours)	0.09	0.04	0.49
	6.45	3.38	39.08
ITS15	0.18	0.05	1.36
	11.74	5.25	61.03
ManualCalib	0.02	0.01	0.15
	1.80	1.26	10.98

3.4.1 *Evaluation of VP Estimation – Camera Calibration Error*

To evaluate the camera calibration itself (the obtained vanishing points), we follow the evaluation metric proposed with the BrnoCompSpeed dataset [SJŠ+18]. The evaluation measures the difference between ratios of distances between markings towards the first vanishing point (red lines in Figure 3.6) and the distances between markers towards the second vanishing point (green lines in Figure 3.6). As the ratio does not depend on scale, this metric considers only the camera calibration in the form of two detected vanishing points.

Since we do not require any parameter tuning for the camera calibration method, we report the results on all videos in the BrnoCompSpeed dataset (including extra sessiono). The results (reported in Table 3.1) show that our automatic calibration method Edgelets outperforms calibration method ITS15 almost twice in mean error. It should be noted that the same distances that were used to obtain the manual calibration were evaluated by the calibration error metric based on distance ratios; this gives the manual calibration an unfair advantage in the comparison.

The significant improvement of our method is caused by more precise acquisition of **v**; position of **u** stays the same for our method as for the ITS15 calibration method. The important role of vanishing points is given by two reasons. The first one is that the vanishing points are directly used for estimating the focal length; the second one is that they are used for computation of the viewpoint on the vehicle for scale estimation. Therefore, if the viewpoint is computed imprecisely, the alignment of the rendered 3D model is also imprecise.

3.4.2 *Evaluation of Distance Measurement in the Road Plane*

The next step is to evaluate the camera calibration together with the obtained scale. We use manual annotations of distances on the road plane which are going towards the first or the second vanishing point, respectively (red and green in Figure 3.6).

First, we evaluated the distance measurement only towards the first vanishing point as it is the direction in which the vehicles are going and it is more important for speed

Table 3.2: Distance measurement errors on the road plane for different calibrations. Only distances towards the first vanishing point (red in Figure 3.6) were used for this evaluation. The first row for each calibration method contains absolute errors in meters; the relative errors in percents are in the second row.

system	mean	median	99 %
Edgelets + BBScale + reg (ours)	0.26	0.17	1.08
	2.33	2.06	5.49
ITS15 + BMVC14	1.23	0.81	5.40
	9.62	10.65	21.07
Edgelets + ManualScale (ours)	0.10	0.06	0.57
	0.98	0.62	4.46
ITS15 + ManualScale	0.25	0.14	1.54
	2.11	1.66	8.07
ManualCalib + ManualScale	0.10	0.08	0.32
	1.08	0.65	3.59

Table 3.3: Distance measurement errors on the road plane for different calibrations. Each segment of the table represents a different level of supervision in the calibration. The first row for each calibration method contains absolute errors in meters and the relative errors in percents are in the second row.

system	mean	median	99 %
Edgelets + BBScale + reg (ours)	0.34	0.18	2.29
	3.47	2.28	30.49
ITS15 + BMVC14	1.17	0.72	5.82
	9.79	9.00	55.89
Edgelets + ManualScale (ours)	0.24	0.10	2.60
	2.66	1.00	34.75
ITS15 + ManualScale	0.57	0.20	5.43
	5.84	2.07	52.19
ManualCalib + ManualScale	0.07	0.04	0.30
	0.84	0.50	3.47

measurement. The results are shown in Table 3.2 for different combinations of calibrations and scale estimations. The table shows several things. First, our fully automatic method for camera calibration (Edgelets) and scale inference (BBScale + reg) significantly outperforms the previous automatic method ITS15 + BMVC14. Second, when we use our automatically computed calibration and scale obtained with manual annotations, we achieve almost the same results as ManualCalib + ManualScale, which required much more manual effort than our automatic system.

When we evaluated the same metric with all the distances, the results are similar (see Table 3.3). Again, our method significantly outperforms the previous automatic method. Considering the calibrations with manually obtained scale, our system has a slightly higher error then the manual calibration. However, this is caused by the fact that the manual calibration is optimized directly to the evaluation metric by Equation (3.16) and thus gets an unfair and unrealistic advantage.

To summarize the distance measurement results: our method significantly outperforms previous automatic state-of-the-art for speed measurement – the mean error for distance measurement in the direction of vehicles' flow (which is important for speed measurement) was reduced by 79 % (1.23 m to 0.26 m).

3.4.3 *Evaluation of Speed Measurement*

The most important part of the evaluation is the speed measurement itself. We used the same vehicle detection and tracking system in all experiments so that the results for different calibrations and scales are directly comparable.

We show both quantitative results in the form of Table 3.4. The table is divided into several parts where we compare similar levels of supervision.

The first level of supervision is fully automatic; in the second level, known ground truth dimensions on the road plane are used. In the third and final level of supervision, we use known ground truth speeds to form the lower error bound for different calibration methods.

Regarding the first level of supervision, our system Edgelets + BBScale + reg significantly outperforms the previous automatic method ITS15 + BMVC14 and we reduce the mean speed measurement error by 86 % (7.98 km/h to 1.10 km/h) . Another important fact is that our fully automatic method for camera calibration and scale inference also outperforms manual calibration and scale inference (1.35 km/h mean error) while the error is reduced by 19 % (1.35 km/h to 1.10 km/h). This improvement is important as in the previous approaches, the automation always compromised the accuracy, forcing the system developer to trade off between them. Our work shows that our proposed fully automatic method based on computer vision is superior to manual calibration.

When it comes to the second and the third level of supervision, the results follow the same trend with our calibration outperforming all of them (manual and automatic). The fact that manual calibration is better on the calibration metric (Section 3.4.1) and distance measurement (Section 3.4.2), while our method outperforms the manual calibration at the speed measurement task, is caused by the fact that the manual calibration uses the same data which are then used for the evaluation of the calibration metric and distance measurement. The achieved accuracy is very close to meeting the standards for speed measurements accuracy required for enforcement (typically 3 % in many European countries). The accuracy is definitely comparable to measurements achievable by radars [SJŠ+18], while being considerably cheaper, more flexible, and passive.

Table 3.4: Evaluation of speed measurement errors; all the systems are different only in the calibration and scale inference, with the same tracking of vehicles. Each segment represents one level of supervision in the calibration (automatic, known ground truth distances on road, known ground truth speeds). The first row for each calibration method contains absolute errors in km/h; the relative errors in percents are in the second row.

system	mean	median	99 %
Edgelets + BBScale + reg (ours)	1.10	0.97	3.05
	1.39	1.22	4.13
ITS15 + BMVC14	7.98	8.18	18.58
	10.15	11.45	19.22
Edgelets + ManualScale (ours)	1.04	0.83	3.48
	1.31	1.04	4.61
ITS15 + ManualScale	1.44	1.17	5.43
	1.76	1.50	6.16
ManualCalib + ManualScale	1.35	0.95	4.84
	1.64	1.18	5.40
Edgelets + SpeedScale (ours)	0.52	0.35	2.57
	0.66	0.44	3.71
ITS15 + SpeedScale	0.80	0.57	3.70
	0.99	0.72	4.68
ManualCalib + SpeedScale	0.56	0.38	2.73
	0.71	0.48	3.63

3.5 CONCLUSIONS

We propose a fully automatic method for traffic surveillance camera calibration. It does not have any constraints on camera placement and does not require any manual input whatsoever. The results show that our system decreases the mean speed measurement error by 86 % (7.98 km/h to 1.10 km/h) compared to previous automatic state-of-the-art method and by 19 % (1.35 km/h to 1.10 km/h) compared to manual calibration method. This improvement is important as in the previous approaches, the automation always compromised the accuracy, forcing the system developer to trade off between them. Our work shows that our proposed fully automatic method based on computer vision algorithms is superior to the manual calibration. This result can be important beyond the field of traffic surveillance, since different forms of manual camera calibration are often considered the “ground truth”, but our work shows that automatic calibration from statistics of repeated inaccurate measurements can be more precise, despite requiring no user input. Our method removes the necessity of per-camera setting or calibration, but it still requires some human annotations per coarse geographic region (e.g. European Union or the USA) and per time period when the car models get vastly replaced (e.g. per decade).

In the experiments, we also showed that our method is able to calibrate real world traffic surveillance cameras and our proposed method for vehicle detection and tracking significantly reduces the number of false positives compared to the previous method. In future work, we would like to simplify the system and remove the necessity to render the vehicles by approximation of the bounding box size by a function parametrized by viewpoint and image location.

4 CONCLUSIONS

This thesis presents contributions to the state of the art in **Intelligent Transportation Systems** and **Computer Vision**. Specifically, the work is focused on two tasks – **automatic speed measurement** of vehicles from videos and **fine-grained recognition** of vehicles from images and videos. The papers with core contributions of the thesis were published at top conferences and journals (CVPR, CVIU, IEEE T-ITS).

The first addressed problem is fine-grained recognition of vehicles. In the first paper [SHH16], an image normalization method exploiting automatically extracted 3D bounding boxes around vehicles is proposed. The results show that the method significantly improves classification and verification accuracy. The biggest improvements are for images of vehicles taken from viewpoints unseen during training, therefore the results show that the proposed images normalization improves **generalization to unseen viewpoints**. Further improvements and analysis of the approach were published in my second paper [SŠH18] dealing with the problem. The improved approach eliminates the necessity to know the vanishing points a priori – it is possible to construct the 3D bounding boxes of the vehicles from a single image. The results show that the proposed method consistently improves classification accuracy **by up to 12 percentage points** with different CNNs [15, 25, 11, 8]. The classification error was also **reduced by up to 50 %**.

Currently, we are exploring using the 3D bounding boxes for vehicle re-identification and as the results show [SŠJH18], normalization of vehicles by the proposed "unpacking" of vehicles by their 3D bounding box improve even the re-identification performance. Therefore, the applicability of the image normalization by the 3D bounding boxes goes beyond fine-grained classification of vehicles.

The other addressed problem is automatic speed measurement of vehicles. First, we had to collect a large dataset with precise ground truth speed measurements [SJŠ$^+$18] as there was no dataset with large number of vehicles with precise ground truth speeds. The dataset contains over 20 000 vehicles with ground truth speed measurements acquired from two synchronized LIDAR optical gates. Furthermore, we proposed a method for fully automatic traffic surveillance camera calibration enabling precise speed measurement of vehicles. The approach is based on vanishing point estimation and 3D model alignment of several common fine-grained models. Thus, instead of using artificial calibration patterns or measurements on the road plane, we use the recognized vehicles with known 3D models as "calibration objects". The experimental results show that the method achieves **1.10 km/h** mean speed measurement error while outperforming both state-of-the-art methods and manual calibration in the speed measurement task. This is important because in the previous approaches, the automation always compromised the accuracy, forcing the system developer to trade off between them.

Currently, we are working on fully automatic camera calibration method applicable outside the scope of road surveillance. For example, the method targets calibration of a camera on a parking lot or a square. The approach is based on detected keypoints on fine-grained recognized vehicles and a priori known 3D position of the keypoints within the vehicles' 3D models.

PUBLICATIONS

Publications containing the core of the thesis

[SHH16] Jakub Sochor, Adam Herout, and Jiří Havel. BoxCars: 3D Boxes as CNN Input for Improved Fine-Grained Vehicle Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas: IEEE Computer Society, 2016, pp. 3006-3015. ISBN 978-1-4673-8851-1. ISSN 1063-6919.

[SJH17] Jakub Sochor, Roman Juránek, and Adam Herout. Traffic Surveillance Camera Calibration by 3D Model Bounding Box Alignment for Accurate Vehicle Speed Measurement. In *Computer Vision and Image Understanding*. 2017, vol. 2017, no. 161, pp. 87-98. ISSN 1077-3142.

[SJŠ$^+$18] Jakub Sochor, Roman Juránek, Jakub Špaňhel, Lukáš Maršík, Adam Široký, Adam Herout, and Pavel Zemčík. Comprehensive Dataset for Automatic Single Camera Visual Speed Measurement. In *IEEE Transactions on Intelligent Transportation Systems (to be accepted as regular paper after minor revision)*, 2018. ISSN 1558-0016.

[SŠH18] Jakub Sochor, Jakub Špaňhel, and Adam Herout. BoxCars: Improving Vehicle Fine-Grained Recognition using 3D Bounding Boxes in Traffic Surveillance In *IEEE Transactions on Intelligent Transportation Systems*, 2018. ISSN 1558-0016. DOI: 10.1109/TITS.2018.2799228.

My other publications

[Soc14] Jakub Sochor. Fully automated real-time vehicles detection and tracking with lanes analysis. In *Proceedings of The 18th Central European Seminar on Computer Graphics*. Technical University Wien, 2014.

[DSH14] Markéta Dubská, Jakub Sochor, and Adam Herout. Automatic camera calibration for traffic understanding. In *British Machine Vision Conference*, 2014.

[DHJS15] Markéta Dubská, Adam Herout, Roman Juránek, and Jakub Sochor. Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1162–1171, June 2015.

[SH15] Jakub Sochor and Adam Herout. Unsupervised processing of vehicle appearance for automatic understanding in traffic surveillance. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8, Nov 2015.

[SZK$^+$15] István Szentandrási, Michal Zachariáš, Rudolf Kajan, Jan Tinka, Markéta Dubská, Jakub Sochor, and Adam Herout. INCAST: Interactive camera streams for surveillance cams ar. In *Proceedings of the 2015 14th IEEE International Symposium on Mixed and Augmented Reality*, pages 1–5. Institute of Electrical and Electronics Engineers, 2015.

[ŠSJ$^+$17] Jakub Špaňhel, Jakub Sochor, Roman Juránek, Adam Herout, Lukáš Maršík, and Pavel Zemčík. Holistic Recognition of Low Quality License Plates by CNN using Track Annotated Data. In *International Workshop on Traffic and Street Surveillance for Safety and Security (AVSS 2017)*, 2017. ISBN 978-1-5386-2939-0.

[SŠJH18] Jakub Sochor, Jakub Špaňhel, Roman Juránek, and Adam Herout. Learning Feature Aggregation in Temporal Domain for Re-Identification. In *European Conference on Computer Vision (submitted).*

REFERENCES

[1] 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.

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[3] Viet-Hoa Do, Le-Hoa Nghiem, Ngoc Pham Thi, and Nam Pham Ngoc. A simple camera calibration method for vehicle velocity estimation. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on*, pages 1–5, June 2015.

[4] P. Dollár, R. Appel, S. Belongie, and P. Perona. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8): 1532–1545, August 2014. ISSN 0162-8828.

[5] Markéta Dubská and Adam Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2013.

[6] United Nations Economic Commision for Europe. *Intelligent Transport Systems (ITS) for sustainable mobility*. February 2012. ISBN 978-8897212034.

[7] 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.

[8] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. Compact bilinear pooling. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[9] Timnit Gebru, Jonathan Krause, Yilun Wang, Duyun Chen, Jia Deng, Erez Lieberman Aiden, and Li Fei-Fei. Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the united states. *Proceedings of the National Academy of Sciences*, page 201700035, 2017.

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

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] Xiao Chen He and Nelson H. C. Yung. New method for overcoming ill-conditioning in vanishing-point-based camera calibration. *Optical Engineering*, 46(3):037202–037202–12, 2007.

[13] Roman Juránek, Adam Herout, Markéta Dubská, and Pavel Zemčík. Real-time pose estimation piggybacked on object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[14] 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.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. ISSN 0018-9219.

[17] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN models for fine-grained visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[19] D. C. Luvizon, B. T. Nassu, and R. Minetto. A video-based system for vehicle speed measurement in urban roadways. *IEEE Transactions on Intelligent Transportation Systems*, 18(6):1393–1404, June 2017. ISSN 1524-9050.

[20] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[21] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Deep expectation of real and apparent age from a single image without facial landmarks. *International Journal of Computer Vision*, pages 1–14, 2016. ISSN 1573-1405.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Dec 2015. ISSN 1573-1405.

[23] Jianbo Shi and C. Tomasi. Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, Jun 1994.

[24] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[26] I. Sina, A. Wibisono, A. Nurhadiyatna, B. Hardjono, W. Jatmiko, and P. Mursanto. Vehicle counting and speed measurement using headlight detection. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, pages 149–154, September 2013.

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

[28] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[29] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[30] Jimei Yang, Brian Price, Scott Cohen, Honglak Lee, and Ming-Hsuan Yang. Object contour detection with a fully convolutional encoder-decoder network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[31] Xinhua You and Yuan Zheng. An accurate and practical calibration method for roadside camera using two vanishing points. *Neurocomputing*, 2016. ISSN 0925-2312.

[32] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[33] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000. ISSN 0162-8828.

[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.

[35] Yuan Zheng and Silong Peng. A practical roadside camera calibration method based on least squares optimization. *IEEE Transactions on Intelligent Transportation Systems*, 15:831–843, 2014.

[36] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *International Confernece on Pattern Recognition*, pages 28–31, 2004.