A Semi-Automatic 2D solution for Vehicle Speed Estimation from Monocular Videos

Amit Kumar*, Pirazh Khorramshahi*, Wei-An Lin* , Prithviraj Dhar, Jun-Cheng Chen and Rama Chellappa
Center for Automation Research , UMIACS
University of Maryland, College Park
{akumar14,pirazhkh,walin,prithvi,rama}@umiacs.umd.edu,pullpull@cs.umd.edu

Abstract

In this work, we present a novel approach for vehicle speed estimation from monocular videos. The pipeline consists of modules for multi-object detection, robust tracking, and speed estimation. The tracking algorithm has the capability for jointly tracking individual vehicles and estimating velocities in the image domain. However, since camera parameters are often unavailable and extensive variations are present in the scenes, transforming measurements in the image domain to real world is challenging. We propose a simple two-stage algorithm to approximate the transformation. Images are first rectified to restore affine properties, then the scaling factor is compensated for each scene. We show the effectiveness of the proposed method with extensive experiments on the traffic speed analysis dataset in the NVIDIA AI City challenge. We achieve a detection rate of 1.0 *in vehicle detection and tracking, and Root Mean Square Error of* 9.54 *(mph) for the task of vehicle speed estimation in unconstrained traffic videos.*

1. Introduction

With increased popularity of deep learning and autonomous driving technologies, intelligent traffic analytics, including vehicle speed estimation, anomaly event detection on streets (*e.g.*, traffic accidents), and vehicle re-identification, have become active research areas. In this work, we focus on estimating the vehicle speed, a crucial information for traffic analysis which can be applied to detect traffic congestions or other anomalous events. Specifically, we consider the problem of vehicle speed estimation from monocular videos. The task is challenging in two aspects. First, a robust vehicle detection and tracking algorithm is required to localize individual vehicles over time under varying car orientations or lighting conditions. Incorrect localization and tracking can lead to catastrophic results. Second, the transformation from image space to real world is often unavailable and requires expensive measuring equipments such as LIDARs.

To address these issues, we present a semi-automatic approach for vehicle speed estimation which works reliably for monocular videos and only requires minimal amount of measurements. While deep learning has achieved state-of-the art results for many vision tasks, including object detection [14, 17], image recognition [10, 13], and object tracking [22], its application in traffic surveillance domain is still under-studied. Many existing Intelligent Traffic Systems (ITS) are still based on traditional techniques such as background subtraction and vehicle segmentation using hand-crafted features which are sensitive to noise. In this work, we employ the state-of-the-art object detector, Mask-RCNN [9], to generate reliable vehicle bounding boxes. Given these bounding boxes, two efficient tracking algorithms, SORT [1] and DeepSORT [21] are applied to track individual vehicles and estimate velocities in the image domain.

Arguably, the most challenging task in vehicle speed estimation is to model the transformation from the image domain to the real world such that the speed of the vehicles can be inferred from measurements taken in the image domain. While most of the previous methods resort to obtaining accurate correspondences between points in the 3D world and image plane through extensive measurements and camera calibration, we propose a simple two-stage algorithm to approximate the transformation. We first estimate a rectifying transformation using vanishing points in the image to restore the affine properties. Velocities measured in the image domain can be rectified correspondingly. We then measure the scaling factor from the image domain to the real world by comparing the actual lane width to the one in the rectified image. Linear interpolation is applied to the scaling factors to compensate for non-planar regions in the scene. An overview of the entire system is shown in Figure 1.

To summarize, the main contribution of this work is to propose a semi-automatic vehicle speed estimation ap-

*The first three authors equally contribute to this work.

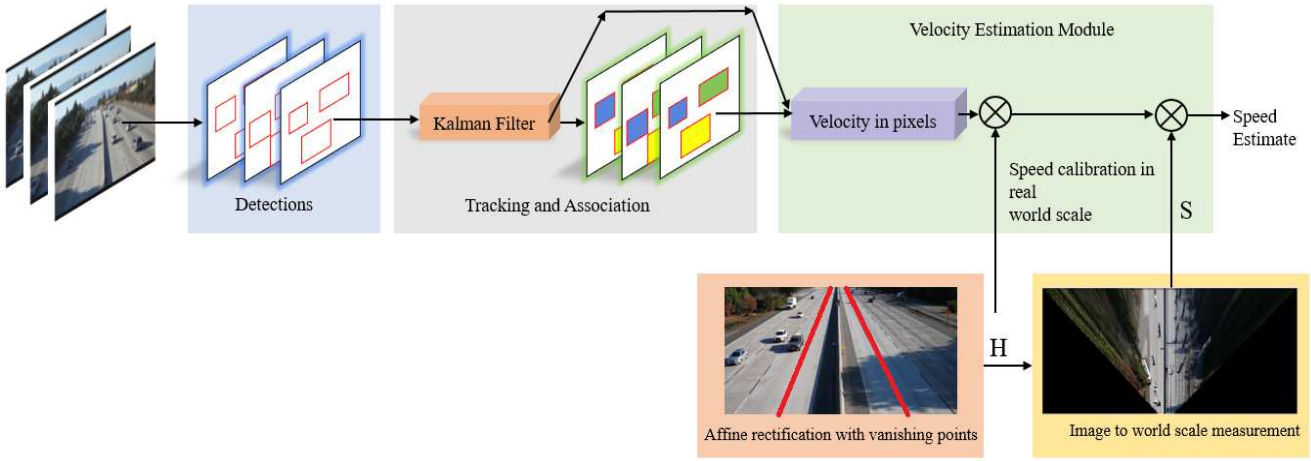


Figure 1: Overview of the proposed system for vehicle speed estimation from monocular videos.

proach, which does not require expensive steps such as in camera calibration and measuring 3D distance s using lasers and LIDARS. For the traffic speed analysis dataset in the NVIDIA AI City Challenge, the proposed framework achieves an RMSE of 9.54 mph.

The rest of the paper is organized as follows: We briefly summarize some related works in Section 2. The proposed method is presented in Section 3. Finally, experimental evaluations are detailed in Section 4.

2. Related Works

In vehicle speed estimation, the most challenging task is to relate measurements taken in the image domain to real world metrics. In the following, we categorize previous works into methods that require camera calibration (obtaining intrinsic and extrinsic parameters of the surveillance camera) and methods that directly estimate the transformation from image domain to 3D world.

Most of the approaches such as [2,8,12] estimate camera parameters by assuming the knowledge of exact point correspondences between the image plane and 3D world are provided. Other methods [3, 5] are based on 3D models and vehicle motion to calibrate the camera. [3, 4] are fully automatic as they use mean vehicle dimensions for camera calibration. [18] aligns 3D bounding boxes for vehicles for camera calibration. However without prior knowledge about the types of vehicles or extensive distance measurement on the road, these approaches are not applicable.

In [7] the authors assume that the camera is only tilted along the axis perpendicular to the road. By locating the vanishing point in the direction of the road axis, the image can be rectified. Vehicle tracking and speed estimation are performed in the rectified image domain. Cathey and Dailey [2] used a method based on detecting the vanishing point in the direction of vehicle movements. This vanishing point is detected as the intersection of line markings with least squares adjustment. The scale (pixels/meters ratio) for the camera is computed from average line marking stripe length and known stripe length in the real world. We refer readers to [19] for a more detailed summary. In this work, we use a combination of the above two approaches, where the transformation from the image domain to 3D world is measured by rectification and prior knowledge from Google maps.

3. Proposed Approach

The proposed vehicle speed estimation system consists of three components: (1) vehicle detection (2) tracking, and (3) speed estimation. We describe the details of each component in the following sections. A flowchart of the proposed approach is shown in Figure 1.

3.1. Vehicle Detection

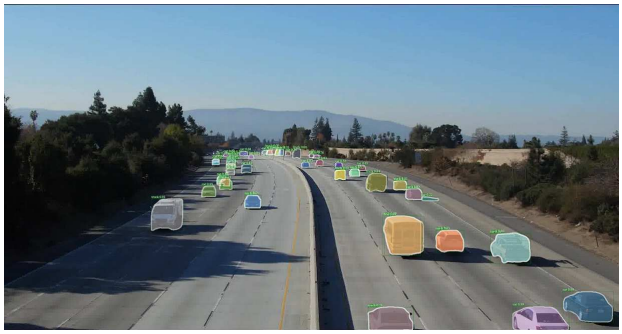


Figure 2: Vehicle detection and segmentation results using Mask-RCNN [9] for a Track 1 video frame of NVIDIA AI City Challenge.

To localize vehicles in traffic videos, we apply Mask R-CNN [9], an extension of the well-known Faster R-CNN [17] framework. In addition to the original classification and bounding box regression network of Faster R-CNN, the Mask-RCNN adds another branch to predict segmentation masks for each Region of Interest (RoI). This is done by applying a small fully convolutional network (FCN) to each RoI to predict a pixel-level segmentation mask. The joint learning of detection and segmentation allows the Mask R-CNN to localize the vehicle with reliable and tight bounding boxes. In Figure 2, we demonstrate a sample result from one of the video frames from Track 1 of NVIDIA AI City Challenge. We observe that Mask-RCNN is able to detect the vehicles at different scales, which is crucial for this challenge as vehicles can appear at any scale in the image. The tracking and speed estimation module use the bounding boxes from Mask-RCNN.

3.2. Tracking

In this section, we briefly describe the algorithm used to track vehicles. We compare and contrast two different tracking algorithms: Simple online and real time tracking (SORT) [1] and DeepSORT [21]. It is noteworthy to mention that the main advantage of employing SORT and DEEPSORT for tracking, is that the pixel velocity can be jointly estimated with the tracking. Even though some of the deep learning based approaches have motion models, they do not have explicit relations with velocity. Moreover, these approaches require training sequences which are not available.

SORT. SORT is a real time, online tracking algorithm which has the accuracy comparable to the state of the art online trackers while supporting higher update rates. It associates detections in every frame using Kalman Filters, specifically, SORT approximates the dynamics of each target vehicle with a linear Gaussian state space model. The state of each target is modeled as:

$$[x, y, s, r, \dot{x}, \dot{y}, \dot{s}] \tag{1}$$

where x and y represent the horizontal and vertical pixel location of the center of the bottom two coordinates of the bounding box, while s and r represent the area and the aspect ratio of the bounding box respectively. When a detection is associated to a target, the detected bounding box is used to update the state vector of the target.

Deep SORT. Deep SORT is another online tracker with competitive performance to the state of the art online trackers. In DeepSORT, the state vector is defined as:

$$[x, y, h, r, \dot{x}, \dot{y}, \dot{h}, \dot{r}] \tag{2}$$

where x, y, r are same as in (1), while h represents the height of the bounding box.

We track the center of the bottom two corners as these points are close to the ground plane and have the least amount of distortion under the affine rectification presented in Section 3.3. The vector $(\dot{x}, \dot{y})$ is used to approximate the velocity of a vehicle in the image domain.

Data Association in SORT v.s. DeepSORT. The major difference between SORT and DeepSORT is the way they perform data association. In SORT, the new bounding box location for each target is predicted by the Kalman Filter. The assignment cost is then computed as the intersection-over-union (IOU) between each incoming detection and all predicted bounding boxes of the existing targets. The assignment is solved optimally using the Hungarian algorithm. Additionally, a minimum IOU is imposed to reject assignments when the overlap between the detection and the target is less than IOU_{min}. In the case of DeepSORT, for each detected bounding box, an appearance feature is extracted by a pretrained CNN. The training detail is presented in Section 4.1. For each track, a gallery of associated appearance features is maintained. An incoming detection is assigned to the track with the smallest cosine distance.

3.3. Speed Estimation

After obtaining pixel velocities from the Kalman filter in the image domain, our speed estimation approach has two steps: (1) affine rectification to restore affine-invariant properties of the scene, and (2) scale recovery which estimates vehicle speed in the real world.

3.3.1 Affine Rectification

The traffic videos are captured by uncalibrated cameras. Therefore, each frame in the video is the projection of the 3D world on the image plane under a projective transformation. Since ratios between segments are not preserved under projective transformations, estimating speed directly in the image domain is difficult. In this work, we assume most of the roads can be well-approximated by a plane. For these planar regions, we apply the rectification technique which estimate a homography $\mathbf{H}$ that maps points $\mathbf{x} = [x, y, 1]^T$ in the image domain to points $\mathbf{X} = [X, Y, 1]^T$ in the rectified domain. That is,

$$\mathbf{Hx} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{X}. \tag{3}$$

Non-planar regions would result in degraded speed estimation, which will be compensated by the scale recovery presented in Section 3.3.2. We observe that once $\mathbf{H}$ is determined, the velocity in the rectified domain can be obtained by differentiating both sides of (3):

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} \frac{h_{11}+C_{2,3}y}{(h_{31}x+h_{32}y+1)^2} & \frac{h_{12}-C_{2,3}x}{(h_{31}x+h_{32}y+1)^2} \\ \frac{h_{21}+C_{1,3}y}{(h_{31}x+h_{32}y+1)^2} & \frac{h_{22}-C_{1,3}x}{(h_{31}x+h_{32}y+1)^2} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix}, \tag{4}$$

where $C_{i,j}$ represents the minor of $\mathbf{H}$ which corresponds to $h_{i,j}$.

One classical approach of estimating $\mathbf{H}$ is based on detecting vanishing points for two perpendicular directions. We denote the two vanishing points as $\mathbf{v}_1$ and $\mathbf{v}_2$ in homogeneous coordinates, where $\mathbf{v}_1$ corresponds to the direction of the road axis, and $\mathbf{v}_2$ corresponds to the direction perpendicular to $\mathbf{v}_1$. Then the following equations hold:

$$\mathbf{H}\mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{H}\mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{5}$$

where $\mathbf{H}$ has the form

$$\begin{bmatrix} a & b & 0 \\ c & d & 0 \\ - & \mathbf{v}_1 \times \mathbf{v}_2 & - \end{bmatrix}. \tag{6}$$

Parameters in (3) can be solved using (5) and (6). For each location, we manually detect two vanishing points by selecting points in the scene. In locations 1, 2, and 3, since lines that are perpendicular to the road axis are almost parallel in the image domain, we set $\mathbf{v}_2 = [0, 1, 0]^T$ for these locations.

3.3.2 Scale Recovery

Image rectification proposed in Section 3.3 can result in distortion of non-planar regions. Hence, after image rectification, it is important to recover the scale in order to translate the speed from pixel space into real world domain. This recovery needs to be done in both horizontal and vertical directions. We use the lane widths and road white strips length for the purpose of scale recovery in the horizontal and vertical directions respectively. Using Google Maps, we obtained a rough estimate of the real world distances. After rectification road lanes become parallel which ensures a constant lane width in the x direction in planar regions. Therefore, we choose the following as the scaling factor in x direction:

$$s_x = \frac{W}{w} \tag{7}$$

In (7) W and w denote the actual lane width in meters and pixels respectively. For the vertical direction, we propose another approach. After projection and rectification, the pixels along the vertical direction gets stretched and this effect is more prominent in the pixels near the detected vanishing point. Therefore, the scale along the y direction changes non-linearly. To compensate for this scale variation, we employed a linear compensator as follows:

$$s(y) = \frac{s_2 - s_1}{y_{max} - y_{min}} y + s_1, \tag{8}$$

$$s_1 = \frac{L_1}{l_1}, \tag{9}$$

$$s_2 = \frac{L_2}{l_2}, \tag{10}$$

where L_1 and L_2 are lengths of two white strips at two different heights in meters. l_1 and l_2 are the corresponding lengths in pixels in the transformed image. Finally, The speed estimate of the vehicle is then given by

$$\sqrt{(s_x\dot{X})^2 + (s_y\dot{Y})^2}. \tag{11}$$

We estimate the speed of all the vehicles in the window whose heights in the image space are from y_{min} to y_{max}.

4. Experimental Evaluations

In this section, we evaluate the proposed algorithm on the NVIDIA AI City Challenge Dataset. We first conduct qualitative and quantitative evaluations for our tracking algorithms. We then report the results of our vehicle speed estimation from monocular videos.

4.1. Implementation Details

Vehicle Appearance Model. To learn discriminative vehicular representations for DeepSORT, we train a DCNN for the task of fine grained vehicle identification. The network architecture is shown in Figure 3. We train the network on the CompCars dataset which consists of 136,726 images for 163 car makes and 1,716 car models. The network is trained with batch size 128. The learning rate is set to 0.01 and is halved after 50K iterations. We initialize $\alpha = 40$ for L_2-softmax.

Vehicle Detector. We use Mask-RCNN implemented in Detectron [6] which is trained on the MS-COCO dataset [15]. It is based on a feature pyramid network [14] and a ResNet-101 as the backbone.

4.2. Evaluation Metric

The performance of vehicle speed estimation is calculated using the $S1$ metric.

$$S1 = DR \times (1 - NRMSE), \tag{12}$$

in which DR and $NRMSE$ represents the detection rate and the normalized root mean square error (RMSE) with respect to other participating teams respectively. DR is defined as the ratio between detected and ground truth vehicle;

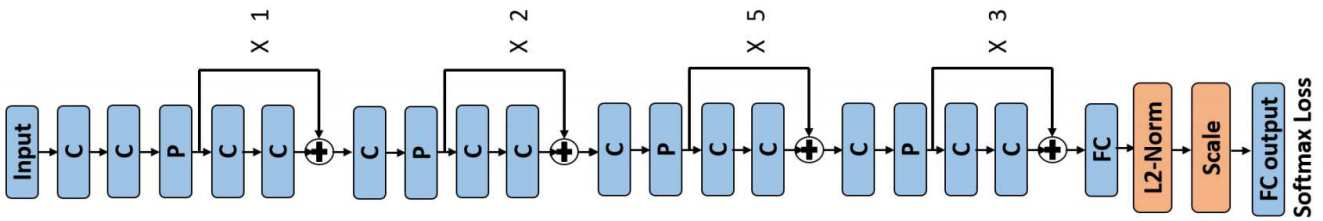
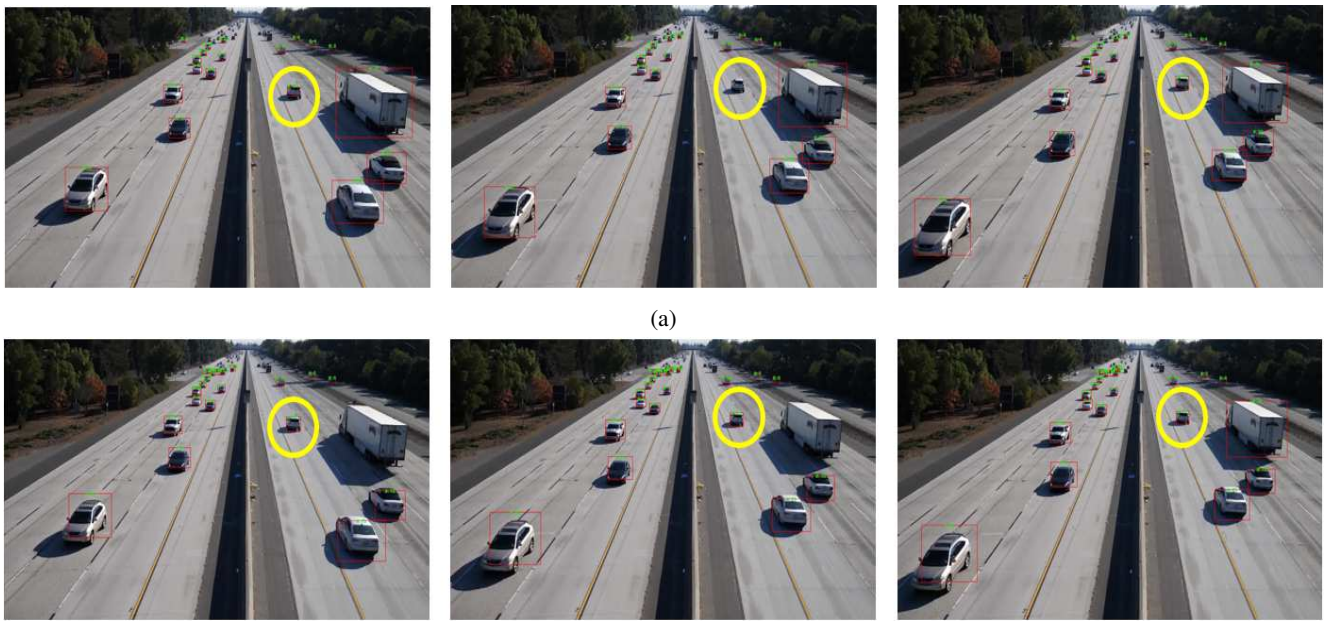


Figure 3: Network architecture proposed in [16,20]. C denotes a Convolution Layer followed by a PReLU nonlinearity [11]. P denotes a Max Pooling Layer. Each pooling layer is followed by a set of residual connections. The number of residual connections is presented alongside. After the fully-connected layer (FC), an L_2-Normalization layer and Scale Layer are added and followed by a softmax loss layer. The figure is cited from [16].



(a)

(b)

Figure 4: Comparison between tracking results obtained from SORT and DeepSORT. (a) In SORT a track is lost and reinitialized after a few frames. (b) DeepSORT is able to effectively track the vehicle without losing the track.

moreover, a vehicle is alleged to be detected if it is at least localized in 30% of all the time it shows up in the video. Additionally, a vehicle is localized if at least one predicted bounding box exists with IOU of 0.5 or higher relative to the annotated bounding box for the vehicle. The speed estimate error is computed as the RMSE of the ground truth vehicle speed and predicted speed for all correctly localized ground-truth vehicles.

$$NRMSE_i = \frac{RMSE_i - RMSE_{min}}{RMSE_{max} - RMSE_{min}}, \qquad (13)$$

where $RMSE_{min}$ and $RMSE_{max}$ are the smallest and largest estimation error among all participating teams for NVIDIA AI City Challenge, respectively.

Method	Detection Rate	RMSE
SORT	1.00	9.54
DeepSORT	1.00	10.10

Table 1: Comparison of speed estimation on the NVIDIA AI City Challenge dataset using different tracking algorithms.

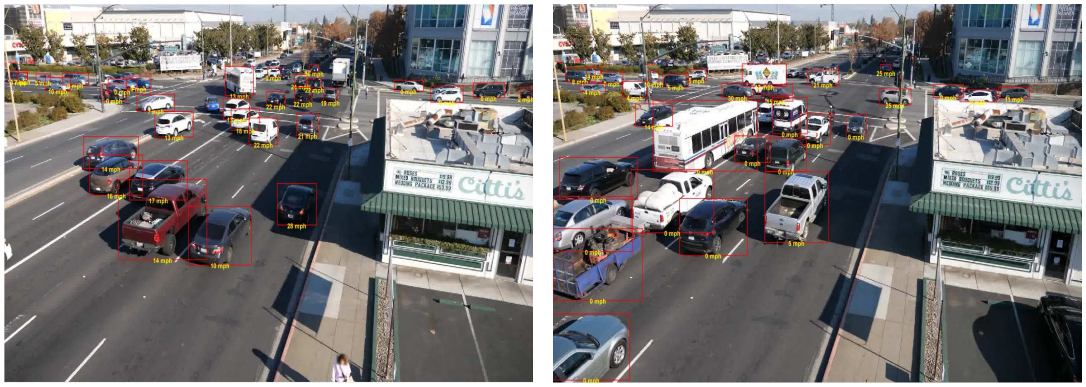
4.3. Vehicle Tracking and Speed Estimation

In Figure 4 and Table 1, we compare the tracking performance of SORT and DeepSORT. It can be observed from Figure 4 that since DeepSORT is assisted by the learned deep representation, the tracking results are more stable and consistent than SORT which only uses bounding box infor-

(a)

(b)

(c)



(d)

Figure 5: Comparison between tracking results obtained from 4 different locations from NVIDIA AI City Challenge dataset. (Best viewed in color when zoomed in)

mation for association and usually misses the track when the detections of the tracked vehicle are not available for several consecutive frames. This demonstrates a tracking method based on similarity model can achieve stable tracking results with fewer id switches. This effect is shown in Figure 4. However, quantitative results from the evaluation server show that RMSE obtained using SORT is slightly lower when DeepSORT is used as a tracking algorithm. We hypothesize that since the condition to declare a vehicle detection only requires a vehicle to be localized for 30% of frames through the entire video, both approaches can achieve similar DR with proper hyper-parameter settings. However, due to the design of the evaluation metric, if a vehicle is tracked for a long time, it is more likely to make errors in speed estimation.

Finally, we visualize the speed estimation for several sampled frames in Figure 5. For the Track 1 of NVIDIA AI City Challenge, the proposed system achieved $S1 = 0.7654$ which puts this approach in 7^{th} position among all participating teams.

5. Conclusions and Future Work

In this paper, we presented a semi-automatic system for vehicle speed estimation from monocular videos. The proposed approach can reliably estimate the speed without explicit camera calibration and 3D modeling for vehicles or buildings. For our future work, key-point detection and tracking can be aggregated in the pipeline for improved speed measurements.

6. Acknowledgement

This research is based upon work supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number D17PC00345. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes not withstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied of IARPA, DOI/IBC or the U.S. Government.

References

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016. 1, 3

[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] 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

[4] M. Dubská and A. Herout. Real projective plane mapping for detection of orthogonal vanishing points. In *BMVC*, 2013. 2

[5] M. Dubská, A. Herout, R. Juránek, and J. Sochor. Fully automatic roadside camera calibration for traffic surveillance. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1162–1171, 2015. 2

[6] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár, and K. He. Detectron. https://github.com/facebookresearch/detectron, 2018. 4

[7] L. Grammatikopoulos, G. Karras, and E. Petsa. Automatic estimation of vehicle speed from uncalibrated video sequences. In *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pages 332–338, 2005. 2

[8] L. Grammatikopoulos, G. Karras, and E. Petsa. An automatic approach for camera calibration from vanishing points. *ISPRS journal of photogrammetry and remote sensing*, 62(1):64–76, 2007. 2

[9] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 1, 2, 3

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, Dec 2015. 5

[12] X. He and N. H. C. Yung. New method for overcoming ill-conditioning in vanishing-point-based camera calibration. *Optical Engineering*, 46(3):037202, 2007. 2

[13] A. Krizhevsky, I. Sutskever, and G. 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. 1

[14] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, volume 1, page 4, 2017. 1, 4

[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. 4

[16] R. Ranjan, C. D. Castillo, and R. Chellappa. L2-constrained softmax loss for discriminative face verification. *CoRR*, abs/1703.09507, 2017. 5

[17] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1, 3

[18] J. Sochor, R. Juranek, 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. 2

[19] J. Sochor, R. Jurnek, J. pahel, L. Mark, A. irok, A. Herout, and P. Zemk. Brnocompspeed: Review of traffic camera calibration and comprehensive dataset for monocular speed measurement, 2017. 2

[20] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016. 5

[21] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017. 1, 3

[22] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, and J. Yan. POI: multiple object tracking with high performance detection and appearance feature. *CoRR*, abs/1610.06136, 2016. 1