This CVPR workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Geometry-aware Traffic Flow Analysis by Detection and Tracking

[1,2]Honghui Shi, [1]Zhonghao Wang, [1,2]Yang Zhang, [1,3]Xinchao Wang, [1]Thomas Huang
[1]IFP Group, Beckman Institute at UIUC, [2]IBM Research, [3]Stevens Institute of Technology
{hshi10, zwang246, yzhan143, xinchao, t-huang1}@illinois.edu

Abstract

In the second Nvidia AI City Challenge hosted in 2018, the traffic flow analysis challenge proposes an interest task that requires participants to predict the speed of vehicles on road from various traffic camera videos. We propose a simple yet effective method combing both learning based detection and geometric calibration based estimation. We use a learning based method to detect and track vehicles, and use a geometry based camera calibration method to calculate the speed of those vehicles. We achieve a perfect detection rate of target vehicles and a root mean square error (RMSE) of 6.6674 in predicting the vehicle speed, which rank us the third place in the competition.

1. Introduction

To accurately estimate the speed of vehicles from traffic camera videos is an interesting research topic for computer vision. The track 1 of Nvidia AI City Challenge [14] in 2018, namely the traffic flow analysis challenge, deals with predicting the speed of vehicles on road in various traffic camera videos. This paper introduces our method for tackling the speed estimation task in this challenge. We use state-of-the-art region-based detector to detect vehicles on road and the simple Medianflow algorithm to track vehicles detected in the video. We propose a method involving camera calibration method to estimate the speed of those vehicles. Though simple, our method effectively achieves high accuracy among submissions from all participating teams.

2. Related Work

Object Detection: State-of-the-art region-based object detectors [15, 3, 8, 1, 2] trained on auxiliary large-scale detection datasets [12, 4, 16, 5] are able to detect vehicles from traffic camera videos accurately without learning from scratch, weak supervision, or specific domain adaptation [17, 18, 9, 19]. Multiple bounding boxes are drawn to where vehicles are present in frames extracted from the traffic camera videos. To infer the vehicle instances in each frame, we run our detection model mainly based on the newly released deep learning frameworks [6]. We use the model ResNet-101 Feature Pyramid Network [11] and the weights downloaded from Mask R-CNN source file for the Mask R-CNN structure. As the model returns bounding boxes for all instances it recognizes in a frame, we filter out the bounding boxes of other instances but vehicles including cars, trucks and buses. We consider the bounding box with inference confidence higher than certain threshold as valid bounding boxes for vehicles. An example of accurate vehicle detection can be seen in Figure 1 and Figure 2.

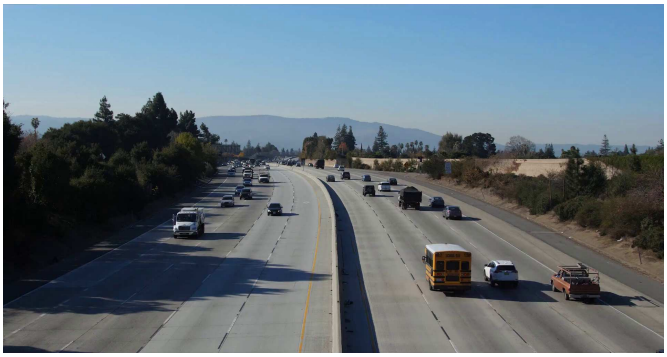

Figure 1. An example of original frames from location 1

Figure 2. An example of recognition results

Object Tracking: Even though we can use Mask R-CNN detect vehicles in a frame with good effects. However, there

Figure 3. An example showing detection failure

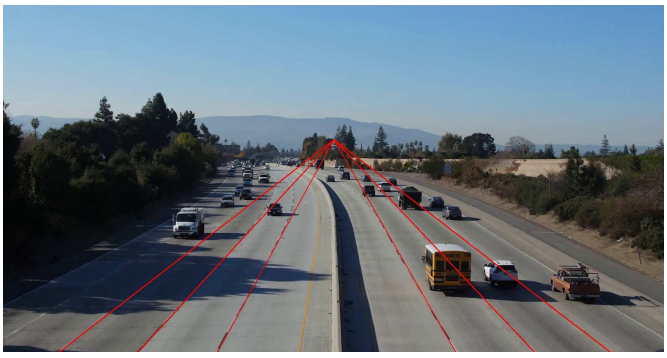

Figure 4. Illustration of finding vanishing point

still can be vehicles close to the camera but not recognized. Shown in Figure 3, the white van is such an example. If we only calculate the speed of vehicles based on the detected instances, then we may easily lose track of a vehicle due to such a problem. Therefore, to increase the probability that we know the positions of a vehicle in at least a fixed amount of frames, we use a tracking algorithm to track a vehicle detected in the following frames. In this way, we can calculate the speed of that vehicle based on 4 frames at least in most cases. Due to a small and predictable motion of the same vehicle between adjacent frames, Medianflow tracker [10] can efficiently track that vehicle. This tracker is a key point based tracker, and uses Lucas-Kanade algorithm [13] to produce a trajectory of its tracking object. It also filters invalid tracking points and takes the scale variance of an object into an account. Thus, the scale of the bounding box generated by the Medianflow tracker varies with respect to the scale changes of a vehicle moving close to or away from the camera. Using the centers of the bounding boxes generated for a vehicle by both Mask R-CNN and the Medianflow tracker, we can get the trajectory of that vehicle in continuing frames at least with high probability. Alternatively, we can also use post-processing techniques such as Seq-NMS [7] to enhance the per-frame detection results to solve the detection failure case. However, such method requires us to detect on each frame and are more computationally expensive.

Geometry Understanding: We first find the vanishing point of the first frame of a video. We observe that the lanes of the road are parallel to each other in a large area of each frame. Therefore, we first find the dash line which separates two lanes in a frame where straight segments of the dash line can be seen. Then use a math formula to express the extended lines of those straight dash line segments. We let the vanishing point be the point in the frame which has the lowest mean square distance to all those straight dash line segments. This is shown in Figure 4. Even though the camera is shaky sometimes, the vanishing point does not change its position in frames very much. For now, we manually find the straight dash line segments. In the future, we can develop a more comprehensive way in detecting the straight dash line segments. Then we construct a distance measurement formula to measure the displacement of a vehicle between frames in real world. Given this distance and frame rate of the video, we can compute the speed of that vehicle. This is further introduced in the method section.

3. Proposed Method

3.1. Detection and Tracking

We use the pretrained model of Mask R-CNN to detect vehicles in frames with an interval of three frames. The model we use is ResNet 101 Feature Pyramid Network. The structure of this neural network and its weights are available on the source repository of Mask R-CNN. After we get the detection bounding boxes for the vehicles, we use Medianflow algorithm to track the detected vehicles in the following three frames. Then, we use centers of those bounding boxes to represent the positions of those vehicles in the frame. To know the trajectory of a vehicle, we find the closest vehicle position in the next frame to the position of that vehicle in the current frame. If the distance between these two positions is above a certain bound, then we say we lose the track of that vehicle.

3.2. Geometry-aware Traffic Flow Analysis

To understand the distance relationship between a line segment in a frame and the distance that line segment indicates in real world, we construct a formula to convert the distance in pixels in a frame to the distance in real world. We first find the vanishing point of the lane separation dash lines. Thus, all lines in the image that go through that vanishing point are parallel to each other. We observe the camera plane is relatively perpendicular to the lanes on road. Thus, in real world, the distance between any point in the image and the vanishing point equals the distance between the vanishing point and the intersection of a perpendicular

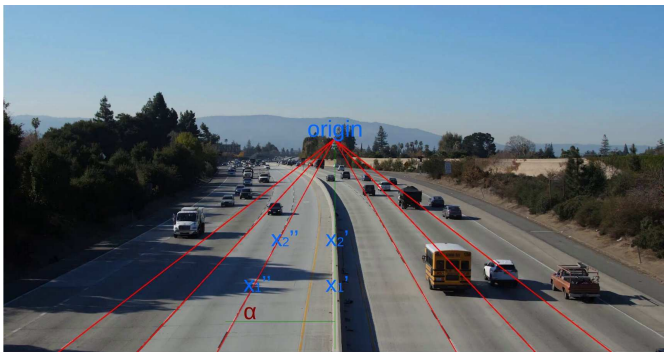

Figure 5. Illustration of distance relationship

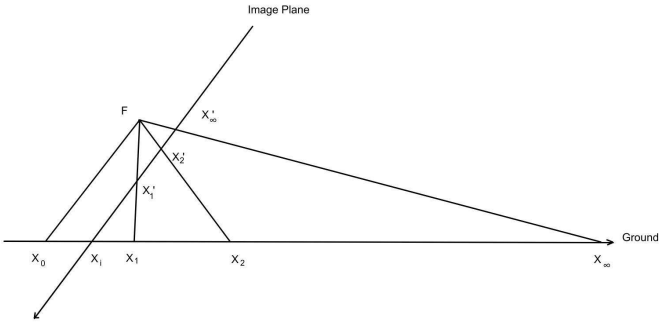

Figure 6. Illustration of vertical relationship

line going through the point of interest to the vertical line going through the vanishing point. Shown in Figure 5, This relationship can be approximately expressed by

$$d_{x_1"x_2"} = d_{x_1'x_2'} \tag{1}$$

where $d_{x_1"x_2"}$ and $d_{x_1'x_2'}$ are the distances of $x_1"x_2"$ and $x_1'x_2'$ in real world respectively.

Let's consider the relationship of the distance between the vanishing point and a point on the vertical line across the vanishing point in real world. As shown in Figure 6, x_∞ is infinitely far away from the image plane and on the ground. Map this infinity point to the image plane, we get x'_∞. For points on the ground, x_1 and x_2, the mapping from the ground to the image plane are x'_1 and x'_2. according to triangular relationships, we get

$$d_{x_1'x_2'} = d_{Fx_0}\frac{d_{x_1x_\infty}}{d_{x_0x_\infty}}\frac{d_{x_0x_i}}{d_{x_0x_1}} - d_{Fx_0}\frac{d_{x_2x_\infty}}{d_{x_0x_\infty}}\frac{d_{x_0x_i}}{d_{x_0x_2}} \tag{2}$$

Because $\frac{d_{x_1x_\infty}}{d_{x_0x_\infty}} = \frac{d_{x_2x_\infty}}{d_{x_0x_\infty}} = 1$, we can get

$$d_{x_1'x_2'} = \frac{C}{d_{x_0x_1}} - \frac{C}{d_{x_0x_2}} \tag{3}$$

where $C = d_{Fx_0}d_{x_0x_i}$. Because d_{Fx_0} and $d_{x_0x_i}$ are constants, C is also a constant. Thus, the distance between x_1 and x_2 can be expressed by

$$d_{x_1x_2} = \frac{C'}{d'_{x'_\infty x'_2}} - \frac{C'}{d'_{x'_\infty x'_1}} \tag{4}$$

where C' is some constant, and d' is a distance measured in the number of pixels.

After finding the vanishing point of a frame, if we know C', then we can get the real distance between two points in the image. We measure the distance of several line segments that are parallel to the lane on road and are visible in the video. Extend these line segments in a frame and they will intersect at the vanishing point we find. Now we can calculate the displacement distance of an object by

$$d = |\frac{C}{x_1\cos\theta_1} - \frac{C}{x_2\cos\theta_2}| \tag{5}$$

where d is the displacement distance of an object in real world, x_1 and x_2 are the distances between object positions in different frames with the vanishing point respectively, and θ_1 and θ_2 are the corresponding angles between the vertical line across the vanishing point and the line segment from the vanishing point to the object position. C is calculated by

$$C = \underset{C}{\operatorname{argmin}} \sum_{i=1}^{n} \left(|\frac{C}{x_{i1}\cos\theta_{i1}} - \frac{C}{x_{i2}\cos\theta_{i2}}| - d_i\right)^2 \tag{6}$$

where x_{i1} and x_{i2} are marked line segment in the frame and d_i is the corresponding distance in real world measured from Google Maps.

Then, we can compute the speed of a vehicle by

$$s = d\frac{f_r}{f_n} \tag{7}$$

where s is the speed of the vehicle, f_r is the frame rate and f_n is the difference of frame indexes between two frames. An example of speed calculated for a frame is shown in figure 3.

4. Experiments

We experiment our method for the 27 videos provided by track 1 of Nvidia AI City Challenge 2018, and achieve 6.6674 root mean square error (RMSE) compared to the ground truth. Figures 7-10 show the speeds estimated for the vehicles recognized in that frame for 4 different locations.

5. Discussions

Our method can achieve relatively good results in measuring the speed of vehicles whose trajectories are parallel

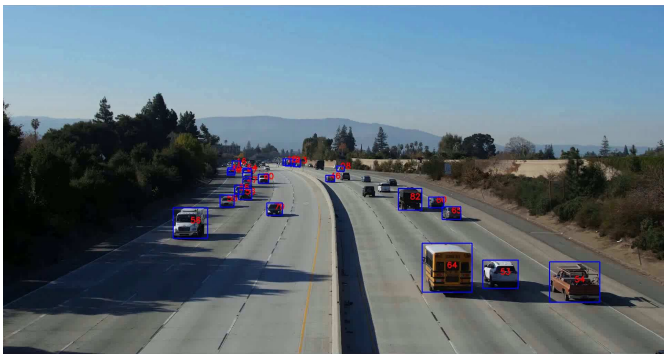

Figure 7. Speed estimation results for location 1

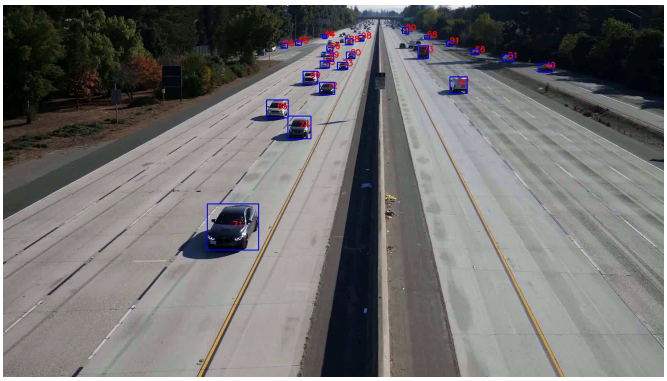

Figure 8. Speed estimation results for location 2

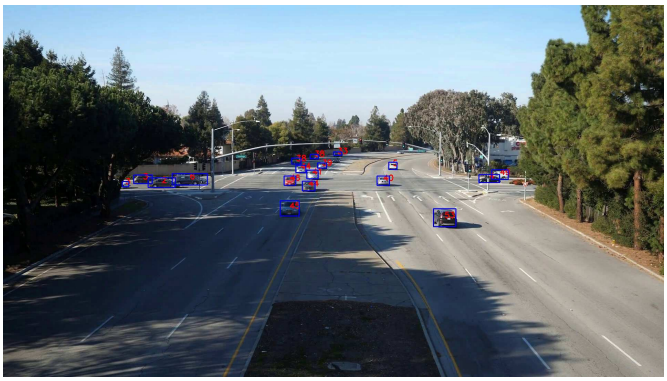

Figure 9. Speed estimation results for location 3

to the lane of road and vanish at the vanishing point preset. However, if vehicles have horizontal displacement that are orthogonal to the lane within two frames, then this method is not capable of measuring the horizontal speed.

We observe that the camera plane is approximately perpendicular to lanes on road. Thus, the width of the road in a frame linearly decreases and reaches 0 as the road vanishes at the vanishing point. Measuring the road width from Google Maps, we can calculate the horizontal distance in a frame by

$$d_h = d'_h \frac{d_s}{d'_s} \frac{d_p}{d'_p} \tag{8}$$

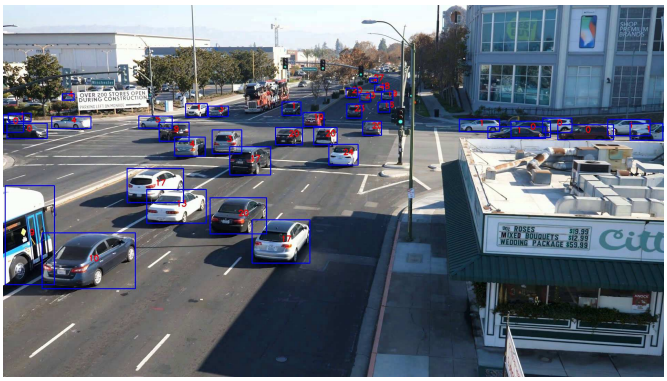

Figure 10. Speed estimation results for location 4

where d_h is the horizontal distance in real world we need to calculate, d_s is the number of pixels between the apexes of the horizontal line segment to be measured in a frame, d_p is the number of pixels from the vanishing point to the horizontal line segment to be measured, d'_h is the width of the road in real world, d'_s is the number of pixels between the apexes of the line segment which we mark as the width of the road, and d'_p is the number of pixels from the vanishing point to the line segment which we mark as the width of the road. However, after we use this method to measure the speed of vehicles in horizontal direction and add it to the total speed, the RMSE increases.

We did not use the UADETRAC videos [20]. On the one hand, we are not certain if the speed given for each vehicle is accurate in the UADETRAC videos after careful examination. On the other hand, some important information (the angle and the height of the camera for example) is not given from both the UADETRAC videos and the challenge videos. We cannot get accurate speed estimation without such information.

6. Conclusion

To tackle the traffic speed estimation problem from traffic camera videos, we use learning based method to detect and track vehicles in the given traffic camera videos, and use a geometric processing method to calculate the speed of vehicles. Our submission achieves a perfect detection rate of target vehicles and RMSE of 6.6674. Note we entered the challenge at a very late stage and many future work would be pursued especially in accurate speed estimation after detection and tracking with high performance.

Acknowledgements. This work is in part supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network.

References

[1] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. *arXiv preprint arXiv:1712.00726*, 2017.

[2] B. Cheng, Y. Wei, H. Shi, R. Feris, J. Xiong, and T. Huang. Revisiting rcnn: On awakening the classification power of faster rcnn. *arXiv preprint arXiv:1803.06799*, 2018.

[3] 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.

[4] 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] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

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

[7] W. Han, P. Khorrami, T. L. Paine, P. Ramachandran, M. Babaeizadeh, H. Shi, J. Li, S. Yan, and T. S. Huang. Seq-nms for video object detection. *arXiv preprint arXiv:1602.08465*, 2016.

[8] 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.

[9] Z. Jie, Y. Wei, X. Jin, J. Feng, and W. Liu. Deep self-taught learning for weakly supervised object localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1377–1385, 2017.

[10] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 2756–2759. IEEE, 2010.

[11] 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.

[12] 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.

[13] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[14] M. Naphade, D. C. Anastasiu, A. Sharma, V. Jagrlamudi, H. Jeon, K. Liu, M.-C. Chang, S. Lyu, and Z. Gao. The nvidia ai city challenge. *IEEE Smart-World, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, 2017.

[15] 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.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[17] Z. Shen, Z. Liu, J. Li, Y.-G. Jiang, Y. Chen, and X. Xue. Dsod: Learning deeply supervised object detectors from scratch. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 3, page 7, 2017.

[18] Z. Shen, H. Shi, R. Feris, L. Cao, S. Yan, D. Liu, X. Wang, X. Xue, and T. S. Huang. Learning object detectors from scratch with gated recurrent feature pyramids. *arXiv preprint arXiv:1712.00886*, 2017.

[19] H. Shi, Z. Liu, Y. Fan, X. Wang, and T. Huang. Effective object detection from traffic camera videos. 2017.

[20] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu. Ua-detrac: A new benchmark and protocol for multi-object detection and tracking. *arXiv preprint arXiv:1511.04136*, 2015.