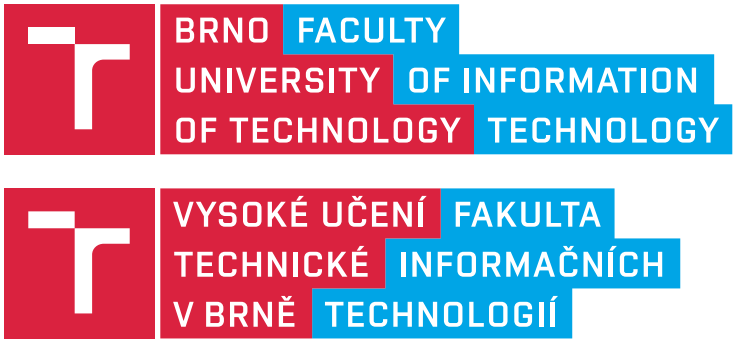


DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

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

AUTOR PRÁCE

ING. JAKUB SOCHOR

SUPERVISOR

VEDOUCÍ PRÁCE

PROF. ADAM HEROUT, PH.D.

BRNO 2018

Ing. Jakub Sochor: *Automatic Traffic Video Surveillance: Fine-Grained Recognition of Vehicles and Automatic Speed Measurement*, doctoral thesis Brno, Brno University of Technology, Faculty of Information Technology, 2018.

Dedication

To my parents, family, and friends for their support in my life,
to my supervisor Adam Herout for the guidance and mentoring,
to my colleagues for their help and cooperation.

ABSTRACT

This thesis presents my 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 and **fine-grained recognition** of vehicles.

The problem of vehicle fine-grained recognition can be defined as a task where the system is supposed to produce exact fine-grained type (e.g. "Škoda Octavia combi mk2") for a presented vehicle. In my first paper on this topic, a method exploiting automatically constructed 3D bounding boxes around vehicles is proposed. The results show that the method significantly improves classification and verification accuracy. Further improvements and analysis of the approach was published in my second paper dealing with the problem. The improved approach eliminates necessity to know vanishing points a priori – it is possible to construct the 3D bounding box of the vehicle from a single image of the vehicle. The results show that our proposed method consistently improves classification accuracy **by up to 12 percentage points** with different Convolutional Neural Networks. The classification error was also **reduced by up to 50 %**.

The second addressed problem is automatic speed measurement of vehicles. The proposed system should work from a single camera without any manual calibration or input. First, we had to collect a large dataset with precise ground truth speed measurements as there was no such dataset. The dataset contains over 20 000 vehicles with ground truth speed measurement 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 vehicle models. The experimental results show that our 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.

KEYWORDS

fine-grained recognition, traffic surveillance, camera calibration, speed measurement

ABSTRAKT

V rámci této dizertační práce se zaměřuji na **Inteligentní dopravní systémy** a **Počítačové vidění** – především **automatické měření rychlosti** a **rozpoznání automobilů podle typů**.

Rozpoznání automobilů podle typů je úkol, ve kterém system má predikovat přesný typ (např. „Škoda Octavia combi mk2“) pro daný obrázek automobilu. Publikoval jsem dva články, které popisují navržený přístup k tomuto problému a tvoří jádro této dizertace. Prezentovaná metoda je založena na 3D obalových kvádrech postavených okolo automobilů, které jsou následně využity pro rozbalení obrázku automobilu do roviny a tudíž normalizaci vstupu neuronové sítě, která dělá následné rozpoznání. Přístup byl dále rozpracován v druhé publikaci, kde je navržena metoda pro určení tohoto 3D obalového kvádru z jediného obrázku – tudíž není nutné mít zkalibrovanou kameru. Experimentální výsledky ukazují, že navržená metoda zlepšuje úspěšnost rozpoznání **o 12 procentních bodů** – chyba rozpoznání je **redukována o 50 procent**.

Při měření rychlosti má systém za úkol odhadnout rychlost projíždějících aut z videa. Cílem je také, ať měření probíhá plně automaticky bez jakékoli manuální kalibrace. Jelikož neexistoval žádný dataset, který by obsahoval velké množství průjezdů s přesně změřenou rychlostí, tak jsme nejprve takovýto dataset pořídili. Dále jsem navrhnul metodu pro plně automatickou kalibraci dopravní dohledové kamery což umožňuje měřit rychlost automobilů pozorovaných touto kamerou. Metoda je založena na odhadu kalibrace pomocí detekovaných úběžníků scény a následného zarovnání 3D modelů několika běžných typů automobilů. Experimentální výsledky ukazují, že navržená metoda dosahuje průměrné chyby měření rychlosti **1,10 km/h**.

KLÍČOVÁ SLOVA

rozpoznání typů automibilů, dohled dopravy, kalibrace kamery, měření rychlosti

BIBLIOGRAPHIC CITATION

Ing. Jakub Sochor: *Automatic Traffic Video Surveillance: Fine-Grained Recognition of Vehicles and Automatic Speed Measurement*, doctoral thesis, Brno University of Technology, Faculty of Information Technology, 2018.

DECLARATION

I declare that this dissertation thesis is my original work and that I have written it under the guidance of prof. Adam Herout, Ph.D. All sources and literature that I have used during my work on the thesis are correctly cited with complete reference to the respective sources.

Brno, 2018

Ing. Jakub Sochor, February 26, 2018

ACKNOWLEDGMENTS

I would like to thank my supervisor Adam Herout for his mentoring, guidance, inspiration, and always positive and constructive feedback to my work. It was also pleasure to work and cooperate with my colleagues Roman Juránek, Jakub Špaňhel, Markéta Dubská, and all the other people I have worked with during my studies. Also, I would like to acknowledge everyone who supported me through my life. So many thanks to my family, parents, siblings, friends, and everyone else I have met in my life, for their support and encouragement.

CONTENTS

1

INTRODUCTION

United Nations Economic Commission for Europe in document *Intelligent Transport Systems (ITS) for sustainable mobility* [42, 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. [50] 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.

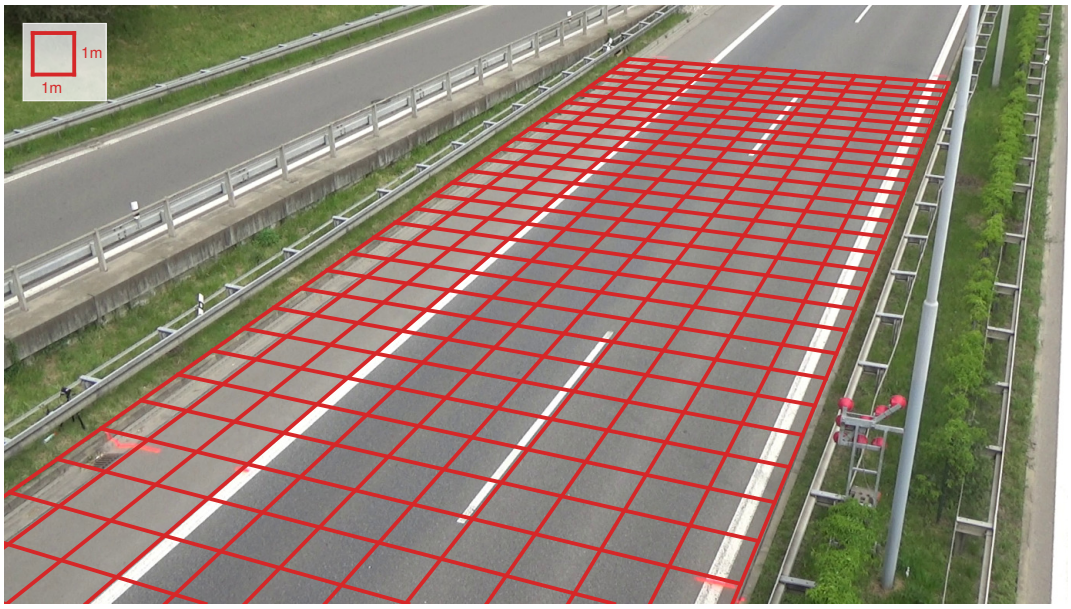


Figure 1.1: An example of calibration for speed measurement obtained by a fully automatic algorithm proposed in the thesis. The calibration is represented by an orthogonal regular grid with 1 m sides.

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 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 [85] 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 [82, 143, 62, 47]. 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

[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

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].

1.3 OTHER PUBLICATIONS

I have also authored and co-authored other publications during my studies while all of them are directly connected to the topic of the dissertation thesis. Therefore, I provide the list of the papers in chronological order of publication:

- *Fully Automated Real-Time Vehicles Detection and Tracking with Lanes Analysis* – **CESCG[5], 2014** [Soc14]. Paper on a student conference based on my Master's thesis. The paper was accepted for oral presentation and I have received award for the 3rd best paper on the conference.

- *Automatic Camera Calibration for Traffic Understanding* – **BMVC[6], 2014** [DSH14] Our first publication dealing with automatic speed measurement of vehicles from traffic surveillance camera.

- *Fully Automatic Roadside Camera Calibration for Traffic Surveillance* – **IEEE T-ITS[7], 2014** [DHJS15]. Publication introducing traffic camera calibration by detection of vanishing points.

---

[4] Computer Vision and Image Understanding – IF: 2.498

[5] Central European Seminar on Computer Graphics

[6] British Machine Vision Conference

[7] IEEE Transactions on Intelligent Transportation Systems – IF: 3.724

- *Unsupervised Processing of Vehicle Appearance for Automatic Understanding in Traffic Surveillance* – **DICTA[8], 2015** [SH15]. Clustering of fine-grained vehicle samples from a camera. Method proposed in the paper was used for construction of the BoxCars21k dataset [SHH16]. The paper was accepted for oral presentation.

- *INCAST: Interactive Camera Streams for Surveillance Cams AR* – **ISMAR[9], 2015** [SZK+15]. Paper dealing with Augmented Reality with static cameras and dynamic content. Our automatic traffic camera calibration methods [DSH14, DHJS15] were used for the static camera calibration for the paper.

- *Holistic Recognition of Low Quality License Plates by CNN using Track Annotated Data* – **IWT4S-AVSS[10], 2017** [ŠSJ+17]. In this paper, we propose to use single CNN for recognition of license plates. As the method does not require any segmentation of characters, it is able to recognize even license plate of low quality. The results show that our method outperforms other solutions by an order of magnitude.

- *Learning Feature Aggregation in Temporal Domain for Re-Identification* – **ECCV[11], 2018, submitted** [SŠJH18]. This paper is focused on aggregation of learned features in temporal domain. We propose an end-to-end approach which produces different weights for different elements of the feature vectors. Furthermore, the experimental results show that 3D bounding boxes [SHH16, SŠH18] improve performance even for the vehicle re-identification task.

1.4 THESIS ORGANIZATION

This dissertation thesis consists of several parts. Part I introduces relevant background and state-of-the-art methods for both fine-grained recognition of vehicles and automatic speed measurement of vehicles. My contributions to the fine-grained recognition and speed measurement are described in Parts II and III. Both these parts consist of re-formated copies of papers [SHH16, SJŠ+18, SJH17, SŠH18] which form the core of the dissertation thesis. Finally, discussion about the impact of the work is presented in Part IV together with thesis conclusions.

[8] International Conference on Digital Image Computing: Techniques and Applications

[9] IEEE International Symposium on Mixed and Augmented Reality

[10] International Workshop on Traffic and Street Surveillance for Safety and Security (in conjuction with IEEE AVSS)

[11] European Conference on Computer Vision

Part I

STATE OF THE ART

This part of dissertation thesis provides overview of state of the art methods for vehicle fine-grained recognition and speed measurement of vehicles using traffic surveillance cameras. Different methods for both these tasks are introduced and described. Limitations of these methods and implications for the direction of my work are also discussed.

2

FINE-GRAINED RECOGNITION OF VEHILES

Fine-grained object recognition received increasing levels of attention by the computer vision community in the past few years [76, 46, 49, 38, 160, 7, 71, 157, 191]. The fine-grained recognition is a task where different subtypes (e.g. animal breeds, vehicle types) should be classified. This is usually harder than standard classification (e.g. is that a building or a person?) as there is much smaller inter-class variation and still relatively high intra-class variation due to pose variance.

More specifically, fine-grained recognition of vehicles is a task of determining what model is present on a given image. The fine-grained recognition can have different granularities. In our work, we consider the finest and complex granularity where we want to distinguish even different model years of the same vehicle model. Therefore, our task is to recognize the make, model, submodel, and model year of a given vehicle (e.g. "Škoda Octavia sedan mk1").

To provide context to the proposed method, we provide a summary of existing fine-grained recognition methods (both general and focused on vehicles). We also briefly describe recent advancements in Convolutional Neural Networks as they are used for the fine-grained recognition in the proposed method.

2.1 FINE-GRAINED OBJECT RECOGNITION IN GENERAL

A number of papers addressing this topic was published and some of them are discussed in this part of the text. Methods used also on other fine-grained recognition tasks (not only vehicles) are described in this section. However, some of these algorithms described in this part were used *also* for fine-grained recognition of vehicles. Methods focused *only* on fine-grained recognition of vehicles are described in the following section.

Quite a large number of papers [117, 14, 43, 149, 164, 107, 139, 179, 183, 142, 54, 115, 170, 34, 171, 180, 181] propose to use parts of the object for fine-grained recognition. Several papers [149, 164] use saliency masks to obtain the location of the parts. Other papers [139, 54] use transfer of parts' locations from objects of the same type with annotated parts. Lam et al. [107] propose to use HSNet for an iterative search of discriminative parts. Zhang et al. [179] use detected parts in a dense manner with superpixels corresponding to different parts of birds. Several papers [178, 72] propose to unify fine-grained classification and semantic parts

detection into one end-to-end trainable network. Zhang et al. [183] propose to iteratively train and pick deep filters which correspond to parts. Simon et al. [142] use deep neural activation maps to detect parts of objects which are used to build a star shape constellation model. Xiao et al. [163] proposed to use two nets – one for object level classification and the second one for part level classification. Zhang et al. [182] proposed to use part-based fine-grained recognition with R-CNN [52]. Lin et al. [91] use three neural networks for simultaneous localization, alignment, and classification of images. Liu et al. [95] focused on dog breed classification and proposed to use parts localized on dogs' faces (e.g. nose, ears, nose).

Other papers focused on exploiting segmentation for the fine-grained recognition. Chai et al. [19] propose to use segmentation to identify discriminative foregrounds of objects. In their later paper [20], the segmentation is used in combination with Deformable Part Models [40]. The segmentation approach was also proposed by Li et al [89]; however, they use a segmentation algorithm which is optimized and fine-tuned for the purpose of localization of discriminative parts for fine-grained recognition. Krause et al. [81] proposed to use co-segmentation and automatic part localization in combination with R-CNN to overcome missing annotations of parts. Gavves et al. [48] propose to detect objects, for each pixel compute probability that the pixel is a part of the object based on the detections, and use GrabCut [129] for segmentation of the objects which is initialized by the probabilities. The authors then encode the shape of the object by Histograms of Oriented Gradients [29], use this for alignment, and finally encode the object appearance by Fisher vectors [118].

Sanchez et al. [132] also use Fisher vectors [133] for fine-grained recognition. There are also improvements to this approach using Fisher vectors; Gosselin et al. [55] propose to use late fusion of two different Fisher vector systems, Nakayama [108] augment descriptors by adding polynomials of feature vector elements and reduce dimensionality by canonical correlation analysis [65].

Some authors use bilinear classification [120] for fine-grained recognition. Lin et al. [92] use only convolutional layers from a Convolutional Neural Network for feature extraction which are classified by a bilinear classifier [120]. Gao et al. [47] followed the path of bilinear pooling [92] and proposed a method for Compact Bilinear Pooling which achieves same accuracy as the full bilinear pooling [92] with a lower number of features.

There were also attempts to address a hierarchy of fine-grained classes. Xie et al. [166] proposed to use hyper-class for data augmentation and regularization of multi-task fine-grained deep learning. Zhou et al. [189] use CNN with Bipartite-Graph Labeling to achieve better accuracy by exploiting the fine-grained annota-

tions and vehicle body type (e.g. van, sedan). Some authors [76, 46] addressed fine-grained recognition with a high number of classes.

Chabot et al. [18] constructed a single network for estimation of 3D structure of the objects in images. The network is used for alignment of several typical simple 3D models. Such information can be used for feature localization on the objects and other related tasks. Farrel et al. [39] use pose normalization based on Poselet [10, 11] in cooperation with Random Forests [12] for fine-grained recognition of birds.

Also, an analysis of importance of image resolution in fine-grained recognition was performed by Chevalier et al. [105]. Cai et al. [13] propose to use one Convolutional Neural Network which first estimates super-resolution version of the image and the enhanced image is then used for the fine-grained classification.

Finally, Gebru et al. [49] propose a method for model adaptation from other image domain (e.g. e-commerce presentation images) to target domain for fine-grained recognition.

2.2 FINE-GRAINED RECOGNITION OF VEHICLES

Fine-grained recognition of vehicles is discussed in this section as it is a special case of fine-grained recognition based on visual appearance. The goal of the recognition is, in an ideal case, to identify the exact type of the vehicle (e.g. make, model, submodel, model year). The recognition system focused only on vehicles (in relation to general fine-grained classification of birds, dogs, etc.) can benefit from that the vehicles are rigid, have some distinguishable landmarks (e.g. license plates) or there are 3D CAD models of the vehicle.

2.2.1 *Methods Limited to Frontal Images of Vehicles*

Petrovic et al. [119] propose an algorithm for fine-grained recognition of vehicles based on that the vehicle is seen from the front and therefore it is possible to detect the license plate. The license plate is used for extraction of a patch around the license plate. Different types of features are extracted for the patch and classified by a nearest neighbor framework with euclidean or cosine distance.

Dlagnekov et al. [31] also use detection of license plates to obtain region of interest for make & model recognition. The license plate is detected by AdaBoost [156] and the surroundings of the license plate is extracted as the region of interest. For the recognition of vehicles' types, the authors propose to use SIFTs [101] and match their location with all other images of vehicles.

He et al. [61] focus on surveillance cameras which are located above the road in a way that vehicles are seen from the frontal side. The authors detect vehicles by

DPM [40], use different types of anchors (license plates, headlights, statistical) to rectify and normalize the image. Illumination is also normalized to achieve better performance and edges are extracted from the frontal rectified image as features for each vehicle. Finally, the features are classified by a standard neural network.

Liao et al. [90] propose to use Strongly Supervised DPM [3] to identify different parts on a frontal image of a vehicle. A representation of the vehicle is extracted from the detected parts, which are weighted by their discriminative power. The authors show that using the proposed weighting scheme improve the recognition accuracy. The results also show that the most discriminative parts are air grille and headlamps.

Baran et al. [4] use different descriptors to achieve real-time recognition performance. The authors use SURF [5], SIFT [101] or Edge Histogram [114] descriptors classified by binary SVM [137] in one-vs-all approach. The experiments show that the best results are achieved by weighting these descriptors. This approach is not inherently limited only to frontal viewpoints; however, it was only used on frontal images of vehicles, thus it is hard to generalize the achieved results to an arbitrary viewpoint.

Zhang et al. [175, 176] propose to use a different technique for extraction of feature vectors for vehicles. They used a modified version of HMAX descriptor [125]. The classification itself was performed by numerous classifiers (k-nearest neighbor, decision trees, multilayer perceptron [127], Random Forests [12], and SVM [27]). The experiments were carried out on frontal images of vehicles taken during different illumination conditions (day, night, etc.).

Hu et al. [69] propose a novel real-time method for recognition of vehicle brands. The authors detect vehicles by DPM [40] and learn the most discriminative parts by a new algorithm which the authors named Spatially Coherent Discriminative Pattern Learning. The results show that the authors achieve a high accuracy in the brand recognition task (0.94); however, as the authors process relatively high resolution (for surveillance cameras) frontal images of vehicles, the brand recognition is easier because brand logos are visible and recognizable.

Hsieh et al. [68] propose a make & model recognition method which builds on the symmetry property of vehicles. The authors propose to use a novel version of SURF [5] descriptor called Symmetrical SURF and use this novel descriptor for detection and recognition of vehicles. The classification itself is performed by SVM. Unfortunately, the usage of Symmetrical SURF limits the method to frontal/rear images of vehicles, as vehicles are not symmetrical from sides or different viewpoints.

Pearce et al. [116] use license plate detection to obtain ROI for fine-grained recognition. The authors use different features (Canny edges [15], Square mapped gra-

dients [119] and Harris corners [59]) and recursively sum features from different areas for better alignment. The final class is obtained by k-nearest neighbor classifier or SVM.

2.2.2 *Methods Based on 3D CAD Models*

Lin et al. [93] propose to jointly optimize 3D model fitting and fine-grained classification of vehicles to overcome pose variation of different vehicle types. First, the initial pose is obtained by DPM [40] and positions of landmarks are estimated by a linear Support Vector Regressor which is used for the alignment in further processing. Then, 3D model represented by Active Shape Model [25] is fitted to the vehicle image with average pixel distance used as the alignment metric between the vehicle and the model. Model with the lowest average pixel distance is selected for the fitting (the models represent all sedans, pickups, etc.). Finally, HOG [29] features and Fisher vectors [118] are used for classification by linear multi-class SVM.

Hsiao et al. [67] propose to use 3D CAD model to deal with viewpoint variance. The authors propose to detect edges on the vehicle and use the 3D model (formulated as Active Shape Model [25, 88]) for alignment which is done by 3D chamfer matching. In the recognition step, the alignment is done for every 3D model present in the dataset and the vehicle is classified by logistic regression with average chamfer distances for each model as inputs.

Krause et al. [80] propose to use synthetic data to train geometry and viewpoint classifiers for 3D model and 2D image alignment. The authors directly sample patches from the 3D surface of aligned vehicles; rectify the patches and compute RootSIFT [2, 101] for each rectified patch. For the final feature representation, the authors use novel 3D versions of Spatial Pyramid [84] and BubbleBank [30]. The classification is performed by SVM.

Prokaj et al. [121] use synthetic 3D CAD models to deal with viewpoint variations. The authors propose to detect SIFT [101] features on the vehicle image and on every 3D model seen from a set of discretized viewpoints. The final type is obtained as the type with the highest similarity under the recognized viewpoint. This simple method has, however, several limitations and one of the biggest of them is that it is computationally expensive and does not scale with the number of vehicle types.

2.2.3 *Other Methods*

Yang et al. [169] propose to use Deep Convolutional Neural Networks [85, 82] for fine-grained classification of vehicles. This possibility was enabled by their rela-

tively big dataset of vehicles (see Section 2.4). The authors also focused on maximal speed estimation of vehicles, number of doors, and other features of vehicles.

Similarly, Zwemer et al. [191] use CNN without any modification. However, the authors acquired training data automatically by using license plates and a public database of vehicles which includes the license plate and make & model.

Gu et al. [58] propose a pipeline for recognition of vehicles from different viewpoints. First, they extract the center and roughly estimate the viewpoint from the bounding aspect ratio. Then, they use a different Active Shape Model for alignment in different viewpoints and they use segmentation for background removal; finally, the authors use template matching for obtaining the final vehicle type.

Stark et al. [146] propose to use an extension of DPM [40] to be able to handle multi-class recognition. The model is represented by latent linear multi-class SVM with HOG [29] features. The authors show that the system outperforms different methods based on LLC [158] and HOG [29]. The recognized vehicles are used for eye-level camera calibration.

Boonsim et al. [9] propose a method for fine-grained recognition of vehicles at night. The authors use relative position and shape of features visible at night (e.g. lights, license plates) to identify the make & model of a vehicle, which is visible from the rear side.

Lee et al. [86] try to overcome small rotation variance ($0° - 15°$) by detection of license plate and normalizing the image to the frontal view of vehicle. For classification, the authors use Region of Intereset (ROI) extracted around the license plate.

Fang et al. [38] propose to use an approach based on detected parts. The parts are obtained in an unsupervised manner as high responses from mean response map of the last convolutional layer of used CNN. The classification itself is then done by merging all feature vectors from whole image and parts and classifying by SVM. Although the method is not explicitly limited to frontal/rear viewpoints, the design and evaluation of the method was done only on frontal images of vehicles.

Wang et al. [160] propose to use detected parts of vehicles and extracted features by CNN and HOG followed by SVM classification and voting scheme between different parts of the vehicle. Hu et al. [71] introduce spatially weighted pooling of convolutional features in CNNs to extract important features from the image. Birglari et al. [7] propose to use discriminative parts and use classification model formulated as Latent SVMs.

A different approach was proposed by Wang et al. [157]. The authors propose to train the classifier on data acquired from web and use Maximum Mean Discrepancy [57] to match the feature domains of the web nature source training data and surveillance nature target data.

Table 2.1: Summary of different algorithms for fine-grained recognition of vehicles. The **view** flag indicates whether the method is suitable for images from arbitrary viewpoint. The **surv** determines whether the method is suitable for surveillance images (e.g. the method does not require a 3D model, it is able to work with low-resolution images). The table continues on the next page.

author	method	view	surv
Petrovic, 2004 [119]	ROI extracted around license plate, multiple features computed for the ROI, classified by nearest neighbor	✗	✓
Dlagnekov, 2005 [31]	License plate detection by AdaBoost for ROI extraction, SIFT matching for recognition	✗	✓
Prokaj, 2009 [121]	Direct comparison of SIFT similarity detected on vehicle image and 3D model under different viewpoint	✓	✗
Pearce, 2011 [116]	Recursively sum different features for better alignment, classification by k-NN or SVM	✗	✓
Stark, 2012 [146]	Extension of DPM [40] to multi-class problem, HOG [40] features	✓	✓
Gu, 2013 [58]	Viewpoint estimation, active shape model alignment, segmentation to background and vehicle, template matching for final class recognition	✓	✗
Krause, 2013 [80]	Alignment of images by viewpoint classifiers trained on synthetic data from 3D CAD models, 3D version of [84] and [30] for feature extraction, SVM for classification	✓	✗
Lee, 2013 [86]	Detection of license plate corners and normalization of the license plate position	✗	✓
Zhang, 2013 [175, 176]	Detection of license plate for ROI extraction, ensemble by majority voting of different features and different classifiers for recognition	✗	✓
Hsiao, 2014 [67]	3D chamfer matching of 2D edges and 3D CAD model for alignment, classification by logistic regression with average chamfer distance as input	✓	✗
Hsieh, 2014 [68]	Novel version of SURF [5] called Symmetrical SURF used for detection and recognition of vehicles, classification done by SVM	✗	✓
Lin, 2014 [93]	Simultaneous optimization of predicted class label and 3D Active Shape Model [25] for alignment, classification by SVM	✓	✓

author	method	view	surv
Baran, 2015 [4]	Weighting different descriptors (SURF [5], SIFT [101], Edge Histogram [114]), classification by SVM	✗	✓
He, 2015 [61]	DPM detection of vehicles seen from the frontal side, different anchors for position normalization, edges as features classified by neural network	✗	✓
Hu, 2015 [69]	Detection of vehicles by DPM [40], learning the most discriminative patches and their classification by linear multi-class SVM	✗	✓
Liao, 2015 [90]	SSDPM [3] parts identification on frontal image, parts weighting scheme to improve accuracy of recognition	✗	✓
Yang, 2015 [169]	Convolutional Neural Network without any modification	✓	✓
Boonsim, 2016 [9]	Detection of lights at night and extraction of features from ROI relative to the detected lights	✗	✓
Fang, 2016 [38]	Detection of parts discovered from convolutional layers high responses	✗	✓
Wang, 2016 [160]	HOG and CNN features with SVM and voting scheme	✗	✓
Biglari, 2017 [7]	Localisation of discriminative parts and classification by Latent SVMs	✓	✓
Hu, 2017 [71]	Spatially weighted pooling of CNN convolutional features	✓	✓
Wang, 2017 [157]	CNN training on web-nature images and with Maximum Mean Discrepancy [57] to match the target domain	✓	✓
Zwemer, 2017 [191]	CNN without any modifications, training on automatically acquired data using license plates	✓	✓

2.2.4 *Summary*

Quite a large number of methods for recognition of vehicle make & model were described. One large group [119, 31, 90, 61, 4, 176, 69, 68, 116, 175, 9, 86] is limited to frontal/rear images of vehicles. One part [119, 31, 116, 175, 86] of this group uses license plates for ROI localization and extraction, other methods [90, 61, 69, 146, 7] use Deformable Part Models [40] (or modifications of DPM) to detect and recognize vehicles and there are also methods [4, 176, 68] using directly different descriptors and feature point detectors.

Other methods [93, 67, 80, 121] utilize available 3D CAD models to deal with viewpoint variations. The modeling is done usually by Active Shape Models [25] (in case of methods [58, 93, 67]) or other feature point detectors and descriptors (methods [80, 121]). Then, some works [169, 191, 157] propose to use Deep Convolutional Neural Networks [85, 82] for the classification of fine-grained vehicles. There were also papers [38, 160] exploiting features extracted from CNN for the classification.

A summary of all the methods can be found in Table 2.1 and as the table shows, only a small number of methods are able to deal with images taken from reasonably arbitrary viewpoint and at the same time to be usable in surveillance scenarios where it is impossible or impractical to provide other data (such as 3D CAD models). Thus, my work in the fine-grained recognition is focused on this area – recognition of vehicles' make & model from images taken by a surveillance camera from an arbitrary viewpoint.

2.3 DEEP CONVOLUTIONAL NEURAL NETWORKS

The first version of Convolutional Neural Networks was proposed by LeCun et al. [85] in 1998 and the authors used it for on-line handwriting recognition. Much more recently, Deep Convolutional Neural Networks got much attention than before, thanks to the paper by Krizhevsky et al. [82]. This network was used for ImageNet classification [130] and decreased Top-5 error by 38 % (from 26.2 % to 16.4 %). After the network by Krizhevsky et al. [82], deeper and more complex CNNs such as the GoogLeNet by Szegedy et al. [151] or ResNets by He et al. [62] seem to be consistently winning the ImageNet contest [130].

Besides the image classification, the deep CNNs can be used recognition of people [153, 150], verification [153] object detection [52, 184, 123], text debluring [66], deconvolution [136, 167], video analysis [77, 144, 148], text recognition [73], people counting [177], metric learning [70], edge detection [138], pose estimation [37, 155], and other computer vision [161, 165, 141] and non computer vision tasks [24].

Recently, authors also used input normalization to improve performance of CNN [153] and adding additional training data to CNN [82]. Also, parts of the CNN can be viewed as feature extractors and independently reused. These trained feature extractors outperform the hand-crafted features [8, 153].

To sum up, CNNs are currently used for a wide variety of tasks and they seem to outperform previous solutions by a great margin. However, these nets require a large amount of training data and long training (usually on GPU).

2.4 EXISTING DATASETS FOR FINE-GRAINED RECOGNITION OF VEHICLES

For accurate fine-grained vehicle recognition, it is also important to have a good dataset, besides the algorithm itself. This importance grows with the last years with boost of Deep Convolutional Neural Networks as to train these networks, it is necessary to have a high number of training samples. For the purposes of traffic surveillance, it is necessary to have a dataset containing vehicles captured from multiple viewpoints (to eliminate limitations to frontal or rear viewpoint) and also the vehicles must not be only from eye-level viewpoints, as surveillance cameras are typically above observed vehicles.

There is a large number of datasets of vehicles [130, 1, 113, 36, 162, 16, 111, 87, 53, 134, 51, 112, 106] which are usable mainly for vehicle detection, pose estimation, and other tasks. However, these datasets do not contain annotation of the precise vehicles' make & model; thus these datasets cannot be used for fine-grained vehicle recognition.

When it comes to the publicly available fine-grained datasets, a few of them exist and all are quite recent. Lin et al. [93] published *FG3DCar* dataset (300 images, 30 classes), Stark et al. [146] made another dataset containing 1,904 vehicles from 14 classes. Krause et al. [80] published two datasets; one of them, called *Car-197*, contains 16k of images and 196 classes. The other one, *BMW 10*, is made of 10 models of BMW vehicles and 500 images. Finally, Liao et al. [90] created a dataset of 1,482 vehicles from 8 classes. All these datasets are relatively small for training the CNN for real-world surveillance tasks.

Yang et al. [169] published a large dataset *CompCars* in 2015. The dataset consists of a web-nature part, made of 136k of vehicles from 1,600 classes taken from different viewpoints. Then, it also contains a surveillance-nature part with 44k frontal images of vehicles taken from surveillance cameras. Finally, Tafazzoli et al. [152] collected quite large *VMMR* dataset. However, the dataset was acquired by web crawling, therefore it does not contain images taken from surveillance cameras and viewpoints.

A summary of the dataset attributes can be found in Table 2.2. As the table shows, none of these datasets can be directly used for traffic surveillance applications which are not limited to frontal viewpoint. Exemplar images from the datasets are in Figure 2.1.

Table 2.2: Overview of publicly available datasets for fine-grained vehicle classification. **View** denotes whether the dataset contains multiple viewpoints and **surv** flag defines whether the dataset contains images from surveillance cameras.

name	granularity	images	classes	view	surv
Stark et al., [146]	model	1,904	14	✓	✗
BMW-10 [80]	model year	512	10	✓	✗
Car-197 [80]	model year	16,185	196	✓	✗
FG3DCar [93]	model year	300	30	✓	✗
Liao et al. [90]	make	1482	8	✗	✓
CompCars-web [169]	model year	136,727	1,687	✓	✗
CompCars-surv [169]	model year	44,481	281	✗	✓
VMMR [152]	model year	291,752	9,170	✓	✗

(a) Car-197 [80]

(b) FG3DCar [93]

(c) CompCars – web nature [169]

(d) CompCars – surveillance nature [169]

Figure 2.1: Examples of vehicles from different datasets for fine-grained recognition of vehicles.

3

AUTOMATIC VEHICLE SPEED MEASUREMENT FROM TRAFFIC SURVEILLANCE CAMERAS

An important part of speed measurement of vehicles from a single monocular camera is calibration of the camera. This includes in a general case dealing with perspective projection, different rotations of the camera, and it is also necessary to deal with unknown distance from the camera to the ground plane of a road, and possibly with radial and tangential distortion of camera.

Generally, in Computer Vision, the task of camera calibration is a task computing intrinsic and extrinsic camera parameters [60]. These parameters are usually represented by $\mathbf{K}$ $[\mathbf{R}\ \mathbf{T}]$ matrices, where $\mathbf{K}$ denotes intrinsic parameters, and rotation and translation of the camera is represented by $\mathbf{R}$ and $\mathbf{T}$.

However, in the area of traffic surveillance, it is also usually necessary to obtain a representation of the road plane. Therefore, in the previous works and in ours as well, the vanishing points are usually used for the calibration and representation of the road plane. In a general case, the vanishing points may be ideal points (lie in infinity). An example of camera model used for traffic surveillance can be found in Figure 3.1.

With the traffic surveillance camera calibrated, the task of speed measurement becomes rather straightforward. It is necessary to track the vehicle in the image and estimate the passed distance using the calibration. The passed distance and elapsed time can directly be used for the estimation of vehicles' speed. Therefore, the review of state-of-the-art methods for speed measurement is focused primarily on the camera calibration.

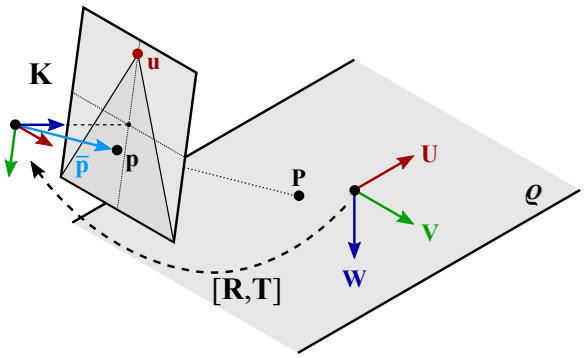


Figure 3.1: Possible camera model for traffic surveillance. Road plane is represented by $\boldsymbol{\rho}$ and vanishing points by $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$. A point on the road plane is denoted by $\mathbf{P}$ and its projection to image space by $\mathbf{p}$.

3.1 METHODS FOR CAMERA CALIBRATION AND SPEED MEASUREMENT

One important attribute of the camera calibration algorithm which should be kept in mind is that whether the algorithm works automatically in the sense that there is no manual input required per installed camera. The automaticity became more important as the number of installed cameras grows. There is a large number of papers and approaches to this problem and they will be discussed in detail in the following text.

Another important aspect of the camera calibration methods for traffic surveillance are their limitations. For example, some methods assume one vanishing point in infinity or that the camera has zero pan angle with respect to the road plane. An ideal camera calibration algorithm would be free of all these camera placement limitations.

As we will need to differentiate different vanishing points in the following text, we define the following naming convention. The **first vanishing point** is the one in the direction of vehicle movement, the direction towards **second vanishing point** is perpendicular to this direction and it is parallel to the road plane in 3D. Finally, the direction towards the **third vanishing point** is perpendicular to the road plane.

3.1.1 *Methods Based on Acquired Line Markings*

He and Yung [63] proposed a method for speed estimation of vehicles which is based on calibration using a calibration pattern formed by lane markings on the road [64]. The authors of the paper use a rectified image in further processing to deal with perspective projection. To obtain points of vehicles on the ground plane, shadows cast by rear bumpers are used. The vehicles and shadows are detected by background subtraction and binary block matching. The method achieves mean error rate 3.27 % for day and 8.51% at night. The speeds were compared with a ground truth obtained by a radar and ranges from 41 to 122 km per hour.

Cathey and Dailey [17] used a method based on detection of the vanishing point which is in the direction of vehicles movement. To obtain this vanishing point, detected line markings are used and their intersection in the least squares manner. The scale (pixels/meters ratio) for the camera is computed from average line marking stripe length and known stripe length in the real world. Finally, the authors used cross correlation to compute the number of pixels which vehicles passed between consecutive frames.

Grammatikopoulos et al. [56] use the assumption that the camera is only tilted along x-axis; thus they assume that the second vanishing point is in infinity. The first vanishing point is detected as the intersection of line markings with least

squares adjustment. The vehicles are detected by background subtraction and tracked by normalized cross-correlation. The authors require that one distance in the scene is known for the speed measurement. The reported mean error is 3 km/h.

Filipiak et al. [41] propose to use sequences of detected license plates of vehicles for finding intrinsic and extrinsic camera parameters by an evolutionary algorithm. The method was evaluated on a dataset captured by surveillance cameras with a small field of view on the road.

You et al. [172] propose to use detection of vanishing point in the direction of vehicles' movements from lane markings and vanishing point perpendicular to road plane from detected poles and pedestrians. The authors obtain the scale from known height of camera above the road or known dimensions on the road.

3.1.2 *Methods Based on Vehicles' Movement*

Dubská et al. [DSH14] published a speed measurement system using calibration method by detection of two vanishing points [DHJS15]. The first vanishing point (which is in the direction of vehicles' movement) is computed from tracked feature points on the vehicles and Hough transformation-based line-to-line mapping to obtain intersection of the lines. The second vanishing point is extracted from edges present on vehicles with the same mapping. The authors propose an algorithm for 3D bounding box computation around the vehicle blobs and measure their dimensions and match mean of these measured dimensions with the mean dimensions of vehicles for a given country. The reported mean error rate is 1.99 %.

Schoepflin et al. [135] use an activity map to obtain lane boundaries and the intersection of the boundaries treat as vanishing point in the direction of vehicle motion. The second vanishing point is detected as the intersection of lines formed by the bottom edges of the vehicles. One known length in the image is used for scale inference.

3.1.3 *Methods using Manual Measurements*

Maduro et al. [104] assume a known angle on the ground plane to calibrate the camera and use lengths of line markings' stripes to obtain the camera scale for a given scene. The authors used background subtraction to detect vehicles and Kalman filter [75] for tracking of the vehicles. The reported error rate is 2 % relative to ground truth speed obtained by GPS.

Nurhadiyatna et al. [110] used GMM background subtraction [190] for detection of vehicles and tracked them by Kalman filter. The authors use a calibrated pinhole

camera with zero pan and known distances in the real world. The mean error of their method is 7.63 km/h.

Sina et al. [145] focus on speed measurement at night. They used detected and paired headlights to detect vehicles, track them and measure their speed. The camera calibration is based on manual measurements of camera angles and distance of the camera from the ground plane. The average error is 3.3 km/h relative to ground truth obtained by GPS.

Luvizon et al. [103, 102] used a different approach and they propose to detect and track license plates in order to obtain movement of vehicles in the scene. The movement is then converted to real world passed distance by rectifying and scaling. The scale inference is based on a priori known real world measures. The reported mean error is 1.63 km/h.

Furthermore, Ke et al. [78] propose to use measurements on standard shipping container which was observed on a passing vehicle by the camera. Although it eliminates the requirement to stop the traffic, it is still necessary to use a vehicle with the container and drive it in front of the camera.

3.1.4 *Other Methods*

Dailey et al. [28] proposed a method for vehicles speed measurements based on tracking of vehicle blobs and constraining them to move along a line. The blobs are detected as inter-frame differences followed by Sobel edge detector. The authors assume that the vehicles are moving towards or from the camera and use mean length of vehicles to obtain the scale of the scene. The mean reported error is 6.5 km/h relative to manual measurements.

Do et al. [32] proposed a camera calibration method for speed measurement based on artificial markers drawn on the road. They assume that the camera has zero pan angle and that markers determining vertices of equilateral triangle with a known distance between vertices which are visible on the road. They used the triangle to obtain the scale factor and the tilt angle. The reported mean error from three scenarios with relatively low speed is 2.9 %.

Lan et al. [83] use optical flow to compute speed of different points of vehicle and the authors average this speed to get the speed of the vehicle in image units. However, to convert them into kilometers per hour, the authors assume that there is no perspective projection effect and the width of the ROI (width of lanes) is known. The reported mean error is between 0.9 % and 2.5 %.

Llorca et al. [99] propose to use two synchronized cameras, with different focal lengths focused on different parts of the road. The cameras are manually calibrated using a calibration pattern and the speed is computed as a section speed between

these cameras. The authors use detected license plates for re-identification of the vehicles.

Zheng et al. [188] use a simplified 3D model with known distances between markers on the model. The model is aligned with observed vehicle and scene geometry extracted from the distances between the landmarks. However, the method has high mean error in distance measurement in the traffic flow direction (5.14 %).

Very recently, Bhardwaj et al. [6] proposed to use detected keypoints (e.g. mirrors, lights) on the observed vehicles. The keypoints with known geometry are used as 3D points for the PnP problem [60]. The acquired calibrations are then averaged over multiple observations of different vehicles.

3.1.5 *Summary*

Summary of the presented camera calibration methods can be found in Table 3.1. As the table shows, some of the approaches have different limitations and do not work under all conditions, and the reported mean error varies greatly – it should be noted that the error is not directly comparable, as it was evaluated by the authors on different datasets and by different protocols.

To sum up the overview of camera calibration methods, some of them [28, 83] do no take perspective projection into account, some algorithms [28, 56, 110, 83, 32] have limitations about the camera placement. Quite a large number of approaches [104, 110, 145, 103, 41, 172, 78] use measurements in the scene which enable direct camera calibration. Methods [63, 32, 99] using a calibration pattern (virtual or drawn on the road) have been proposed. Another set of methods use vanishing points to obtain camera calibration [135, 17, 56, DSH14].

Several approaches to scale calibration have been proposed. Besides the multiple manual measurements on the road [104, 110, 145, 103, 41, 172] and calibration patterns [63, 32], there are two groups of methods. The algorithms from the first group [135, 17, 56, 83] use one known distance in the scene (e.g. length of line marking stripe). The other methods use dimensions of vehicles [28, DSH14] to obtain proper scale calibration.

One important attribute of the calibration methods is whether they work fully automatically and do not require any manual per camera calibration input. This helps reduce the cost of the camera installation and the automatic methods have better scaling properties. Only a small number approaches are fully automatic and do not require any manual camera calibration. Two of these methods [28, DSH14] use mean dimension of vehicles to obtain proper scaling factor for the given camera. Another, very recent, fully automatic approach was proposed by

Table 3.1: Summary of different camera calibration methods for speed measurement. It should be noted that the reported errors are only informative as all the methods are evaluated on different datasets and by different protocols. We consider a system to be automatic if it does not require any manual calibration for each individual camera. **auto** – denotes whether the system works fully automatically, **view** – denotes whether the system is usable from arbitrary viewpoint.

	camera calibration method	**auto**	**view**	**mean error**
Dailey, 2000 [28]	multiple assumptions on vehicle movements and known mean length of vehicles	✓	✗	6.5 km/h
Schoepflin, 2003 [135]	detection of two vanishing points, one known length	✗	✓	N/A
Cathey, 2005 [17]	vanishing point obtained from detected line markings, scale computed from lengths of stripes	✗	✓	N/A
Grammatik., 2005 [56]	one vanishing point obtained from detected line markings, second assumed in infinity, one known distance is required	✗	✗	3 km/h
He, 2007 [63]	calibration by pattern formed by lane markings	✗	✓	3.27 %
Maduro, 2008 [104]	known angle of the ground plane, lengths of line markings' stripes	✗	✓	2 %
Nurhadiyatna, 2013 [110]	known distances in the real world and in the scene, zero pan assumption	✗	✗	7.63 km/h
Sina, 2013 [145]	manual measurements	✗	✓	3.3 km/h
Dubská, 2014 [DSH14]	detection of two vanishing points, scale computed by matching of statistics of vehicles' dimensions to mean dimensions of vehicles	✓	✓	1.99 %
Lan, 2014 [83]	relaxation of perspective projection, known width of lanes	✗	✗	0.9 % – 2.5 %
Do, 2015 [32]	zero pan assumption, equilateral triangle drawn on the road	✗	✗	2.91 %
Filipiak, 2016 [41]	constant speed assumption, evolutionary algorithm to recover intrinsic and extrinsic parameters from detected license plate sequences	✓	✗	2.3 km/h
Llorca, 2016 [99]	two synchronized cameras with speed section measurement based on license plate detection	✗	✓	1.44 km/h
You, 2016 [172]	first vanishing point from lane markings and third vanishing point from poles and pedestrians, known height of camera installation	✗	✓	N/A
Zheng, 2016 [188]	alignment of simplified 3D model to vehicles	✓	✓	5.14 %
Bhardwaj, 2017 [6]	automatically detected keypoints on vehicles	✓	✓	12 %
Ke, 2017 [78]	measurements on standard shipping container	✗	✓	N/A
Luvizon, 2017 [103, 102]	known real world measures	✗	✓	1.63 km/h

Table 3.2: Summary of datasets used for evaluation of visual speed measurement methods.

dataset	videos	vehicles	source of GT	resolution	evaluation metrics
Dailey, 2000 [28]	1	532	induction loops	N/A	speed measurement error
Schoepflin, 2003 [135]	2	1 015	induction loops	320 × 240	speed measurement error
Grammatik., 2005 [56]	1	20	manual measurements	768 × 576	speed measurement error
He and Yung, 2007 [63]	1	64	RADAR	1280 × 1024	speed measurement error
Maduro, 2008 [104]	2	few	GPS	N/A	speed measurement error
Nurhadiyatna, 2013 [110]	10	15	GPS	320 × 240	speed measurement error
Sina, 2013 [145]	13	13	GPS	N/A	speed measurement error, vehicle counting
Dubská, 2014 [DSH14]	6	29	GPS	864 × 480	speed measurement error, distance measurement error
Lan, 2014 [83]	1	2 010	RADAR	640 × 480	speed measurement error
Luvizon, 2014 [103]	1	75	induction loops	768 × 480	speed measurement error, license plate detection
Do, 2015 [32]	1	3	speedometer	N/A	speed measurement error
Filipiak, 2016 [41]	2	955	induction loops	1280 × 720	speed measurement error

Bhardwaj et al. [6], and it is based on the geometry of the detected keypoints on the observed vehicles.

Another important attribute is whether the camera can be placed in any position above the road, as some papers require for example that the camera has zero pan. In real world scenarios, this can be hard to guarantee if the camera will not be placed on a portal above the road.

Thus we want to take an approach for traffic camera calibration which will be fully automatic and it will not have any placement constraints. There are only two papers [DSH14, 6] which satisfy both these conditions and we want to build on the approach proposed by Dubská et al. [DSH14] and extend the camera and scale calibration method.

3.2 EVALUATION DATASETS USED IN EXISTING WORKS

The described methods usually used different methods for evaluation of the speed measurement and ground truth speed acquisition. Some methods [28, 135, 103, 41] use inductive loops for ground truth acquisition, other methods [104, DSH14] GPS or RADAR [83]. Do et al. [32] used the speedometer on a motorbike, which should be considered very imprecise.

When it comes to the number of evaluated speed measurements, Lan et al. [83] used 2 010 ground truth speeds (only one video sequence), others [28, 135, 41] have hundreds of vehicles with known ground truth. And there are also works [56, 63, 104, 110, 145, DSH14, 83, 103, 32] that use at most tens of ground truth speeds with the lowest number in [32] (one ground truth speed) and the highest number of 75 measurements in [103]. Cathey at al. [17] have no evaluation at all. A summary of existing datasets can be found in Table 3.2. It should be noted that with the exception of [110, DSH14], the datasets are not publicly available which makes comparison of the methods impossible.

Almost every mentioned dataset (except [145] and a part of [63]) is recorded in daylight as the methods usually become unusable at night when only headlights of vehicles are visible. Existing datasets usually evaluate only the speed measurement error (with different statistics – mean, deviation etc.) and some exceptions (see Table 3.2) evaluate also other tasks.

The existing evaluation of algorithms should be considered insufficient as existing works use a relatively low number of observed vehicles and scenes. Also, for GPS and speedometer, the ground truth is imprecise as in our evaluation, GPS has mean error over 2 % and the speedometer reports a higher speed then the actual. Therefore, we created our novel dataset [SJŠ+18] with precise ground truth and 20 865 of vehicles with ground truth speed. It is also possible to evaluate other camera calibration aspects such as calibration error and distance measurement on the road plane with the computed scale. These two metrics can provide interesting insights into properties of camera calibration algorithms as they are needed and harnessed in the intelligent transportation surveillance.

Part II

FINE-GRAINED RECOGNITION OF VEHICLES

This part of the dissertation thesis contains re-formatted copies of my two papers dealing with fine-grained recognition of vehicles. The first paper is a CVPR conference paper which introduces the idea of using 3D bounding boxes of vehicles to improve the fine-grained recognition. As experimental results in the paper show, exploiting the 3D bounding boxes improves classification and verification accuracy.

In the second paper, which is published in IEEE Transactions on Ingelligent Transportation Systems (IF: 3.724), we explore the 3D bounding boxes for fine-grained classification further. Also, a method for estimation of the 3D bounding box is provided and evaluated. The experimental evaluation is significantly extended and as the results show, using the 3D bounding boxes improves classification accuracy consistently with different Convolutional Neural Networks and we also outperform other state-of-the-art methods for fine-grained recognition.

4

BOXCARS: 3D BOXES AS CNN INPUT FOR IMPROVED FINE-GRAINED VEHICLE RECOGNITION

CITATION SOCHOR Jakub, HEROUT Adam a HAVEL Jiří. 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, s. 3006-3015. ISBN 978-1-4673-8851-1. ISSN 1063-6919.

ABSTRACT We are dealing with the problem of fine-grained vehicle make & model recognition and verification. Our contribution is showing that extracting additional data from the video stream – besides the vehicle image itself – and feeding it into the deep convolutional neural network boosts the recognition performance considerably. This additional information includes: 3D vehicle bounding box used for "unpacking" the vehicle image, its rasterized low-resolution shape, and information about the 3D vehicle orientation. Experiments show that adding such information decreases classification error by 26 % (the accuracy is improved from 0.772 to 0.832) and boosts verification average precision by 208 % (0.378 to 0.785) compared to baseline pure CNN without any input modifications. Also, the pure baseline CNN outperforms the recent state of the art solution by 0.081. We provide an annotated set "BoxCars" of surveillance vehicle images augmented by various automatically extracted auxiliary information. Our approach and the dataset can considerably improve the performance of traffic surveillance systems.

4.1 INTRODUCTION

We are developing a system for traffic surveillance from roadside cameras. It is meant to be fully automatic (not requiring manual per-camera configuration) and tolerant to sub-optimal camera placement (the cameras will not be placed above the lanes, but on the road side, wherever it is naturally possible).

One important component of such a system is recognition of vehicle make & model – as accurate as possible. This fine-grained recognition serves multiple purposes. Besides obvious collection of statistics and demographic information and verification of license plate authenticity, recognition of dominant and characteristic types can establish a highly accurate scale calibration of the camera, much

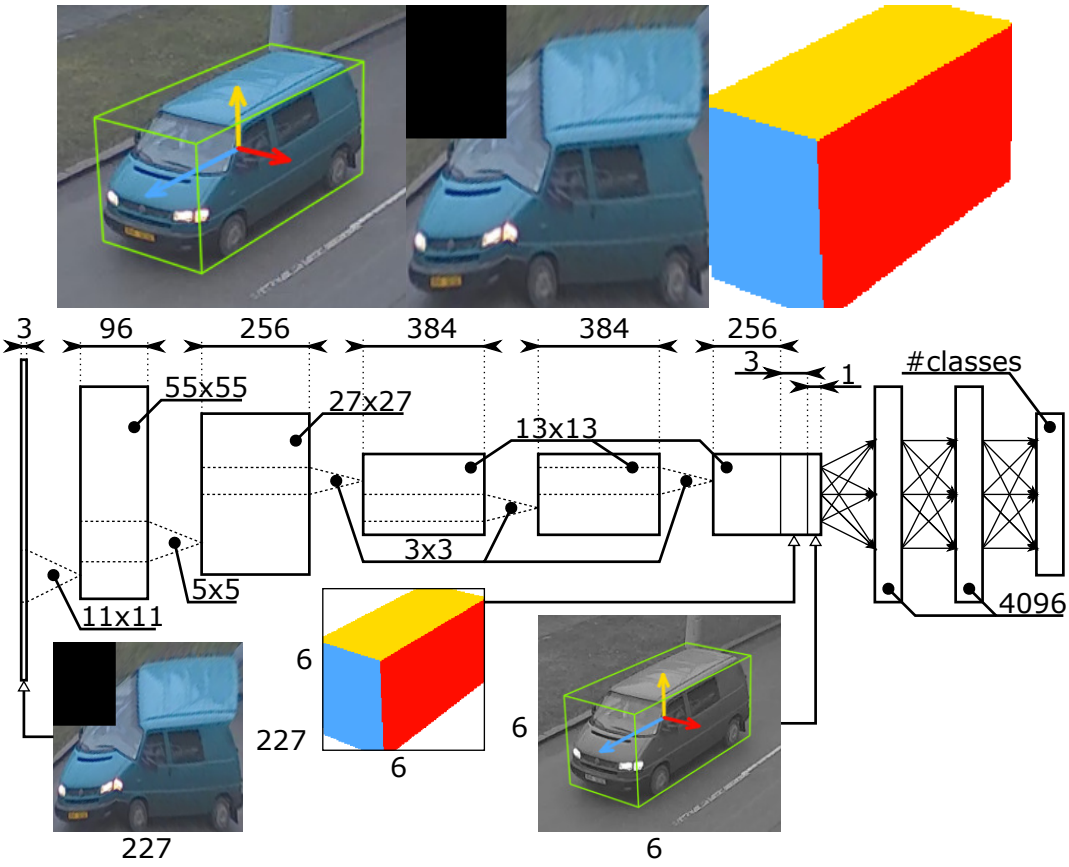


Figure 4.1: We take advantage of the surveillance camera being fixed, use its automatically obtained calibration to *unpack* the input image so that it is better aligned, and we add additional inputs to the CNN. These modified inputs boost the performance of vehicle recognition and especially vehicle make & model verification.

more precise than a statistic of undifferentiated cars [DSH14]. The system should also be able to adapt to new models of cars on its own. It should therefore not only recognize the pre-trained set of models, but also *verify* whether two given vehicle samples are of the same make & model or not – without previously seeing these particular vehicle types.

Fine-grained vehicle recognition has been receiving increased research attention recently. Many works and datasets focus on recognition of "web images" shot from a limited set of viewpoints, typically from eye-level views [169, 68, 80, 58, 93, 146]. Some works also deal with data of surveillance nature [169, 90, 61].

Our work goes beyond a recent work by Yang et al. [169]. They collected presumably the first dataset of sufficient proportion for training convolutional neural networks (the surveillance-nature portion of their dataset contains around 50k samples). They also propose a CNN architecture for fine-grained vehicle recognition and publish benchmarking results.

Since we aim at a fixed-camera surveillance system, we take advantage of fully automatic camera calibration including scale [DSH14] and we use the automatically extracted information for improving the recognition system (Fig. 4.1). The

automatically calibrated camera allows us to extract a 3D bounding box of the passing vehicle. The system then "unpacks" the bounding box to get a better aligned image representation. The shape and location of the bounding box is also input to the CNN and helps it to reference the relevant information. Lastly, the view direction extracted for each vehicle sample is also encoded and input to the fully connected CNN layers, further boosting the performance. The whole algorithm is designed to work with low-resolution vehicle images taken from arbitrary viewpoints of surveillance nature (frontal, sideways, varying elevation, etc.).

We collected a dataset *BoxCars* from a network of surveillance cameras and we make it publicly available for training and benchmarking. The cameras are automatically calibrated and the vehicle samples are automatically augmented by the 3D bounding box information. This information is easily obtainable in real time and it can be a part of any surveillance system.

The experiments show that the proposed enhanced information boosts the average precision of vehicle recognition considerably (0.772 to 0.832 for medium difficulty, 0.733 to 0.804 for hard cases). The same modification helps even much more for the *vehicle type verification* task: given observations of two vehicles, tell if they are of the same type (in the fine-grained sense, i.e. including make, model, year). The particular vehicle types have not been necessarily seen by the classifier during training. The improvement in this task was from 0.378 to 0.785 for medium difficulty samples and 0.353 to 0.710 for difficult cases. This verification task is important for growing the set of vehicles recognizable by the system in an unsupervised manner – without collecting and annotating the samples in advance.

The **contributions of this paper** are the following: **i)** We show that additional information easily obtainable in real time for static surveillance cameras can boost the CNN verification performance greatly (by 208 %), **ii)** The vehicle fine-grained classification error was decreased by 26 %, **iii)** We collected a dataset of vehicle samples accompanied with the 3D bounding boxes (*BoxCars*, 21,250 samples, 63,750 images, 27 different makes, 148 make & model + submodel + model year classes).

4.2 RELATED WORK

When it comes to fine-grained vehicle classification, many approaches are limited to frontal or rear viewpoint and they are based on detection of the license plate for ROI extraction [119, 31, 116, 109, 175, 4]. Authors of these papers are using different schemes for extracting the feature vectors and for the classification itself. Stark et al. [146] use fine-grained categorization of cars by DPM in order to obtain metric information and get a rough estimate of depth information for single images (containing cars in usable poses). Another approach proposed by Prokaj and Medioni

[121] is based on pose estimation and it is able to handle any viewpoint. The authors suggest to use 3D models of vehicles, fit them to the recognized pose, project them to 2D and use SIFT-like features for the comparison of the vehicles. Krause et al. [80] used 3D CAD models to train geometry classifiers and improve results of 3D versions of Spatial Pyramid and BubbleBank [30] by 3D patch sampling and rectification. Lin et al. [93] proposed to use 3D Active Shape Model fitting to obtain positions of landmarks and achieved much better results than other methods on their own dataset FG3DCar. Authors of [79] propose to learn discriminative parts of vehicles with CNN and use the parts for fine-grained classification. Gu et al. [58] used pose estimation and active shape matching to deal with pose variation and normalization. Hsiao et al. [67] use 3D chamfer matching of backprojected curves on an automatically generated visual hull of the vehicle. However, the authors assume to have shots of vehicles against clean background and that the shots are taken under regular intervals.

Very recent work by He et al. [61] focuses on surveillance images; however, the authors assume to have high-resolution frontal image of the vehicle to correctly detect license plate and other artificial anchors. Liao et al. [90] used Strongly Supervised DPM (SSDPM) to categorize frontal images of vehicles and classification based on discriminative power of different parts of SSDPM. Hsieh et al. [68] proposed a new symmetrical SURF keypoint detector to detect and segment frontal vehicle images into several grids for fine-grained classification. Very recent work by Yang et al. [169] proposed to use Convolutional Neural Networks for fine-grained classification, regression of parameters etc. Krause et al. [81] proposed to use co-segmentation and automatic part localization in combination with R-CNN to overcome missing parts annotations.

Recently, Deep Convolutional Neural Networks (CNN) consistently succeed in hard image recognition tasks such as the ImageNet [130] contest. After the network by Krizhevsky et al. [82], deeper and more complex CNNs such as the GoogLeNet by Szegedy et al. [151] seem to be consistently winning the contest. Authors also used input normalization to improve performance of CNN [153] and adding additional training data to CNN [82]. Parts of the CNN can be viewed as feature extractors and independently reused. These trained feature extractors outperform the hand-crafted features [8, 153]. Recently, a relatively large number of authors proposed to use Deep Neural Networks for fine-grained classification in general [153, 163, 169, 91, 166, 105, 92].

To sum up, in most cases, the existing approaches either use 2D frontal images, or 3D CAD models to allow viewpoint invariance. We propose to extract and use 3D information based on video data from the surveillance camera at general view-

points. This information is fed to a CNN as additional input, leading to better car classification and especially type verification.

4.3 FINE-GRAINED VEHICLE CLASSIFICATION AND VERIFICATION METHODOLOGY

In agreement with the recent progress in the Convolutional Neural Networks [153, 82, 22], we propose to use CNN for both classification and verification. The classification task will be done directly by the net and for the verification task, we use features (activations) extracted from the last-but-one layer and cosine distance. We enhance the input of the net by several techniques using automatically extracted 3D bounding boxes [DSH14]. We focus on vehicle images obtained from surveillance cameras where the automatic extraction of 3D bounding boxes is possible cheaply in real time. We used BVLC Reference CaffeNet [82] pretrained on ImageNet [130] and then fine-tuned on our dataset as a baseline from which we improve.

4.3.1 *Unpacking the Vehicles' Images*

We based our work on 3D bounding boxes [DSH14] (Fig. 4.2) which can be automatically obtained for each vehicle seen by a surveillance camera (see our original paper [DSH14] for further details). These boxes allow us to identify side, roof, and front/rear side of vehicles in addition to other information about the vehicles. We use these localized segments to normalize the image of observed vehicles.

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. 4.3) and they are organized into the final matrix **U**:

$$\mathbf{U} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{F} & \mathbf{S} \end{pmatrix} \tag{4.1}$$

The unpacking itself is done by obtaining homography between points b_i (Fig. 4.3) and perspective warping parts of the original image. The left top submatrix is filled with zeros. This unpacked version of the vehicle can be 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 all that without the necessity to use DPM or other algorithms for part localization.

Figure 4.2: Examples of vehicles (top), their 3D bounding boxes (middle) and unpacked version of the vehicles (bottom).

4.3.2 *Viewpoint Encoding*

We also found out that it improves the results when the net is aware of the viewpoint of the vehicles. The viewpoint is extracted from the orientation of the 3D bounding box – Fig. 4.4. 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 $v_i = \overrightarrow{C_b C_i}$. Point C_b 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. The vectors are normalized to have 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.

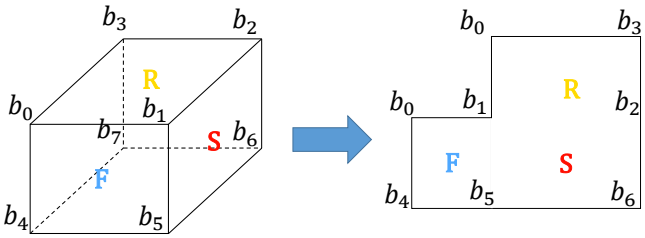


Figure 4.3: Unpacking the input vehicle image based on its bouding box. Points b_i are vertices of the 3D bounding box [DSH14].

Figure 4.4: Examples of vectors encoding the viewpoint.

4.3.3 *Rasterized Bounding Boxes*

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 three-channel image containing each visible face rasterized in a different channel. Formally, point (x, y) of the rasterized bounding box $\mathbf{T}$ is obtained as

$$\mathbf{T}_{x,y} = \begin{cases} (1,0,0) & (x,y) \in \square b_0 b_1 b_4 b_5 \\ (0,1,0) & (x,y) \in \square b_1 b_2 b_5 b_6 \\ (0,0,1) & (x,y) \in \square b_0 b_1 b_2 b_3 \\ (0,0,0) & \text{otherwise} \end{cases} \tag{4.2}$$

where $\square b_0 b_1 b_4 b_5$ denotes the quadrilateral defined by points b_0, b_1, b_4 and b_5 in Figure 4.3.

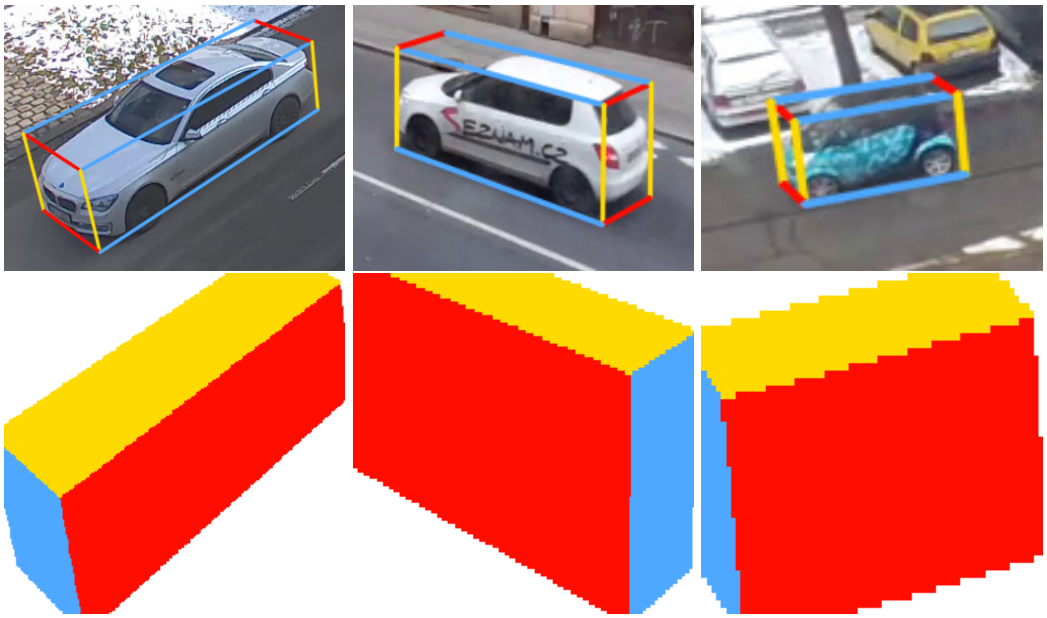


Figure 4.5: Examples of rasterized bounding boxes for CNN (colors are R,G,B in the actual computation, but are changed here for aesthetic reasons).

Finally, the 3D rasterized bounding box is cropped by the 2D bounding box of the vehicle. For an example, see Figure 4.5, showing rasterized bounding boxes for different vehicles taken from different viewpoints.

4.3.4 *Final CNN Using Images + Auxiliary Input*

All this information is finally passed to the CNN (Fig. 4.1). The unpacked version of vehicles is used directly as the image input instead of the original image. The rasterized bounding box and encoded viewpoints are added to the net after the convolutional layers. We experimented with changing the layer where the information is added but different positions did not improve the results further and the mentioned setting is easiest with regard to pre-training the network.

As the auxiliary input is added after the convolutional layers, it needs to be passed in 6×6 matrices. The rasterized bounding box is rescaled (Lanczos interpolation) to 6×6 and added to the net. The encoded viewpoints are added to the net in 6×6 one-channel matrix with zeros everywhere except for the first row which contains normalized vectors encoding the viewpoint. The first row $\mathbf{t}$ of this matrix contains all three 2D vectors: $\mathbf{t} = (v_f^x, v_f^y, v_r^x, v_r^y, v_s^x, v_s^y)$.

For better understanding of the text, we define labels for the nets with different input modifications. The original CNN processing cropped images of vehicles without any modifications is referenced as **baseline**. Network denoted as **Rast** contains the rasterized bounding boxes, **View** net is augmented by the encoded viewpoints, and in **Unp** version of the net, the original image is replaced by the

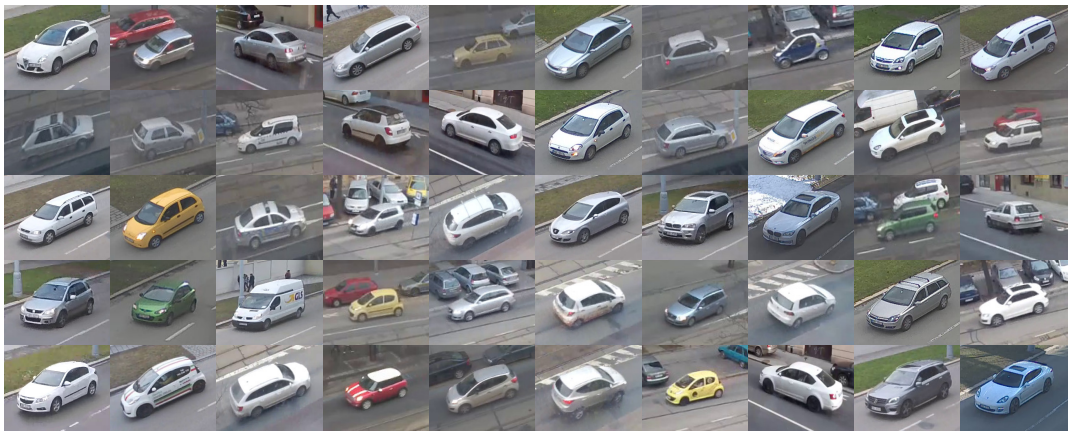


Figure 4.6: A sample of the novel *BoxCars* dataset. In total, it captures 21,250 vehicles in 63,750 images, from 27 different makes (148 fine-grained classes).

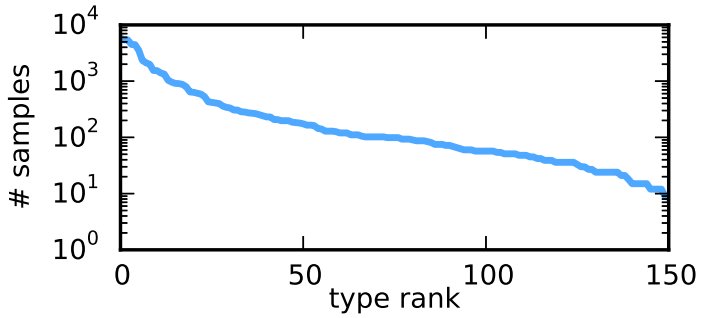


Figure 4.7: Distribution of samples in our *BoxCars* dataset across vehicle types. The distribution corresponds to real-life occurence of the models.

unpacked image of vehicles. All these input modifications can be combined, yielding **RastView**, **RastUnp** and **RastViewUnp** nets.

4.4 BOXCARS: NEW DATASET FOR SURVEILLANCE VEHICLE VERIFICATION

There is a large number of datasets of vehicles [130, 1, 113, 36, 162, 16, 111, 87, 53, 134, 51, 112, 106] which are usable mainly for vehicle detection, pose estimation, and other tasks. However, these datasets do not contain annotation of the precise vehicles' make & model.

When it comes to the fine-grained datasets, a few of them exist and all are quite recent. Lin et al. [93] published FG3DCar dataset (300 images, 30 classes), Stark et al. [146] made another dataset containing 1,904 vehicles from 14 classes. Krause et al. [80] published two datasets; one of them, called *Car Types*, contains 16k of images and 196 classes. The other one, *BMW 10*, is made of 10 models of BMW vehicles and 500 images. Finally, Liao et al. [90] created a dataset of 1,482 vehicles from 8 classes. All these datasets are relatively small for training the CNN for real-world surveillance tasks.

Yang et al. [169] published a large dataset *CompCars* this year (2015). The dataset consists of a web-nature part, made of 136k of vehicles from 1,600 classes taken from different viewpoints. Then, it also contains a surveillance-nature part with 50k frontal images of vehicles taken from surveillance cameras.

We collected and annotated a new dataset *BoxCars*. 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. 4.6). We used surveillance cameras mounted near streets and tracked the passing vehicles. Each correctly detected vehicle is captured in 3 images, as it is passing by the camera; therefore, we have more visual information about each vehicle. The dataset contains 21,250 vehicles (63,750 images) of 27 different makes. The vehicles are divided into classes: there are 102 make & model classes, 126 make & model + submodel classes, and 148 make & model + submodel + model year classes. The distribution of types in the dataset is shown in Figure 4.7 and samples from the dataset are in Figure 4.6. The data include information about the 3D bounding box [DSH14] for each vehicle and an image with a foreground mask extracted by background subtraction [147, 190]. The dataset is made publicly available[1] for future reference and evaluation.

Our proposed dataset is difficult in comparison with other existing datasets in size of the images (thousands of pixels, min/mean/max): *CompCars* – 107/503/1114, *Cars-196* – 4/479/42120, *BoxCars* – 8/39/253. Also, the samples in our dataset are compressed by realistic h264 codec settings, and unlike most existing surveillance datasets, our viewpoints are diverse and not just frontal/rear.

4.5 EXPERIMENTAL RESULTS

The evaluation of the improvement caused by our modifications of the CNN input can be only done on our *BoxCars* dataset as other fine-grained datasets listed in Section 4.4 do not include the information about the bounding boxes. However, to put the performance of the system into context with other published methods, we evaluated the BVLC Reference net [82] on the most recent dataset *CompCars* and the improvement will be measured relatively to this baseline.

4.5.1 *Evaluation on the CompCars Dataset*

We trained the baseline net on the *CompCars* dataset [169] and evaluated its accuracy. As Table 4.1 shows, this net significantly outperforms the CNN used by Yang

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

Table 4.1: Comparison of **classification** results on the *CompCars* [169] dataset (accuracy).

	Top-1	Top-5
[169]	0.767	0.917
Ours	**0.848**	**0.954**

Table 4.2: Comparison of **verification** results on the *CompCars* dataset (accuracy).

	Easy	Medium	Hard
[169]	0.833	0.824	0.761
Ours	**0.850**	**0.827**	**0.768**

et al. [169] in their paper in both Top-1 and Top-5 categories. We used the All-view split as the authors achieved the best results if they did not take the viewpoint into account, but instead, they trained a common net on all the viewpoints at once.

We also evaluated the make & model verification accuracy using the activations extracted from the baseline net and cosine distance. The results are shown in Table 4.2 and our system outperforms the original paper in verification as well. It should be noted that the *CompCars* verification dataset has a high random baseline (0.5).

Since the baseline net outperforms the method published by Yang et al. [169], we measure the improvement achieved by our modifications of the CNN input relatively to the performance of this baseline net on our *BoxCars* dataset.

4.5.2 *Classification Results*

We defined two ways of splitting the *BoxCars* dataset into the training and testing parts. In both of them, *medium* and *hard*, vehicles taken from 70 % of cameras are included in the training part and the vehicles taken by the rest of the cameras are in the test set. The difference of viewpoints between the training and the test sets is not too large, as it would be if for example the rear views would be in the training set and frontal views in the test set. This kind of splitting is suitable for benchmarking surveillance algorithms because real-life applications would also use cameras placed in roughly predictable viewpoints. The difference between the *medium* and *hard* splittings is that we consider vehicles of the same make+model+submodel but differing in their model year as the same types in the *medium* dataset. In the *hard* dataset, we differentiate also the model year. For stability of the classification,

Table 4.3: Training and testing set sizes for the classifications task. Numbers of samples represent the amount of all images used. The number of unique vehicles is one third of these counts.

	# types	# training samples	# test samples
medium	77	40,152	19,590
hard	87	37,689	18,939

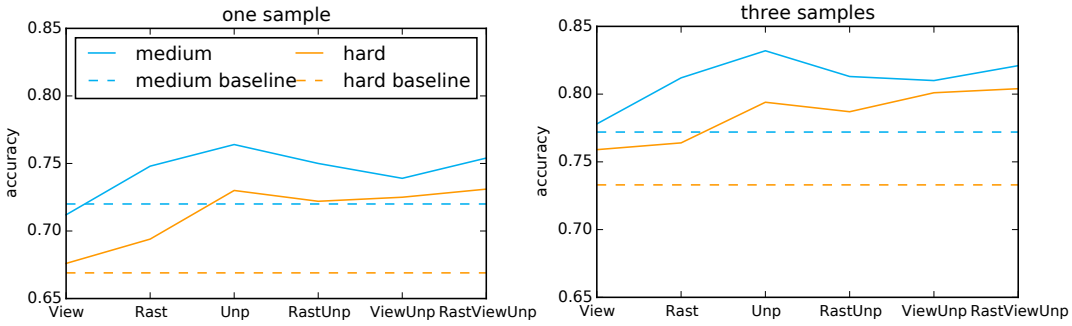


Figure 4.8: Top-1 accuracies for different input modifications. The dashed line represent the baseline accuracy achieved by the CNN without any input modification. The method identifiers are explained in Sec. 4.3.4.

types with too few samples were omitted and the training/testing set sizes can be found in Table 4.3 (this approach is consistent with [169]).

All the CNNs were pre-trained on ImageNet [130] and then fine-tuned on one of the *medium* or *hard* datasets. When the rasterized bounding boxes or encoded viewpoints are introduced to the nets, the weights of fully connected layers are randomly re-initialized and in that case we do not use the pre-trained weights on ImageNet in those layers.

We evaluated all the net's input modifications and also their combinations. The results are shown in Table 4.4 and Figure 4.8. As we have multiple samples for each vehicle, we can use mean probability for each vehicle type and achieve better results, see Table 4.5. The improvement between one sample and three samples is 0.073 (0.731 to 0.804 in Top-1 accuracy). Also, Table 4.5 shows that the improvement achieved by the modified CNN in the *medium* dataset is 0.060 (0.772 to 0.832) and 0.071 (0.733 to 0.804) in the *hard* case.

We consider the improvement in classification accuracy as interesting because the task itself is complex and difficult even for a human. Also, the classification error was reduced by 26 % for both splittings. Consider the examples of the most confused types shown in Figure 4.9, where *Volkswagen Up* and *Skoda Citigo* are manufactured in the same production plant and they differ only in subtle branding parts in the region of the frontal mask. Also, Figure 4.10 shows examples of probabilities obtained for different vehicles. These graphs indicate that the net is aware

Table 4.4: Comparison of classification accuracy results for one sample per vehicle. The method identifiers are explained in Sec. 4.3.4.

	baseline		Unp		Rast		View		RastUnp		ViewUnp		RastViewUnp	
	med	hard	med	hard	med	hard	med	hard	med	hard	med	hard	med	hard
Top-1	0.720	0.669	**0.764**	0.730	0.748	0.694	0.712	0.676	0.750	0.722	0.739	0.725	0.754	**0.731**
Top-5	0.910	0.883	**0.915**	**0.897**	0.903	0.872	0.885	0.865	0.891	0.882	0.894	0.883	0.901	0.890

Table 4.5: Comparison of classification accuracy results for three samples per vehicle, the final probability for a class is obtained as mean probability over the samples.

	baseline		Unp		Rast		View		RastUnp		ViewUnp		RastViewUnp	
	med	hard	med	hard	med	hard	med	hard	med	hard	med	hard	med	hard
Top-1	0.772	0.733	**0.832**	0.794	0.812	0.764	0.778	0.759	0.813	0.787	0.810	0.801	0.821	**0.804**
Top-5	0.924	0.903	**0.945**	0.926	0.927	0.906	0.913	0.901	0.928	0.923	0.930	0.919	0.937	**0.929**

of the sample being similar to multiple types and that it can safely distinguish from completely disparate models.

The experimental results indicate that the most important improvement is unpacking the image (Section 4.3.1), presumably because it leads to better alignment of the vehicle features on the input CNN level. The further input modifications help only in the *hard* splitting, where subtler details make a difference.

4.5.3 *Verification Results*

Verification of pairs of vehicle types (for two vehicle samples decide: same types / different types) is as important as classification, especially when it comes to reasoning about unseen and untrained vehicle types. We selected even more difficult splitting for evaluation of the verification performance. Only some cameras are present in the training set (the same ones as in the classification task) and only some vehicle types are present in the training set. The testing is done on pairs of randomly selected 3,000 vehicles mainly taken from cameras which were not present during training (over 80 % of vehicles is from unseen cameras) and the testing set of vehicles also contains types which were not seen during training (over 10 % of samples, approximately 25 % of pairs contain at least one vehicle of an unseen type). Thus, the algorithm is required to verify unseen types of vehicles taken from unseen viewpoints and it has to generalize well.

We have three splittings for the verification task. *Easy* contains pairs of vehicles from the same unseen camera, *medium* contains pairs from different unseen cameras and finally, *hard* contains pairs of vehicles from different unseen cameras and

Figure 4.9: Most misclassified vehicle types for the RastViewUnp version of the net. **left:** Volkswagen Up and Skoda Citigo, **middle:** Volkswagen Caddy and Citroen Berlingo, **right:** Kia Ceed and Renault Megane.

Table 4.6: Training and test sets sizes for the verification task. The number of training samples represent the number of images used in training. The number of unique training vehicles is one third of this number.

	training		testing	
	# types	# samples	# types	# pairs
easy	113	34 929	100	1 394 008
medium	113	34 929	99	1 435 532
hard	126	32 658	113	1 501 156

the model year is also taken into account. The training/testing set sizes can be found in Table 4.6.

Again, we used nets pre-trained on ImageNet [130], fine-tuned them during the training and then used the features from the last fully connected layer (fc7) and compared them by cosine distance. Two different training passes were done, one for the *easy* and *medium* splitting (both splittings do not take model year into account) and one for the *hard* one.

We evaluated the algorithm on the three dataset splittings using only one sample for each vehicle and the results are in Figure 4.11. When the three samples are taken into account by working with the median cosine distance, the results improve as shown in Figure 4.12. Using the median cosine distance improved the average precision on average by 0.094.

Figure 4.10: Examples of types probabilities for different vehicles for the RastViewUnp version of the net. Only one sample was used for the classification and the model year was differentied in the first example.

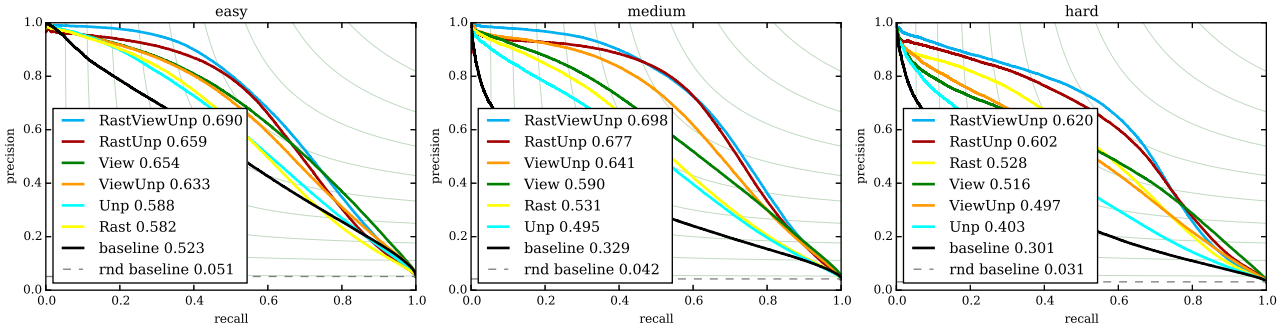


Figure 4.11: Precision-Recall curves for different verification dataset splittings. Only one sample was used for the verification. Numbers denote Average Precision. The method identifiers are explained in Sec. 4.3.4; *rnd baseline* denotes random baseline based on the number of positive pairs. Numbers in legend denote Average Precision.

The plots show that our CNN input modifications have a huge impact on the average precision in the verification task. For example, considering the *medium* set and the median cosine distance over the three samples, *RastViewUnp* improved AP of the baseline CNN by 208 %. Figure 4.13 shows what improved the average precision in verification. The numbers gradually increase as we add more and more modifications of the CNN input. It is rather interesting that both rasterized bounding boxes and orientation encoding help the net, even in combination; we expected that these two would be alternative ways of encoding the same information, but apparently, they encode it slightly differently and both turn out to be helpful.

We also obtained vehicle type verification precision and recall of human subjects for the *BoxCars* dataset. We randomly selected 1,000 pairs of vehicles; one third of the pairs included vehicles of different types, one third of pairs had the same

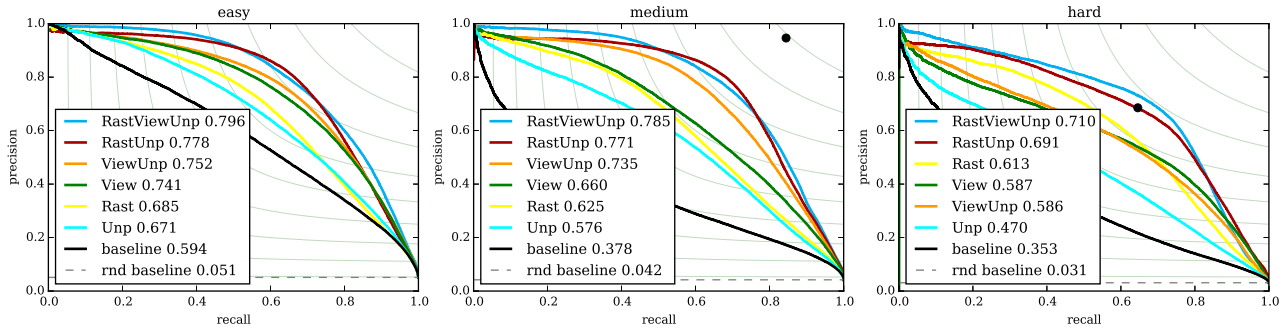


Figure 4.12: Precision-Recall curves for different verification dataset splittings. Median cosine distance over three vehicle samples was used in this case. **Black dots** represent mean precision and recall obtained by human annotators, see text for details. Numbers in legend denote Average Precision.

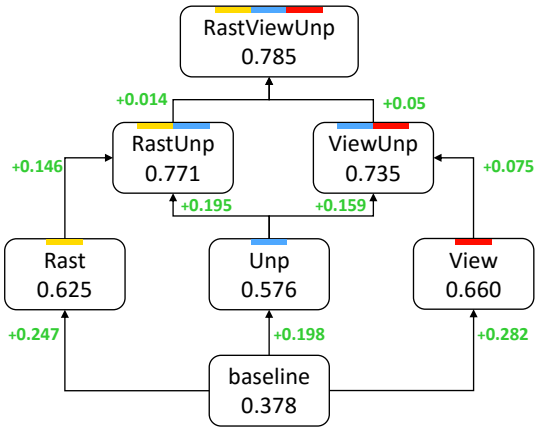


Figure 4.13: Schematic image of improvements in the verification AP for different CNN input modifications. The image is based on the *medium* splitting with median distance over the samples.

make & model + submodel, but differed in the model year, and the last third contained pairs of vehicles of the same type including model year. Participants were requested to manually indicate one of these three situations for each given pair of vehicles. All three captured images of the vehicles, taken by different cameras, were shown to the participants (that is why the human data is present in *medium* and *hard* cases in Fig. 4.12 but not in Fig. 4.11). We received a total of 8,011 inputs (8 per pair) with mean precision 0.946 and mean recall 0.844 for the *medium* case. On the other hand, the results show that the human annotators have problems with correctly distinguishing different model years (the *hard* case), with mean precision 0.685 and mean recall 0.646. These results are shown in Figure 4.12 as black dots; note that in the *hard* case, the system outperforms the human annotators.

4.6 CONCLUSIONS

Surveillance systems can and should benefit from the camera being fixed. The camera can be fully automatically calibrated and more information can be extracted for the passing vehicles. We show that this information considerably improves the fine-grained recognition by CNN, and tremendously boosts the verification task average precision.

Our dataset *BoxCars* is meant to help experiments in this direction by providing sufficient amount of data enriched by information which can be automatically extracted in real time in actual surveillance systems. We keep collecting samples from new surveillance cameras, so that the size of the dataset will gradually increase in near future.

5

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

CITATION 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.

ABSTRACT In this paper, we focus on fine-grained recognition of vehicles mainly in traffic surveillance applications. We propose an approach that is orthogonal to recent advancements in fine-grained recognition (automatic part discovery, bilinear pooling). Also, in contrast to other methods focused on fine-grained recognition of vehicles, we do not limit ourselves to a frontal/rear viewpoint, but allow the vehicles to be seen from any viewpoint. Our approach is based on 3D bounding boxes built around the vehicles. The bounding box can be automatically constructed from traffic surveillance data. For scenarios where it is not possible to use precise construction, we propose a method for an estimation of the 3D bounding box. The 3D bounding box is used to normalize the image viewpoint by "unpacking" the image into a plane. We also propose to randomly alter the color of the image and add a rectangle with random noise to a random position in the image during the training of Convolutional Neural Networks. We have collected a large fine-grained vehicle dataset BoxCars116k, with 116k images of vehicles from various viewpoints taken by numerous surveillance cameras. We performed a number of experiments which show that our proposed method significantly improves CNN classification accuracy (the accuracy is increased by up to 12 percentage points and the error is reduced by up to 50 % compared to CNNs without the proposed modifications). We also show that our method outperforms state-of-the-art methods for fine-grained recognition.

5.1 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

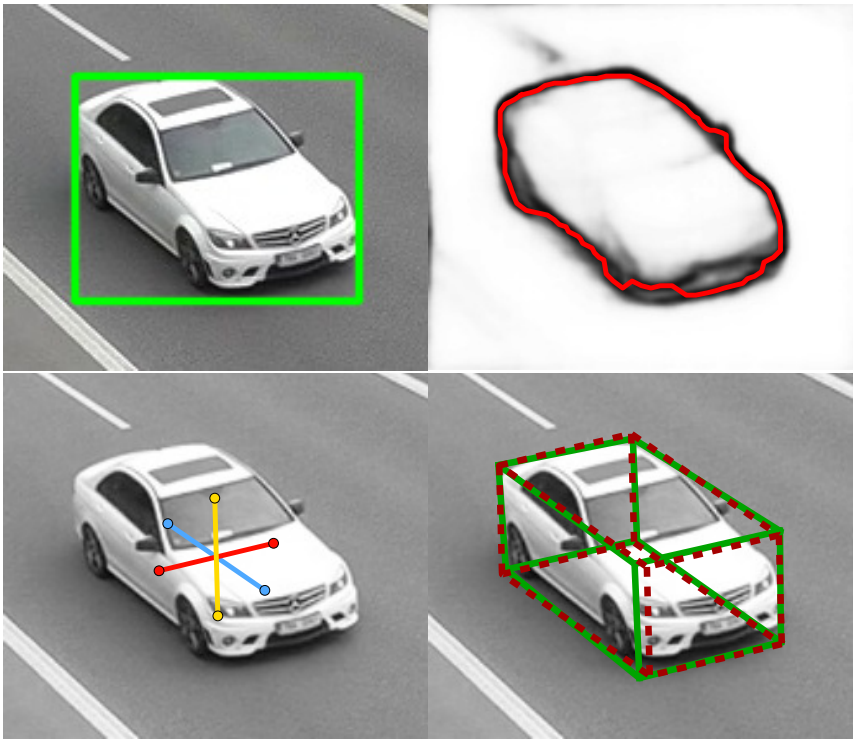


Figure 5.1: Example of automatically obtained 3D bounding box used for fine-grained vehicle classification. **Top left:** vehicle with 2D bounding box annotation, **top right:** estimated contour, **bottom left:** estimated directions to vanishing points, **bottom right:** 3D bounding box automatically obtained from surveillance video (green) and our estimated 3D bounding box (red).

et al. [50] 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.

Methods focused on the fine-grained recognition of vehicles usually have some limitations – they can be limited to frontal/rear viewpoints or use 3D CAD models of all the vehicles. Both these limitations are rather impractical for large scale deployment. There are also methods for fine-grained recognition in general which were applied on vehicles. The methods recently follow several main directions – automatic discovery of parts [81, 142], bilinear pooling [92, 47], or exploiting struc-

ture of fine-grained labels [166, 189]. Our method is not limited to any particular viewpoint and it does not require 3D models of vehicles at all.

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 5.4 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 5.1). Our input modifications are able to significantly increase the classification accuracy (up to **12 percentage points**, classification error is reduced by up to **50 %**).

The key contributions of the paper are:

- Complex and thorough evaluation of our previous method [SHH16].
- Our novel data augmentation techniques further improve the results of the fine-grained recognition of vehicles relative both to our previous method and other state-of-the-art methods (Section 5.3.3).
- We remove the requirement of the previous method [SHH16] to know the 3D bounding box by estimating the bounding box both at training and test time (Section 5.3.4).
- We collected more samples to the BoxCars dataset, increasing the dataset size almost twice (Section 5.4).

We will make the collected dataset and source codes for the proposed algorithm publicly available[1] for future reference and comparison.

5.2 RELATED WORK

In order to provide context to the proposed method, we present a summary of existing fine-grained recognition methods (both general and focused on vehicles).

5.2.1 *General Fine-Grained Object Recognition*

We divide the fine-grained recognition methods from recent literature into several categories as they usually share some common traits. Methods exploiting annotated model parts (e.g. [72, 179]) are not discussed in detail as it is not common in fine-grained datasets of vehicles to have the parts annotated.

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

5.2.1.1 *Automatic Part Discovery*

Parts of classified objects may be discriminatory and provide lots of information for the fine-grained classification task. However, it is not practical to assume that the location of such parts is known a priori as it requires significantly more annotation work. Therefore, several papers [170, 34, 171, 79, 142, 81, 183] have dealt with this problem and proposed methods how to automatically (during both training and test time) discover and localize such parts. The methods differ mainly in the ways in which they are used for the discovery of discriminative parts. The features extracted from the parts are usually classified by SVMs.

5.2.1.2 *Methods using Bilinear Pooling*

Lin et al. [92] use only convolutional layers from the net for extraction of features which are classified by a bilinear classifier [120]. Gao et al. [47] followed the path of bilinear pooling and proposed a method for Compact Bilinear Pooling getting the same accuracy as the full bilinear pooling with a significantly lower number of features.

5.2.1.3 *Other Methods*

Xie et al. [166] proposed to use a hyper-class for data augmentation and regularization of fine-grained deep learning. Zhou et al. [189] use CNN with Bipartite Graph Labeling to achieve better accuracy by exploiting the fine-grained annotations and coarse body type (e.g. Sedan, SUV). Lin et al. [91] use three neural networks for simultaneous localization, alignment and classification of images. Each of these three networks does one of the three tasks and they are connected into one bigger network. Yao et al. [171] proposed an approach which uses responses to random templates obtained from images and classifies merged representations of the response maps by SVM. Zhang et al. [180] use pose normalization kernels and their responses warped into a feature vector. Chai et al. [19] propose to use segmentation for fine-grained recognition to obtain the foreground parts of an image. A similar approach was also proposed by Li et al. [89]; however, the authors use a segmentation algorithm which is optimized and fine-tuned for the purpose of fine-grained recognition. Finally, Gavves et al. [48] propose to use object proposals to obtain the foreground mask and unsupervised alignment to improve fine-grained classification accuracy.

5.2.2 *Fine-Grained Recognition of Vehicles*

The goal of fine-grained recognition of vehicles is to identify the exact type of the vehicle, that is its make, model, submodel, and model year. The recognition system focused only on vehicles (in relation to general fine-grained classification of birds, dogs, etc.) can benefit from that the vehicles are rigid, have some distinguishable landmarks (e.g. license plates), and rigorous models (e.g. 3D CAD models) can be available.

5.2.2.1 *Methods Limited to Frontal/Rear Images of Vehicles*

There is a multitude of papers [119, 31, 23, 116, 122, 86, 175, 98] using a common approach: they detect the license plate (as a common landmark) on the vehicle and extract features from the area around the license plate as the front/rear parts of vehicles are usually discriminative.

There are also papers [176, 68, 69, 90, 4, 61] directly extracting features from frontal images of vehicles by different methods and optionally exploiting the standard structure of parts on the frontal mask of car (e.g. headlights).

5.2.2.2 *Methods based on 3D CAD Models*

There were several approaches on how to deal with viewpoint variance using synthetic 3D models of vehicles. Lin et al. [93] propose to jointly optimize 3D model fitting and fine-grained classification, Hsiao et al. [67] use detected contour and align the 3D model using 3D chamfer matching. Krause et al. [80] propose to use synthetic data to train geometry and viewpoint classifiers for the 3D model and 2D image alignment. Prokaj et al. [121] propose to detect SIFT features on the vehicle image and on every 3D model seen from a set of discretized viewpoints.

5.2.2.3 *Other Methods*

Gu et al. [58] propose extracting the center of a vehicle and roughly estimate the viewpoint from the bounding box aspect ratio. Then, they use different Active Shape Models for alignment of data taken from different viewpoints and use segmentation for background removal.

Stark et al. [146] propose using an extension of Deformable Parts Model (DPM) [40] to be able to handle multi-class recognition. The model is represented by latent linear multi-class SVM with HOG [29] features. The authors show that the system outperforms different methods based on Locally-constrained Linear Coding [158] and HOG. The recognized vehicles are used for eye-level camera calibration.

Liu et al. [94] use deep relative distance trained on a vehicle re-identification task and propose training the neural net with Coupled Clusters Loss instead of

triplet loss. Boonsim et al. [9] propose a method for fine-grained recognition of vehicles at night. The authors use relative position and shape of features visible at night (e.g. lights, license plates) to identify the make&model of a vehicle, which is visible from the rear side.

Fang et al. [38] propose using an approach based on detected parts. The parts are obtained in an unsupervised manner as high activations in a mean response across channels of the last convolutional layer of used CNN. The authors in [71] introduce spatially weighted pooling of convolutional features in CNNs to extract important features from the image.

5.2.2.4 *Summary of Existing Methods*

Existing methods for the fine-grained classification of vehicles usually have significant limitations. They are either limited to frontal/rear viewpoints [119, 31, 23, 116, 122, 86, 175, 98, 176, 68, 69, 90, 4, 61] or require some knowledge about 3D models of the vehicles [121, 80, 67, 93] which can be impractical when new models of vehicles emerge.

Our proposed method does not have such limitations. The method works with arbitrary viewpoints and we require only 3D bounding boxes of vehicles. The 3D bounding boxes can either be automatically constructed from traffic video surveillance data [DSH14, DHJS15] or we propose a method on how to estimate the 3D bounding boxes both at training and test time from single images (see Section 5.3.4).

5.2.3 *Datasets for Fine-Grained Recognition of Vehicles*

There is a large number of datasets of vehicles (e.g [130, 106]) which are usable mainly for vehicle detection, pose estimation, and other tasks. However, these datasets do not contain annotations of the precise vehicles' make and model.

When it comes to the fine-grained recognition datasets, there are some [146, 80, 93, 90] which are relatively small in number of samples or classes. Therefore, they are impractical for the training of CNN and deployment of real world traffic surveillance applications.

Yang et al. [169] published a large dataset *CompCars*. The dataset consists of a web-nature part, made of 136k of vehicles from 1 600 classes taken from different viewpoints. It also contains a surveillance-nature part with 50k frontal images of vehicles taken from surveillance cameras.

Liu et al. [97] published dataset *VeRi-776* for the vehicle re-identification task. The dataset contains over 50k images of 776 vehicles captured by 20 cameras covering an 1.0 km^2 area in 24 hours. Each vehicle is captured by $2 \sim 18$ cameras under

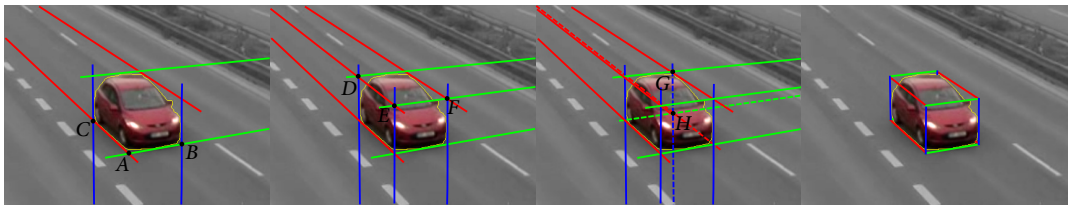


Figure 5.2: 3D bounding box construction process. Each set of lines with the same color intersects in one vanishing point. See the original paper for full details [DSH14]. The image was adopted from the paper with the authors' permission.

different viewpoints, illuminations, resolutions and occlusions. The dataset also provides various attributes, such as bounding boxes, vehicle types, and colors.

5.2.4 *Vehicle Detection*

In traffic surveillance applications, it is common that prior fine-grained vehicle classification is necessary to detect vehicles; therefore, we include a brief overview of existing methods for vehicle detection. It is possible to use standard object detectors – either based on convolutional neural networks [124, 123], AdaBoost [33], Deformable Part Models [40] or Hough Transformation [45]. There were also attempts to improve specifically vehicle detection based on geometric information [159], during night [131], or to increase the accuracy of localization of occluded vehicles [26].

5.3 PROPOSED METHODOLOGY FOR FINE-GRAINED RECOGNITION OF VEHICLES

In agreement with recent progress in the Convolutional Neural Networks [153, 82, 22], 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.

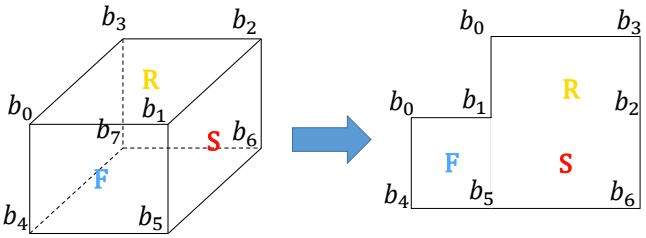


Figure 5.3: 3D bounding box and its unpacked version.

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

We based our work on 3D bounding boxes proposed by [DSH14] (Fig. 5.4) which can be automatically obtained for each vehicle seen by a surveillance camera (see Figure 5.2 for schematic 3D bounding box construction process or the original paper [DSH14] for further details). 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. 5.3) and they are organized into the final matrix **U**:

$$\mathbf{U} = \begin{pmatrix} \mathbf{o} & \mathbf{R} \\ \mathbf{F} & \mathbf{S} \end{pmatrix} \tag{5.1}$$

The unpacking itself is done by obtaining homography between points b_i (Fig. 5.3) 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 DPM or other algorithms for part localization. Later in the text, we will refer to this normalization method as **Unpack**.

5.3.2 *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.

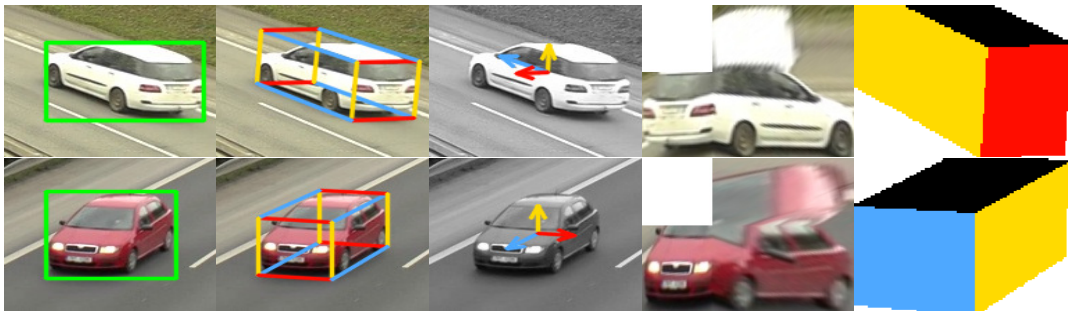


Figure 5.4: Examples of data normalization and auxiliary data fed to nets. **Left to right:** vehicle with 2D bounding box, computed 3D bounding box, vectors encoding viewpoints on the vehicle (**View**), unpacked image of the vehicle (**Unpack**), and rasterized 3D bounding box fed to the net (**Rast**).

Figure 5.5: 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**).

View. The viewpoint is extracted from the orientation of the 3D bounding box – Fig. 5.4. 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 $\mathbf{T}$ is obtained as

$$\mathbf{T}_p = \begin{cases} (1,0,0,0) & p \in \angle b_0 b_1 b_4 b_5 \text{ and } d = 1 \\ (0,1,0,0) & p \in \angle b_0 b_1 b_4 b_5 \text{ and } d = 0 \\ (0,0,1,0) & p \in \angle b_1 b_2 b_5 b_6 \\ (0,0,0,1) & p \in \angle b_0 b_1 b_2 b_3 \\ (0,0,0,0) & \text{otherwise} \end{cases} \tag{5.2}$$

where $\angle b_0 b_1 b_4 b_5$ denotes the quadrilateral defined by points b_0, b_1, b_4 and b_5 in Figure 5.3.

Finally, the 3D rasterized bounding box is cropped by the 2D bounding box of the vehicle. For an example, see Figure 5.4, showing rasterized bounding boxes for different vehicles taken from different viewpoints.

5.3.3 *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 5.5. 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.

The results presented in Section 5.5.5 show that both these modifications improve the classification accuracy both in combination with other presented techniques or by themselves.

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. [174], 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.

Figure 5.6: Estimation of 3D bounding box. **Left to right:** image with vehicle 2D bounding box, output of contour object detector [168], our constructed contour, estimated directions towards vanishing points, ground truth (**green**) and estimated (**red**) 3D bounding box.

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

As the results (Section 5.5) 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 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. [168] 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 5.6 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. [128], we formulated the regression of the direction towards the vanishing points as a classification task into bins corresponding to angles and we used ResNet50 [62] 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 (see Figure 5.7). 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).

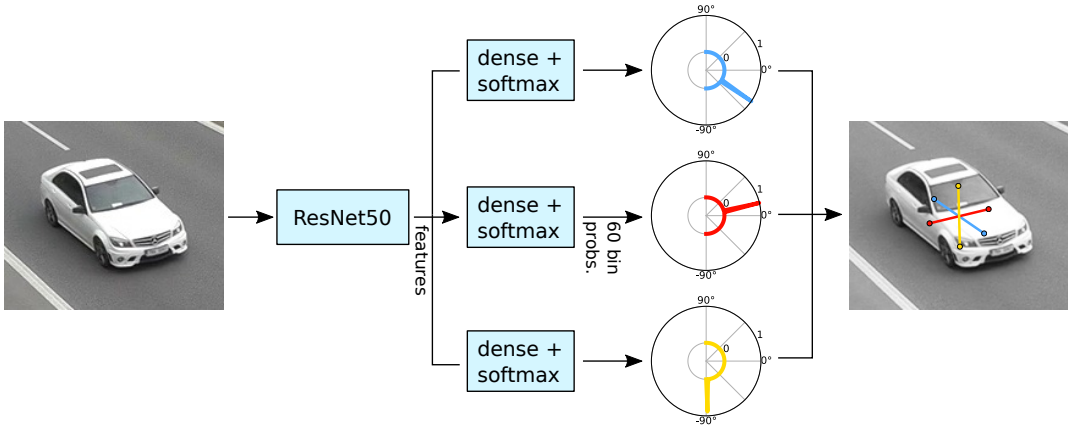


Figure 5.7: Used CNN for estimation of directions towards vanishing points. The vehicle image is fed to ResNet50 with 3 separate outputs which predict probabilities for directions of vanishing points as probabilities in a quantized angle space (60 bins from $-90°$ to $90°$).

As the training data for the regression we used BoxCars116k dataset (Section 5.4) 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. 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 5.5.3 show that by using this estimated 3D bounding boxes, our method still significantly outperforms other convolutional neural networks without input modification.

Figure 5.8: Collate of random samples from the BoxCars116k dataset.

5.4 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. 5.8). 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 [124] trained on COD20k dataset [74]) is captured in multiple images, as it passes by the camera; therefore, we have more visual information about each vehicle.

5.4.1 *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

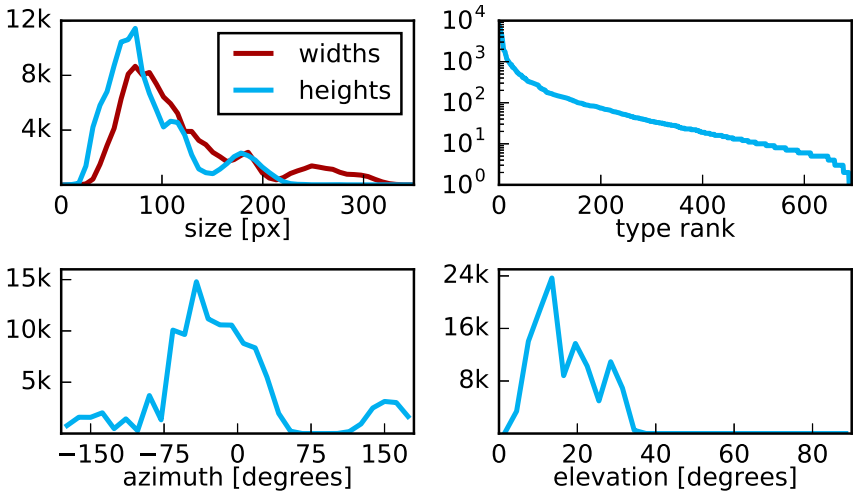


Figure 5.9: BoxCars116k dataset statistics – **top left:** 2D bounding box dimensions, **top right:** number of fine-grained types samples, **bottom left:** azimuth distribution (0° denotes frontal viewpoint), **bottom right:** elevation distribution.

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).

5.4.2 *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. Detailed statistics about the dataset can be found in Figure 5.9 and Appendix A. The distribution of types in the dataset is shown in Figure 5.9 (**top right**) and samples from the dataset are in Figure 5.8. 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 [147, 190]. The dataset has been made publicly available[2] 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 (see Figure 5.9 and 3D plots in Appendix A).

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

5.4.3 *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. Detailed split statistics can be found in Appendix A.

5.5 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** [82]
- **VGG16**, **VGG19** [143]
- **ResNet50**, **ResNet101**, **ResNet152** [62]
- CNNs with Compact Bilinear Pooling layer [47] in combination with VGG nets denoted as **VGG16+CBL** and **VGG19+CBL**.

Table 5.1: 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. The raw numbers can be found in Appendix A.

	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

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).

5.5.1 *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 [130] 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 [143]. 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.

The summary results for both medium and hard splits are shown in Table 5.1 and the raw results are in Appendix A. 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.

5.5.2 *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 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.

Table 5.2: 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 [82]	66.65/77.75	963
VGG16 [143]	77.26/86.71	173
VGG19 [143]	76.74/86.06	146
Resnet50 [62]	75.48/84.61	155
Resnet101 [62]	76.46/85.31	95
Resnet152 [62]	77.68/86.20	66
BCNN (VGG-M) [92]	64.83/72.22	87*
BCNN (VGG16) [92]	69.64/78.56	10*
CBL (VGG16) [47]	70.38/80.11	165
CBL (VGG19) [47]	70.69/80.26	141
PCM (AlexNet) [142]	63.24/73.94	15
PCM (VGG19) [142]	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

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. [92] 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 20^{th} epoch). We also used image flipping to augment the training set.

CBL. We modified compatible nets with **C**ompact **BiL**inear Pooling proposed by [47] which followed the work of [92] 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 5.5.1.

PCM. Simon et al. [142] 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 5.2. As the table shows, our method significantly outperforms both standard CNNs [82, 143, 62] and methods for fine-grained recognition [92, 142, 47]. 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 5.2; 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.

5.5.3 *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 5.3.4) would be used instead.

The classification results are shown in Table 5.3; 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).

Table 5.3: 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 5.3.4).

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

5.5.4 *Impact of Training/Testing Viewpoint Difference*

We were also interested in finding out the main reason why the classification accuracy is improved. We have analyzed several possibilities and found out that the most important aspect is viewpoint difference.

For every training and testing sample we computed the viewpoint (unit 3D vector from vehicles' 3D bounding boxes centers) and for each testing sample we found one training sample with the lowest viewpoint difference (see Figure 5.11). Then, we divided the testing samples into several bins based on the difference angle. For each of these bins we computed the accuracy for the standard nets without any modifications and nets with the proposed modifications. There is 56% of the test samples in the first bin ($0° - 2°$), and in the middle bins there are 22% and 17% of test data. In the last bin, there are 5% of the test data. Finally, we obtained an improvement in percentage points for each modification and bin, by comparing the net's performance on the data in the bin with and without the modification harnessed. The results are displayed in Figure 5.10.

There are several facts which should be noted. The first and most important is that the **Unpack** modification alone improves significantly the accuracy for larger viewpoint differences (the accuracy is improved by more than 20 percent points for the last bin). The other important fact, which should be noted, is that the other modifications (mainly **Color** and **ImageDrop**) improve the accuracy furthermore. This improvement is independent on the training-testing viewpoint difference.

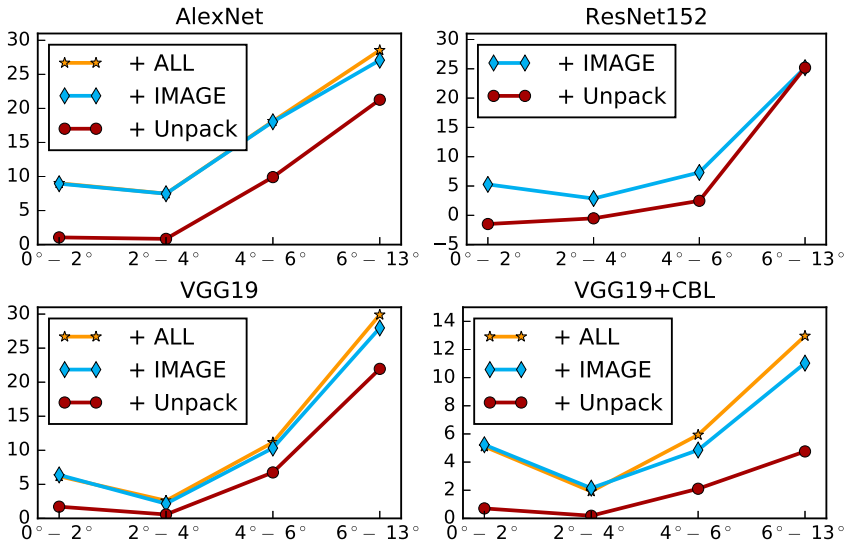


Figure 5.10: Correlation of improvement relative to CNNs without modification with respect to train-test viewpoint difference. The x-axis contains bins viewpoint difference bins (in degrees), and the y-axis denotes improvement compared to base net in percent points, see Section 5.5.4 for details. The graphs show that with increasing viewpoint difference, the accuracy improvement of our method increases. Only one representative of each CNN family (AlexNet, VGG, ResNet, VGG+CBL) is displayed – results for all CNNs are in Appendix A.

Figure 5.11: Examples of viewpoint difference between the training and testing sets. Each pair shows a testing sample (left) and its corresponding "nearest" training sample (right); by "nearest" we mean the sample with the lowest angle between its viewpoint and the test sample's viewpoint.

5.5.5 *Impact of Individual Modifications*

We were also curious how different modifications by themselves help to improve the accuracy. We conducted two types of experiments which focus on different aspects of the modifications. The evaluation is not done on ResNets, as we only use **IMAGE** level modifications with ResNets; thus, we cannot evaluate Rast and View modifications with ResNets.

Table 5.4: Summary of improvements for different nets and modifications computed as [*base net* + *modification*] − [*base net*]. The raw data can be found in Appendix A.

	mean	**best**
Unpack	+2.11/+2.55	+3.47/+4.37
View	−0.32/−0.35	+0.19/+0.31
Rast	−0.03/−0.04	+0.30/+0.72
Color	+3.17/+2.03	+4.80/+3.60
ImageDrop	+0.70/+0.20	+1.53/+0.96

Table 5.5: Summary of improvements for different nets and modifications computed as [*base net* + *all*] − [*base net* + *all* − *modification*]. The raw data can be found in Appendix A.

	mean	**best**
Unpack	+3.41/+3.48	+6.93/+7.60
View	−0.14/−0.15	+0.36/+0.18
Rast	−0.03/−0.08	+0.30/+0.20
Color	+3.42/+2.43	+6.34/+6.18
ImageDrop	+1.32/+0.77	+4.24/+3.54

The first experiment is focused on the influence of each modification by itself. Therefore, we compute the accuracy improvement (in accuracy percent points) for the modifications as [*base net* + *modification*] − [*base net*], where [...] stands for the accuracy of the classifier described by its contents. The results are shown in Table 5.4. As it can be seen in the table, the most contributing modifications are **Color**, **Unpack**, and **ImageDrop**.

The second experiment evaluates how a given modification contributed to the accuracy improvement when all of the modifications are used. Thus, the improvement is computed as [*base net* + *all*] − [*base net* + *all* − *modification*]. See Table 5.5 for the results, which confirm the previous findings and **Color**, **Unpack**, and **ImageDrop** are again the most positive modifications.

5.5.6 *Vehicle Type Verification*

Lastly, we evaluated the quality of features extracted from the last layer of the convolutional nets for the verification task. Under the term *verification*, we understand the task to determine whether a pair of vehicle tracks share the same fine-grained

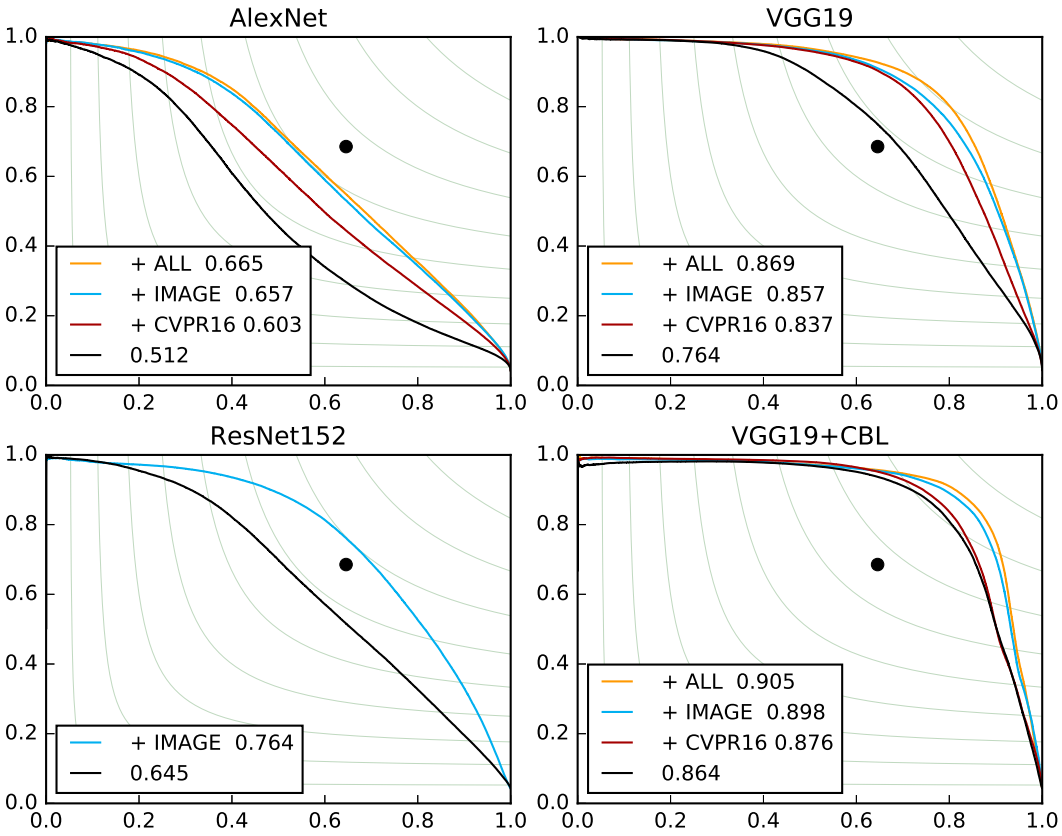


Figure 5.12: Precision-Recall curves for verification of fine-grained types. Black dots represent the human performance [SHH16]. Only one representative of each CNN family (AlexNet, VGG, ResNet, VGG+CBL) is displayed – results for all CNNs are in Appendix A.

type or not. In agreement with previous works in the field [153], we use cosine distance between the features for the verification.

We collected 5 million random pairs of vehicle tracks from the test part of *BoxCars116k* splits and evaluate the verification on these pairs. As we used tracks which can have a different number of vehicle images, we used 9 random pairs of images for each pair of tracks and then used median distance between these image pairs as the distance between the whole tracks.

Precision-Recall curves and Average Precisions are shown in Figure 5.12. As the results show, our modifications significantly improve the average precision for each CNN in the given task. Moreover, as the figure shows, the method outperforms human performance (black dots in Figure 5.12), as reported in the previous paper [SHH16].

5.6 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 [143, 62], 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 [92, 142, 47] 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.

Part III

AUTOMATIC SPEED MEASUREMENT OF VEHICLES

This part of the dissertation thesis contains re-formatted copies of two my papers on the topic of fully automatic speed measurement of vehicles. The first paper introduces a new comprehensive dataset for speed measurement of vehicles. The dataset, BrnoCompSpeed, is the largest publicly available dataset for speed measurement of vehicles from video with very precise ground truth information about the speed of passing vehicles. The videos are recorded at different locations and under various viewpoints.

In the second paper, we propose a novel method for fully automatic speed measurement of vehicles from video. The method is based on detection of two vanishing points and it also uses 3D model bounding box alignment of several common vehicle types for scene scale estimation. The results on BrnoCompSpeed dataset show that our method significantly outperforms previous state-of-the-art methods and manual calibration from manual distance measurements on the road plane.

6

COMPREHENSIVE DATASET FOR AUTOMATIC SINGLE CAMERA VISUAL SPEED MEASUREMENT

CITATION 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.

ABSTRACT In this paper, we focus on traffic camera calibration and visual speed measurement from a single monocular camera, which is an important task of visual traffic surveillance. Existing methods addressing this problem are hard to compare due to a lack of a common dataset with reliable ground truth. Therefore, it is not clear how the methods compare in various aspects and what are the factors affecting their performance. We captured a new dataset of 18 full-HD videos, each around one hour long, captured at 6 different locations. Vehicles in the videos (20 865 instances in total) are annotated with precise speed measurements from optical gates using LIDAR and verified with several reference GPS tracks. We made the dataset available for download and it contains the videos and metadata (calibration, lengths of features in image, annotations, etc.) for future comparison and evaluation. Camera calibration is the most crucial part of the speed measurement; therefore, we provide a brief overview of the methods and analyze a recently published method for fully automatic camera calibration and vehicle speed measurement and report the results on this dataset in detail.

6.1 INTRODUCTION

Speed measurement is one of the crucial problems in traffic surveillance. So far, the field is dominated by radar and section speed measurements because they meet tight methodological requirements and standards. However, these methods are limited in the information they provide and they may be expensive. For example, in radar measurement it is impossible to recognize the fine-grained models of passing cars and the radar antenna must be placed at a specific position regarding the traffic. Section speed measurement requires two cameras for each position and a complex infrastructure for processing the data. Speed measurement from

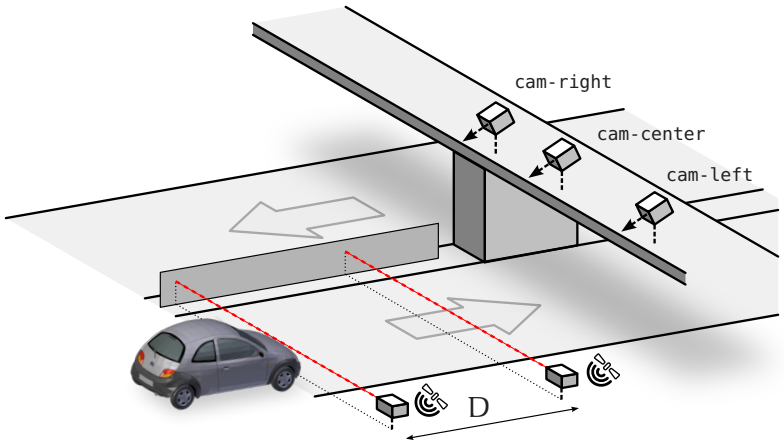


Figure 6.1: The data recording setup. We use two LIDARs synced by GPS time, and three cameras recording the highway from different surveillance viewpoints.

a single monocular camera is not typically used for surveillance; however it can be beneficial – one camera can be used for surveillance on multiple lanes, it is possible to use the data for fine-grained make & model recognition of the vehicles [61, 4, 69, 68] and other tasks. Another interesting aspect is that it is possible to use already installed monitoring/security cameras for speed measurement and other traffic analysis tasks.

A number of works dealing with monocular speed measurement can be found in the literature [135, 28, 56, 63, 104, 110, 145, DSH14, 83, 103, 32] (detailed individually below). Such systems are on the rise especially recently, with the growing number of IP cameras, with increase of their resolution, and with the development of computer vision algorithms used for their processing. Our aim is to provide an important missing piece: a dataset which would allow for reliable comparison between the approaches. These systems are described in detail in the following section.

We captured a new benchmark dataset of 18 full-HD videos taken from surveillance viewpoints on the traffic (see Figure 6.1). Each of the videos is around one hour long to allow for even lengthy calibration procedures and self-adjustment of the surveillance system. Triplets of videos are observing the same time interval at the same location from different angles. These shots were captured at 6 different locations. Vehicles in the videos (20 865 instances in total) are annotated with precise speed measurements from optical gates using LIDAR and verified with several reference GPS tracks. We provide[1] the videos and metadata (calibration, distances measured on the road plane, annotations, etc.) for future comparison and evaluation. To illustrate the properties of the dataset and to establish a first baseline, we analyze the data by a recently published method for fully automatic camera cal-

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

ibration and vehicle speed measurement [DSH14] and we report the quantitative results.

Although the dataset is focused on speed measurement, it can be used also for different traffic surveillance tasks, for example vehicle counting, tracking, vehicle classification and other.

We consider the camera calibration algorithm to be the most crucial part of speed measurement. It defines how well the speed measurement is done as it is impossible to measure speed accurately with a poorly calibrated camera. The used algorithm also defines whether it is usable with a camera observing the road from arbitrary viewpoint and it determines whether the method can be used fully automatically which is important for large scale deployment. Therefore, we include a brief overview of existing camera calibration algorithms for traffic surveillance applications.

The key contributions of this paper are: **a)** Novel, publicly available dataset for evaluation of camera calibration in traffic surveillance and speed measurement. The dataset contains 18 videos and 20 865 vehicles with known precise ground truth. **b)** Thorough and complex evaluation of a recent fully automatic method for traffic camera calibration [DSH14].

6.2 RELATED WORK – CAMERA CALIBRATION FOR SPEED MEASUREMENT OF VEHICLES

One of the most important parts of speed measurement of vehicles from a single monocular camera is calibration of the camera. In a general case, this includes dealing with perspective projection and different rotations of the camera; it is also necessary to deal with unknown distance from the camera to the ground plane of the road and possibly with radial and tangential distortion. It is usually necessary to obtain intrinsic and extrinsic camera parameters together with the scene scale (or the distance of the camera from the road/ground plane). Therefore, we include also a brief overview of the typical solutions of camera calibration for speed measurement of vehicles. However, first we include the definition of traffic surveillance camera calibration.

6.2.1 *Traffic Surveillance Camera Calibration*

General mathematical model for camera calibration is represented by a projection matrix $\mathbf{P} = \mathbf{K}\,[\mathbf{R}\;\mathbf{T}]$, where $\mathbf{K}$ denotes intrinsic camera parameters, $\mathbf{R}$ stands for camera rotation, and $\mathbf{T}$ represent camera translation. The extrinsic parameters (rotation and translation) are relative to defined world coordinate system (see Figure

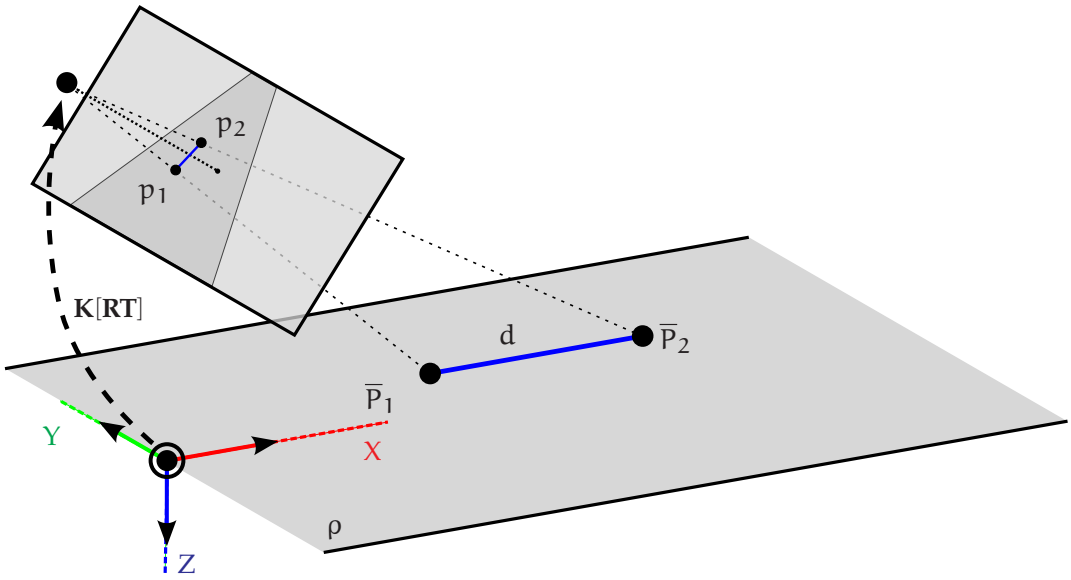


Figure 6.2: The essential goal of traffic surveillance camera calibration is to be able measure real world distance d between two points ($\overline{P}_1$, $\overline{P}_2$) on road plane given their projection to the image (p_1, p_2). X,Y, and Z axes represent a real world coordinate system and **K** represent intrinsic camera parameters, while **R** and **T** are extrinsic camera parameters.

6.2). Since the calibration for traffic surveillance is specific, we describe all these aspects with **application to vehicle speed measurement** in mind.

GOAL The essential goal of traffic surveillance camera calibration is to measure speed of vehicles. For the speed measurement, it is required to be able measure time and distances on the road plane. The time measurement part is rather trivial. However, for the distance measurement it is necessary to measure the distance between two points on the road plane (or any other plane parallel to the road plane and with known distance from the road plane) given their projection to the image. See Figure 6.2 for an example.

INPUT For fully automatic methods, the camera calibration input is usually a video of the observed traffic scene. However, for methods which include manual steps, part of the input are also usually distance measurements on the road plane.

ASSUMPTIONS Zero pixel skew is generally used as an assumption about the camera model. Another widely used assumption is that the camera's principal point is in the center of the image. Also, there is usually the assumption that the road can be approximated by a plane. The authors usually assume that the observed road segment is approximately straight, that the vehicles move straight and their velocity is constant on the measured segment (no acceleration).

MATHEMATICAL MODEL As the standard camera model with $\mathbf{K}$ $[\mathbf{R}\ \mathbf{T}]$ matrices is sufficiently described in existing literature [60]; we refer the readers there. However, it is also possible to use a different formulation based on vanishing points of the road plane [DHJS15]. This formulation is easily convertible to the standard one. Finally, for computation of 3D real world coordinates on the road plane of a point in image space, it is necessary to compute intersection of the road plane (e.g. $z = 0$ as shown in Figure 6.2) and a ray defined by the camera optical center and the coordinates on the image plane.

ATTRIBUTES One important attribute of the camera calibration algorithm, which should be kept in mind, is whether the algorithm works automatically in the sense that there is no manual input required per installed camera. The property of being automatic becomes more important as the number of installed cameras grows. A number of papers and approaches to solving this problem exist and they will be discussed in detail in the following text. Another important attribute is whether the algorithm works from arbitrary viewpoint, as it is a significant drawback of a method if it requires specific camera placement relative to the observed road.

6.2.2 *Methods Based on Acquired Line Markings*

He and Yung [63] proposed a method for speed estimation of vehicles which is based on calibration using a calibration pattern formed by lane markings on the road [64]. The authors use a rectified image in further processing in order to deal with perspective projection. To obtain the locations of the vehicles within the ground plane, shadows cast by rear bumpers are used. The vehicles and shadows are detected by background subtraction and binary block matching.

Cathey and Dailey [17] used a method based on detection of the vanishing point which is in the direction of vehicles movement. To obtain this vanishing point, detected line markings are used and their intersection is found in the least squares manner. The scale (pixels/meters ratio) for the camera is computed from average line marking stripe length and known stripe length in the real world. Finally, the authors used cross correlation to compute the number of pixels which vehicles passed between consecutive frames.

Grammatikopoulos et al. [56] use the assumption that the camera is only tilted along Y axis in Figure 6.2; thus they assume that the *second vanishing point* (horizontal and perpendicular to the first one) is in infinity. The first vanishing point is detected as the intersection of the line markings with least squares adjustment.

The vehicles are detected by background subtraction and tracked by normalized cross-correlation.

You et al. [172] propose to use detection of vanishing point in the direction of vehicles' movements from lane markings and vanishing point perpendicular to road plane from detected poles and pedestrians. The authors obtain the scale from known height of the camera above the road or known dimensions on the road.

By definition, this class of methods based on observed line markings is usable only when the line markings are present, visible, and recognizable. This fact can be limiting on local roads, where the line markings are not drawn or on highways during work on the road with additional temporary line markings. Also, some of the methods require measurements on the road which is a great disadvantage.

6.2.3 *Methods Based on Vehicles' Movement*

Dubská et al. [DSH14] published a speed measurement system using a calibration method by detection of two vanishing points [DHJS15]. We give the details on this method below in Section 6.2.5.

Schoepflin et al. [135] use an activity map (by detecting the vehicles as the moving foreground) to obtain lane boundaries and the intersection of the boundaries treat as the first vanishing point in the direction of the vehicle motion. The second vanishing point is detected as the intersection of lines formed by the bottom edges of the vehicles. One known length (manually measured and entered per camera) in the image is used for scale inference.

Filipiak et al. [41] propose to use sequences of detected license plates of vehicles for finding intrinsic and extrinsic camera parameters by an evolutionary algorithm. The method was evaluated on a dataset captured by zoomed surveillance cameras with a small field of view on the road.

The methods based on vehicles' movement no longer need visible road markings; however, when used on small local roads, the calibration may take some time as it usually improves with more observed vehicles.

6.2.4 *Methods Using Manual Measurements*

Maduro et al. [104] assume two known arbitrary angles on the ground plane to calibrate the camera and use lengths of line markings' stripes to obtain the camera scale for the given scene. The authors used background subtraction for detecting the vehicles and Kalman filter [75] for tracking them.

Nurhadiyatna et al. [110] used GMM background subtraction [190] for detection of vehicles and tracked them by Kalman filter. They use a calibrated pinhole camera with zero pan and known distances in the real world.

Sina et al. [145] focus on speed measurement at night. They used detected and paired headlights to detect vehicles, track them and measure their speed. The camera calibration is based on manual measurements of camera angles and distance of the camera from the ground plane. The reported average error is 3.3 km/h relative to ground truth obtained by GPS.

Luvizon et al. [103] used a different approach and they propose to detect and track license plates in order to obtain motion of vehicles in the scene. The motion is then converted to the real world distance by rectifying and scaling. The scale inference is based on a priori known real world measures.

Methods using manual measurements on the road have the biggest disadvantage that it is necessary to do the manual measurements, which potentially can mean stopping traffic on the road. The advantage of the methods may be (in some cases) that they are more accurate than automatic or semi-automatic ones.

6.2.5 *Automatic Calibration Method based On Statistics of Dimensions*

Here we give details on the speed measurement method of Dubská [DSH14], as it meets all our requirements (it is fully automatic and it is usable from arbitrary viewpoint) and we use it later in the experiments. In principle, the method relies on camera calibration from two automatically detected vanishing points.

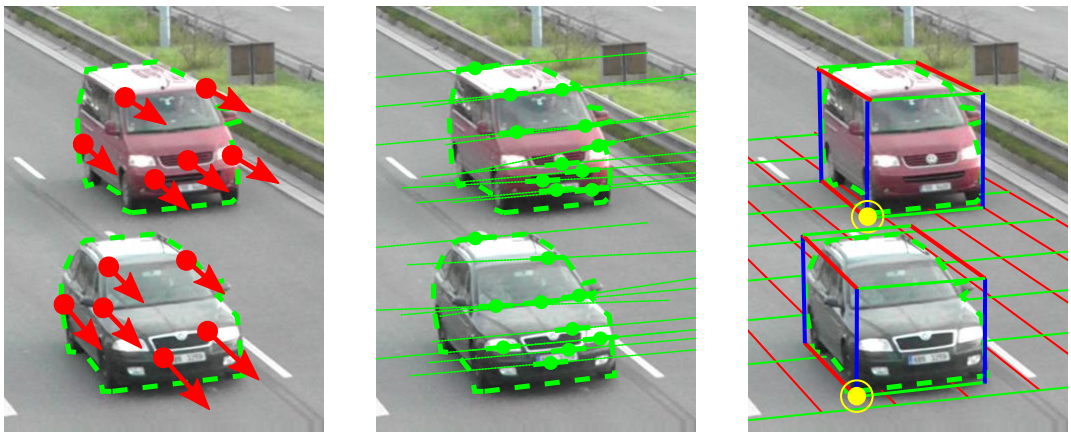


Figure 6.3: Automatic camera calibration according to Dubska [DSH14]. From left to right: Tracked keypoints for VP1, oriented edges voting for VP2, and road plane with bounding boxes for the cars and reference points for tracking.

The authors use a simple foreground detection model to filter areas with movement. The first vanishing point (VP1, which is in the direction of vehicles' movement) is recovered from tracked feature points on the vehicles using min eigen-

Table 6.1: Summary of different camera calibration methods for speed measurement. It should be noted that the reported errors are only informative as all the methods are evaluated on different datasets and by different protocols. We consider a system to be automatic if it does not require any manual calibration for each individual camera. **auto** – denotes whether the system works fully automatically, **view** – denotes whether the system is usable from arbitrary viewpoint

	camera calibration method	**auto**	**view**	**mean error**
Dailey et al., 2000 [28]	multiple assumptions on vehicle movements and known mean length of vehicles	✓	✗	6.5 km/h
Schoepflin et al., 2003 [135]	detection of two vanishing points, one known length	✗	✓	N/A
Cathey et al., 2005 [17]	vanishing point obtained from detected line markings, scale computed from lengths of stripes	✗	✓	N/A
Grammatik. et al., 2005 [56]	one vanishing point obtained from detected line markings, the second one assumed in infinity, one known distance is required	✗	✗	3 km/h
He and Yung, 2007 [63]	calibration by pattern formed by lane markings	✗	✓	3.27 %
Maduro et al., 2008 [104]	known angle of the ground plane, lengths of line markings' stripes	✗	✓	2 %
Nurhadiyatna et al., 2013 [110]	known distances in the real world and in the scene, zero pan assumption	✗	✗	7.63 km/h
Sina et al., 2013 [145]	manual measurements	✗	✓	3.3 km/h
Dubská et al., 2014 [DSH14]	detection of two vanishing points, scale computed by matching of statistics of vehicles' dimensions to mean dimensions of vehicles	✓	✓	1.99 %
Lan et al., 2014 [83]	relaxation of perspective projection, known width of lanes	✗	✗	0.9 % – 2.5 %
Luvizon et al., 2014 [103]	known real world measures	✗	✓	1.63 km/h
Do et al., 2015 [32]	zero pan assumption, equilateral triangle drawn on the road	✗	✗	2.91 %
Filipiak et al., 2016 [41]	constant speed assumption, evolutionary algorithm to recover intrinsic and extrinsic parameters from detected license plate sequences	✓	✗	2.3 km/h
You et al., 2016 [172]	detection of vanishing point in the direction of vehicles' movements from lane markings and vanishing point perpendicular to road plane from poles and pedestrians, the scale is obtained from known height of camera above the road	✗	✓	N/A

value detector and KLT tracker. The tracked points' motion is transformed using a line-to-line Hough transformation parametrized by parallel coordinates [35] where the global maximum corresponds to the image of the first vanishing point. The second vanishing point (VP2) is extracted from strong edges present on the moving vehicles meeting some conditions given by the position of the VP1. The edges (and their orientations) are, again, transformed to the Hough space where the strongest maximum accounts for the vanishing point. From these two vanishing points, the camera intrinsics and extrinsics can be recovered (assuming principal point in the image center, square pixels and zero skew).

The authors propose an algorithm for computing the 3D bounding box around the vehicle blobs. Mean size of the bounding boxes and known mean dimensions of the vehicles for a given country accounts for the scene scale. Vehicle speed is measured simply by tracking 3D bounding boxes around the blobs using Kalman filter and measuring the travel distance in the real world.

The authors evaluated their method on several videos with several car passes with ground truth speed obtained from GPS.

6.2.6 *Other Methods*

Dailey et al. [28] proposed a method for vehicles speed measurement based on tracking of vehicle blobs and constraining them to move along a line. The blobs are detected as inter-frame differences followed by Sobel edge detector. The authors assume that the vehicles are moving towards or from the camera and use mean length of vehicles to obtain the scene scale.

Do et al. [32] proposed a camera calibration method for speed measurement based on artificial markers drawn on the road. They assume that the camera has zero pan angle and that markers determining vertices of an equilateral triangle with a known distance between vertices which are visible on the road. They used the triangle to obtain the scale factor and the tilt angle.

Lan et al. [83] use optical flow to compute the speed of different points of a vehicle and they average this speed to get the speed of vehicle in image units. However, to convert them into kilometers per hour, the authors assume that there is no perspective projection effect and the width of the ROI (width of lanes) is known.

6.2.7 *Summary and Analysis of the Methods*

A summary of the presented camera calibration methods can be found in Table 6.1. As the table shows, some of the approaches have different limitations and they

Table 6.2: Summary of datasets used for evaluation of visual speed measurement methods.

dataset	videos	vehicles	source of gt	resolution	evaluation metrics
[28]	1	532	induction loops	N/A	speed measurement error
[135]	2	1 015	induction loops	320 × 240	speed measurement error
[56]	1	20	manual measurements	768 × 576	speed measurement error
[63]	1	64	RADAR	1280 × 1024	speed measurement error
[104]	2	few	GPS	N/A	speed measurement error
[110]	10	15	GPS	320 × 240	speed measurement error
[145]	13	13	GPS	N/A	speed measurement error, vehicle counting
[DSH14]	6	29	GPS	864 × 480	speed measurement error, distance measurement error
[83]	1	2 010	RADAR	640 × 480	speed measurement error
[103]	1	75	induction loops	768 × 480	speed measurement error, license plate detection
[32]	1	3	speedometer	N/A	speed measurement error
[41]	2	955	induction loops	1280 × 720	speed measurement error
proposed	**18**	**20 865**	**LIDAR gates**	**1920 × 1080**	**calibration error, distance measurement error, speed measurement error, vehicle counting recall, false positives vehicles per minute**

do not work under all conditions. The reported mean error varies greatly – it should be noted that the error is not directly comparable, as it was evaluated by the authors on different datasets (generally not publicly available) and by different protocols.

To sum up the camera calibration methods, some of them [28, 83] do no take perspective projection into account, some algorithms [28, 56, 110, 83, 32] have limitations in camera placement. Quite a large number of approaches [104, 110, 145, 103] use measurements in the scene which enable direct camera calibration. Methods [63, 32] using a calibration pattern (virtual or drawn on the road) have been proposed. Another set of methods use vanishing points to obtain camera calibration [135, 17, 56, DSH14].

Several approaches to scale calibration have been proposed. Besides the multiple manual measurements on the road [104, 110, 145, 103] and calibration patterns [63, 32], two groups of methods exist. The algorithms from the first one [135, 17, 56, 83] use one known distance in the scene (e.g. length of line marking stripe). The other methods use dimensions of vehicles [28, DSH14] to obtain a proper scale calibration.

One important attribute of the calibration methods is whether they work fully automatically and do not require any manual per camera calibration input. The automation helps reduce the cost of camera installation and the automatic methods have better scaling properties. Only two approaches are fully automatic and do not require any manual camera calibration. Both of these methods [28, DSH14] use mean dimension of vehicles to obtain a proper scaling factor for the given camera.

Methods [104, 110, 145, 103, 63, 32] which require measurements of physical dimensions on the road have even more significant drawbacks with respect to the scaling properties. To perform the measurements, it is usually necessary to stop (or limit) traffic on the road increasing installation time and costs.

Another important attribute is whether the camera can be placed at any position above the road, as some methods require for example that the camera has zero pan. In real world scenarios, this can be hard to guarantee when the camera is not placed on a portal above the road. The only method that satisfies the conditions of automatic calibration and arbitrary view is [DSH14] which we use later in the experiments.

6.2.8 *Evaluation Datasets Used in Existing Works*

The described methods usually used different methods for evaluation of the speed measurement and ground truth speed acquisition. Some methods [28, 135, 103, 41] use inductive loops for ground truth acquisition, other methods [104, DSH14] GPS or RADAR [83]. Do et al. [32] used the speedometer on a motorbike, which should be considered very imprecise.

When it comes to the number of evaluated speed measurements, Lan et al. [83] used 2 010 ground truth speeds (only one video sequence), others [28, 135, 41] have hundreds of vehicles with known ground truth. And there are also works [56, 63, 104, 110, 145, DSH14, 83, 103, 32] that use at most tens of ground truth speeds with the lowest number in [32] (one ground truth speed) and the highest number of 75 measurements in [103]. Cathey at al. [17] have no evaluation at all. A summary of existing datasets can be found in Table 6.2. It should be noted that with the exception of [110, DSH14], the datasets are not publicly available which makes comparison of the methods impossible.

Almost every mentioned dataset (except [145] and a part of [63]) is recorded in daylight as the methods usually become unusable in the night when only headlights of vehicles are visible. Existing datasets usually evaluate only speed measurement error (with different statistics – mean, deviation etc.) and some exceptions (see Table 6.2) evaluate also other tasks.

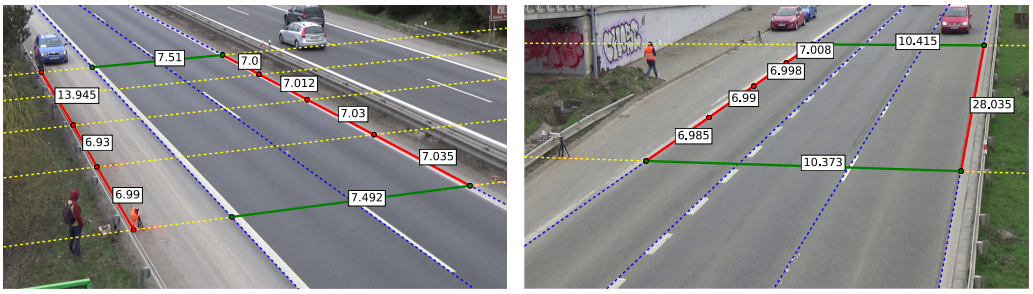


Figure 6.4: Markings and measured distances on the road plane. **Blue dashed lines** – annotated lane dividing lines, **yellow dashed lines** – measurement lines, **red line segments** – measured distances towards the first vanishing point, **green line segments** – measured distances towards the second vanishing point. Images with these annotations for all videos can be found in Appendix B. Best viewed on screen.

The existing evaluation of algorithms should be considered insufficient as existing works use a small number of observed vehicles and scenes. Also, for GPS and speedometer, the ground truth is imprecise as in our evaluation GPS has mean error over 2 % and speedometer reports higher speed then the actual. Therefore, we created our novel dataset with precise ground truth and 20 865 of vehicles with ground truth speed. It is also possible to evaluate other camera calibration aspects such as calibration error and distance measurement on the road plane with the computed scale. These two metrics can provide interesting insights into properties of camera calibration algorithms as they are needed and harnessed in the intelligent transportation surveillance.

6.3 DATASET ACQUISITION METHODOLOGY

We performed six recording sessions at different locations with free flow traffic. For each session, we obtained three videos (approximately one hour long) from different positions by different video cameras (Panasonic HC-X920, Panasonic HDC-SD90, Sony Handycam HDR-PJ410). The videos were recorded in full-HD resolution and with 50 frames per second progressive scan. The recording setup is schematically shown in Figure 6.1 and an example of the scene is in Figure 6.4.

Reference speed values of passing vehicles were obtained from a pair of experimental setups, containing a LIDAR (LaserAce® IM HR 300), a GPS module (Leadtek LR9540D), and a PC. These were placed on the side of the road perpendicular to the direction of traffic flow at a defined distance D between them. It was important to place the lasers to the same height and parallel in the vertical and horizontal axes (see Figure 6.1). This requirement guarantees that an incoming vehicle always disturbs the laser beams at the same point.

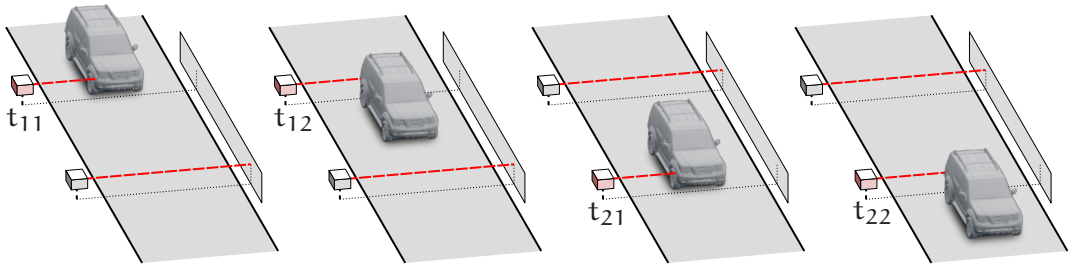


Figure 6.5: The four phases of a passing car which are used for ground truth speed annotation. Best viewed on screen.

The LIDAR works in the single shot mode (one laser pulse per range measurement). The sampling rate is 1 kHz and maximal measurement range is 300 m. GPS receiver synchronizes times on PC using TIMEMARK signal (1 pulse per second with 1 s precision). The data from each LIDAR and GPS module were recorded by the computer and each measurement was assigned with a high resolution timestamp obtained from the operating system.

Distance logs from both LIDARs are pre-processed individually. We search for timestamps t_{xy} (see Figure 6.5) which correspond to car entering/leaving first/second laser beam. And each excitation is assigned with the lane based on the measured distance from the LIDAR. Excitations generated by the same car on the first and second LIDAR need to be matched. The matching is based on the correspondence of lane with limits on speed and acceleration of cars. We calculate immediate speed when entering the first laser v_{11} (at the time t_{11}), length of the vehicle L, and its average acceleration a over measured span of known length D by the following set of equations:

$$v_{11} + \frac{1}{2}a(t_{12} - t_{11}) = \frac{L}{t_{12} - t_{11}} \tag{6.1}$$

$$v_{21} + \frac{1}{2}a(t_{22} - t_{21}) = \frac{L}{t_{22} - t_{21}} \tag{6.2}$$

$$v_{11} + \frac{1}{2}a(t_{22} - t_{11}) = \frac{D + L}{t_{22} - t_{11}} \tag{6.3}$$

Then, it is possible to compute immediate speed at any point of the measured span. Unfortunately, when a car is partially occluded by another vehicle, the equations above cannot be used for the calculation (as some timestamps are unknown). If at least timestamps t_{11} and t_{21} are known, the average speed can be computed as

$$v_{avg} = \frac{D}{t_{21} - t_{11}}. \tag{6.4}$$

As we are using LIDARs (instead of e.g. optical gates), we are able to detect situations when a vehicle is partially occluded by a closer vehicle using the measured distances by the LIDARs. See Figure 6.6 for examples of all possible occlusion types. There are several possibilities of occlusion on the pair of LIDARs:

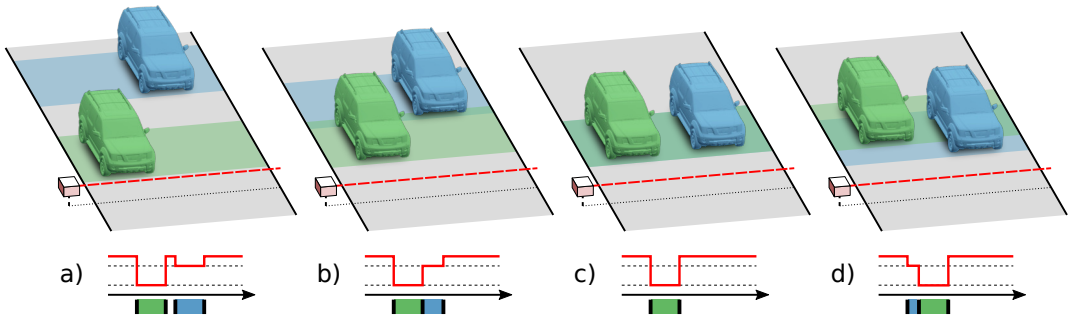


Figure 6.6: Possible variants of occlusion. **a)** the vehicles are not occluded at all, **b)** the closer vehicle is occluding the frontal part of the farther vehicle **c)** the farther vehicle is fully covered, **d)** the farther vehicle's rear part is covered. The graphs below represent LIDAR responses with different levels for empty road, fast lane (top dashed line), and slower lane (bottom dashed lane). See text for description of how all these situations are handled. Best viewed on screen.

1. Occlusion situations on both the LIDARs are either **a)** or **d)** – in these situations we are able to detect that there is a occluded vehicle and measure their speed.

2. Occlusion situations on at least one LIDAR is of type **b)** or **c)** – we are able to detect that there is a second "shadowed" vehicle; the speed measurement is not reliable and the second vehicle is omitted from the dataset and evaluation.

3. Occlusion situations on **both** LIDARs are **c)** – the second "shadowed" vehicle cannot even be detected. This situation is very unlikely, as the vehicle in the fast lane would have to be smaller, precisely aligned, and maintain the same speed as the closer vehicle.

In summary, we either measure the speed accurately, or we know that the speed measurement is not precise and we ignore such a measurement. Therefore, besides the 20 865 vehicles with precise ground truth speed, the dataset contains 2 779 instances of vehicles which are marked as invalid for speed measurement evaluation.

We also performed manual verification of the matched timestamps t_{11} and t_{21} by checking that they correspond to the same vehicle in the video.

6.3.1 *Accuracy of the Acquired Dataset*

The distance D between LIDARs is 28 meters (21 meters in one case), and the LIDARs have 1 kHz sampling rate. The actual value of D for every recording session was measured by handheld laser distance meter, and we assume that upper bound of the distance measurement error is $e_d = 0.05$ m. Time measurement error caused

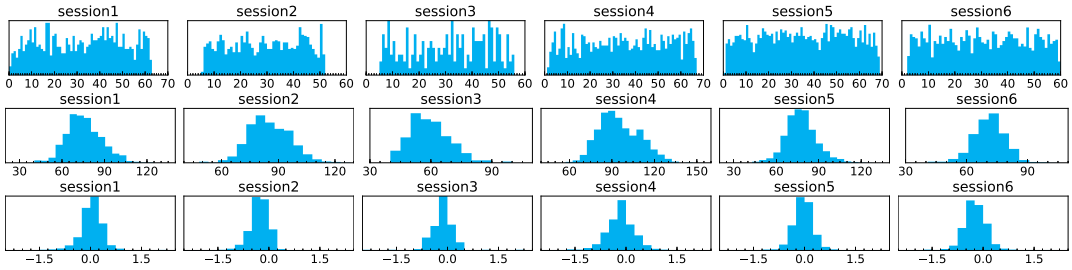
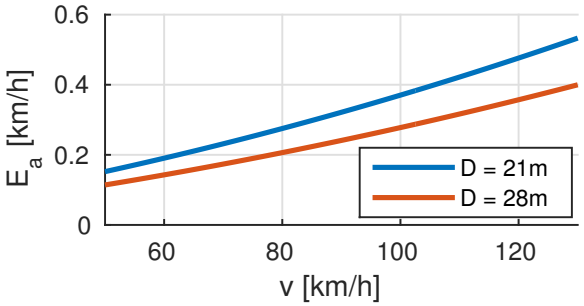


Figure 6.7: **top:** Histograms of density of vehicles for each session; one minute granularity in x-axis. **middle:** Ground truth speeds measured by the LIDAR setup (Section 6.3); speed in km/h on x-axis. **bottom:** Ground truth accelerations; m/s^2 on the x-axis.

by improper synchronization of LIDARs is at most $e_t = 1\,\text{ms}$. Both, e_d and e_t are exaggerated and in reality they are lower. The upper bound of speed measurement error E_r (relative) and E_a (absolute) for the given speed v can be computed as:

$$E_r = \frac{e_d + e_t \cdot v}{D}$$

$$E_a = E_r \cdot v$$



For a vehicle going $v = 20\,\text{m/s}$ (72 km/h), the resulting maximum possible errors are $E_r = 0.25\,\%$ and $E_a = 0.05\,\text{m/s}$ (0.18 km/h). We consider these errors to be small enough as the errors of the methods presented in Section 6.5 are much higher than this error of measurement.

6.4 DATASET STATISTICS AND EVALUATION PROTOCOL

The dataset consists of 18 videos (6 sessions on different locations, 3 videos from different angles for each location) and there is totally 20 865 vehicles with known ground truth speed.

To provide statistics about the dataset we report the total number of cars with ground truth speed for each video in Table 6.3. We also report histograms of speeds, accelerations, and traffic density in Figure 6.7.

The dataset is (to our knowledge) by far larger than other datasets serving similar purpose reported in the literature. It covers views typical for traffic surveillance from arbitrary cameras. It provides high quality videos with various traffic conditions (low traffic in Session 3, high traffic in Sessions 5 and 6). However, it is quite limited in lighting and weather conditions. Almost all videos were taken in cloudy

Table 6.3: Numbers of vehicle passes with known ground truth for each video.

	left	center	right
session1	854	848	849
session2	1 163	1 258	1 583
session3	193	193	193
session4	1 188	1 192	1 177
session5	2 021	2 027	2 030
session6	1 358	1 353	1 358
TOTAL		20 865	

weather (except some parts of Session 3) with no distracting phenomena (fog, rain, etc.).

6.4.1 *Evaluation Protocol*

For future comparison of methods, we provide an evaluation script[2] which automatically evaluates all the used metrics. It requires two vanishing points of the road plane, principal point of the camera and scale of the scene as the calibration parameters. Then, the systems are supposed to report for each observed vehicle a track of one arbitrary reference point on the road plane (frame numbers + image coordinates). In our case, the point is obtained by the constructed 3D bounding boxes (see Figure 6.3). The point must be on the road plane for proper projection; however it can be any point on the road plane which the authors are able to localize – it is not necessary to use the 3D bounding boxes.

To compare vehicles with the ground truth, we match the time when a vehicle passed the measurement line to the time reported by LIDAR and the lane in which the vehicle is. As the vehicles are sometimes not tracked correctly and the tracking can be lost, we extrapolate the vehicle trajectory in order to get the correct time.

For each vehicle, we calculate tentative speeds between the positions K frames apart (approximately 0.1 s, $K = 5$ for 50 fps video) by projecting the image point coordinates to the road plane using the provided calibration. The resulting speed is then median of the tentative speeds. We found out that this method is more robust than measuring the full section speed due to possible tracking errors.

The computation of distance between two points p_1 and p_2 is schematically shown in Figure 6.2 with general model for traffic surveillance camera and it is described in detail in Appendix B.

[2] The evaluation code is available together with the dataset at https://medusa.fit.vutbr.cz/traffic

As methods may require different training sets we define three train/test splits. Split **A** uses all videos for testing, split **B** has Session 1 and Session 2 reserved for training, and finally, split **C** has Session 1, Session 2, and Session 3 for training. Whenever it is possible, the results should be reported on the splitting with the lowest number of training sessions.

6.5 EXPERIMENTS

On the above described dataset we evaluate recent method [DSH14] described in Section 6.2.5; we use this method for the evaluation because it works fully automatically (contrary to [104, 110, 145, 103]) and it is not limited to some viewpoints (contrary to [28, 56, 110, 83, 32]). The method is able to automatically recover camera calibration and scene scale. However, our dataset provides data in the form of measured distances on the road plane usable for computing camera calibration (vanishing points and scene scale). Therefore, we also report the performance of the semi-automatic variants of the method.

We defined labels for different camera calibrations (the vehicle detection and tracking stays the same for all of the methods):

FullACC [DSH14] – unmodified system from [DSH14], as it is Fully Automatic Camera Calibration.

OptScale,**OptScaleVP2** – Keep calibration (vanishing points) from FullACC and calculate optimal scale using lengths in direction to VP1 (OptScale) or VP2 (OptScaleVP2). The scale is computed as a mean of scale values obtained from the distance measurements on the road.

OptCalib,**OptCalibVP2** – The first vanishing point is kept from the FullACC. And as the second vanishing point is selected a point which minimizes the calibration error (see Section 6.5.1). The minimization is done by a grid search in space of feasible vanishing points. The scale is computed the same way as for OptScale and OptScaleVP2.

On these five variants we report the calibration error, distance measurement error, and speed measurement error. The speed error is additionally compared to the GPS speed. The evaluation in this chapter is done on split **A**, as the method does not require any training so we can use all the videos for evaluation. Evaluation for each video separately can be done directly from the published dataset as we included also the results.

As the camera calibration algorithm is the most sensitive part of a speed measurement system, we provide also an evaluation of the calibration itself based on two detected vanishing points and evaluation of distance measurement accuracy

Table 6.4: Errors for distance ratios (see text for details). The first row for each calibration method contains absolute errors and the relative errors in percents are in the second row.

system	mean	median	95 %
FullACC [DSH14], OptScale, OptScaleVP2	0.15	0.04	0.56
	10.89	4.52	40.24
OptCalib, OptCalibVP2	0.03	0.01	0.09
	2.62	1.58	8.79

on the road plane. These two metrics are also important as they compare directly the camera calibrations without any influence of vehicle detection and tracking.

For each presented evaluation metric we propose to report mean, median and 95 percentile and where it is possible (speed measurement error), we also report absolute and relative cumulative histograms of errors. These statics correspond to used evaluation metrics and methods shown in Table 6.2.

6.5.1 *Calibration Error*

The first evaluation experiment is focused on the calibration itself (detected two vanishing points) excluding the scale. We measure the ratio between every pair of distances measured on the road plane (see Figure 6.4) and compare it with the ratio of the dimensions measured using the calibration. The scale is therefore omitted from this evaluation and the results depend only on the two vanishing points.

For each system, we measure mean, median and 95 percentile error for both absolute units ($err = |r_{gt} - r_m|$) and relative units ($err = |r_{gt} - r_m|/r_{gt} \cdot 100\%$), where r_{gt} denotes the ground truth ratio, and r_m represents the measured ratio. This computation of absolute and relative error is used also in Sections 6.5.2 and 6.5.3. The results can be found in Table 6.4. As there are two groups of methods (**FullACC+OptScale+OptScaleVP2** and **OptCalib+OptCalibVP2**) which share the calibration (vanishing points) and are differentiated only in the scale which is not used in this experiment, the results are the same for methods within each group.

The results show that the camera calibration automatically obtained by [DSH14] is far from perfect. The biggest error is caused by inaccurate localization of the second vanishing point (VP2). Thus the lengths in the direction to VP2 (i.e. widths of vehicles) are unreliable for scale computation.

Table 6.5: Errors for distance measurement **towards the first vanishing point** (see text for details). 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	95 %
FullACC [DSH14]	1.41	1.06	4.45
	12.32	12.00	25.13
OptScale	0.23	0.13	1.18
	1.94	1.45	5.05
OptScaleVP2	2.61	1.60	8.53
	21.86	20.21	57.62
OptCalib	0.14	0.09	0.41
	1.43	0.92	3.56
OptCalibVP2	0.34	0.14	1.74
	2.46	1.54	8.05

6.5.2 *Distance Measurement Error*

To evaluate the distance measurement including the scale, we carried out the next experiment where we compared ground truth distances on the road plane (see Figure 6.4) and distances obtained using the camera calibration converted to meters using the scale.

We divided the experiment into two parts. The first one is focused only on distances towards the first vanishing point, as these are the most important for the speed measurement. The results can be found in Table 6.5. In the second part of the experiment, we evaluated all the distances measured on the road plane and the results can be found in Table 6.6.

The results in Table 6.5 show many different interesting aspects of the algorithm. The first one is that if we use the original calibration and use scale computed from distances towards the first vanishing point (**OptScale**) it improves the results significantly. However, when we use the same calibration and scale computed from distances towards the second vanishing point (**OptScaleVP2**), then the results are even worse than the original ones. This implies that the error is in the localization of the second vanishing point. The table also shows that in a situation when we use the correctly localized second vanishing point, the results significantly improve for both **OptCalib** and **OptCalibVP2** relative to **OptScale** and **OptScaleVP2**.

Table 6.6: Errors for **all** distance measurements (see text for details). 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	95 %
FullACC [DSH14]	1.30	0.85	4.47
	12.11	10.91	25.64
OptScale	0.62	0.17	2.47
	6.83	2.05	32.04
OptScaleVP2	1.98	1.21	6.41
	16.84	12.94	49.98
OptCalib	0.14	0.07	0.60
	1.58	0.68	5.24
OptCalibVP2	0.30	0.12	1.12
	2.37	1.44	8.47

Results in Table 6.6 support the hypothesis that the second vanishing point is incorrectly detected by the method [DSH14] in some cases, as the distance measurement performance significantly drops for **OptScale** when distances towards the second vanishing point are added to the evaluation (Table 6.5 vs Table 6.6). Also, it shows that **OptCalib** and **OptCalibVP2** are not affected by this. Also, as we expected, the performance of **OptScaleVP2** increases when the dimensions towards the second vanishing point are added to the evaluation. The original system **FullACC** does not have this significant drop in performance as the scale is determined either from widths, lengths, or heights; so it can be matched to the the correct scale for measurements of the distances towards the second vanishing point.

6.5.3 *Speed Measurement Error*

For the speed measurement task itself, we evaluate mainly the error between the ground truth speed and the measured one. This metric does not include statistics about incorrectly detected and tracked vehicles.

The results on the evaluation videos can be found in Table 6.7 and cumulative histograms of errors are shown in Figure 6.8. To compare the results with another non-visual speed measurement method, we also drove a car with GPS

Table 6.7: Errors for speed measurements (see text for details). The first row for each calibration method contains absolute errors in km/h and the relative errors in percents are in the second row.

system	mean	median	95 %
GPS	1.64	1.19	—
	2.18	1.42	—
RADAR	1.07	0.89	2.69
	1.33	1.14	3.23
FullACC [DSH14]	8.59	8.45	17.14
	10.89	11.41	19.84
OptScale	1.71	1.17	4.69
	2.13	1.51	5.56
OptScaleVP2	15.66	13.09	47.86
	19.83	17.51	59.25
OptCalib	1.43	0.83	3.89
	1.81	1.05	5.07
OptCalibVP2	2.43	1.40	6.66
	3.08	1.76	8.00

system with enabled raw logging to be seen in the recordings multiple times and computed the speed of these observations from the GPS logs by averaging over a longer period. We used nVidia Shield tablets as our GPS loggers in "Device only" mode (GPS localization ON, Wi-Fi and GSM localization OFF) and logged the GPS data in NMEA format using a standard logging application available in the application store. We process offline RMC messages, distance and velocity between each two following points are computed using Haversine formula described by Robusto [126]. For each evaluation video we have approximately 20 passes with the GPS speed measured.

The results in Table 6.7 and Figure 6.8 show that the systems **OptCalib**, **OptScale** and **OptCalibVP2** work relatively well (with **OptCalib** being the best). Also, the results show that the **FullACC** has lower errors than **OptScaleVP2** implying that the biggest problem is in bad localization of the second vanishing point as was described in Section 6.5.2. The results also show that when the original vanishing points from **FullACC** are used and the scale is computed to optimize the error

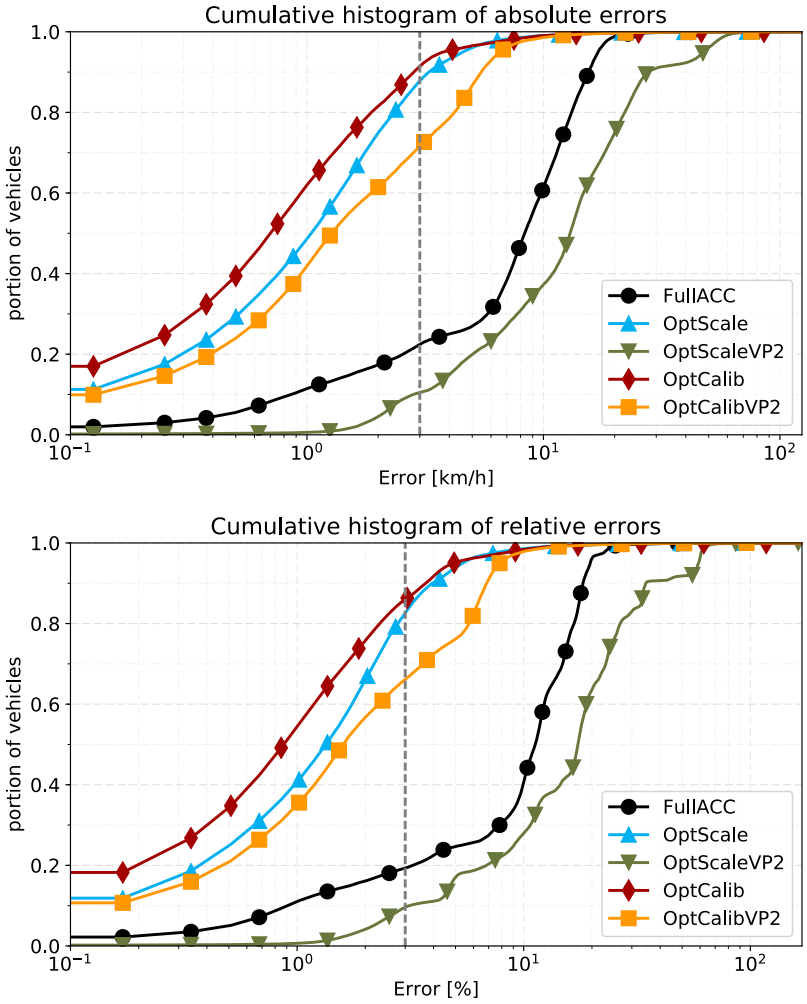


Figure 6.8: Cumulative histograms of distribution of errors. The dashed vertical line represents 3 km/h (or 3 %) threshold. See text for details. (Line markers represent every 1500th data sample.)

in distance measurement towards the first vanishing point (**OptScale**), the performance increase significantly. Also, when the second vanishing point is correctly localized (**OptCalib** and **OptCalibVP2**) the results improves furthermore.

Table 6.7 also shows that the optimal system **OptCalib** outperforms the results obtained from the GPS speed measurements. The 95 percentile is not reported for the GPS as there is a much smaller number of measurements for the GPS than in the other cases; thus the numbers are not comparable. Also, we evaluated the speed measurement done by RADARs (2D microwave FM-CW radar module RFbeam K-MC4 operating in K-band) and we used one RADAR for each lane for each evaluation session in order to compare them with results obtained from the methods using video; see Table 6.7.

We also evaluated the number of false positives per minute of video (9.745) and recall (0.872) for all the videos. In the case of vehicle counting, false positives represent reported vehicle tracks which are not present in the dataset; recall denotes the fraction of correctly matched ground truth vehicle tracks with reported vehicle

tracks. The results are the same for all the systems as they share the vehicle detection and tracking part and they are only different in the camera calibration. The false positives are caused mainly by lost tracking and re-initialization. Another important drawback of the current method is that the motion is not detected correctly in some cases and the motion mask is divided into several contours.

Although the systems with some manual calibration (**OptScale** and **OptCalib**) have relatively low speed measurement errors in comparison with GPS and RADAR, the fully automatic system still has too large errors and the automatic traffic surveillance camera calibration methods need improvement.

6.6 CONCLUSIONS

We collected and processed a dataset for evaluation of purely visual speed measurement by a single monocular camera. Cameras are becoming ubiquitous and a considerable portion of them observe traffic. By providing this dataset we intend to encourage research of fully automatic traffic camera calibration methods, which could be used for mining valuable automatic traffic surveillance data from existing and new camera infrastructure.

On the collected data, we evaluated an approach which is both fully automatic and can process virtually arbitrary views. The evaluation shows its weaknesses (localization of the VP2 and scale inference), which can encourage further research in this area, which we will focus on. The measurements also established a first baseline to be outperformed by future works.

7

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

CITATION 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.

ABSTRACT In this paper, we focus on fully automatic traffic surveillance camera calibration which we use for speed measurement of passing vehicles. We improve over a recent state-of-the-art camera calibration method for traffic surveillance based on two detected vanishing points. More importantly, we propose a novel automatic scene scale inference based on matching bounding boxes of rendered 3D models of vehicles with detected bounding boxes in the image. The proposed method can be used from an arbitrary viewpoint, and thus it has no constraints on camera placement. We evaluate our method on the recent comprehensive dataset for speed measurement BrnoCompSpeed. Experiments show that our automatic camera calibration method by detecting two vanishing points reduces the error by 50 % compared to the previous state-of-the-art method. We also show that our scene scale inference method is more precise (mean speed measurement error 1.10 km/h) outperforming both state-of-the-art automatic calibration method (error reduction by 86 % – mean error 7.98 km/h) and manual calibration (error reduction by 19 % – mean error 1.35 km/h). We also present qualitative results of the proposed automatic camera calibration method on video sequences obtained from real surveillance cameras on various places and under different lighting conditions (night, dawn, day).

7.1 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 [185]) unusable. That is why many solutions place markers to the observed scene and/or measure existing geometric features [145, 32, 172, 102]. These

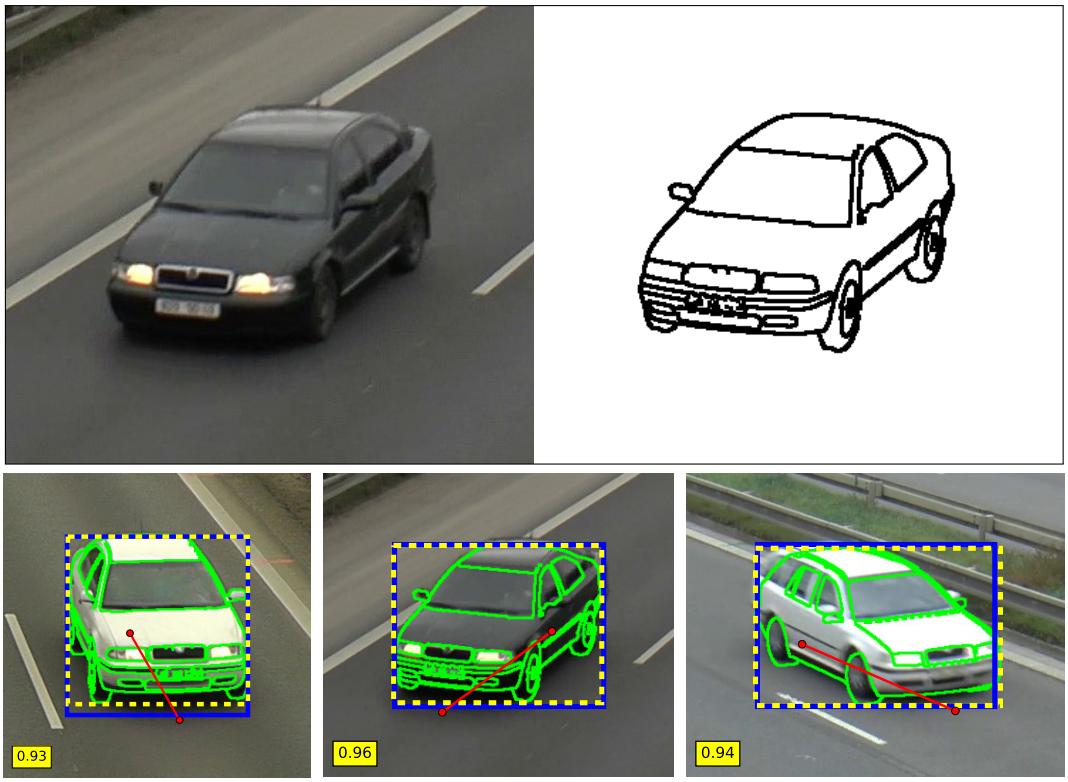


Figure 7.1: Example of detected vehicles and 3D model bounding box aligned to the vehicle detection bounding box. **top:** detected vehicle and corresponding 3D model (edges only), **bottom:** examples of aligned bounding boxes with shown 3D model edges (green), its bounding box (yellow) and vehicle detection (blue).

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 [124], classify them into a few common fine-grained types by a CNN [82] 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 7.1 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.

Existing solutions for traffic surveillance camera calibration [28, 135, 17, 56, 63, 104, 145, 110, DSH14, 83, 103, DHJS15, 32, 102, 172] (see Section 7.2 for detailed analysis) usually have limitations for real world applications. They are either limited to some viewpoints (zero pan, second vanishing point at infinity), or they require some per-installed-camera manual work. To our knowledge, there is only one work [DSH14] which does not have these limitations and therefore we compare our results with this solution. For a brief description of the method, see Section 7.2; a more comprehensive review can be found in a recent dataset paper BrnoCompSpeed by Sochor et al. [SJŠ+18].

The key contributions of this paper are:

- Improved camera calibration method by detection of two vanishing points – camera calibration error reduced by 50 %.
- Novel method for scene scale inference significantly outperforming automatic traffic camera calibration methods (error reduced by 86 % – 7.98 km/h to 1.10 km/h) and also manual calibration method (error reduced by 19 % – 1.35 km/h to 1.10 km/h) in automatic speed measurement from a monocular camera.
- The results show that when used for the speed measurement task, the automatic (zero human input) method can perform better than the laborious manual calibration, which is generally considered accurate and treated as the ground truth. This finding can be important also in other fields than only in traffic surveillance.

7.2 RELATED WORK

The camera calibration algorithm (obtaining intrinsic and extrinsic parameters of the surveillance camera) is critical for the accuracy of vehicle speed measurement

by a single monocular camera, as it directly influences the speed measurement accuracy. There is a very recent comprehensive review of the traffic surveillance calibration methods [SJŠ+18], so for detailed information we refer to the review and we include only a brief description of the methods.

Several methods [63, 17, 56] are based on detection of vanishing points as an intersection of road markings (lane dividing lines). Other methods [DSH14, DHJS15, 135, 28] use vehicle motion to calibrate the camera. Then there is also a set of methods which use some form of manually measured dimensions on the road plane [104, 110, 145, 103, 102, 32, 83].

An important attribute of the calibration methods is whether they are able to work automatically without any manual per-camera calibration input. Only two methods [28, DSH14] are fully automatic and both of them use mean vehicles' dimensions for the camera calibration. Another attribute that is important for real-world deployment is whether the camera can be placed at an arbitrary position above the road, which is not true for some methods as they assume to have zero pan or other constraints.

Regarding fine-grained vehicle classification, there are several approaches. The first one is based on detected parts of vehicles [81, 142, 38], another approach is based on bilinear pooling [92, 47]. There is also an approach based on Convolutional Neural Networks (CNN) and input modification [SHH16]. For object detection, it is possible to use boosted cascades [33], HOG detectors [29], or Deformable Parts Models (DPMs) [40]. Also, there was a recent advancement in object detection based on CNNs [52, 124, 96].

Several authors dealt with alignment of 3D models and vehicles and used this technique for gathering data in the context of traffic surveillance. [93] propose to jointly optimize 3D model fitting and fine-grained classification, [67] align edges formulated as Active Shape Model [25, 88]. [80] and propose to use synthetic data to train geometry and viewpoint classifiers for 3D model and 2D image alignment. [121] use detected SIFT features [101] to align 3D vehicle models and the vehicle's observation. They use the alignment mainly to overcome vehicle appearance variation under different viewpoints. However, in our case, as the precise viewpoint on the vehicle is known (Section 7.4.3), such alignment does not have to be done. Thus we adopt much simpler and more efficient method based on 2D bounding boxes – simplifying the procedure considerably without sacrificing the accuracy.

When it comes to camera calibration in general, various approaches exist. The widely used method by Zhang [185] uses a calibration checkerboard to obtain intrinsic and extrinsic (relative to the checkerboard) camera parameters; [95] use controlled panning or tilting with stereo matching to calibrate the camera. Correspondences of lines and points are used by Chaperon et al. [21]. Yu et al. [173]

focus on automatic camera calibration for tennis videos from detected lines on the tennis court.

7.3 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 7.2 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

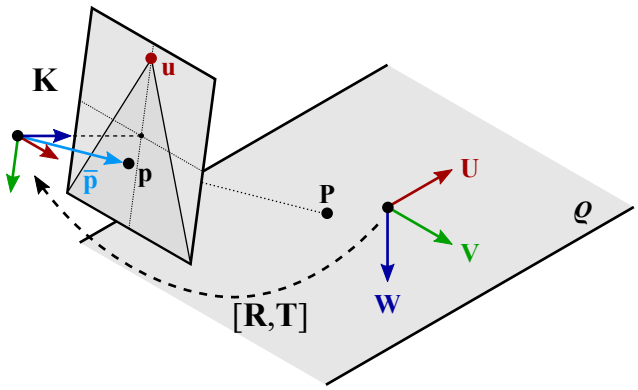


Figure 7.2: Camera model and coordinates. Points denoted by small letters represent points in image space while points in the world space on the road plane ρ are represented by capital letters. The representation stays the same for both finite and ideal points.

the vanishing points) and therefore, we add arbitrary value $\delta = 1$ as the constant term in Equation (7.6).

$$f = \sqrt{-\mathbf{u}^T \cdot \mathbf{v}} \tag{7.1}$$

$$\overline{\mathbf{u}} = [u_x, u_y, f]^T \tag{7.2}$$

$$\overline{\mathbf{v}} = [v_x, v_y, f]^T \tag{7.3}$$

$$\overline{\mathbf{w}} = \overline{\mathbf{u}} \times \overline{\mathbf{v}} \tag{7.4}$$

$$\mathbf{n} = \frac{\overline{\mathbf{w}}}{\|\overline{\mathbf{w}}\|} \tag{7.5}$$

$$\rho = \left[\mathbf{n}^T, \delta\right]^T \tag{7.6}$$

With known road plane ρ, 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:

$$\overline{\mathbf{p}} = [p_x, p_y, f]^T \tag{7.7}$$

$$\mathbf{P} = -\frac{\delta}{\left[\overline{\mathbf{p}}^T, 0\right] \cdot \rho} \overline{\mathbf{p}} \tag{7.8}$$

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 [186, 44, 187], therefore we refer to them.

7.4 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 [124] and tracked by a combination of background subtraction and Kalman filter [75] 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 7.5.

7.4.1 *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 7.5.3 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 [140] and the KLT tracker [154].

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

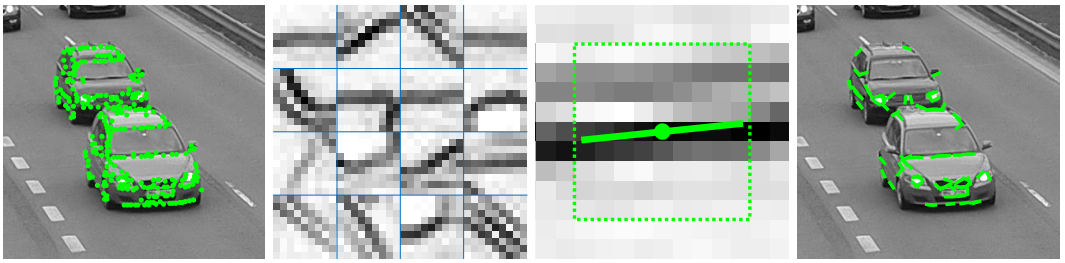


Figure 7.3: Visualization of edgelet detection. From left to right – Seed points $\mathbf{s}_i$ as local maxima of image gradient (foreground mask was used to filter interesting areas); Patches gatherded around the seed points from which is computed the edge orientation; Detail of an edgelet and its orientation superimposed on the gradient image; Top 25 % of edgelets detected in the image.

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 7.3.

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 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:

$$\mathbf{X}_i = \begin{bmatrix} w_1(m_1 - x_i) & w_1(n_1 - y_i) \\ w_2(m_2 - x_i) & w_2(n_2 - y_i) \\ \vdots & \vdots \\ w_k(m_k - x_i) & w_k(n_k - y_i) \end{bmatrix} \tag{7.9}$$

where $[m_k, n_k, 1]^T$ are coordinates of the neighboring pixels ($k = 1 \ldots 81$) 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 (7.10), singular vectors and values of $\mathbf{X}_i$ can be computed as:

$$\mathbf{W}_i \mathbf{\Sigma}_i^2 \mathbf{W}_i^T = \mathrm{SVD}\left(\mathbf{X}_i^T \mathbf{X}_i\right), \tag{7.10}$$

where

$$\mathbf{W}_i = [\mathbf{a}_1, \mathbf{a}_2] \tag{7.11}$$

$$\mathbf{\Sigma}_i = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}. \tag{7.12}$$

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 (7.11) and the edge quality is the ratio of singular values $q_i = \frac{\lambda_1}{\lambda_2}$ from (7.12). Each edgelet is then represented as a triplet $\mathcal{E}_i = (\mathbf{s}_i, \mathbf{d}_i, q_i)$.

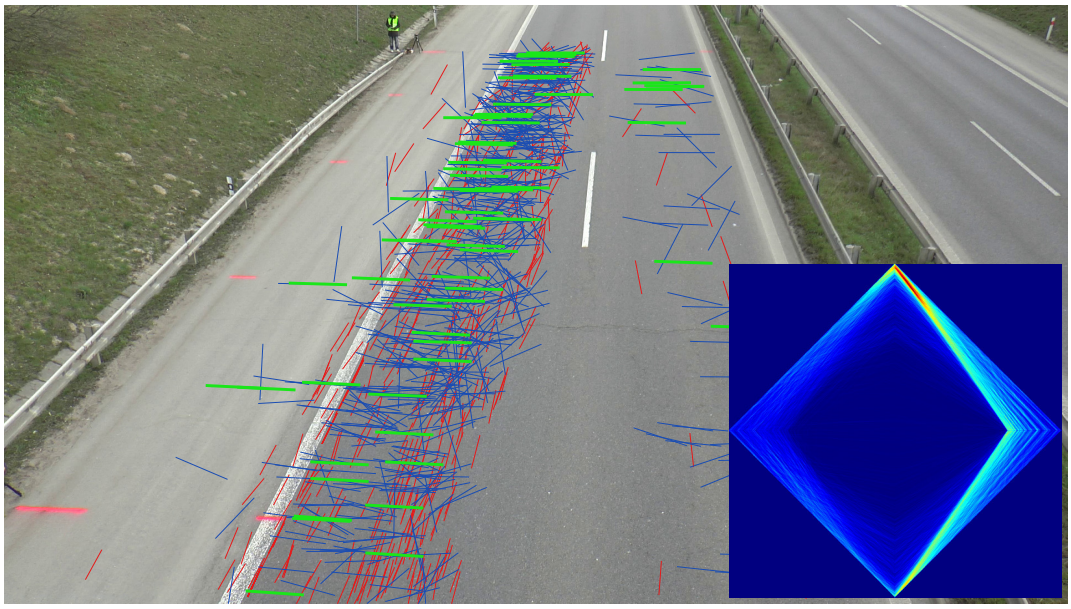


Figure 7.4: Visualization of edges gathered from a video – **(red)** edges that pass close to the first vanishing point, **(blue and green)** edges accumulated to the Diamond Space, and **(green)** edges supporting the detected second vanishing point. The corresponding Diamond Space is shown in bottom-right corner.

We gather the edgelets from the input video (see Figure 7.4), keeping only the strong ones which do not coincide with already estimated $\mathbf{u}$, and accumulate them to the Diamond Space accumulator [35]. 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 (7.1)), 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.

7.4.2 *Vehicle Detection and Tracking*

During the speed measurement, passing cars are detected in each frame by the Faster-RCNN (FRCN) detector [124] but any detector can be used as well (e.g. ACF, LDCF [33]). We trained the detector on COD20K dataset [74] 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

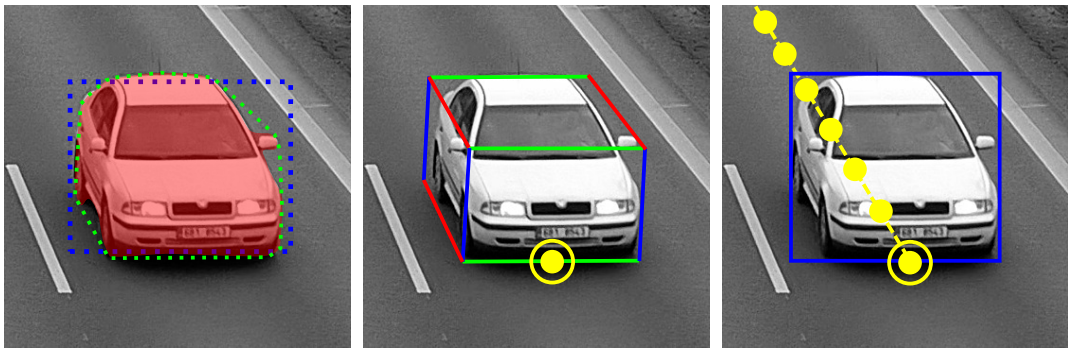


Figure 7.5: Car detection and tracking. From left to right: Car detected by FRCN (blue), its foreground mask and convex hull (green); 3D bounding box constructed around the convex hull and tracking point on the bottom front edge; Car bounding box (from the convex hull) tracked by Kalman filter.

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 an example, see Figure 7.5.

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 [168], or semantic segmentation methods such as [100]). In Section 7.5.6, we are showing a number of examples of real-world surveillance cameras under bad conditions, where the calibration algorithm works well.

7.4.3 *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 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 [82] 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 7.6. 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 7.7. 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. (7.13), where l_{t_i} is the real

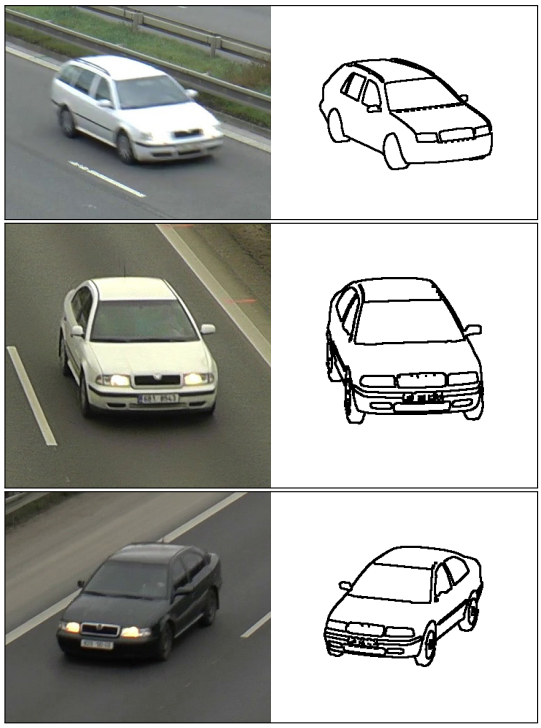


Figure 7.6: Examples of used 3D models (showing only edges) render under the same viewpoint as the corresponding real vehicle on the road. The left image show model which we will refer as Combi and the other two images show 3D model Sedan. Both the models are for Skoda Octavia mk1 which is common on the observed streets.

world length of the type t_i). For all considered combinations of i and j, the IoU matching metric m_{ij} is computed.

$$\lambda_{ij} = \frac{l_{t_i}}{\|\mathbf{F}_{ij} - \mathbf{R}_{ij}\|} \tag{7.13}$$

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 (7.14). The probability $p\left(\lambda \,|\, (\lambda_{ij}, m_{ij})\right)$ is computed by kernel density estimation with a discretized space.

$$\lambda^* = \arg\max_\lambda \; p\left(\lambda \,|\, (\lambda_{ij}, m_{ij})\right) \tag{7.14}$$

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,

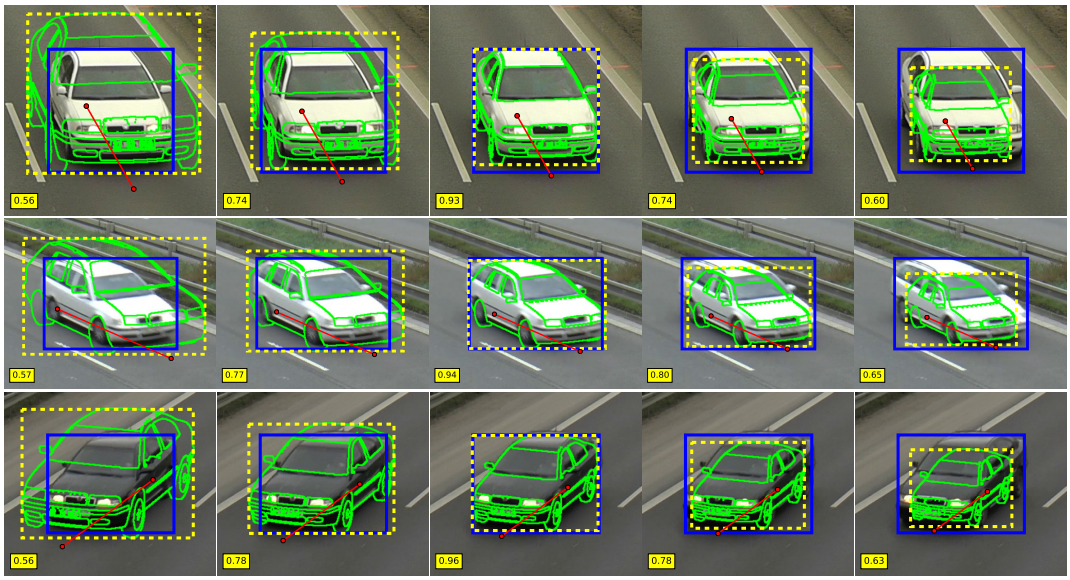


Figure 7.7: Development of IoU (yellow boxes) metric for different scales (**left to right**), vehicle types and viewpoints (**top to bottom**). The left two images show larger rendered vehicle, the middle one show the best match, and the right two images show smaller rendered vehicle. The rendered vehicle 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.

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 [124] 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.

7.4.4 *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 $i = 1 \dots N$, the speed v is calculated from Equation (7.15) by projecting the reference points $\mathbf{p}_i$ to the ground plane $\mathbf{P}_i$ (see Equation (7.8)).

$$v = \underset{i=1\dots N-\tau}{\text{median}} \left(\frac{\lambda^*_{reg} \|\mathbf{P}_{i+\tau} - \mathbf{P}_i\|}{t_{i+\tau} - t_i} \right) \tag{7.15}$$

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).

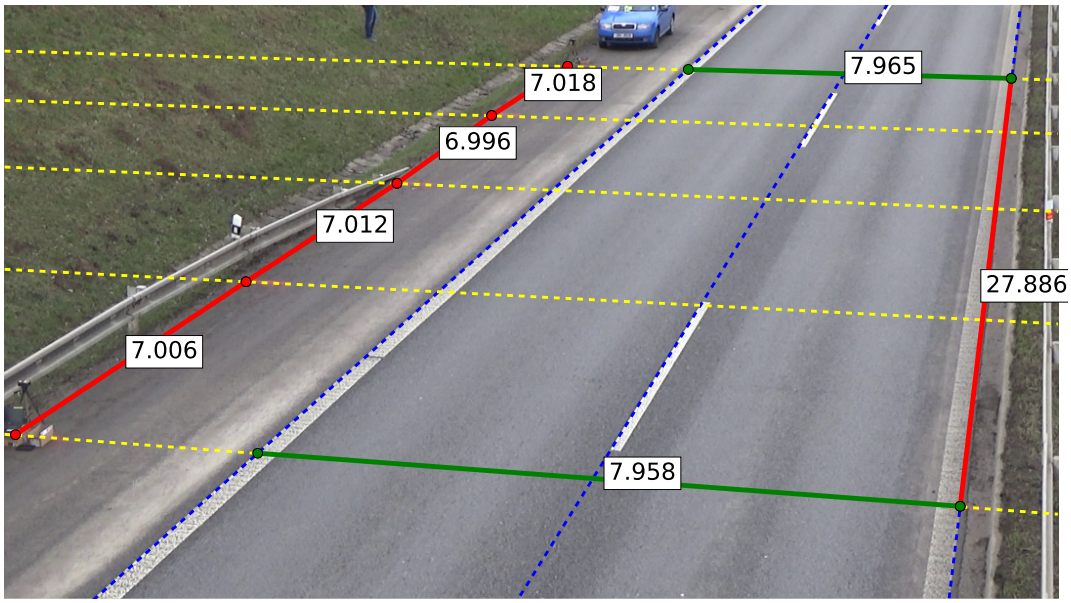


Figure 7.8: An example of manually measured distances between markers on the road plane. Other examples can be found in the original BrnoCompSpeed publication [SJŠ+18]. Blue lines denote the lane dividing lines, lines perpendicular to the vehicles direction are shown in yellow. Finally, measured distances between two points towards the first (second) vanishing point are shown by red (green) color.

7.5 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 7.8. 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]. Brief outline of the method is in Sections 7.2 and 7.4.1.
- **Edgelets** – Camera calibration method proposed in this paper, Section 7.4.1.
- **ManualCalib** – We use known distances (Figure 7.8) on the road for manual calibration of the camera. In agreement with the previous papers [17, 56, 64] we use intersection lanes dividing lines (blue dashed lines in Figure 7.8) 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 $\mathbf{u}$ by solving $\mathbf{A}\mathbf{u} = -\mathbf{b}$ in a least squares manner, where rows of $\mathbf{A}$ contain transposed normal vectors of the lines and rows of $\mathbf{b}$ contain constant terms of the lines.

The second vanishing point $\mathbf{v}$ can be obtained in the same manner (as the intersection of yellow dashed lines in Figure 7.8, 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 (7.16).

$$\mathbf{v}^* = \arg\min_{\mathbf{v}} \left(\sum_{(\mathbf{p}_1, \mathbf{p}_2, d) \in \mathcal{D}_2} |\lambda \|\mathbf{P}_1 - \mathbf{P}_2\| - d| \right), \tag{7.16}$$

where set $\mathcal{D}_2$ contains image endpoints and distances measured on the road towards the second vanishing point (green line segments in Figure 7.8) and scale λ is computed for the given vanishing points $\mathbf{u}, \mathbf{v}$ by Equation (7.17). 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 (7.8) 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 (7.16)).

The usage of standard manual methods based on calibration patterns (e.g checkerboards) proposed by [185] 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]. Brief outline of the method is in Section 7.2.
- **BBScale + reg** – Our method for scale calibration using bounding box matching (Section 7.4.3) 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 (7.17) 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 7.8).

$$\lambda = \mathbb{E}\left[\frac{d_i}{\|\mathbf{P}_{i,1} - \mathbf{P}_{i,2}\|} \right] \tag{7.17}$$

- **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 (7.18) 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.

$$\lambda = \mathbb{E}\left[\frac{\hat{v}_i}{v_i}\right] \tag{7.18}$$

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.

7.5.1 *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 7.8) and the distances between markers towards the second vanishing point (green lines in Figure 7.8). 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 7.1) 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 $\mathbf{v}$; position of $\mathbf{u}$ 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.

Table 7.1: Errors of distance measurement ratios (see Section 7.5.1 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

Table 7.2: Distance measurement errors on the road plane for different calibrations. Only distances towards the first vanishing point (red in Figure 7.8) 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

7.5.2 *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 7.8).

Table 7.3: 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

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 measurement. The results are shown in Table 7.2 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 7.3). 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 (7.16) 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

Table 7.4: 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

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).

7.5.3 *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 (see Section 7.5) in all experiments so that the results for different calibrations and scales are directly comparable.

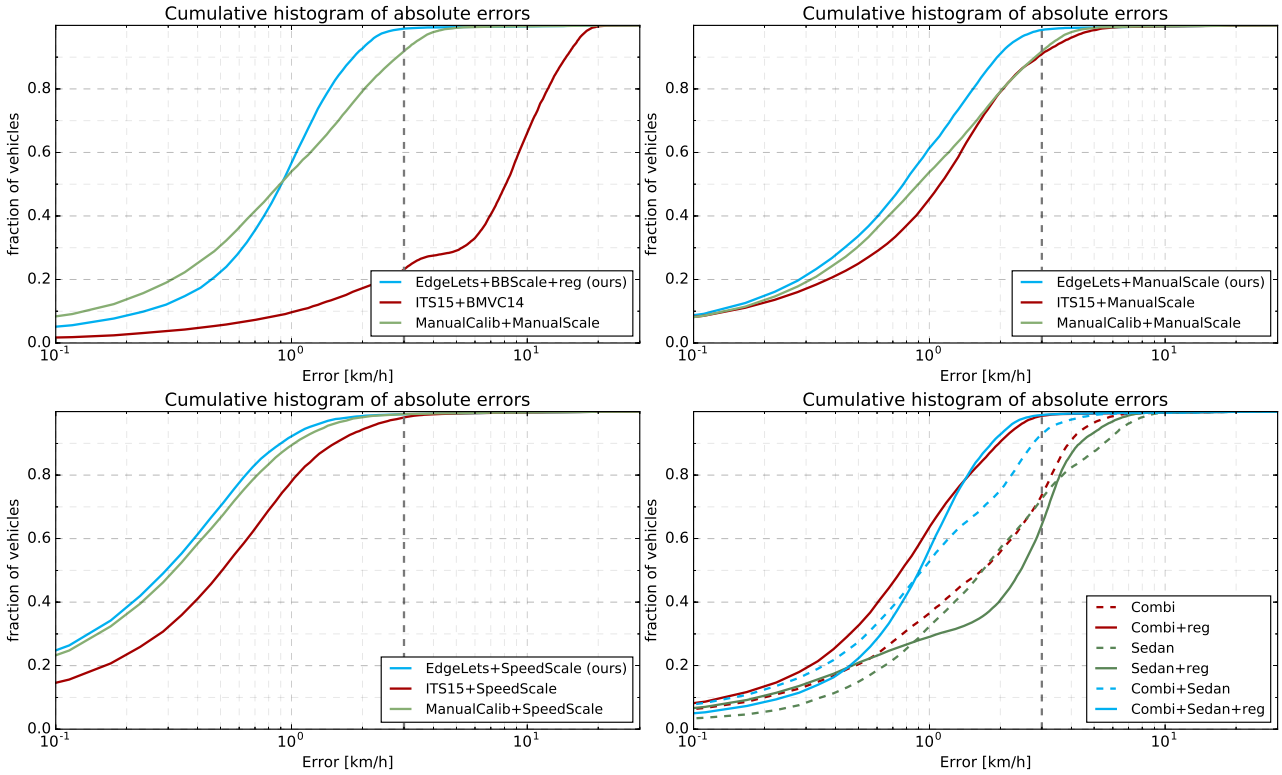


Figure 7.9: Evaluation of speed measurement – cumulative histograms of errors. The gray dashed vertical lines represent 3 km/h error. **top left:** comparison of automatic methods and a manual method for camera calibration, **top right:** calibrations obtained with known ground truth distances on the road plane, **bottom left:** calibrations with scale obtained by minimizing the speed measurement error, thus forming a lower bound error for speed measurement with given camera calibration and tracking algorithm, **bottom right:** analysis of influence of different aspects of used 3D car models evaluated on speed measurement, see Section 7.5.4. The cumulative histogram is suitable for directly obtaining the "success rate" for a given error tolerance.

We show both quantitative results in the form of Table 7.4 and plots with cumulative error histograms in Figure 7.9. The table and the figures are 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 7.5.1) and distance measurement (Section 7.5.2), 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.

7.5.4 *Sensitivity to Selection of the 3D Model*

We also evaluated how using different 3D models of vehicles influences the speed measurement results. The results are shown in Table 7.5 and Figure 7.9 (bottom right). We tested several combinations of used vehicles: use of only one of the models (Combi, Sedan) or both of them together (Combi + Sedan), forming the first segment of the table. It shows that using both the models significantly improves the results, as the errors in geometry of the 3D models cancel out. We consider using only a few (as few as two) fine-grained models as beneficial because it is not necessary to obtain more 3D models and training data for fine-grained recognition. The experiments show that having two models is sufficient for obtaining usable results; using more than two models in practice would follow the same principles and could increase the robustness further.

The second segment of the table shows the performance of the system with scale correction regression to overcome the inaccuracies of the 3D models. The results show that for model Combi, the error significantly decreases. However, for the Sedan model, the results stay more or less the same. This paradox is caused by the smaller number of training data for Sedan version as for some training videos, no Sedan vehicle was detected. The results also show that if we use both models, the performance drop is not that significant (1.10 km/h to 1.38 km/h) and therefore, it is possible to use the scale inference without the scale correction regression.

Table 7.5: Analysis of influence of different aspects of used 3D car models. It shows that it is best to use both models. The second segment of the table also shows that it is useful to use scale correction regression as described in Section 7.4.3. The first row for each 3D model combination method contains absolute errors in km/h; the relative errors in percents are in the second row.

system	mean	median	99 %
Sedan	2.39	1.74	8.67
	2.82	2.14	7.74
Combi	2.03	1.72	6.51
	2.48	2.14	5.94
Combi + Sedan	1.38	0.99	5.18
	1.70	1.23	4.94
Sedan + reg	2.43	2.49	7.26
	2.97	3.17	6.56
Combi + reg	1.03	0.82	3.29
	1.33	1.04	4.49
Combi + Sedan + reg	1.10	0.97	3.05
	1.39	1.22	4.13

7.5.5 *Vehicle Detection and Tracking Evaluation*

Since we use a different vehicle detection and tracking method than [DSH14], we evaluate also this part of the solution. We compare the methods on all videos of BrnoCompSpeed (including extra sessiono) with exactly the same calibration (ManualCalib + ManualScale) to isolate the influence of vehicle detection and tracking.

We report the number of False Positives Per Minute and mean recall in vehicles counting. The results can be found in Table 7.6 and as the table shows, our method considerably reduces the number of false positives with essentially the same recall.

A tracked vehicle is matched to the ground truth if it passes through the correct lane and the time difference of pass through the measurement line (yellow line in Figure 7.8 which is closest to the camera) compared to the ground truth is less than 0.2 s. This threshold is used by [SJŠ+18] to safely match the vehicles as a higher threshold could lead to mismatches between the detected track and ground truth.

As we use the same calibration, we can also compare directly the speed measurement error which is influenced (with the same calibration) only by the tracking. As

Table 7.6: Evaluation of differences between vehicle detection and tracking proposed by [DSH14] and our detection and tracking method. FPPM denotes the number of False Positives Per Minute, recall was computed as mean recall across all videos and speed error denotes mean speed measurement error.

method	FPPM	recall	speed error [km/h]
[DSH14]	9.77	0.885	1.46
ours	1.91	0.863	1.21

the table shows, our tracking method yields slightly reduced speed measurement error for the same scale and camera calibration.

For the tracking and speed measurement, we use the point at the front of the vehicle on the road plane (using 3D bounding box), which is geometrically correct, as the point is on the road plane. We evaluated how the choice of the tracking point influences the measurement error, comparing to a naive solution which takes the center of the bottom edge of the 2D bounding box for the tracking, and we found out that the difference to the correct solution was negligible.

7.5.6 *Camera Calibration on Real Surveillance Cameras*

The automatic calibration from vehicle movement can be justifiably suspected of requiring idealized conditions and to be sensitive to bad lighting, etc. In order to verify the usability of our camera calibration method in real-world conditions, we obtained data from surveillance cameras in production use at 9 different locations. The videos were captured both at day and night conditions. The data are rather of a poor quality (704×576 px or 704×288 px) with 6 frames per second and mean length of 40s. As the ground truth calibration is not available for the data, we report only qualitative results in the form of equilateral grid projected on the road plane. Despite the challenging character of the sequences (poor video quality and lighting conditions), we were able to correctly detect the vanishing points, as can be seen in Figure 7.10 on a few examples, and thus find the camera parameters and its orientation, which is important in many real-world surveillance applications (e.g estimation of vehicle viewpoints or image rectification).

7.6 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

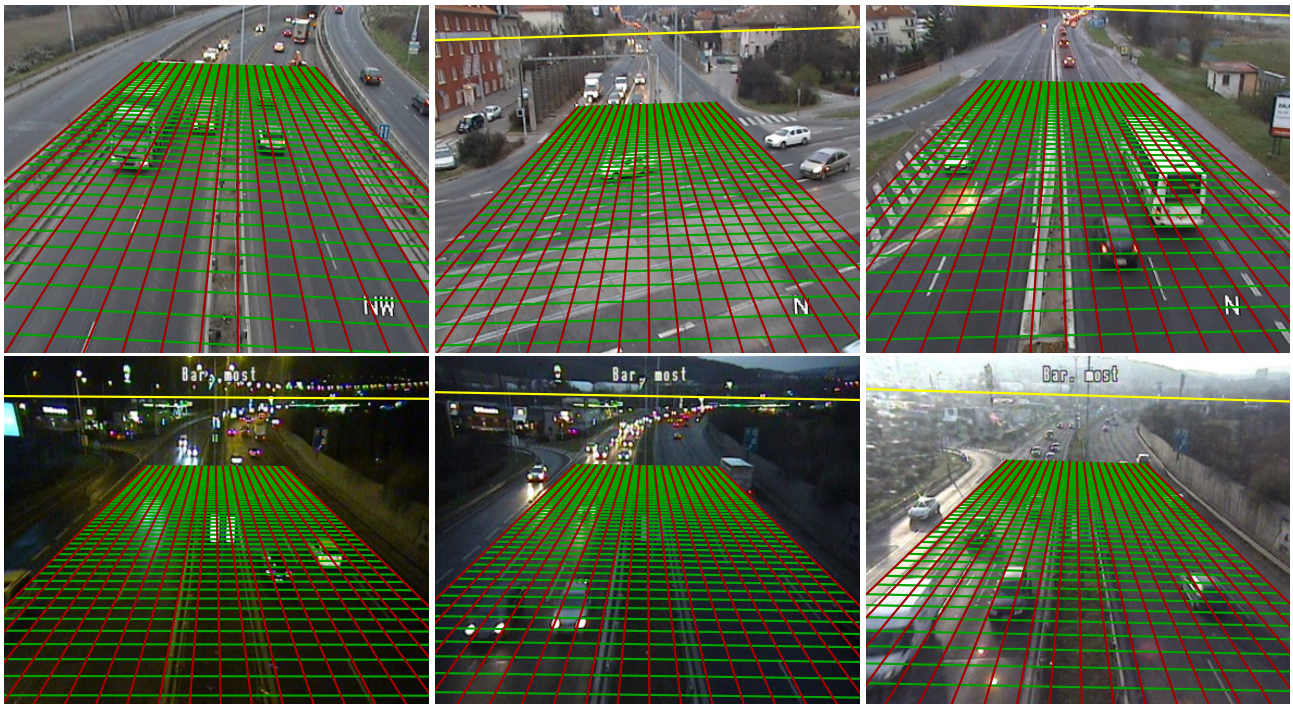


Figure 7.10: Example of camera calibration (two vanishing points) for real world surveillance cameras. The first row shows different locations, while the second one show the same locations at night, dawn, and daylight. The yellow line denotes the detected horizon (if present inside the frames) and red-green grid is formed by lines going to the first vanishing point (red) and to the second one (green). In an ideal case the grid is perpendicular in the real world and the lines are parallel to the features which define the vanishing points on the ground (e.g. line marking). Also, it should be noted that the method is able to work even on an intersection (top center).

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.

Part IV

CONCLUSIONS

8

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 [82, 143, 62, 47]. 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.

Part V

APPENDICES

A

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

A.1 ADDITIONAL BOXCARS116K DATASET STATISTICS

Table A.1: **Top:** Statistics of our new *BoxCars116k* dataset. **Bottom:** Statistics about splits with different difficulty (*hard* and *medium*).

# tracks	27 496
# samples	116 286
# cameras	137
# make	45
# make & model	341
# make & model & submodel	421
# make & model & submodel & model year	693

	hard	medium
# classes	107	79
# train+val cameras	81	81
# test cameras	56	56
# training tracks	11 653	12 084
# training samples	51 691	54 653
# validation tracks	637	611
# validation samples	2 763	2 802
# test tracks	11 125	11 456
# test samples	39 149	40 842

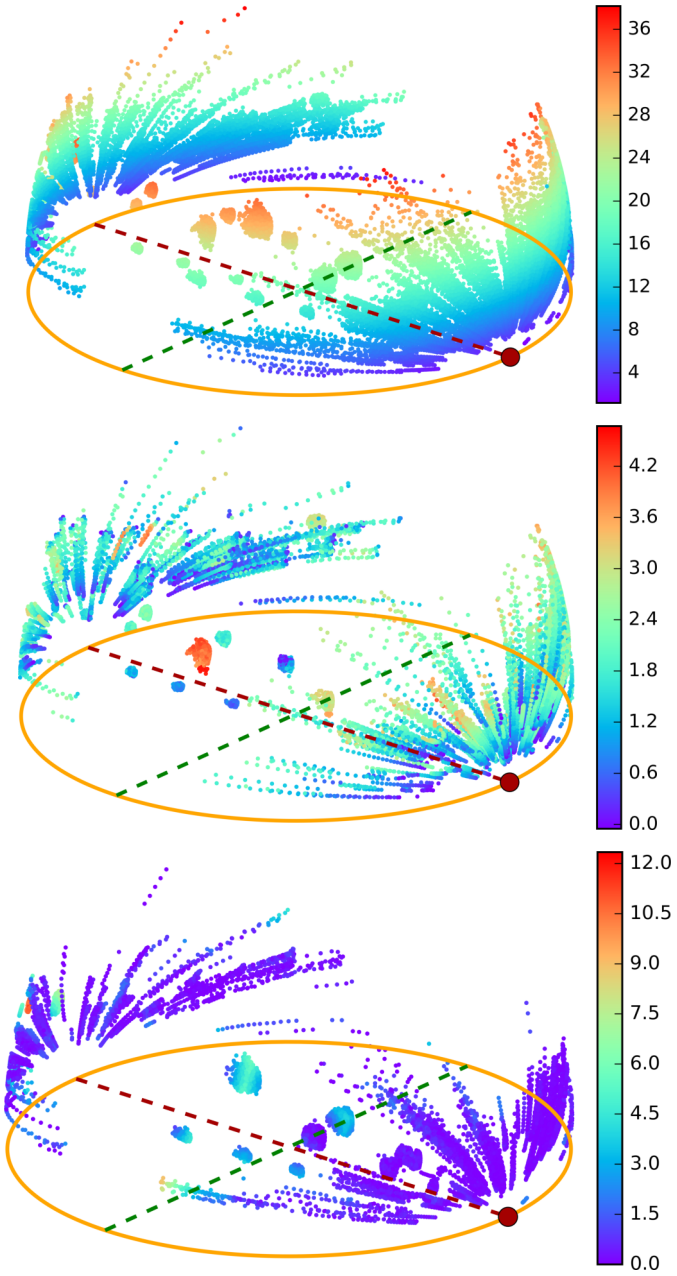


Figure A.1: Viewpoints to dataset samples (horizontal flips are not included). Red dot on the unit circle denotes the frontal viewpoint. **Top:** all samples with elevation color coding (in degrees), **middle:** training samples for hard split with color coded by 2D BB area (in thousands of pixels), **bottom:** test samples for hard split color coded by angle to the nearest training viewpoint sample (in degrees).

Table A.2: Comparison of accuracy with all types and 8 merged types into supertypes.

	accuracy [%]	
net	all types	merged types
AlexNet + ALL	77.79/88.60	79.08/89.70
VGG16 + ALL	84.13/92.27	85.42/**93.28**
VGG16+CBL + ALL	75.06/83.42	76.82/85.07
VGG19 + ALL	84.12/92.00	**85.51**/92.97
VGG19+CBL + ALL	75.62/83.76	78.56/86.62
ResNet50 + IMAGE	82.27/90.79	83.51/91.79
ResNet101 + IMAGE	83.41/91.59	84.65/92.55
ResNet152 + IMAGE	83.74/91.71	85.10/92.84

Figure A.2: Example of vehicle types merged into one supertype. **Left:** Renault Traffic, **right:** Opel Vivaro.

A.2 ADDITIONAL EXPERIMENTAL DATA

A.2.1 *Vehicle Types Resisting to Fine-Grained Recognition*

As possible applications of the fine-grained recognition may vary, we merged pairs of fine-grained classes during testing into one supertype. The merge was done for vehicles which are made by the same concern, have the same dimensions and shape, and which are only differentiated by subtle branding details on the mask. This merge can be beneficial if the task is for example determining the dimensions of the vehicle.

We merged 8 pairs of vehicle types (see Figure A.2 for an example) affecting 1 034 tracks and 5 567 image samples. We show the results in Table A.2; the accuracy improves only slightly – by ~ 1 percent point.

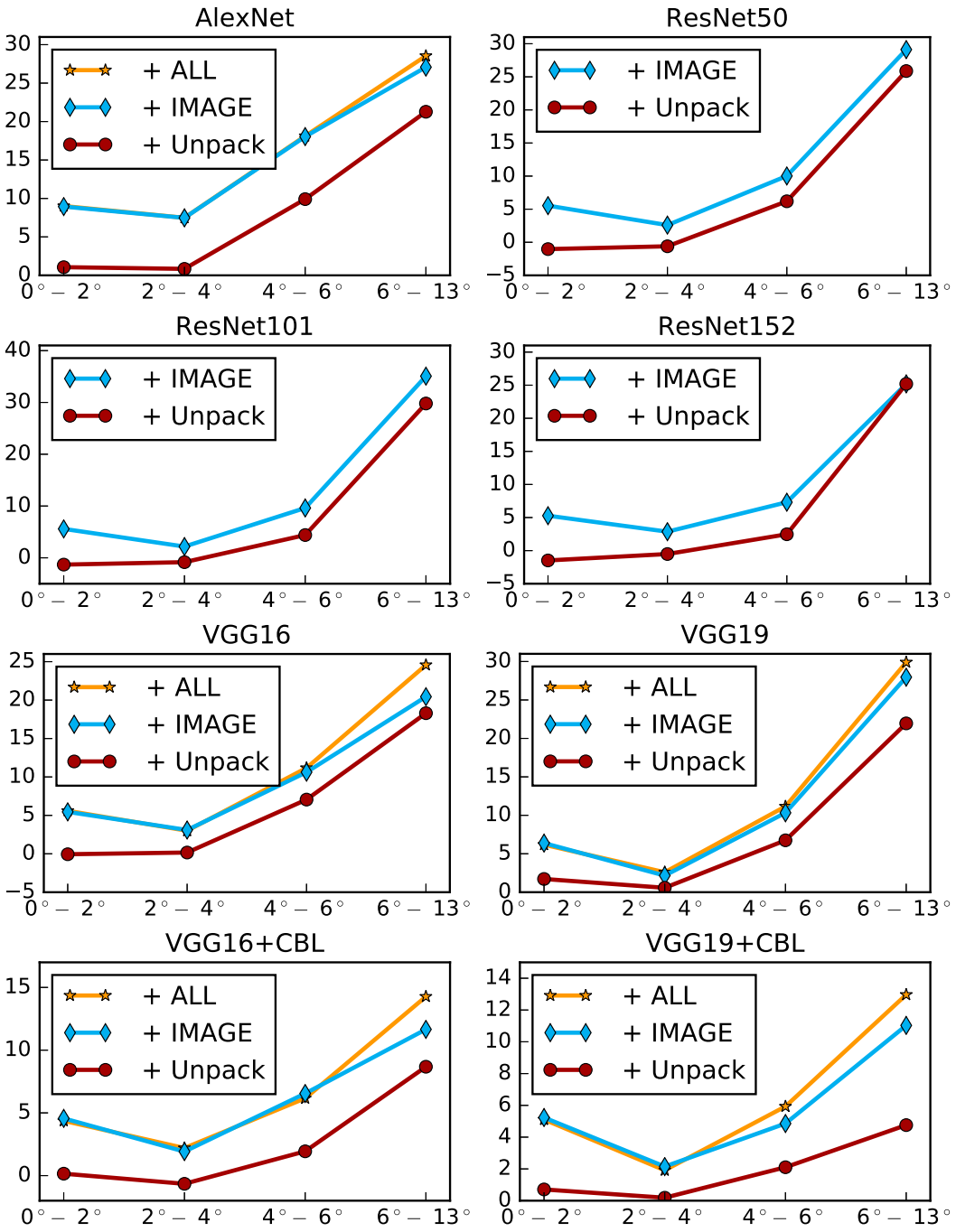


Figure A.3: **All results for Figure 5.10 of the main document.** Correlation of improvement relative to CNNs without modification with respect to train-test viewpoint difference. The x-axis contains bins viewpoint difference bins (in degrees), and the y-axis denotes improvement compared to base net in percent points. The graphs show that with increasing viewpoint difference, the accuracy improvement of our method increases.

Table A.3: **Raw data for Table 5.4 of the main document.** Improvements for different nets and modifications computed as [*base net* + *modification*] − [*base net*], where [...] stands for the accuracy of the classifier described by its contents.

	AlexNet	VGG16+CBL	VGG19+CBL	VGG16	VGG19	**mean**	**best**
Unpack	+3.47/+4.37	+0.69/+1.06	+1.02/+1.31	+2.07/+2.51	+3.29/+3.48	+2.11/+2.55	+3.47/+4.37
View	−0.96/−1.20	−0.19/−0.19	+0.19/+0.31	−0.46/−0.93	−0.19/+0.26	−0.32/−0.35	+0.19/+0.31
Rast	−0.80/−1.18	+0.30/+0.27	+0.28/+0.72	−0.20/−0.08	+0.28/+0.09	−0.03/−0.04	+0.30/+0.72
Color	+4.80/+3.60	+2.08/+0.97	+2.47/+1.65	+2.72/+1.38	+3.79/+2.55	+3.17/+2.03	+4.80/+3.60
ImageDrop	+0.05/−0.47	+0.29/−0.43	+1.53/+0.96	+0.63/+0.07	+1.00/+0.84	+0.70/+0.20	+1.53/+0.96

Table A.4: **Raw data for Table 5.5 of the main document.** Improvements for different nets and modifications computed as [*base net* + *all*] − [*base net* + *all* − *modification*], where [...] stands for the accuracy of the classifier described by its contents.

	AlexNet	VGG16+CBL	VGG19+CBL	VGG16	VGG19	**mean**	**best**
Unpack	+6.93/+7.60	+2.18/+2.22	+2.06/+2.32	+2.82/+2.46	+3.07/+2.82	+3.41/+3.48	+6.93/+7.60
View	+0.09/+0.18	−0.41/−0.19	−0.78/−0.64	+0.36/+0.15	+0.05/−0.27	−0.14/−0.15	+0.36/+0.18
Rast	+0.22/+0.17	+0.11/−0.08	−0.76/−0.58	+0.30/+0.20	−0.01/−0.11	−0.03/−0.08	+0.30/+0.20
Color	+6.34/+6.18	+2.54/+1.28	+2.21/+1.31	+3.08/+1.73	+2.92/+1.67	+3.42/+2.43	+6.34/+6.18
ImageDrop	+1.07/+0.79	+4.24/+3.54	−0.79/−1.21	+0.89/+0.05	+1.19/+0.68	+1.32/+0.77	+4.24/+3.54

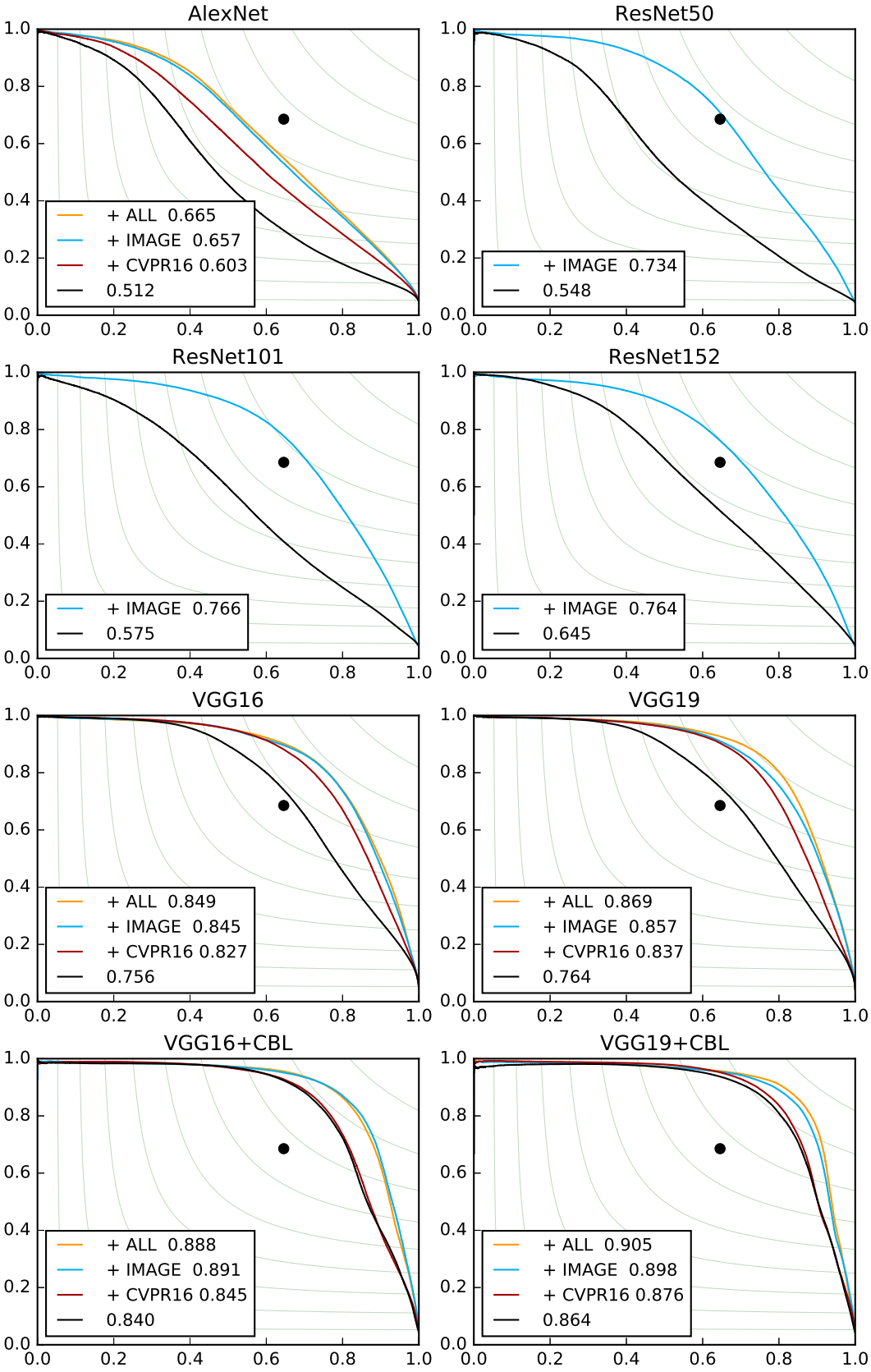


Figure A.4: **All results for Figure 5.12 of the main document.** Precision-Recall curves for verification of fine-grained types. Black dots represent the human performance.

Table A.5: **Raw data for Table 5.1 of the main document.** Improvements of our proposed modifications for different CNNs. The accuracy is reported as single sample accuracy/track accuracy. We also present improvement in percentage points and classification error reduction in the same format.

SPLIT: MEDIUM	accuracy [%]	improvement [pp]	error reduction [%]
AlexNet + IMAGE	77.77/88.16	+12.09/+11.64	35.21/49.57
AlexNet + ALL	77.52/87.52	+11.84/+10.99	34.49/46.82
AlexNet + CVPR16	70.90/82.18	+5.23/+5.65	15.22/24.06
AlexNet	65.68/76.53	—	—
VGG16 + ALL	83.89/91.75	+7.93/+6.36	32.99/43.55
VGG16 + IMAGE	83.93/91.69	+7.96/+6.30	33.13/43.13
VGG16 + CVPR16	79.50/88.58	+3.54/+3.19	14.71/21.86
VGG16	75.96/85.39	—	—
VGG16+CBL + IMAGE	75.67/83.49	+4.93/+3.27	16.84/16.55
VGG16+CBL + ALL	75.47/83.23	+4.73/+3.01	16.15/15.23
VGG16+CBL + CVPR16	71.07/81.02	+0.33/+0.80	1.12/4.06
VGG16+CBL	70.74/80.22	—	—
VGG19 + ALL	84.43/92.22	+9.03/+7.88	36.70/50.33
VGG19 + IMAGE	83.98/91.71	+8.58/+7.37	34.88/47.05
VGG19 + CVPR16	80.26/89.39	+4.87/+5.05	19.78/32.27
VGG19	75.40/84.34	—	—
VGG19+CBL + IMAGE	76.88/84.63	+5.34/+3.95	18.75/20.46
VGG19+CBL + ALL	75.47/83.88	+3.92/+3.20	13.79/16.58
VGG19+CBL + CVPR16	72.53/81.90	+0.98/+1.22	3.46/6.32
VGG19+CBL	71.54/80.67	—	—
ResNet50 + IMAGE	82.28/90.63	+7.21/+7.09	28.90/43.08
ResNet50	75.07/83.55	—	—
ResNet101 + IMAGE	83.10/90.80	+6.05/+5.19	26.37/36.08
ResNet101	77.05/85.61	—	—
ResNet152 + IMAGE	83.80/91.38	+5.36/+4.40	24.85/33.78
ResNet152	78.44/86.98	—	—

SPLIT: HARD	accuracy [%]	improvement [pp]	error reduction [%]
AlexNet + ALL	77.79/88.60	+11.15/+10.85	33.42/48.77
AlexNet + IMAGE	77.67/88.28	+11.02/+10.53	33.04/47.31
AlexNet + CVPR16	70.21/81.67	+3.56/+3.92	10.68/17.62
AlexNet	66.65/77.75	—	—
VGG16 + ALL	84.13/92.27	+6.88/+5.56	30.24/41.85
VGG16 + IMAGE	83.79/92.23	+6.53/+5.53	28.71/41.58
VGG16 + CVPR16	79.58/89.27	+2.32/+2.56	10.22/19.27
VGG16	77.26/86.71	—	—
VGG16+CBL + ALL	75.06/83.42	+4.67/+3.31	15.78/16.63
VGG16+CBL + IMAGE	75.04/83.16	+4.66/+3.05	15.73/15.32
VGG16+CBL + CVPR16	70.94/81.08	+0.56/+0.97	1.88/4.88
VGG16+CBL	70.38/80.11	—	—
VGG19 + IMAGE	83.91/92.17	+7.17/+6.11	30.83/43.84
VGG19 + ALL	84.12/92.00	+7.38/+5.94	31.74/42.62
VGG19 + CVPR16	79.69/89.42	+2.95/+3.36	12.69/24.11
VGG19	76.74/86.06	—	—
VGG19+CBL + ALL	75.62/83.76	+4.93/+3.50	16.82/17.71
VGG19+CBL + IMAGE	75.47/83.56	+4.78/+3.30	16.31/16.71
VGG19+CBL + CVPR16	71.92/81.64	+1.23/+1.38	4.20/6.97
VGG19+CBL	70.69/80.26	—	—
ResNet50 + IMAGE	82.27/90.79	+6.79/+6.18	27.69/40.13
ResNet50	75.48/84.61	—	—
ResNet101 + IMAGE	83.41/91.59	+6.95/+6.27	29.52/42.72
ResNet101	76.46/85.31	—	—
ResNet152 + IMAGE	83.74/91.71	+6.06/+5.51	27.16/39.93
ResNet152	77.68/86.20	—	—

B

COMPREHENSIVE DATASET FOR AUTOMATIC SINGLE CAMERA VISUAL SPEED MEASUREMENT – SUPPLEMENTARY MATERIAL

Algorithm B.1: Computation of distance of two points (p_1, p_2) on the road plane for given calibration (u, v, c, λ). The focal length is represented f, capital variables denote coordinates of points on the sensor in 3D world and $\overline{P}_i$ represent coordinates of points p_i on the road plane.

Input: Points $p_1 = [p_1^x, p_1^y, 1], p_2 = [p_2^x, p_2^y, 1]$

Input: Vanishing points $u = [u_x, u_y, 1], v = [v_x, v_y, 1]$

Input: Principal point $c = [c_x, c_y, 1]$, scene scale λ

Output: Distance d between p_1 and p_2 in meters

$f = \sqrt{-(u - c) \cdot (v - c)}$

$U = [u_x, u_y, f], V = [v_x, v_y, f], C = [c_x, c_y, 0]$

$W = [W_x, W_y, W_z] = (U - C) \times (V - C)$

$$w = \left[\frac{W_x}{W_z} \cdot f + c_x, \frac{W_y}{W_z} \cdot f + c_y, 1 \right] = [w_x, w_y, 1]$$

$\rho = [w_x, w_y, f] - C$

$\rho = \rho / \|\rho\| = [a, b, c]$

$\rho = [a, b, c, 10]$

for all $i \in 1, 2$ **do**

$P_i = [p_i^x, p_i^y, f]$

$g_i = P_i - C$

$$t_i = -\frac{\rho \cdot [c_x, c_y, 0, 1]}{[a, b, c] \cdot g_i}$$

$\overline{P}_i = C + t_i \cdot g_i$

end for

$d = \lambda \cdot \|\overline{P}_1 - \overline{P}_2\|$

	GPS		RADAR		
	mean	**med.**	**mean**	**med.**	**95 %**
Session 1	1.94	1.11	1.05	0.86	2.62
	2.67	1.44	1.37	1.14	3.37
Session 2	2.11	1.63	1.14	0.94	2.80
	2.35	2.06	1.33	1.12	3.22
Session 3	1.75	1.15	0.78	0.66	1.94
	4.94	2.43	1.31	1.19	3.26
Session 4	1.43	1.03	1.22	1.01	3.17
	1.68	1.10	1.28	1.15	3.34
Session 5	1.30	1.19	1.04	0.87	2.63
	1.71	1.40	1.33	1.15	3.34
Session 6	1.52	0.76	0.96	0.83	2.29
	2.43	1.35	1.33	1.13	3.23
TOTAL	1.64	1.19	1.07	0.89	2.67
	2.18	1.42	1.33	1.14	3.23

Table B.1: Results of GPS and RADAR speed measurement errors for every session separately. The first row for each session contains absolute errors in km/h and the relative errors in percents are in the second row. We do not report the 95 percentile for the GPS measurement as there is low number of measurements (maximally twenty).

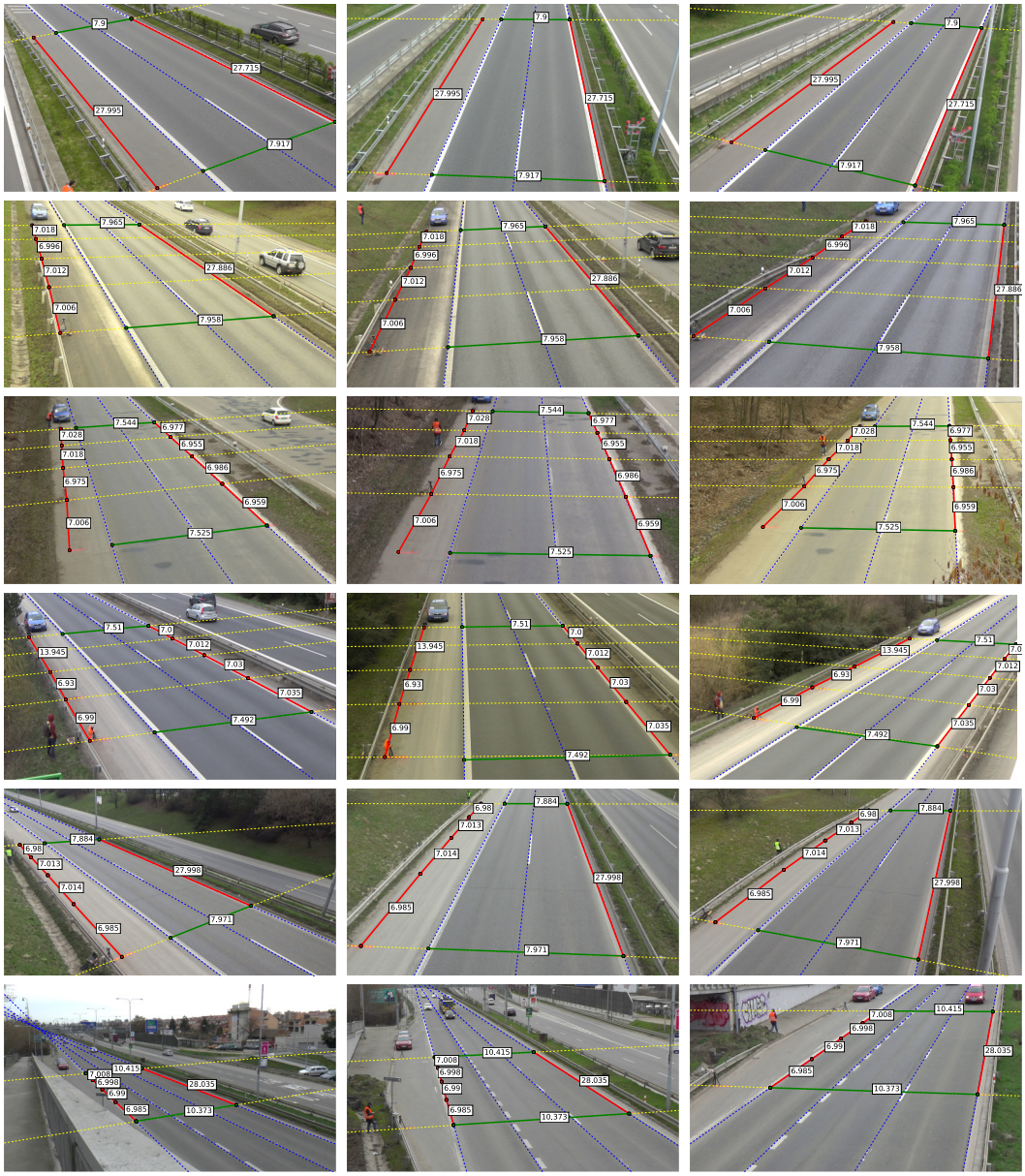


Figure B.1: Top to bottom: Session 1 – Session 6, left to right: cameras from left to right for given session

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] Shivani Agarwal, Aatif Awan, , and Dan Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE PAMI*, 26(11):1475–1490, November 2004.

[2] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2911–2918. IEEE, 2012.

[3] Hossein Azizpour and Ivan Laptev. Object detection using strongly-supervised deformable part models. In *Computer Vision–ECCV 2012*, pages 836–849. Springer, 2012.

[4] Remigiusz Baran, Andrzej Glowacz, and Andrzej Matiolanski. The efficient real- and non-real-time make and model recognition of cars. *Multimedia Tools and Applications*, 74(12):4269–4288, 2015. ISSN 1380-7501.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-33833-8.

[6] Romil Bhardwaj, Gopi Krishna Tummala, Ganesan Ramalingam, Ramachandran Ramjee, and Prasun Sinha. Autocalib: Automatic traffic camera calibration at scale. In *ACM BuildSys*, 2017.

[7] M. Biglari, A. Soleimani, and H. Hassanpour. A cascaded part-based system for fine-grained vehicle classification. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–11, 2017. ISSN 1524-9050.

[8] T. Bluche, H. Ney, and C. Kermorvant. Feature extraction with convolutional neural networks for handwritten word recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 285–289, August 2013.

[9] Noppakun Boonsim and Simant Prakoonwit. Car make and model recognition under limited lighting conditions at night. *Pattern Analysis and Applications*, pages 1–13, 2016. ISSN 1433-755X.

[10] Lubomir Bourdev and Jitendra Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1365–1372. IEEE, 2009.

[11] Lubomir Bourdev, Subhransu Maji, Thomas Brox, and Jitendra Malik. Detecting people using mutually consistent poselet activations. In *ECCV 2010*, pages 168–181. Springer, 2010.

[12] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[13] Dingding Cai, Ke Chen, Yanlin Qian, and Joni-Kristian Kämäräinen. Convolutional low-resolution fine-grained classification, 2017.

[14] Sijia Cai, Wangmeng Zuo, and Lei Zhang. Higher-order integration of hierarchical convolutional activations for fine-grained visual categorization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[15] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986. ISSN 0162-8828.

[16] Claudio Caraffi, Tomas Vojir, Jura Trefny, Jan Sochman, and Jiri Matas. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera. In *ITS Conference*, pages 975–982, September 2012.

[17] 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.

[18] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Y. Chai, E. Rahtu, V. Lempitsky, L. Van Gool, and A. Zisserman. Tricos: A tri-level class-discriminative co-segmentation method for image classification. In *European Conference on Computer Vision*, 2012.

[20] Yuning Chai, V. Lempitsky, and A. Zisserman. Symbiotic segmentation and part localization for fine-grained categorization. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 321–328, December 2013.

[21] Thomas Chaperon, Jacques Droulez, and Guillaume Thibault. Reliable camera pose and calibration from a small set of point and line correspondences: A probabilistic approach. *Computer Vision and Image Understanding*, 115(5):576 – 585, 2011. ISSN 1077-3142. Special issue on 3D Imaging and Modelling.

[22] 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.

[23] Xavier Clady, Pablo Negri, Maurice Milgram, and Raphael Poulenard. Multi-class vehicle type recognition system. In *Proceedings of the 3rd IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, ANNPR '08, pages 228–239, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-69938-5.

[24] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[25] Timothy F Cootes, Christopher J Taylor, David H Cooper, and Jim Graham. Active shape models-their training and application. *Computer vision and image understanding*, 61(1):38–59, 1995.

[26] Eduardo R. Corral-Soto and James H. Elder. Slot cars: 3D modelling for improved visual traffic analytics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

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

[29] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[30] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, June 2013.

[31] Louka Dlagnekov and Serge Belongie. Recognizing cars. Technical report, UCSD CSE Tech Report CS2005-0833, 2005.

[32] 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.

[33] 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.

[34] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering localized attributes for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[35] 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.

[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, June 2010.

[37] Xiaochuan Fan, Kang Zheng, Yuewei Lin, and Song Wang. Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[38] J. Fang, Y. Zhou, Y. Yu, and S. Du. Fine-grained vehicle model recognition using a coarse-to-fine convolutional neural network architecture. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–11, 2016. ISSN 1524-9050.

[39] R. Farrell, O. Oza, Ning Zhang, V.I. Morariu, T. Darrell, and L.S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 161–168, November 2011.

[40] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[41] Patryk Filipiak, Bartlomiej Golenko, and Cezary Dolega. *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part I*, chapter NSGA-II Based Auto-Calibration of Automatic Number Plate Recognition Camera for Vehicle Speed Measurement, pages 803–818. Springer International Publishing, Cham, 2016. ISBN 978-3-319-31204-0.

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

[43] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[44] 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.

[45] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2188–2202, Nov 2011. ISSN 0162-8828.

[46] Jiyang Gao, Zijian, Guo, Zhen Li, and Ram Nevatia. Knowledge concentration: Learning 100k object classifiers in a single CNN, 2017.

[47] 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.

[48] Efstratios Gavves, Basura Fernando, CeesG.M. Snoek, ArnoldW.M. Smeulders, and Tinne Tuytelaars. Local alignments for fine-grained categorization. *International Journal of Computer Vision*, 111(2):191–212, 2015. ISSN 0920-5691.

[49] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. Fine-grained recognition in the wild: A multi-task domain adaptation approach. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[50] 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.

[51] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, June 2012.

[52] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[53] Daniel Glasner, Meirav Galun, Sharon Alpert, Ronen Basri, and Gregory Shakhnarovich. Viewpoint-aware object detection and continuous pose estimation. *Image and Vision Computing*, 30(12):923–933, December 2012. ISSN 0262-8856.

[54] Christoph Göring, Erik Rodner, Alexander Freytag, and Joachim Denzler. Nonparametric part transfer for fine-grained recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2489–2496, 2014.

[55] Philippe-Henri Gosselin, Naila Murray, Hervé Jégou, and Florent Perronnin. Revisiting the fisher vector for fine-grained classification. *Pattern Recognition Letters*, 49:92–98, 2014. ISSN 0167-8655.

[56] 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.

[57] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

[58] Hui-Zhen Gu and Suh-Yin Lee. Car model recognition by utilizing symmetric property to overcome severe pose variation. *Machine Vision and Applications*, 24(2):255–274, 2013. ISSN 0932-8092.

[59] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

[60] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[61] H. He, Z. Shao, and J. Tan. Recognition of car makes and models from a single traffic-camera image. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–11, 2015. ISSN 1524-9050.

[62] 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.

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

[64] 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.

[65] Harold Hotelling. Relations between two sets of variates. *Biometrika*, pages 321–377, 1936.

[66] Michal Hradiš, Jan Kotera, Pavel Zemčík, and Filip Šroubek. Convolutional neural networks for direct text deblurring. In *Proceedings of BMVC 2015*, pages 1–13. The British Machine Vision Association and Society for Pattern Recognition, 2015. ISBN 1-901725-53-7.

[67] E. Hsiao, S.N. Sinha, K. Ramnath, S. Baker, L. Zitnick, and R. Szeliski. Car make and model recognition using 3D curve alignment. In *IEEE WACV*, March 2014.

[68] Jun-Wei Hsieh, Li-Chih Chen, and Duan-Yu Chen. Symmetrical surf and its applications to vehicle detection and vehicle make and model recognition. *Intelligent Transportation Systems, IEEE Transactions on*, 15(1):6–20, February 2014. ISSN 1524-9050.

[69] C. Hu, X. Bai, L. Qi, X. Wang, G. Xue, and L. Mei. Learning discriminative pattern for real-time car brand recognition. *Intelligent Transportation Systems, IEEE Transactions on*, 16(6):3170–3181, December 2015. ISSN 1524-9050.

[70] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Deep transfer metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[71] Q. Hu, H. Wang, T. Li, and C. Shen. Deep CNNs with spatially weighted pooling for fine-grained car recognition. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–10, 2017. ISSN 1524-9050.

[72] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked CNN for fine-grained visual categorization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[73] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *International Journal of Computer Vision*, pages 1–20, 2015. ISSN 0920-5691.

[74] 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.

[75] 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.

[76] Leonid Karlinsky, Joseph Shtok, Yochay Tzur, and Asaf Tzadok. Fine-grained recognition of thousands of object categories with single-example training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[77] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of International Computer Vision and Pattern Recognition (CVPR 2014)*, 2014.

[78] R. Ke, Z. Pan, Z. Pu, and Y. Wang. Roadway surveillance video camera calibration using standard shipping container. In *2017 International Smart Cities Conference (ISC2)*, pages 1–6, Sept 2017.

[79] J. Krause, T. Gebru, J. Deng, L. J. Li, and L. Fei-Fei. Learning features and parts for fine-grained recognition. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 26–33, August 2014.

[80] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshop 3dRR-13*, 2013.

[81] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[82] 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.

[83] Jinhui Lan, Jian Li, Guangda Hu, Bin Ran, and Ling Wang. Vehicle speed measurement based on gray constraint optical flow algorithm. *Optik - International Journal for Light and Electron Optics*, 125(1):289–295, 2014. ISSN 0030-4026.

[84] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[85] 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.

[86] Suhan Lee, Jeonghwan Gwak, and Moongu Jeon. Vehicle model recognition in video. *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 6(2):175, 2013.

[87] B. Leibe, N. Cornelis, K. Cornelis, and L. Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[88] Chunming Li, Chris Gatenby, Li Wang, and John C Gore. A robust parametric method for bias field estimation and segmentation of mr images. In

Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 218–223. IEEE, 2009.

[89] Lingyun Li, Yanqing Guo, Lingxi Xie, Xiangwei Kong, and Qi Tian. Fine-Grained Visual Categorization with Fine-Tuned Segmentation. *IEEE International Conference on Image Processing*, 2015.

[90] Liang Liao, Ruimin Hu, Jun Xiao, Qi Wang, Jing Xiao, and Jun Chen. Exploiting effects of parts in fine-grained categorization of vehicles. In *International Conference on Image Processing (ICIP)*, 2015.

[91] Di Lin, Xiaoyong Shen, Cewu Lu, and Jiaya Jia. Deep lac: Deep localization, alignment and classification for fine-grained recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

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

[93] Yen-Liang Lin, Vlad I. Morariu, Winston Hsu, and Larry S. Davis. Jointly optimizing 3D model fitting and fine-grained classification. In *ECCV*, 2014.

[94] Hongye Liu, Yonghong Tian, Yaowei Yang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[95] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *ECCV 2012*, volume 7572 of *Lecture Notes in Computer Science*, pages 172–185. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-33717-8.

[96] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46448-0.

[97] Xinchen Liu, Wu Liu, Huadong Ma, and Huiyuan Fu. Large-scale vehicle re-identification in urban surveillance videos. In *Multimedia and Expo (ICME), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[98] D. F. Llorca, D. Colás, I. G. Daza, I. Parra, and M. A. Sotelo. Vehicle model recognition using geometry and appearance of car emblems from rear view

images. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 3094–3099, October 2014.

[99] D. F. Llorca, C. Salinas, M. Jimenez, I. Parra, A. G. Morcillo, R. Izquierdo, J. Lorenzo, and M. A. Sotelo. Two-camera based accurate vehicle speed measurement using average speed at a fixed point. In *19th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2016.

[100] 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.

[101] David G Lowe. Object recognition from local scale-invariant features. In *The proceedings of the seventh IEEE international conference on Computer vision, 1999.*, volume 2, pages 1150–1157. Ieee, 1999.

[102] 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.

[103] D.C. Luvizon, B.T. Nassu, and R. Minetto. Vehicle speed estimation by license plate detection and tracking. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6563–6567, May 2014.

[104] C. Maduro, K. Batista, P. Peixoto, and J. Batista. Estimation of vehicle velocity and traffic intensity using rectified images. In *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pages 777–780, October 2008.

[105] Marion Chevalier et al. Lr-CNN for fine-grained classification with varying resolution. In *IEEE International Conference Image Processing (ICIP)*, 2015.

[106] Kevin Matzen and Noah Snavely. NYC3DCars: A dataset of 3D vehicles in geographic context. In *International Conference on Computer Vision (ICCV)*, 2013.

[107] Sinisa Todorovic Michael Lam, Behrooz Mahasseni. Fine-grained recognition as HSnet search for informative image parts. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[108] H. Nakayama. Augmenting descriptors for fine-grained visual categorization using polynomial embedding. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6, July 2013.

[109] P. Negri, X. Clady, M. Milgram, and R. Poulenard. An oriented-contour point based voting algorithm for vehicle type classification. In *International Conference on Pattern Recognition*, pages 574–577, 2006.

[110] A. Nurhadiyatna, B. Hardjono, A. Wibisono, I. Sina, W. Jatmiko, M.A. Ma'sum, and P. Mursanto. Improved vehicle speed estimation using Gaussian mixture model and hole filling algorithm. In *Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on*, pages 451–456, September 2013.

[111] A. Opelt, A. Pinz, M. Fussenegger, and P. Auer. Generic object recognition with boosting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(3):416–431, March 2006. ISSN 0162-8828.

[112] M. Özuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *IEEE CVPR*, pages 778–785, June 2009.

[113] C. Papageorgiou and T. Poggio. A trainable object detection system: Car detection in static images. Technical Report 1673, October 1999. (CBCL Memo 180).

[114] Dong Kwon Park, Yoon Seok Jeon, and Chee Sun Won. Efficient use of local edge histogram descriptor. In *Proceedings of the 2000 ACM workshops on Multimedia*, pages 51–54. ACM, 2000.

[115] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[116] G. Pearce and N. Pears. Automatic make and model recognition from frontal images of cars. In *IEEE International Conference on Advanced Video and Signal-based Surveillance (AVSS)*, pages 373–378, 2011.

[117] Y. Peng, X. He, and J. Zhao. Object-part attention model for fine-grained image classification. *IEEE Transactions on Image Processing*, PP(99):1–1, 2017. ISSN 1057-7149.

[118] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV 2010*, pages 143–156. Springer, 2010.

[119] V.S. Petrovic and T. F. Cootes. Analysis of features for rigid structure vehicle type recognition. In *British Machine Vision Conference*, pages 587–596, 2004.

[120] Hamed Pirsiavash, Deva Ramanan, and Charless C. Fowlkes. Bilinear classifiers for visual recognition. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1482–1490. Curran Associates, Inc., 2009.

[121] J. Prokaj and G. Medioni. 3-D model based vehicle recognition. In *Workshop on Applications of Computer Vision (WACV)*, December 2009.

[122] A. Psyllos, C.N. Anagnostopoulos, and E. Kayafas. Vehicle model recognition from frontal view image measurements. *Computer Standards & Interfaces*, 33(2):142–151, 2011. ISSN 0920-5489.

[123] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016.

[124] 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.

[125] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature neuroscience*, 2(11):1019–1025, 1999.

[126] CC Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

[127] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, DTIC Document, 1961.

[128] 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.

[129] C Rother, V Kolmogorov, and A Blake. Interactive foreground extraction using iterated graph cuts, 2004. In *ACM Transactions on Graphics*. SIGGRAPH, 2004.

[130] 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.

[131] G. Salvi. An automated nighttime vehicle counting and detection system for traffic surveillance. In *2014 International Conference on Computational Science and Computational Intelligence*, volume 1, pages 131–136, March 2014.

[132] Jorge Sánchez, Florent Perronnin, and Zeynep Akata. Fisher vectors for fine-grained visual categorization. In *FGVC Workshop in IEEE Computer Vision and Pattern Recognition (CVPR)*, 2011.

[133] Jorge Sánchez, Florent Perronnin, Thomas Mensink, and Jakob Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, 105(3):222–245, 2013. ISSN 0920-5691.

[134] Silvio Savarese and Li Fei-Fei. 3D generic object categorization, localization and pose estimation. In *The IEEE International Confernece on Computer Vision (ICCV)*. IEEE, 2007.

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

[136] CJ. Schuler, HC. Burger, S. Harmeling, and B. Schölkopf. A machine learning approach for non-blind image deconvolution. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1067–1074. IEEE, 2013.

[137] Bernhard Schölkopf and Alexander J. Smola. *Advances in Kernel Methods: Support Vector Learning*. The MIT Press, 1998. ISBN 0262194163.

[138] Li Shen, Teck Wee Chua, and Karianto Leman. Shadow optimization from structured deep edge detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[139] Zhiqiang Shen, Yu-Gang Jiang, Dequan Wang, and Xiangyang Xue. Iterative object and part transfer for fine-grained recognition. In *The IEEE International Conference on Multimedia and Expo (ICME) – to appear*, 2017.

[140] 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.

[141] Hoo-Chang Shin, Le Lu, Lauren Kim, Ari Seff, Jianhua Yao, and Ronald M. Summers. Interleaved text/image deep mining on a very large-scale radiology database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

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

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

[144] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

[145] 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.

[146] Michael Stark, Jonathan Krause, Bojan Pepik, David Meger, James Little, Bernt Schiele, and Daphne Koller. Fine-grained categorization for 3D scene understanding. In *British Machine Vision Conference*, 2012.

[147] 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.

[148] Lin Sun, Kui Jia, Tsung-Han Chan, Yuqiang Fang, Gang Wang, and Shuicheng Yan. Dl-sfa: Deeply-learned slow feature analysis for action recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[149] Ting Sun, Lin Sun, and Dit-Yan Yeung. Fine-grained categorization via cnn-based automatic extraction and integration of object-level and part-level features. *Image and Vision Computing*, pages –, 2017. ISSN 0262-8856.

[150] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep learning face representation from predicting 10,000 Classes. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1891–1898, June 2014.

[151] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, June 2015.

[152] F. Tafazzoli, H. Frigui, and K. Nishiyama. A large and diverse dataset for improved vehicle make and model recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 874–881, July 2017.

[153] 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.

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

[155] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[156] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004. ISSN 0920-5691.

[157] J. Wang, H. Zheng, Y. Huang, and X. Ding. Vehicle type recognition in surveillance images from labeled web-nature data using deep transfer learning. *IEEE Transactions on Intelligent Transportation Systems*, PP(99):1–10, 2017. ISSN 1524-9050.

[158] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, June 2010.

[159] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye, and X. Xue. Evolving boxes for fast vehicle detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1135–1140, July 2017.

[160] Qi Wang, Zhongyuan Wang, Jing Xiao, Jun Xiao, and Wenbin Li. *Fine-Grained Vehicle Recognition in Traffic Surveillance*, pages 285–295. Springer International Publishing, Cham, 2016. ISBN 978-3-319-48890-5.

[161] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[162] Yu Xiang and Silvio Savarese. Estimating the aspect layout of object categories. In *The IEEE Conference on Computer Vision and Pattern Recognition*, pages 3410–3417, 2012.

[163] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[164] Guo-Sen Xie, Xu-Yao Zhang, Wenhan Yang, Mingliang Xu, Shuicheng Yan, and Cheng-Lin Liu. Lg-cnn: From local parts to global discrimination for fine-grained recognition. *Pattern Recognition*, 71:118 – 131, 2017. ISSN 0031-3203.

[165] Jin Xie, Yi Fang, Fan Zhu, and Edward Wong. Deepshape: Deep learned shape descriptor for 3D shape matching and retrieval. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[166] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[167] Li Xu, Jimmy S Ren, Ce Liu, and Jiaya Jia. Deep convolutional neural network for image deconvolution. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1790–1798. Curran Associates, Inc., 2014.

[168] 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.

[169] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[170] Shulin Yang, Liefeng Bo, Jue Wang, and Linda G. Shapiro. Unsupervised template learning for fine-grained object recognition. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3122–3130. Curran Associates, Inc., 2012.

[171] Bangpeng Yao. A codebook-free and annotation-free approach for fine-grained image categorization. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3466–3473, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-1-4673-1226-4.

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

[173] Xinguo Yu, Nianjuan Jiang, Loong-Fah Cheong, Hon Wai Leong, and Xin Yan. Automatic camera calibration of broadcast tennis video with applications to 3D virtual content insertion and ball detection and tracking. *Computer Vision and Image Understanding*, 113(5):643 – 652, 2009. ISSN 1077-3142. Computer Vision Based Analysis in Sport Environments.

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

[175] Bailing Zhang. Reliable classification of vehicle types based on cascade classifier ensembles. *IEEE Transactions on Intelligent Transportation Systems*, 14(1): 322–332, March 2013. ISSN 1524-9050.

[176] Bailing Zhang. Classification and identification of vehicle type and make by cortex-like image descriptor HMAX. *IJCVR*, 4:195–211, 2014. ISSN 1752-9131.

[177] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[178] Han Zhang, Tao Xu, Mohamed Elhoseiny, Xiaolei Huang, Shaoting Zhang, Ahmed Elgammal, and Dimitris Metaxas. Spda-CNN: Unifying semantic part detection and abstraction for fine-grained recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[179] L. Zhang, Y. Yang, M. Wang, R. Hong, L. Nie, and X. Li. Detecting densely distributed graph patterns for fine-grained image categorization. *IEEE Transactions on Image Processing*, 25(2):553–565, February 2016. ISSN 1057-7149.

[180] Ning Zhang, R. Farrell, and T. Darrell. Pose pooling kernels for sub-category recognition. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3665–3672, June 2012.

[181] Ning Zhang, Ryan Farrell, Forrest Iandola, and Trevor Darrell. Deformable part descriptors for fine-grained recognition and attribute prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[182] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based R-CNNs for fine-grained category detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[183] Xiaopeng Zhang, Hongkai Xiong, Wengang Zhou, Weiyao Lin, and Qi Tian. Picking deep filter responses for fine-grained image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[184] Yuting Zhang, Kihyuk Sohn, Ruben Villegas, Gang Pan, and Honglak Lee. Improving object detection with deep convolutional networks via Bayesian optimization and structured prediction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 249–258, 2015.

[185] 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.

[186] 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.

[187] 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.

[188] Yuan Zheng and Wenyong Zhao. Can vehicle become a new pattern for roadside camera calibration? In *Asian Conference on Computer Vision Workshops*, pages 175–188. Springer, 2016.

[189] Feng Zhou and Yuanqing Lin. Fine-grained image classification by exploring bipartite-graph labels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

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

[191] M. H. Zwemer, G. M. Y. E. Brouwers, R. G. J. Wijnhoven, and P. H. N. de With. *Semi-automatic Training of a Vehicle Make and Model Recognition System*, pages 321–332. Springer International Publishing, Cham, 2017. ISBN 978-3-319-68548-9.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. It is available for LaTeX via CTAN as `classicthesis`.

Final Version as of February 26, 2018 (`classicthesis` version 1.0).