Vehicle Speed Estimation using Evolutionary Optimization for Fish-eye Camera Calibration

Ping-Yang Chen (0886906), You-Jia Lai (309553005), and Yu-Ching Li (609001019)
National Yang Ming Chiao Tung University
Computer Vision
Group 13

1 Introduction

Traffic flow estimation is a key task for monitoring and managing traffic streams in an intelligent transportation system. A prerequisite for enabling this analysis is to accurately locate vehicles in video images so that car attributes can be extracted, compared, and counted.

To estimate car speed from a whole multi-lane intersection, a fish-eye camera will be more suitable compared to an IP camera due to its wider field of view. A fish-eye camera can significantly reduce the number of IP cameras and the requirement to synchronize cameras in the same intersection to maintain a vehicle with the same ID. However, a fish-eye camera always causes hemispherical distortion to the flat ground, thus image of vehicles at different positions (see the top or bottom of Fig. 1(a)) could be quite distorted. The distortion makes vehicle shapes be different between even subsequent two frames and leads to the failure of tracking. It is a basic image process problem which can be flattened by some type of de-warping operation. Then, vehicles can be detected and analyzed with regular shapes for next traffic parameter estimations such as **speed**, volume, density, waiting length, direction, and so on. However, the de-warping operation will increase not only the image size but also the computational cost.

Another challenge is camera calibration which is the key component to build this fish-eye-based car speed surveillance system. Obviously, we can't not just use the 2D bounding box coordinate to calculate the speed for each car. It's better to use 3D coordinate or bird view map to calculate the speed.However, with the installation from higher place, it is not safe and convenient to calibrate the camera with the chessboard, not just for Fish-eye cameras but other surveillance cameras.

We define main requirements for a good Vehicle speed estimation of a multi-lane intersection are as follows:

- A higher accuracy of object detection [1] as described in Sec 2.3 is achieved with the enriched context and semantic information in a feature pyramid network on UAVDT benchmark as shown in table 1 and our home Fish-Eye dataset which was provided by Hsinchu police station as shown in table 2 and 3.;
- An efficient and accurate two-stage tracking by detection method [2] was use to track the vehicles with recursive Kalman filtering and frame-by-frame data association which can solve the occlusion problem as described in Sec 2.4.;
- Introduce a safe and convenient method [3] to calibrate the camera as shown in Fig 3 without the chessboard method in Sec 2.5.
- Use the calculated homograph matrix to project 2D points to 3D points to get the real world latitudes and longitude, providing a robust result of vehicle speed in Sec 2.6.

(a)  (b) 

Figure 1: Above are the comparisons with different cases with the de-warping method. (a) A view from Fish-eye Camera. (b) Fish-eye camera with the de-warping scene.

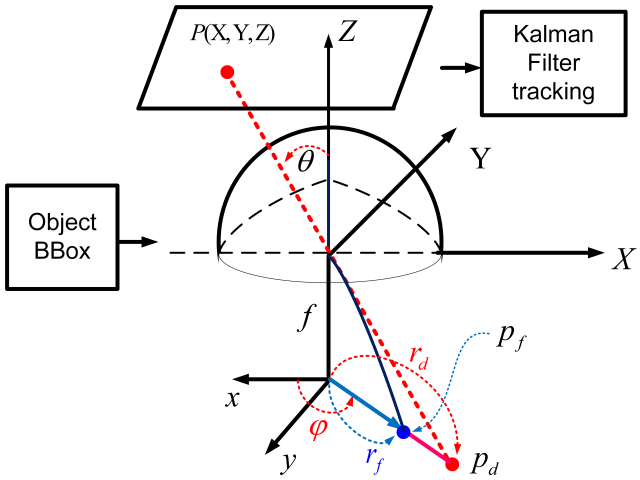

Figure 2: Fisheye camera model.

2 Methods

2.1 Overall Architecture

We first motivate the design of our proposed vehicle Speed estimation for the fish-eye camera in Sec 2.1 by addressing the distortion problem without de-warping. In Section 2.2, before introducing details of our **Vehicle Speed Estimation System**. The relation between distortion images and normal images should be discussed. In section 2.3, details of our vehicle detection model are described. To solve the occlusion problem and prevent the speed calculation from error, section 2.4 adopt the robust DeepSort method to track the vehicle ID. In section 2.5, perspective-n-Point (PnP) can be used to compute the camera homography matrix that contains all the information you need to project points from a 3D plane to the 2D image plane.Finally, Section 2.6 provides the speed calculated method. Details are provided in the following sessions.

2.2 Fisheye Camera De-Warping

As shown in Fig.2, let P be a 3D pint of coordinates $P = (X, Y, Z)^t$. The 2D coordinates of P on the distorted perspective image and the fish-eye image are $p_d = (x_d, y_d)^t$ and $p_f = (x_f, y_f)^t$, respectively. In addition, let f and θ denote the focal length and the angle between the light ray and the Z axis, respectively. In addition, r_d is the distance between p_d and the Z axis. From Fig.2, the perspective effect lets r_d and θ have the following relation:

$$r_d = f \times tan(\theta). \tag{1}$$

This paper assumes the fish-eye camera follows the equidistance formula which makes

$$r_f = f\theta, \tag{2}$$

where r_f is the distance between p_f and the Z axis; that is, the distorted height of the projected point on the image plane. Lastly, suppose (c_x,c_y) and R are the center and radius of circular region in the fisheye image. r_f can be calculated as follows:

$$r_f = \sqrt{(x_f - c_x)^2 + (y_f - c_y)^2}. \tag{3}$$

From Eq.(2), r_d can be calculated as follows:

$$r_d = R \times tan\left(\frac{r_f}{R}\right), \tag{4}$$

where $f = R$. Let φ be the angle between p_f and the x axis. Then, we have

$$\varphi = arctan\left(\frac{y_f - c_y}{x_f - c_x}\right). \tag{5}$$

In addition, x_d and y_d can be obtained as follows:

$$x_d = c_x + r_d \times cos\varphi \quad \text{and} \quad y_d = c_y + r_d \times sin\varphi. \tag{6}$$

Based on Eqs.(3)-(5), x_d and y_d can be rewritten as follows:

$$x_d = c_x + R \times tan\left(\frac{r_f}{f}\right) cos\left(arctan\left(\frac{y_f - c_y}{x_f - c_x}\right)\right) +, \tag{7}$$

and

$$y_d = c_y + R \times tan\left(\frac{r_f}{f}\right) sin\left(arctan\left(\frac{y_f - c_y}{x_f - c_x}\right)\right). \tag{8}$$

2.3 Detection model

We use YoloV4 [1] model which was trained using SGD with an initial learning rate of 0.001, 0.949 momentum, 0.0005 weight decay with a batch size of 64, and mini-batch size 2. During inference, detection results with confidence scores larger than 0.25 are used to generate bounding boxes. The Non-Maximum Suppression (NMS) threshold is set to be 0.7 to ensure the proposals have a large recall rate. The accuracy of object detection is measured by Average Precision (AP) and Mean Average Precision (mAP). All the input images are resized to 896 $\times$ 896. In addition, ImageNet was used to pre-train our detection model.

Since our objects in each classes are imbalanced, we add focal loss [4] during training.

$$FL(p_t) = (1 - p_t)^{log(p_t)} \tag{9}$$

focal loss will reshape the standard cross-entropy loss by down-weighting the loss assigned to well-classified examples, which can prevent the majority of easy negatives to overwhelm the training.

2.4 Tracking model

The remote monitoring system of smart vehicle flow needs to accurately track the starting position and turning position of moving objects. Representative tracking algorithms include the Kalman filter [5–8] and its variants, such as the extended/unscented Kalman [9–12] and particle filters. These can accurately track movement based on adaptive filtering by using a state-space model.

In object tracking, the Deep SORT [2] is a pragmatic approach to multiple objects tracking with a focus on simple, practical algorithms with a deep association metric. However, it is well-known that the deep association metric is useless to associate accurate localization for the vehicle's trajectory due to its CNN architecture. Moreover, its estimation model approximates each object's inter-frame displacements with a "linear constant velocity", which are solved optimally via a Kalman filter framework [14]. The state of each target is modelled as:

$$x = [u, v, r, h, \dot{u}, \dot{v}, \dot{r}, \dot{h}]^T \tag{10}$$

the eight dimensional state space contains the bounding box center position (u, v), aspect ratio r, height h, and their respective velocities in image coordinates. However, the object captured by a fish-eye camera is recorded in a hemispherical representation which makes it be highly distorted, especially for the ones located far from the camera center.

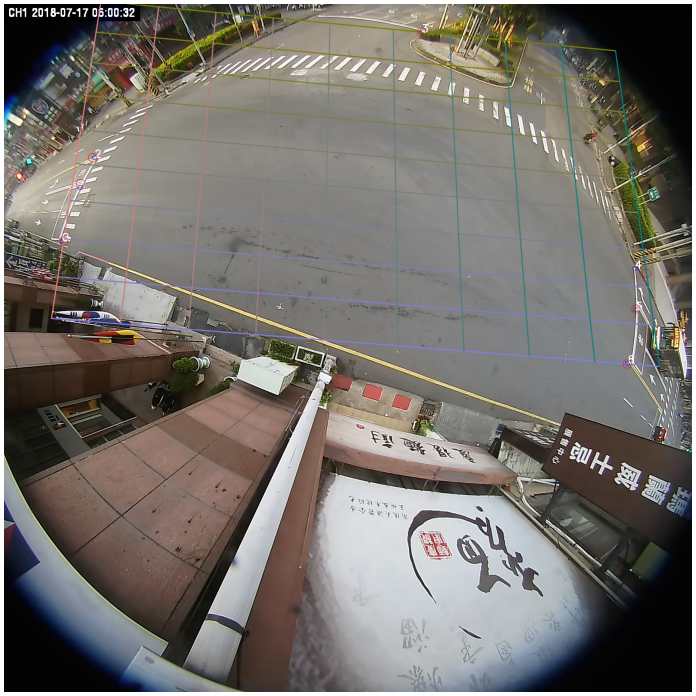

Figure 3: Fisheye camera calibration.

2.5 Evolutionary optimization for camera calibration

With the installation from higher place, it is not safe and convenient to calibrate the camera with the chessboard, not just for Fish-eye cameras but other surveillance cameras.

To tackle this problem, We use this method [3] to solve the calibration and unsafety problem. The repo implements semi-automatic camera calibration based on Perspective-n-Point (PnP). It can compute the camera homography matrix, containing all the information for projecting points from a 3D plane to the 2D image plane.

To relax camera matrix P constraints for camera parameters, they formulate an optimization problem to minimize reprojection.

A set of N_{ls} line segments on the ground plane, each defined by two endpoints, noted P_kQ_k manually selected, whose ground-truth 3D lengths are measured in the Google Maps. Using the calculated camera parameters, the 2D endpoints of the line segments' 2D endpoints noted p_k and q_k could be back-projected to 3D. Their Euclidean distances represent the estimated 3D lengths of the line segments. The absolute differences between estimations and ground truths are summed up to describe the reprojection error. Thus, the objective of our optimization problem is defined as follows:

$$\min_{P} \sum_{k=1}^{N_ls} \left| \|P_k - Q_k\| - \left\|\widehat{P_k} - \widehat{Q_k}\right\| \right| \tag{11}$$

$$p_k = \mathbf{P} \cdot \widehat{P_k}, q_k = \mathbf{P} \cdot \widehat{Q_k} \tag{12}$$

where $\widehat{P_k}$ and $\widehat{Q_k}$ note the estimated endpoints of the selected line segments that are backprojected to the 3D ground plane

They choose the RANSAC-based method for camera calibration and find homography matrix. The figure 3 is the calibration result with a colorful grid on the 3D plane.

2.6 Vehicle Speed Estimation

Use the homography matrix to project 2D points to 3D points. Here the 3D points represent latitude and longitude. Moreover, we can use the haversine formula to calculate the distance between two 3D

Table 1: Comparisons on the UA-DETRAC set with SoTA models.

Method	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
DPM [13]	25.70	34.42	30.29	17.62	24.78	30.91	25.55	31.77
ACF [14]	46.35	54.27	51.25	38.07	58.30	35.29	37.09	66.58
R-CNN [15]	48.95	59.31	54.06	39.47	59.73	39.32	39.06	67.52
CompACT [16]	53.23	64.84	58.70	43.16	63.23	46.37	44.21	71.16
GP-FRCNN [17]	77.96	92.74	82.39	67.22	83.23	77.75	70.17	86.56
EB [18]	67.96	89.65	73.12	53.64	72.42	73.93	53.40	83.73
SSDR [19]	59.07	77.84	64.41	45.98	62.79	60.88	48.55	74.32
FRCNN-Res [19]	61.65	82.90	66.89	48.14	61.97	65.88	59.13	59.17
DFCN [19]	65.82	86.83	72.96	50.47	69.90	69.41	54.11	80.79
FG-BR Net [20]	79.96	93.49	83.60	**70.78**	**87.36**	78.42	70.50	**89.89**
YoloV4* [1]	78.53	93.12	83.87	67.21	83.81	78.14	70.87	86.68

points. The haversine formula determines the great-circle distance between points on a sphere given their latitudes and longitude. After getting the distance, we can divide it by time and get the speed.

3 Experimental Results

3.1 Training dataset

There are two database that we used in this work. We first perform comprehensive studies on the challenging UA-DETRAC dataset [18] to demonstrate the outperformance of the proposed method. UA-DETRAC [18] contains 100 challenging video sequences corresponding to more than 140 000 frames of real-world traffic scenes. There are more than 1.2 million vehicles labeled with bounding boxes in this dataset. It records the videos at 25 frames per second (fps), with a JPEG image resolution of 960 × 540 pixels. The UA-DETRAC detection benchmark's evaluation metric is strict: it adopts a 0.7 IoU threshold for detecting cars. A website is available for performance evaluation of object detection. The results of YoloV4, a single model without any bells and whistles, has been made publicly available. For performance evaluations on UA-DETRAC dataset [18], the used training steps are 200500 with the step decay learning rate 0.001. The learning rate is further multiplied by a factor 0.01 at the 160000 steps and 180000 steps, respectively. Momentum and weight decay rate are set to be 0.9 and 0.0005, respectively.

We second perform comprehensive studies on our challenging non-public FishEye dataset which was provided by Hsinchu police station to demonstrate the outperformance of our proposed method. Our non-public FishEye dataset contains 35 challenging video sequences corresponding to more than 200000 frames of real-world traffic scenes. There are more than 3001 vehicles labeled with bounding boxes in this dataset. It records the videos at 60 frames per second (fps), with a JPEG image resolution of 1920 × 1920 pixels.

3.2 Auto labeling program for higher detection result

We use the YoloV4 model with the DeepSort method to generate more bounding boxes for training data instead of labeling the images from scratch.

3.3 Detection and tracking result

We can see comparison between w/ focal loss and w/o focal loss from table 2 and table 3. The overall result shows in table 4. With focal loss, the recall and MAP are better than the result without focal loss. But the precision and F1 is not better than the result without focal loss. We think the reason is the model with focal loss has more FP results. However, it's better to have a good recall in our task than a good precision cause the precision will be corrected from upon frame by tracking, and we don't want to lose any car in a frame.

Table 2: Comparisons on different categories for accuracy (MAP:50, TP, and FP) in our home Fish-Eye dataset.

Class ID	Class Name	MAP	TP	FP
0	Bus	88.17%	212	21
1	Moto	61.06%	415	320
2	Scooter	96.49%	1199	130
3	Van	1.26%	1	117
4	Car	87.87%	337	119
5	Sedan	90.46%	236	62
6	Truck	82.22%	99	9
7	Pedestrain	53.82%	65	58
8	Bicycle	0.00%	0	1
9	Cart	0.00%	0	0
10	SUV	85.71%	279	134
11	Semi-Trailer	0.00%	0	7
12	Taxi	90.26%	157	46
13	Dump_truck	12.31%	2	22

Table 3: Comparisons on different categories for higher accuracy (MAP:50, TP, and FP) in our home Fish-Eye dataset with focal loss.

id	class	MAP	TP	FP
0	Bus	92.33%	216	46
1	Moto	66.93%	448	389
2	Scooter	96.86%	1197	226
3	Van	1.23%	1	161
4	Car	90.05%	358	162
5	Sedan	92.40%	267	242
6	Truck	88.64%	135	41
7	Pedestrain	54.98%	86	238
8	Bicycle	0.00%	0	9
9	Cart	0.00%	0	0
10	SUV	87.48%	271	143
11	Semi-Trailer	0.00%	0	9
12	Taxi	92.19%	162	180
13	Dump_truck	3.99%	4	49

Table 4: Comparison on w/ focal loss and w/o focal loss

	precision	recall	F1	mAP
w/o focal loss	0.74	0.84	0.79	53.55%
w/ focal loss	0.62	0.88	0.73	54.79%

Figure 4: Vehicle speed. The object farther from the camera likes the leftmost scooter and its speed is 6.79. The object closer to the camera likes the blue car in the image center and its speed is 21.47. The closer object predicts the more reasonable result than the farther object.

3.4 Vehicle speed result

The visualization result shows in figure 4. The unit of speed is kilometer per hour. In the experiment, we find that the objects closer to the camera will predict better results than objects farther from the camera.

4 Acknowledgement

Thank you for supporting and enlightening all in this computer vision course. Walon has been a great mentor, attractive teacher, and guides on the path to knowledge.

5 Discussion and Future work

We are chasing with time. We implement a few parts of our work as shown in the above report and submit. However, it is a massive task for us to implement this work, which consists of object detection, multiple object tracking, camera calibration, and RANSAC, all related to this CV course. We believe reviewers are comfortable with our final effort, and we should be all set now. Although we done the proof of de-warping in Sec 2.2 we hope to compare the performance of vehicle speed between the de-warping and non-de-warping methods(This work) in our future work.

References

[1] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," in *arXiv*, 2020.

[2] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649.

[3] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang, "Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 108–115.

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[5] Bertil Ekstrand, "Some aspects on filter design for target tracking," *Journal of Control Science and Engineering*, vol. 2012, 05 2012.

[6] K. Saho and M. Masugi, "Automatic parameter setting method for an accurate kalman filter tracker using an analytical steady-state performance index," *IEEE Access*, vol. 3, pp. 1919–1930, 2015.

[7] Iyad Hashlamon and Kemalettin Erbatur, "An improved real-time adaptive kalman -lter with recursive noise covariance updating rules," *Turkish Journal of Electrical Engineering and Computer Sciences*, 12 2013.

[8] G. F. Basso, T. Guilherme Silva De Amorim, A. V. Brito, and T. P. Nascimento, "Kalman filter with dynamical setting of optimal process noise covariance," *IEEE Access*, vol. 5, pp. 8385–8393, 2017.

[9] Yu Fan, Fang Lu, Wuxuan Zhu, Guangzhou Bai, and Liang Yan, "A hybrid model algorithm for hypersonic glide vehicle maneuver tracking based on the aerodynamic model," *Applied Sciences*, vol. 7, no. 2, pp. 159, Feb 2017.

[10] Wenling Li, Shihao Sun, Yingmin Jia, and Junping Du, "Robust unscented kalman filter with adaptation of process and measurement noise covariances," *Digital Signal Processing*, vol. 48, pp. 93 – 103, 2016.

[11] G. Vivone, P. Braca, and J. Horstmann, "Knowledge-based multitarget ship tracking for hf surface wave radar systems," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 7, pp. 3931–3949, 2015.

[12] G. Du and P. Zhang, "A markerless human–robot interface using particle filter and kalman filter for dual robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2257–2264, 2015.

[13] P. Dollár *et al*, "Fast feature pyramids for object detection," *IEEE PAMI*, 2014.

[14] C. Zhaowei, S. Mohammad, and V. Nuno, "Learning complexity-aware cascades for deep pedestrian detection," in *ICCV*, 2015.

[15] G. Ross *et al*, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *CVPR*, 2014.

[16] S. Amin and F. Galasso, "Geometric proposals for faster r-cnn," in *AVSS*, 2017.

[17] L. Wang *et al*, "Evolving boxes for fast vehicle detection," in *ICME*, 2017.

[18] L. Wen *et al*, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Comput Vis Image Underst*, 2020.

[19] J.R.R. Uijlings *et al*, "Selective search for object recognition," *Int. J. Comput. Vis.*, 2013.

[20] Z. Fu, *et al*, "Foreground gating and background refining network for surveillance object detection," *IEEE Transactions on Image Processing*, vol. 28, no. 12, pp. 6077–6090, Dec. 2019.