$24^{th}$ *Computer Vision Winter Workshop*
*Friedrich Fraundorfer, Peter M. Roth, Fabian Schenk (eds.)*
*Stift Vorau, Austria, February 6–8, 2019*

DOI: 10.3217/978-3-85125-652-9-04

# Perspective transformation for accurate detection of 3D bounding boxes of vehicles in traffic surveillance

Viktor Kocur
Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava
Mlynská Dolina, Bratislava
`viktor.kocur@fmph.uniba.sk`

**Abstract.** *Detection and tracking of vehicles captured by traffic surveillance cameras is a key component of intelligent traffic systems. In this paper a novel method of detecting 3D bounding boxes of vehicles is presented. Using the known geometry of the surveilled scene, we propose an algorithm to construct a perspective transformation. The transformation enables us to simplify the problem of detecting 3D bounding boxes to detecting 2D bounding boxes with one additional parameter. We can therefore utilize modified 2D object detectors based on deep convolutional networks to detect 3D bounding boxes of vehicles. Known 3D bounding boxes of vehicles can be utilized to improve results on tasks such as fine-grained vehicle classification or vehicle re-identification. We test the accuracy of our detector by comparing the accuracy of speed measurement on the BrnoCompSpeed dataset with the existing state of the art method. Our method decreases the mean error in speed measurement by 22 % (1.10 km/h to 0.86 km/h ) and the median error in speed measurement by 33 % (0.97 km/h to 0.65 km/h mean), while also increasing the recall (83.3 % to 89.3 %).*

## 1. Introduction

Recent development in commercially available cameras has increased the quality of recorded images and decreased the costs required to install cameras in traffic surveillance scenarios. Automatic traffic surveillance aims to provide information about the surveilled vehicles such as their speed, type and dimensions and as such is an important aspect of intelligent traffic system design.

There are two crucial requirements for accurate monitoring of vehicles. Precise detection of their position and accurate camera calibration. Multiple methods of monocular camera calibration exist. However, they usually require manual measurements of some distances in the road plane. Dubská et al. [8] proposed a fully automatic method of camera calibration based on vanishing point detection. Sochor et al. [25] further improved the method. We employ this method for calibration and focus on the accuracy of vehicle detection.

Object detection is one of the key tasks in computer vision. Current state of the art object detectors rely on convolutional neural network backbones to extract feature maps from images. A structure based on so-called anchorboxes is used to determine the position of bounding boxes of detected objects. There are two types of such object detection networks: one-stage detectors and two-stage detectors. One-stage detectors such as SSD [17] or YOLO [22] have shorter inference times, but suffer from worse accuracy than two-stage detectors such as Faster R-CNN [23] and R-FCN [4]. We employ a recently published one-stage detector RetinaNet [14], which managed to bridge the accuracy gap while keeping the inference times low.

In this paper we propose a perspective image transformation, which utilizes the geometry of a common traffic surveillance scenario to rectify the image. Rectification of the image allows us to reduce the problem of detecting 3D bounding boxes of vehicles to detection of 2D bounding boxes with one additional parameter. Any object detector which utilizes anchorboxes can be easily modified to detect 3D bounding boxes of vehicles in the transformed images. We test our method on the task of speed measurement of vehicles captured by a static monocular traffic camera. Compared to existing approaches, our method achieves lower mean error of measured speeds, while being computationally simpler.

## 2. Related work

### 2.1. Camera calibration

Multiple approaches have been proposed to deal with camera calibration in a traffic surveillance scenario. A calibration pattern [7] or manual measurements on the road [19] can be used to determine the camera parameters. Some methods use line segments on the road or traffic signs to find vanishing points [2, 9]. Other methods perform calibration based on vehicle movement [5, 24].

We require the calibration method to be fully automatic and highly accurate. Dubská et al. [8] proposed a fully automatic method of camera calibration by finding three orthogonal vanishing points. The position of the first vanishing point is determined by detecting motion of the surveilled vehicles. Interesting feature points are detected and tracked with a KLT tracker. Hough transform is used to accumulate the found lines of motion in a diamond space based on parallel coordinates. To determine the second vanishing point, the edges of the vehicles which do not correspond to the first vanishing point are again accumulated. The two vanishing points are then determined from the diamond space with the third vanishing point being calculated as orthogonal to the first two.

To detect the scale of the scene, the dimensions of 3D bounding boxes of the passing vehicles are collected. The mean of the collected dimensions is compared against statistical data and thus a scale, which enables measuring distances on the road plane, is determined. This approach has been further improved by Sochor et al. [25] who used edgelets corresponding to relevant car features to detect the second vanishing point more accurately. The detection of the scale was also improved by fitting a 3D model of a common vehicle to the detections of the vehicle in the footage.

### 2.2. Object detection

Most cameras employed in traffic surveillance are static. Many vehicle detection approaches [3, 19] therefore utilize background substraction methods to detect the vehicles. These methods can fail with quickly changing lighting conditions or small camera movements and may result in single detections containing more than one vehicle due to occlusion. Luvizón et al. use motion detector [1] to detect license plates. Daley et al. [5] find edges in the inter-frame differences [29]. Pham and Lee [28] propose a method based on finding the windshields of vehicles. Zhou et al. [31] propose two deep neural networks to propose and verify the detections.

We opt to base our method on recent object detectors based on deep convolutional neural networks, as these, given proper training data, can handle different light conditions and occlusion. These can be divided in two categories: one-stage and two-stage detectors. Two stage methods (Faster R-CNN [23] and R-FCN [4]) utilize a convolutional neural network to extract features from an image. The first stage comprises of determining which regions, called anchorboxes, in the image could potentially contain foreground objects. In the second stage the features corresponding to the proposed anchorboxes are processed further to determine which object, if any, the candidate anchorbox contains. Along with this classification task, the shape and position of the proposed anchorbox is offset via regression to better fit the object. Finally, non-maximum supression is applied to remove all but one bounding box per detected object. An extension of Faster R-CNN called Mask R-CNN [10] additionally enables pixel-wise segmentation of the detected objects.

One-stage detectors differ from two stage detectors by performing classification and regression on all anchorboxes. This results in disproportionate amount of negative (i.e. background) proposals compared to the amount of positive samples (i.e. objects) and biases the training process. To remedy this, SSD [17] uses hard negative mining and YOLO [22] uses a fixed weight on loss contributions by the background anchorboxes. These approaches are faster than two-stage detectors, but have worse results on standard benchmarks [12].

RetinaNet [14] introduced focal loss to address the issue of negative sample bias and utilized a feature pyramid network to further improve its performance. RetinaNet achieves better results on COCO detection task [15] than Faster R-CNN while keeping the benefits of low inference times of the other one-stage detectors.

We compare our results with a method of object detection proposed in [25] in which a Faster R-CNN object detector is used to find 2D bounding boxes. The bounding boxes are then used to join foreground blobs obtained via background subtraction into masks of vehicles.

### 2.3. Object tracking

To allow for vehicle counting and speed measurement, the vehicles have to be tracked from frame to frame. Since object detection may sometimes fail a robust tracker is necessary. Kalman filter [13] has been a reliable tool to tackle the task of object tracking. Recent successes with deep neural nets led to development of object tracking methods which utilize convolutional architectures in combinations with recurrent neural networks [20, 16, 21]. For our case we found that a simple object tracker, which compares the positions of bounding boxes in subsequent frames is sufficient.

### 2.4. Benefits of 3D bounding boxes

The task of detecting 2D bounding boxes of vehicles is in general easier than detecting 3D bounding boxes. However 3D bounding boxes have been shown to be valuable in improving various traffic surveillance algorithms such as fine-grained vehicle classification [27] and vehicle re-identification [30].

### 2.5. Datasets

Sochor et al. [26] performed a survey of existing traffic surveillance datasets. Most publicly available datasets suffer from too few recorded vehicles and inaccurate measurements of ground truth speeds.

Authors of the survey published their own BrnoCompSpeed dataset. The dataset contains videos from 7 locations. Each location is recorded for approximately one hour from three different viewpoints on an overpass above the recorded roads. The dataset contains ground truth data of over 20 thousand vehicle speeds obtained via LIDAR gate measurements. We train and evaluate our method on this dataset.

After the publication of the survey Luvizón et al. [18] published a dataset comprising of 5 hours of video of smaller sections of roads. Speed measurements of cars using inductive loops and labeled license plate numbers are provided as ground truth.

## 3. Proposed method

The goal of our method is to detect 3D bounding boxes of cars recorded with a monocular camera installed above the road plane.

### 3.1. Image transformation

For our method we assume that the camera has been calibrated as per [25]. This calibration method has very few limitations regarding the camera position. The camera has to be positioned above the road plane and the observed road segment has to be straight. The main parameters obtained by the calibration are the positions of the two relevant vanishing points in the image. Assuming that the principal point is in the center of the image, the position of the third vanishing point as well as focal length of the camera can be calculated. This enables us to project any point in the image onto the road plane. To enable measurements of distances on the road plane one additional parameter, denoted as scale, is determined during calibration.

The first detected vanishing point (denoted further as *VP1*) corresponds to the lines on the road plane which are parallel to the direction of the moving vehicles. The second detected vanishing point (*VP2*) corresponds to the lines which lie on the road plane but are perpendicular to the the direction of the moving vehicles. The third vanishing point (*VP3*) corresponds to the lines which are perpendicular to the road plane.

The goal of our transformation is to produce an image in which all lines corresponding to *VP2* will be parallel to the $x$-axis of the transformed image and the lines corresponding to *VP3* will be parallel to the $y$ axis. Thus in the transformed image plane both *VP2* and *VP3* will be ideal points. Additionally, we require the lines corresponding to *VP3* to be preserved and thus we will use perspective transformation. As a result of such transformation the objects which are aligned with the three vanishing points (most importantly the vehicles on the road) will be rectified in the transformed image.

In most cases this transformation can be performed in a way which preserves the whole captured scene in the resulting image. To find the perspective transformation for such cases we employ an algorithm (see Figure 1 for visual reference):

- In the original image plane, for both *VP2* and *VP3*, construct two lines which originate in the vanishing point and are tangent to the viewing window.
- For both lines originating in *VP2* find the intersections with both of the lines originating in *VP3*. This yields four points.
- Determine which of these four intersection points correspond to which corner points of the viewing window.

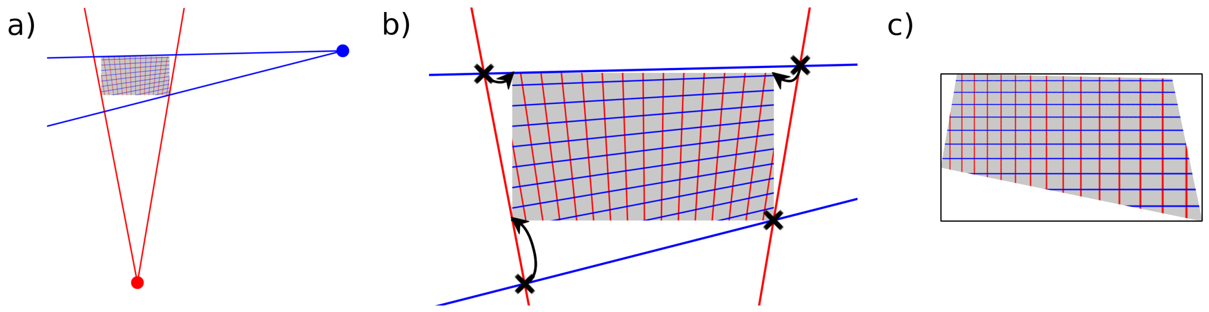

Figure 1. a) Lines starting in *VP2* (blue) and *VP3* (red), which are tangent to the viewing window (gray) are detected. b) Intersections of the lines are found. Points of intersection are paired with the corners of the viewing window. The four pairs are used to find the perspective transform. c) Perspective transform is applied.

- Find a perspective transformation which maps the intersection points to their respective corner points.

This is possible in most cases, however in a case in which the line connecting the two vanishing points intersects the viewing window the algorithm fails. This is expected as in such a case the points on the intersecting line correspond to both vanishing points and thus in the transformed image they would have to form a line which is perpendicular to itself. For this to occur the camera has to be positioned not right above the captured road, but on its side. Usually such situations could be avoided by considering the position in which to install the cameras, for instance in the BrnoCompSpeed dataset there is no such scene where this is the case. Nevertheless it is always possible to crop the viewing window to obtain the transformation.

This strategy can be employed even when the algorithm doesn't fail, but the resulting image is too distorted. In this paper we use this strategy heuristically only for one outlier video and thus we do not specify a rule for its use. Note that the viewing window is reduced only for the calculation of the transformation properties. The pixels which do not fit into the new viewing window can still appear in the transformed image.

### 3.2. Bounding boxes in the transformed image

The 3D bounding boxes we aim to detect are aligned with the vanishing points as in [25]. In their work Sochor et al. first segment the vehicle and then construct the bounding box around the mask. The construction in this manner is problematic, as the resulting bounding box depends on the order in which the vertices of the bounding box are constructed.

The perspective transformation described above enables us to reduce the problem of finding the 3D bounding box to finding a 2D bounding box with one additional parameter. This is possible, since the transformation rectifies the image in a manner in which all the lines corresponding to *VP2* and *VP3* are parallel to either of the image axes. The remaining parameter denoted as $c_c$ is determined by the relative position of the top frontal edge of the 3D bounding box against the 2D bounding box which encloses the whole 3D bounding box. The construction of the 2D bounding box can be seen in Figure 2.

Reconstructing the 3D bounding box from the 2D version can be achieved by considering the position of *VP1* in the transformed image. A point on the side of the 2D bounding box is determined by the parameter $c_c$ and a relative position of the transformed *VP1*. If the transformed *VP1* is to the left of the bounding box, then the point is on the right side and vice versa. If the transformed *VP1* is directly above the bounding box, then the side can be chosen arbitrarily. With the position of this point known, the 3D bounding box can be constructed in the transformed image. To obtain the 3D bounding box in the original image, the positions of the vertices of the 3D bounding box are transformed to the original image space via inverse perspective transform.

### 3.3. Bounding box detection

As shown in the previous subsection we only need to detect 2D bounding boxes with the parameter $c_c$. For this purpose we utilize the RetinaNet object detector [14]. This detector outputs 2D bounding boxes for the detected objects. We modify the method to add $c_c$ to each of the output boxes.

The RetinaNet, as well as other object detecting meta-architectures, use anchorboxes as default positions of bounding boxes to determine where the objects are. The object detection task is separated into

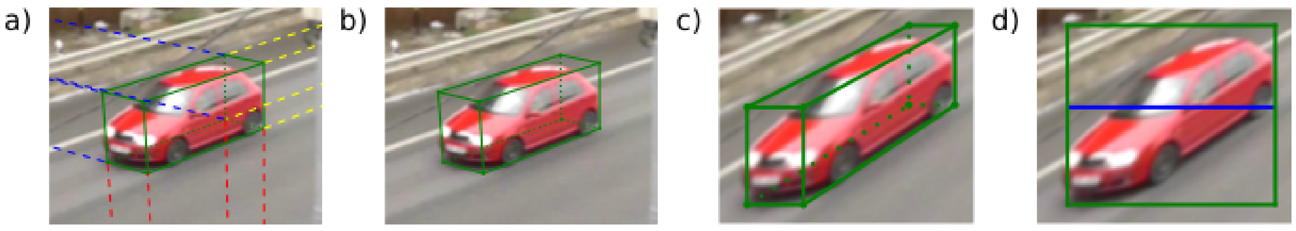

Figure 2. a) 3D bounding box (green) which is aligned with VP1 (yellow), VP2 (blue) and VP3 (red). b) 3D bounding box. c) 3D bouding box after the perspective transform is applied. d) The parametrization of the 3D bouding box as 2D bouding box (green). The parameter $c_c$ is determined as the ratio of the distance from top of the 2D bounding box to the top-front edge of the transformed 3D bounding box (blue) and the height of the 2D bouding box.

three parts: determining which anchorboxes contain which objects, resizing and moving the anchorboxes to better fit the objects and finally performing non-maximum suppression to avoid multiple detections of the same object. To train the network a two-part loss (1) is used.

$$L_{tot} = \frac{1}{N}\left(L_{conf} + L_{loc}\right) \tag{1}$$

The loss is averaged over all $N$ anchorboxes, $L_{conf}$ is the Focal loss used to train a classifier to determine which objects, if any, are in the bounding box. $L_{loc}$ is the regression loss to train the network how to reshape and offset the anchorboxes. To include the parameter $c_c$ we add another regression loss (2).

$$L_c = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} x_{i,j} s_{L1}\left(c^p_{c,i} - c^g_{c,j}\right) \tag{2}$$

In the loss we sum over all of the $N$ anchorboxes and $M$ ground truth bounding boxes. $x_{i,j}$ determines whether the $i$-th anchorbox corresponds to the $j$-th ground truth label [17]. We subtract the ground truth value of $c_c$ denoted as $c^g_{c,j}$ from the predicted value $c^p_{c,i}$ and apply smooth L1 loss.

### 3.4. Training

To obtain training data we use data from two distinct datasets. The first dataset is BoxCars116k [27], the original purpose of this dataset is fine-grained vehicle classification. The dataset contains over 116 thousand images, each containing one car along with make and model labels, information on positions of vanishing points and the 3D bounding box of the car. We transform these images with the proposed transformation and calculate the 2D bounding boxes and $c_c$ based on the provided 3D bounding boxes. Since each image is only of one car we augment the images by rescaling them and placing them randomly on a black background.

The other used dataset is BrnoCompSpeed [26]. We use the split C of this dataset leaving 9 videos for testing, 9 for training and 3 for validation. The original purpose of this dataset is speed measurement, therefore we test our method by measuring speed on the test set. For training and validation we pick every 25-th frame of the videos. Within the dataset a mask of the region of interest (e.g. the surveilled road) is provided, therefore we keep only the pixels from the region of interest to not confuse the network with cars outside the road, which would get transformed in undesirable ways since they may not be aligned with the vanishing points. We run these frames through Mask-RCNN [10] image segmentation network trained on the COCO dataset [15] to obtain masks of the cars. We transform the masks and the images using our transformation and create the 2D bounding boxes with and without $c_c$ as labels for training.

We train the model on the labeled data in a standard procedure. The validation loss is used to choose the best model for inference. We employ ResNet50 [11] pre-trained on Imagenet [6] as our backbone network. The input of the network is an image with 640 by 360 pixels.

### 3.5. Speed measurement

We perform speed measurement on the test videos. For each video the network is used to detect the 3D bounding boxes. Each detected 3D bounding box is compared via its encompassing 2D bounding box to the tracks which have been detected in the previous frames.

For each detection the IoU (intersection over union) metric is calculated against the last 2D bounding box of each track. If IoU is higher than 0.1 for at least one track, then the bounding box is added to the track with highest IoU score. If no track has higher IoU against the detection, then a new track is created. If a track hasn't had any bounding boxes added to it in the last 5 frames, then the track is no longer considered active. To detect speed we filter out bounding boxes too close to the edges of the images. We also discard tracks which have less than 5 detected bounding boxes within them.

For the 3D bounding box the speed is determined using a point which is in the middle of the frontal bottom edge of the 3D bounding box. Since this points should under normal circumstances lie on the road plane, we can use the camera calibration to easily determine the distances between various positions within a track. To detect the average speeds of the vehicles we employ the same method as [26] by calculating median speed over the whole track.

## 4. Results

We compare our results with the results achieved in [25] as these are the best achieved results published in the literature, which we denote as *SochorAuto* for automatic calibration and *SochorManual* for manual calibration. Our method is denoted as *Transform3D*. Examples of the resulting bounding boxes can be seen in Figure 4.

### 4.1. Ablation experiments

To properly gauge the impact of the image transformation we perform two ablation experiments. We train the standard RetinaNet 2D object detector on the same data as the other models, except that the images are not transformed. We refer to this model as *Orig2D*. We also train the standard RetinaNet 2D object detector on the transformed image. We use the same 2D bounding boxes as in *Transform3D*, but without the parameter $c_c$. We refer to this method as *Transform2D*. We train these models with the same hyperparameters as our base model.

For our method as well as the ablation experiments we set the confidence necessary to classify a prediction as true to 0.2. This is an unusually low threshold. However in this case it is beneficial as setting the threshold too high may lead to lower recall, while also producing more false positives due to tracks being possibly split into two. Having more detections in the track even if they have lower confidence scores doesn't hurt the overall resulting speed measurement since we determine the speed as the median of inter-frame speeds.

### 4.2. Cropping the viewing window

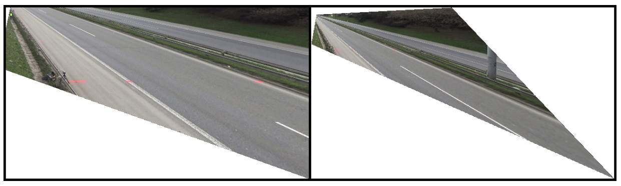
Figure 3. The transformed image when the viewing window is cropped (left) and when the original viewing window is used (right).

| Method | Recall | Precision |
|---|---|---|
| Cropped | 0.8955 | 0.8360 |
| Uncropped | 0.2582 | 0.4595 |

Table 1. Precision and recall achieved on *session 5 left* video for the *Transform3D* utilizing cropping of the viewing window (Cropped) and the baseline *Transform3D* (Uncropped).

In our experiments we noticed that for one of the testing videos (*session 5 left*) the recall values were significantly lower than for the rest of the videos. The reason for this was the fact that the second vanishing point position was too close to the center of the image and the resulting transformed image was too distorted for the image detector to work. To remedy this we employed the strategy described in subsection 3.1. We crop the original 1920 by 1080 pixel viewing window from the left by 100 pixels, from top by 200 pixels, and from the right by 480 pixels and use the reduced viewing window to perform the transformation. See Figure 3 for comparison of the transformed images.

In the Table 1 we show the recall and precision values for both the baseline version and the cropped version. Recall is significantly higher for the cropped version. This indicates that camera placement can affect the algorithm significantly. However, even in a case of bad viewing angle a simple manual setting can provide results similar to changing the camera position.

As the abysmal recall would significantly skew the overall recall of the method, we use the cropped version of the *session 5 left* video in the overall results including the ablation experiments. Other videos could also benefit from employment of this strategy, but we opt to not use cropping when not necessary.

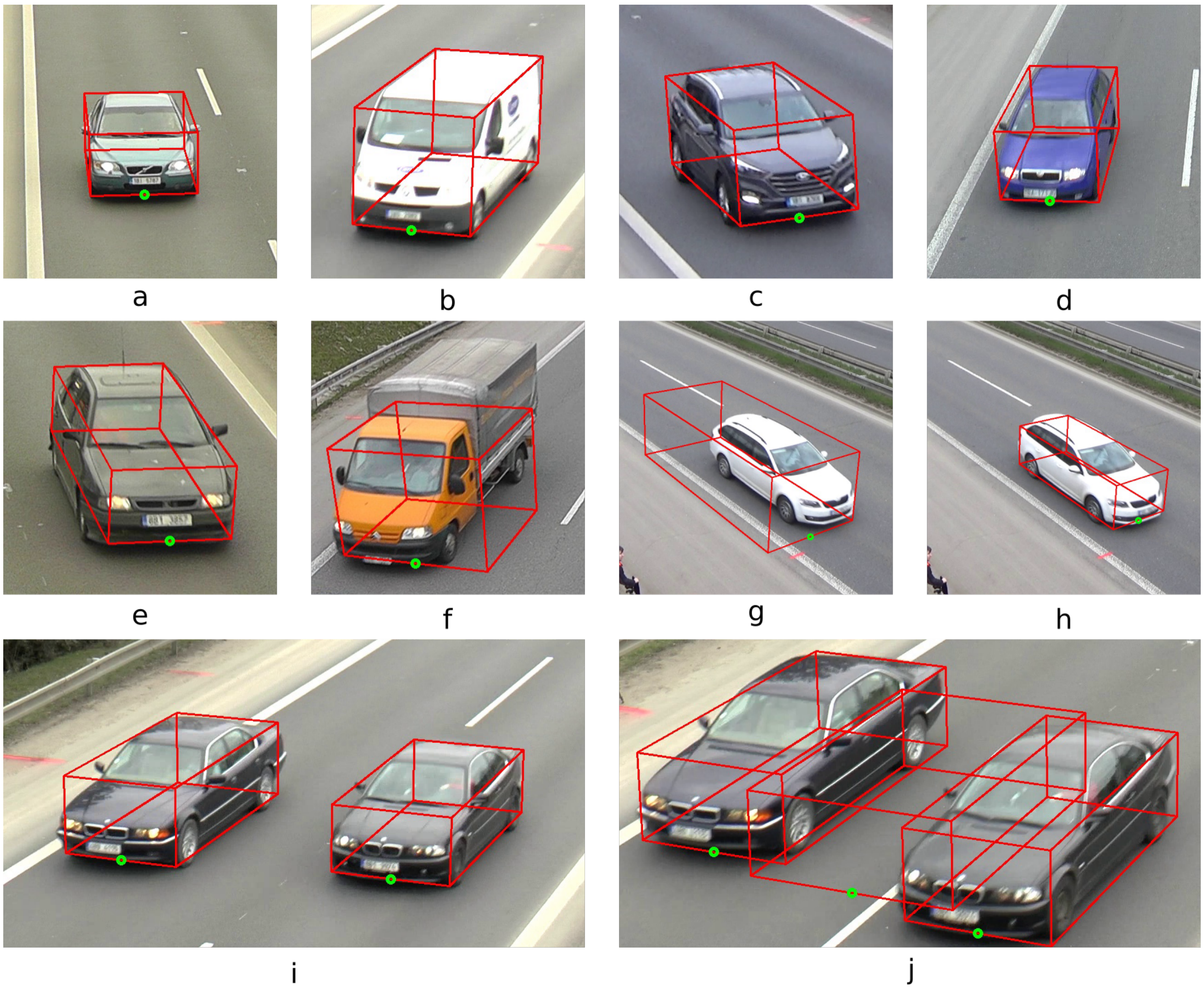

Figure 4. 3D bounding boxes (red) and the reference point for speed measurement (green) detected on the testing data: correct bounding boxes (a-d), bounding boxes with bad dimensions (e,f), the same frame from *session 5 left* without cropping (g) and with cropping (h), a pair of cars detected properly (i), few frames later at the border of the viewing window a false positive appears (j).

| Method | Mean error (km/h) | Median error (km/h) | 95-th percentile (km/h) | Mean Recall (%) | Mean Precision (%) |
|---|---|---|---|---|---|
| Transform3D (ours) | 0.86 | 0.65 | 2.17 | **89.32** | 87.67 |
| Transform2D (ours) | **0.83** | **0.60** | **2.04** | 82.06 | 83.53 |
| Orig2D (ours) | 0.97 | 0.79 | 2.25 | 85.96 | 86.44 |
| SochorAuto [25] | 1.10 | 0.97 | 2.22 | 83.34 | **90.72** |
| SochorManual [25] | 1.32 | 0.95 | 3.45 | 83.34 | **90.72** |

Table 2. The results of the compared methods. Mean, median and 95-th percentile errors are calculated as means of the corresponding error statistics for each video. Recall and precision are averaged over the videos in the test set.

## 4.3. Speed measurement accuracy

In the Table 2 the resulting mean absolute error compared to the measured ground truth speed measurement is shown as well as mean recall and precision for the detected tracks. We make our results available online [1]. Our method achieves lower mean error and significantly lower median error than both the fully automatic method *SochorAuto* and a method using manual calibration *SochorManual*, while also

[1]`https://github.com/kocurvik/CVWW2019_results`

significantly increasing recall and decreasing precision. Since we use the same calibration as *SochorAuto* the only difference is in the detection of vehicles. Our method therefore detects the positions of vehicles more accurately.

From the results of the ablation experiments it can be noted that *Transform2D* outperforms *Transform3D* in terms of lower speed measurement error. There is a tradeoff between accuracy and recall. In some cases *Transform2D* produces a bounding box whose parts lie outside of the original image. The point which is used for tracking can therefore be discarded by the evaluation algorithm provided in [26], which may lead to the whole track being discarded. We opted to not modify the evaluation script to keep the results comparable with other research. This effect usually occurs in videos with significant distortion and therefore we expect the omitted cases to be the difficult ones, thus lowering the speed measurement error. However, we cannot conclude that there is any benefit to using 3D bounding boxes over 2D bounding boxes with regards to the speed measurement task.

The ablation method *Orig2D* also outperforms the methods from [25] with respect to errors, but by a significantly lower margin. This indicates that transforming the image is beneficial to speed measurement, but significant improvements can be obtained just by using a better object detector and training data.

### 4.4. Computational efficiency

Our method runs in real-time (25 FPS) on an Nvidia GTX 970 GPU. We were unable to obtain the FPS of the model we compare against, but we expect it to be lower as it uses the Faster R-CNN object detector, which is in general significantly slower compared to the detector we used [14].

## 5. Conclusion

We propose a method to detect and track 3D bounding boxes of vehicles in a standard traffic surveillance scenario. Our methods are based on applying a deep convolutional neural network for object detection on an image which has been rectified by a perspective transformation based on known positions of vanishing points. 3D bounding boxes can be detected directly without the need of obtaining the contours of the vehicles.

Our method improved the mean absolute error on a speed measurement task by 22 % (1.10 km/h to 0.86 km/h) and median error by 33 % (0.97 km/h to 0.65 km/h) compared to the existing state-of-the art fully automatic method, while also increasing the recall.

## Acknowledgements

I would like to thank my supervisor Milan Ftáčnik for his guidance in this research. I would also like to thank Adam Herout for his valuable comments. I gratefully acknowledge the support of NVIDIA Corporation with the donation of a GPU.

## References

[1] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001. 2

[2] F. Cathey and D. Dailey. A novel technique to dynamically measure vehicle speed using uncalibrated roadway cameras. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 777–782. IEEE, 2005. 2

[3] E. R. Corral-Soto and J. H. Elder. Slot cars: 3d modelling for improved visual traffic analytics. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2017. 2

[4] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 1, 2

[5] 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. 2

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009. 5

[7] V.-H. Do, L.-H. Nghiem, N. P. Thi, and N. P. 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. IEEE, 2015. 2

[8] M. Dubská, A. Herout, and J. Sochor. Automatic camera calibration for traffic understanding. In *BMVC*, volume 4, page 8, 2014. 1, 2

[9] L. Grammatikopoulos, G. Karras, and E. Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Educa-*

*tion and Professional Practice in Geodesy and Related Fields*, pages 332–338, 2005. 2

[10] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017. 2, 5

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5

[12] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*, volume 4, 2017. 2

[13] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960. 3

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 1, 2, 4, 8

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 5

[16] M. Liu and M. Zhu. Mobile video object detection with temporally-aware feature maps. *arXiv preprint arXiv:1711.06368*, 2017. 3

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1, 2, 5

[18] 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, 2017. 3

[19] 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. IEEE, 2008. 2

[20] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler. Online multi-target tracking using recurrent neural networks. In *AAAI*, volume 2, page 4, 2017. 3

[21] G. Ning, Z. Zhang, C. Huang, X. Ren, H. Wang, C. Cai, and Z. He. Spatially supervised recurrent convolutional neural networks for visual object tracking. In *Circuits and Systems (ISCAS), 2017 IEEE International Symposium on*, pages 1–4. IEEE, 2017. 3

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1, 2

[23] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

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

[25] J. Sochor, R. Juránek, and A. Herout. Traffic surveillance camera calibration by 3d model bounding box alignment for accurate vehicle speed measurement. *Computer Vision and Image Understanding*, 161:87–98, 2017. 1, 2, 3, 4, 6, 7, 8

[26] J. Sochor, R. Juránek, J. Špaňhel, L. Maršík, A. Široký, A. Herout, and P. Zemčík. Brnocompspeed: Review of traffic camera calibration and comprehensive dataset for monocular speed measurement. *arXiv preprint arXiv:1702.06441*, 2017. 3, 5, 6, 8

[27] J. Sochor, J. Špaňhel, and A. Herout. Boxcars: Improving fine-grained recognition of vehicles using 3-d bounding boxes in traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 2018. 3, 5

[28] H. Van Pham and B.-R. Lee. Front-view car detection and counting with occlusion in dense traffic flow. *International Journal of Control, Automation and Systems*, 13(5):1150–1160, 2015. 2

[29] C. Vieren, F. Cabestaing, and J.-G. Postaire. Catching moving objects with snakes for motion tracking. *Pattern recognition letters*, 16(7):679–685, 1995. 2

[30] D. Zapletal and A. Herout. Vehicle re-identification for automatic video traffic surveillance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 25–31, 2016. 3

[31] Y. Zhou, L. Liu, L. Shao, and M. Mellor. Dave: a unified framework for fast vehicle detection and annotation. In *European Conference on Computer Vision*, pages 278–293. Springer, 2016. 2