Disentangling and Vectorization: A 3D Visual Perception Approach for Autonomous Driving Based on Surround-View Fisheye Cameras

Zizhang Wu[1]   Wenkai Zhang[1]   Jizheng Wang[1]   Man Wang[1]
Yuanzhu Gan[1]   Xinchao Gou[2]   Muqing Fang[1]   Jing Song[1]

Abstract— The 3D visual perception for vehicles with the surround-view fisheye camera system is a critical and challenging task for low-cost urban autonomous driving. While existing monocular 3D object detection methods perform not well enough on the fisheye images for mass production, partly due to the lack of 3D datasets of such images. In this paper, we manage to overcome and avoid the difficulty of acquiring the large scale of accurate 3D labeled truth data, by breaking down the 3D object detection task into some sub-tasks, such as vehicle's contact point detection, type classification, re-identification and unit assembling, etc. Particularly, we propose the concept of Multidimensional Vector to include the utilizable information generated in different dimensions and stages, instead of the descriptive approach for the bird's eye view (BEV) or a cube of eight points. The experiments of real fisheye images demonstrate that our solution achieves state-of-the-art accuracy while being real-time in practice.

I. INTRODUCTION

In recent years, autonomous driving has attracted more and more attention from both industry and research communities [1]. The 3D perception of the environment is one of the most challenging tasks [2]. Recent advances [3] demonstrate the potential to replace the LIDAR with cheap onboard cameras, which are readily available on most modern vehicles. Particularly, the surround-view fisheye camera system is very popular in mass production, on account of a larger field-of-view (FoV) [4] than the pin-hole cameras.

The surround-view fisheye camera system can provide a 360-degree perception, which makes up for other pinhole cameras' lack of close-range perception around the vehicle, especially in the scene of a traffic jam. So more researches focus on obtaining position and pose information from the above system, which are mostly in the Bird's Eye View (BEV) manner [5]. To obtain a BEV image, previous works either use the Inverse Perspective Mapping (IPM) method [6] to stitch the raw image data from the surrounding cameras or leverage 3D object detection methods [3], [7], [8] to obtain the 3D bounding boxes of the target-vehicles, which are then projected on the BEV image. Despite its simplicity, the IPM method can not directly provide valuable information such as the positions and the heading angles of the target-vehicles [9]. On the contrary, 3D object detection methods [10], [11] can provide more comprehensive information, but a fisheye image dataset with labeled 3D truth data is hardly available [12]. Annotating such datasets is costly and heavily relies on the fisheye cameras used to collect data, which makes

[1]Zongmu Technology
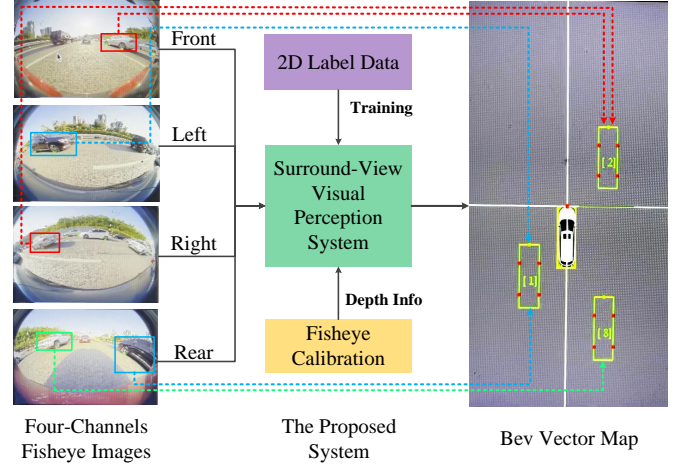[2]Karlsruhe Institute of Technology

Fig. 1: The overview of the proposed method. Best view in color and zoom in.

it unsuitable for mass production. Meanwhile, 3D object detection requires more computing resources compared to 2D [13]. Existing approaches usually utilize the visual features to regress the heading angles of the target-vehicles [14], [15] and achieve great success in many public datasets such as KITTI [16]. However, it remains a challenging task to obtain robust results covering a wide variety of use cases in industrial applications [17].

In autonomous driving and mobile robotics applications, we can generally assume that the most important dynamic objects are on a ground plane, and the camera is mounted at a certain height above the ground [18]. In particular, the assumptions hold well in quite close distance, such as within the perception range of the surround-view fisheye camera system [9]. So we can obtain the 3D positions of certain points on the ground through inverse projection, and these points can serve as useful hints under the condition of being stable and detectable in the image.

In the past few years, the two-dimensional keypoint detection and object detection techniques are gradually mature [13], [19]. Inspired by the above two sides, we define some keypoints of the vehicle in the image, which are also the contact points on the ground [18]. As mentioned above, we can acquire the 3D positions of the vehicle's contact points, such as the contact points of wheels and bumpers. But we can't compute the three-dimensional positions only based on the above information. For example, in some
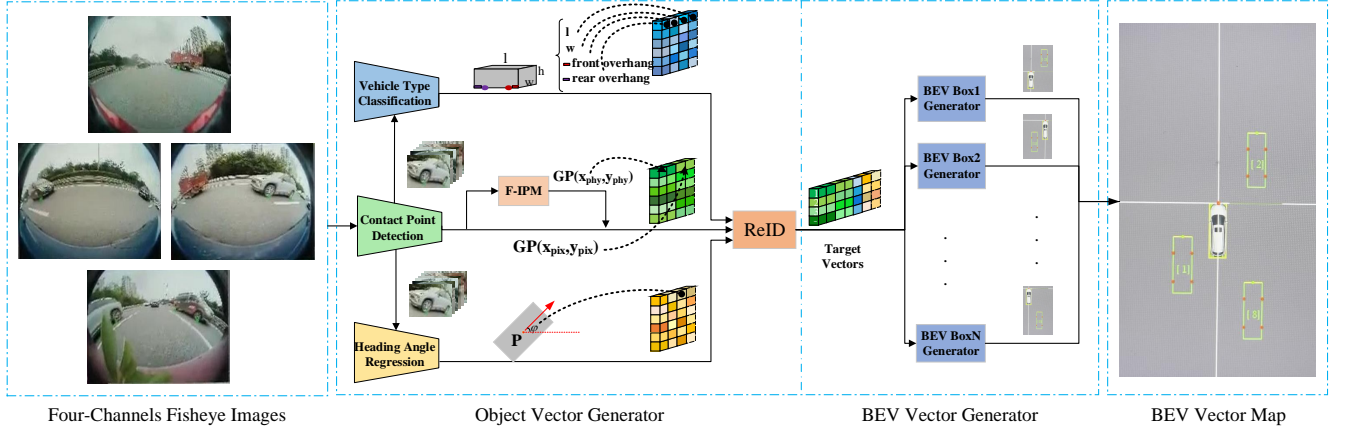
Fig. 2: The overall structure of our proposed approach. The inputs are four-channels fisheye images which construct a surround-view environment for ego-vehicles. The final output is a vector map containing the shape of the object under the bird's-eye view. Best view in color and zoom in.

situations, we also need the heading angle and vehicle type for additional information, which is bounded with its types such as the length, width and height. So we disentangle the 3D object detection task into some sub-tasks, such as vehicle's contact point detection, type recognition, and heading angle regression. Moreover, to maximize the advantages of every algorithm component, we establish a robust mechanism to achieve stable acquisition of 3D information for different situations, combined with geometric constraints. Furthermore, we propose the Multidimensional Vector to improve the prediction effect, which describes the target-vehicle by its center point, heading angle, type, etc. So in this paper, we propose a system that takes raw images from four surrounding fisheye cameras as input and outputs the 3D position of the target-vehicle with only the 2D labeled truth data (including some heading angle truth data) and calibration required, as shown in Fig. 1.

The contributions are summarized as following three-fold: 1. We provide a solution for 3D visual perception of the surrounding environment with a fisheye camera system in autonomous driving and avoid the difficulty of acquiring a large scale of accurate 3D labeled truth data. 2. We improve the prediction effect by the proposed Multidimensional Vector, which includes the utilizable information generated in different dimensions and stages, for more robust re-identification and tracking, etc. 3. Our method demands lower computing resources and achieves state-of-the-art accuracy in real-time scenes, thus has great advantages in mass production.

II. RELATED WORKS

A. Surround-view fisheye cameras and 3D visual perception

Extensive visual perception tasks usually employ surround-view fisheye cameras to capture the surrounding environment due to their large FoV [4] and sufficiently good performance [20], [21]. In particular, ego-vehicles can achieve 360-degree perception using only four fisheye cameras, which is conducive to mass production [22]. [23]

proposes a system to classify track vehicles and pedestrians around the ego-vehicle using only four fisheye cameras. [4] presents a solution to detect and classify moving objects around the vehicle in real-time, which merges four views captured by fisheye cameras into a single frame. Moreover, surround-view fisheye cameras are widely used in various perception tasks, such as semantic segmentation [24], depth estimation [25], [26], object detection [27], [28] and Re-Identification (ReID) [29].

Despite the advantages mentioned above, fisheye cameras have a strong radial distortion and exhibited more complex projection geometry [30], which leads to appearance distortion [31]. Thus, it is hard to generalize the models trained on the fisheye dataset. Besides, the distortion brings additional challenges to annotate fisheye datasets [12], [32], and there are few public 3D visual perception systems for vehicles based on the surround-view fisheye cameras [17]. Moreover, OmniDet [33] presents a multi-task visual perception network to capture the surrounding environment. Their object detection task requires a large annotated dataset of fisheye images which increases the additional cost of the integral task. The dense pixel-wise depth estimation would result in prohibitive computational cost and limited real-time performance.

B. 3D object detection

The visual perception system is a crucial part of autonomous driving, in which 3D object detection is an important component to estimate the pose and location of the surrounding vehicles [3], [34]. Recent developments [11], [8], [10] in 3D object detection mainly utilize various information such as context, geometry, depth map, and so on. They usually regress 3D information of target-vehicles, including the bird's eye view or a cube of eight points, through neural networks [35], [36], [37].

However, previous representations provide only limited information for the visual perception task in autonomous

driving. In this paper, we propose the Multidimensional Vector that includes more usable information, instead of the descriptions of the bird's eye view or a cube of eight points.

III. METHODS

In this section, we introduce the proposed visual perception system in detail, as shown in Fig. 2. First of all, we feed four fisheye images into the Contact Point Detection module, Vehicle Type Recognition module, and Heading Angle Regression module to obtain the required attributes for Multidimensional Vector, including vehicle type, dimensions (l, w, h), front overhang (fo), rear overhang (ro), heading angle (φ) and contact points' coordinates, as shown in Fig. 3. Then these attributes are merged and unified by the ReID module to generate a unique identification number for each target-vehicle. Subsequently, we integrate the intermediate results to generate vectors for the BEV description of target-vehicles and illustrate the final BEV Vector Map. Table I shows the notations used in this paper.

TABLE I: Notations used in this paper.

Notations	Descriptions
A, B, C, D	BEV_box (Left-Front, Left-Rear, Right-Front, Right-Rear)
FW	Front wheel ground contact point
RW	Rear wheel ground contact point
FB	Front bumper contact point
RB	Rear bumper contact point
P	Center point of the target vehicle
O	Origin point of the ego-vehicle coordinate system
GP_{phy}	Physical coordinates of the ground contact point
GP_{pix}	Pixel coordinates of the ground contact poin
l, w, h	length,width and height of the target vehicle
fo	Front overhang of the target vehicle
ro	Rear overhang of the target vehicle
obj_{id}	Id of the target vehicle
θ^*	Azimuth angle of the target-vehicle
φ^*	Heading angle of the target-vehicle
γ^*	Angle of $line(RB, RW)$ in standard position

* All the angles are positive angles along the positive x-axis of the ego-vehicle coordinate system.

A. Object Vector Generator

1) Contact Point Detection: To overcome and avoid the difficulty of acquiring large-scale accurate 3D labeled truth
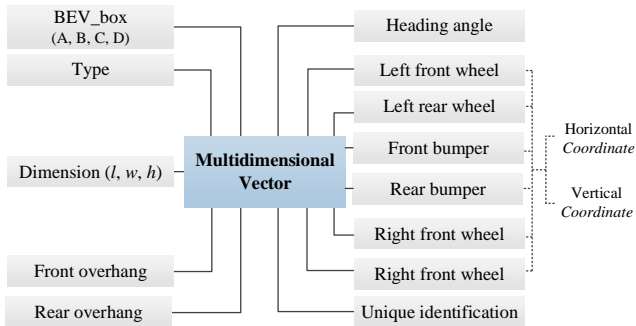
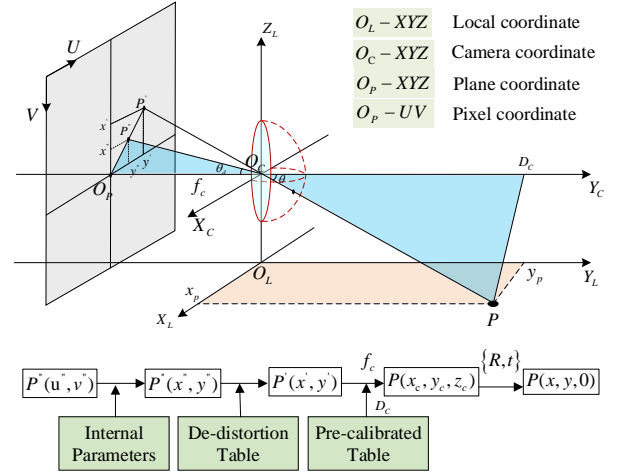Fig. 3: The specific composition of the Multidimensional Vector.

Fig. 4: The diagram of translating the pixel coordinates of the contact points in fisheye image to the physical coordinates. Best view in color and zoom in.

data, we perform 2D contact point detection on fisheye images. For detecting and locating the surrounding vehicles, we need to detect vehicles, front wheels, and rear wheels to compute the contact points of target-vehicles. Considering the trade-off between real-time performance and accuracy in autonomous driving scenes, we employ the CenterNet [19] as the base framework. Specially, we input four fisheye images from different channels into the network, which outputs the predicted bounding boxes of the vehicles, front and rear wheels.

Then we acquire the contact points' pixel coordinates in the image through post-processing. Here we define (ground) contact point as the midpoint of the bottom edge of the bounding box of a vehicle or a wheel. These points are on the ground and their physical coordinates are $P(x, y, 0)$. With the pixel coordinates of the contact points, we can calculate their physical coordinates in the real-world coordinate system through the Fisheye IPM algorithm [6], as shown in Fig. 4. The pixel coordinates of contact points in fisheye images are $P''(u'', v'')$, which are transformed into the distorted plane with coordinates $P''(x'', y'')$ through the internal parameter matrix. By referring to the de-distortion table, they can be converted into the de-distorted plane with coordinates $P'(x', y')$. After introducing the re-calibrated internal parameters and pre-calibrated tables, we get coordinates $P(x, y, z)$ in the camera coordinate system and target depth values D_c. Through the R, t matrix of the external parameters, we obtain coordinates $P(x, y, 0)$ of contact points in the local coordinate system (ego-vehicle coordinate system). In conclusion, we prepare some useful information of detected vehicles in this step, including 2D bounding boxes, pixel coordinates (GP_{pix}) and physical coordinates (GP_{phy}) of the contact points (FB, RB, FW, RW).

2) Vehicle Type Classification: We infer more information on target-vehicles by our vehicle type classification module, which contains a simple classification network [38]. We
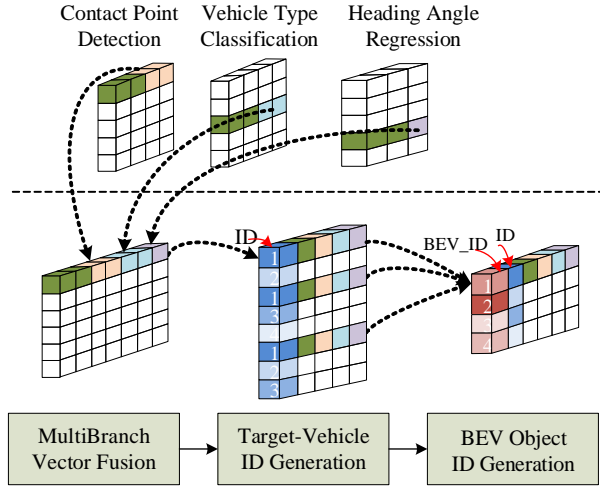
Fig. 5: ReID module is divided into three stages. The first stage fuses the vectors of three branches, the scond stage generates an ID for each object of each channel, and the third stage merges the vectors which describing the same object into one vector. Best view in color and zoom in.

crop vehicle images according to the detected 2D bounding boxes and input them into the network, which outputs the types of vehicles. Particularly, we divide vehicles into 8 types, including car, SUV, MPV, etc. Therefore, we obtain additional attributes for Multidimensional Vector, such as the dimensions (l, w, h), front overhang (fo), and rear overhang (ro), because they are bonded with the type of vehicles.

3) Heading Angle Regression: The Heading Angle Regression module is implemented by the MultiBin [14], which is a method to estimate the pose of objects by the cropped images. We take the output of this module as the estimation of the heading angle in the third case mentioned in III-B.

4) ReID: In our surround-view system, a target-vehicle can be observed from different cameras and described by one or more vectors. To facilitate the subsequent tasks, it is necessary to describe the target-vehicle using a vector with a unique identification number. So, we merge the original vectors from three branches into one vector with the unique identification number, as shown in Fig. 6. This process is target re-identification (ReID) and consists of three different stages.

The first stage is the multi-branch vector fusion. The bounding boxes (X, Y, W, L) information of the target-vehicles is included in the vector of each branch. If two vectors in different branches have the same bounding box, we append one to the other and remove duplicate elements. In this instance, we append the vectors from the vehicle type classification and heading angle regression branches to the contact point detection vector branch.

The second stage is the target-vehicle id generation. Since the same target may be observed in different channels and described with different vectors, the goal of this stage is to assign the same identification number to these vectors. This stage is mainly achieved by some hand-craft rules.
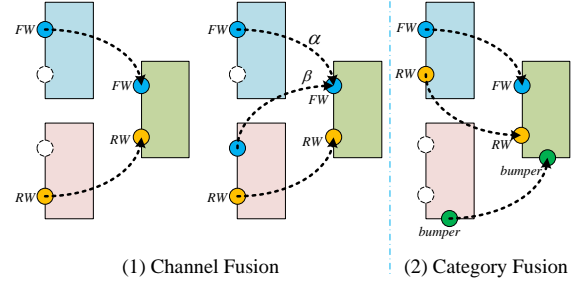

Fig. 6: Channel fusion and category fusion. (1) Wheels from different channels are fused into one vehicle. α and β are 0.5, which indicate weights assigned to the front wheels of two vehicles. (2) Wheels and bumper from different categories are fused into one vehicle. Best view in color and zoom in.

For example, we assign the same identification number to vectors, whose physical coordinates of ground contact points are less than 50 cm away.

The third stage is the BEV target id generation, which aims to merge all the vectors that describe the same target-vehicle and assign them a unique identity. Two situations are considered in this stage. One is the fusion between channels, which is performed in a complementary or weighted way, as shown in Fig. 6(1). The other is the fusion between categories, which is carried out in a complementary way, as shown in Fig. 6(2). Finally, we get the unique vector representation that contains all the information for the BEV Vector Map.

B. BEV Vector Generator

The output of the previous modules is integrated to calculate the BEV Vectors, which are used to construct the BEV Vector Map.

Firstly, we estimate the position of the center point P and the heading angle φ of the target-vehicle. The azimuth angle θ and the heading angle φ (as shown in Table I) are used to determine whether the visible side is left or right side of the target-vehicle via equation (1). The azimuth angle can be calculated by the position of the wheels or the bumpers.

$$\begin{cases} sin(\varphi - \theta) > 0, & left\ side \\ sin(\varphi - \theta) < 0, & right\ side \end{cases} \quad (1)$$

Then, three different cases are considered to estimate the center point P and the heading angle φ of the target-vehicle. In the first case, two wheels on one side of the target-vehicle are visible. The heading angle φ of the target-vehicle is determined by the slope of the $line(RW, FW)$, as shown in Fig. 7(a).

$$\varphi = arctan(\frac{FW_y - RW_y}{FW_x - RW_x}) \quad (2)$$

The positions of the left-front corner A and the left-rear corner B of the target-vehicle as well as their midpoint M are determined with the help of its front overhang fo and
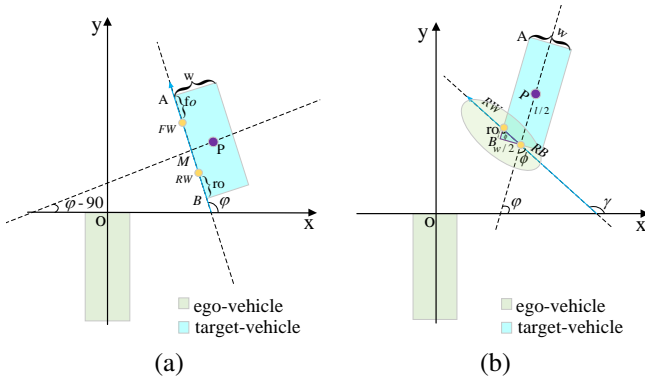
Fig. 7: Two cases of calculating the heading angles of the target-vehicles. (a) two wheels on one side are visible. (b) only one wheel and one bumper are visible. Best view in color and zoom in.

rear overhang ro, which are bounded with its type.

$$\begin{pmatrix} A_x \\ A_y \end{pmatrix} = \begin{pmatrix} FW_x + fo * cos(\varphi) \\ FW_y + fo * sin(\varphi) \end{pmatrix} \quad (3)$$

$$\begin{pmatrix} B_x \\ B_y \end{pmatrix} = \begin{pmatrix} RW_x + ro * cos(\varphi) \\ RW_y + ro * sin(\varphi) \end{pmatrix} \quad (4)$$

$$\begin{pmatrix} M_x \\ M_y \end{pmatrix} = \begin{pmatrix} \frac{A_x+B_x}{2} \\ \frac{A_y+B_y}{2} \end{pmatrix} \quad (5)$$

We acquire the position of center point P by $line(M, P)$ and half of the width w of the target-vehicle.

$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} M_x + \frac{w}{2} * cos(\varphi - 90) \\ M_y + \frac{w}{2} * sin(\varphi - 90) \end{pmatrix} \quad (6)$$

In the second case, one wheel and one bumper can be observed as shown in Fig. 7(b). Similarly, the slope of the $line(RB, RW)$ determines the angle γ.

$$\gamma = arctan(\frac{RW_y - RB_y}{RW_x - RB_x}) \quad (7)$$

The rear overhang ro and half of the width w of the target-vehicle w indicate the angle ϕ.

$$\phi = arctan(\frac{w/2}{ro}) \quad (8)$$

We calculate the heading angle φ of the target-vehicle based on the angle γ and angle ϕ.

$$\varphi = \gamma - \phi \quad (9)$$

The position of the center point P can be determined by the position of the rear bumper RB and half of the length l of the target-vehicle.

$$\begin{pmatrix} P_x \\ P_y \end{pmatrix} = \begin{pmatrix} RB_x + l/2 * cos(\varphi) \\ RB_y + l/2 * sin(\varphi) \end{pmatrix} \quad (10)$$

In the third case, only one bumper is available. So we take the output of the Heading Angle Regression module as the final heading angle of the target-vehicle.

After acquiring the heading angle φ and the center point P of the target-vehicle, we can calculate the positions of its four corners by the width w and length l and illustrate it by a rectangle on the BEV Vector Map.

TABLE II: Comparison results on synthetic panorama dataset

Models	2D-AP	3D-mAP	AOS	IoU	Dist. Err.
[39]	0.447	0.203	0.157	0.265	1.143
[31]	0.472	0.301	0.419	0.36	0.883
Ours	**0.573**	**0.362**	**0.856**	**0.647**	**0.535**

IV. EXPERIMENTS AND RESULTS

In this section, we evaluate the proposed visual perception method on accuracy, robustness and real-time performance.

A. Experimental settings

The 3D detection methods usually require 3D labeled data from the public 3D datasets for training, such as the KITTI dataset [16]. Since we decompose the 3D object detection task into sub-tasks, our method doesn't require 3D labeled data. Currently, there is no public dataset suitable for our setting, so we collect and label the training data manually. The training dataset contains about 140,000 images collected by a surround-view camera system with four fisheye cameras. Our system is deployed on the Qualcomm 820A platform with an Adreno 530 GPU and a Hexagon 680 DSP. It can deliver about 1.2 TOPs (trillion operations per second). In comparison, the NVIDIA Xavier is for 30 TOPs, and Tesla's V3 "Full Self-Driving" Computer is claimed to deliver 144 TOPs. There is no doubt that our system can operate on a platform with rather low computing ability.

B. Results of the synthetic panorama dataset

Following [31], we evaluate our results on the synthetic panorama dataset from [39]. [31] trained their 3D object detector on the KITTI 3D Object training set which is different from ours. The 2D and 3D metrics include 2D-AP (defined using IoU> 0.5 between 2D bounding boxes following [31]), 3D-mAP, average orientation similarity (AOS), mean 3D volumetric IoU, and mean Euclidean distance error. Table II presents these results. Our method has achieved the best performance, especially on AOS and IoU metrics, which proves that our method is effective in decomposing the total task into sub-tasks, even without 3D labeled data. Fig. 8 shows the visualization of the perception output.

C. System positioning error analysis

We conduct three experiments to evaluate the object position accuracy. In these experiments, the numbers of objects perceived by our system are 9382, 11777, and 7911, respectively. In the evaluation of positioning error along the x-direction (Fig. 7), it is eligible when errors are smaller than 25 cm. As shown in Fig. 9(a), the qualification rates of these three tests are 99.82%, 99.50%, and 99.54%.

As for y-direction (Fig. 7), since the strong relation between the positioning error and the distance from the target vehicle to the ego-vehicle, we divide the statistical errors into multiple intervals. In the experiments, the distance range is
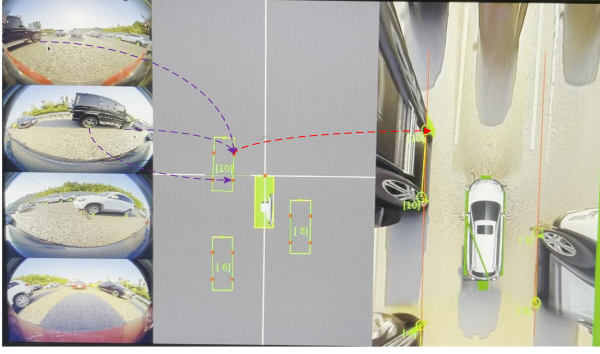
Fig. 8: Visualization results of our perception system. Best view in color and zoom in.
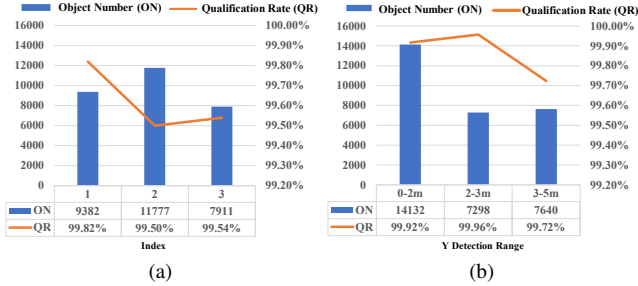

(a)

(b)

Fig. 9: Evaluation of the system positioning qualification rate. Best view in color and zoom in.
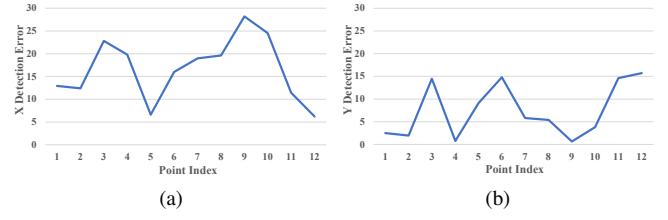

(a)

(b)

Fig. 10: Evaluation of the positional accuracy on the ground with 5% slope gradient. Best view in color and zoom in.
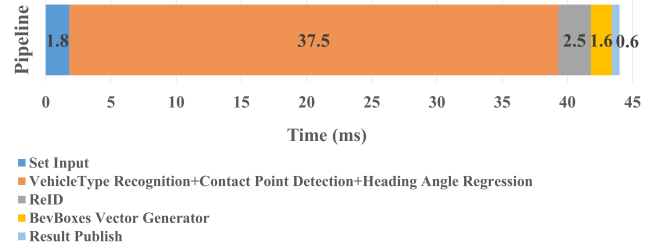

Fig. 11: Time consumption tests on the hardware platform. The total time of our system's test is less than 45 ms, which can meet real-time requirements. Best view in color and zoom in.

divided into 3 intervals of 0-2 meters, 2-3 meters, and 3-5 meters, while the requirements are no larger than 20 cm, 40 cm, and 50 cm. As shown in Fig. 9(b), the qualification rates in the y-direction are 99.92%, 99.96%, and 99.72%. These experiments effectively prove that our system has a high positioning accuracy.

Since there is no ideal horizontal ground plane in practical application, we design an experiment to evaluate the positioning accuracy of our system on the ground with a 5% slope gradient. We randomly select 12 objects distributing from -2.5 to 2.5 meters along the x-direction and calculate their position errors. As shown in Fig. 10(a), the result demonstrates that all the errors are less than 30 cm. Meanwhile, 12 points are randomly selected in the range of 1.5 to 3.5 meters along y-direction to analyze the position errors. As shown in Fig. 10(b), the position errors of objects are less than 20 cm. It can be concluded that our system can work on the ground with a slight slope gradient, which proves the robustness of the proposed system in practical scenarios.

D. System delay analysis

We perform time consumption tests on the hardware platform, and Fig. 11 shows the latency time of each module.

The "Set Input" and "Result Publish" represent the input and output of the proposed system. "Vehicle Type Recognition", "Contact Point Detection", and "Heading Angle Regression" are introduced in the Method section. The total time of our system's test is less than 45 ms. The "Vehicle Type Recognition + Contact Point Detection + Heading

Angle Regression" part takes the most time (37.5 ms) since it's the network inference module. The test is carried out on a serial setting and the total time is the sum of the running time for four input pictures in series.

In real engineering applications, the parallel structure achieves a total time consumption of less than 40 ms, which can meet real-time requirements.

V. CONCLUSIONS

In this paper, we present an accurate and low-latency 3D vehicle perception solution with the surround-view fisheye camera system for autonomous driving. We are the first to propose a method for overcoming and avoiding the acquisition difficulty of the large-scale accurate 3D labeled truth data, by breaking down the 3D object detection task into some sub-tasks and obtain sufficient precision for mass production. The above solution can achieve a sub-120 ms latency from view to multidimensional vector output, with average errors less than 0.25 meters at a distance of 5 meters. Furthermore, the associated fisheye dataset will be public to facilitate the progress of relevant research in this field. In future studies, we will continue to optimize the performance of the 3D perception system.

REFERENCES

[1] J. Monica and M. Campbell, "Vision only 3-d shape estimation for autonomous driving," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020.* IEEE, 2020, pp. 1676–1683.

[2] K. Strobel, S. Zhu, R. Chang, and S. Koppula, "Accurate, low-latency visual perception for autonomous racing: Challenges,mechanisms, and practical solutions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2020.* IEEE, 2020, pp. 1969–1975.

[3] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," *arXiv preprint arXiv:1608.07916*, 2016.

[4] I. Baek, A. Davies, G. Yan, and R. R. Rajkumar, "Real-time detection, tracking, and classification of moving and stationary objects using multiple fisheye images," in *2018 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2018, pp. 447–452.

[5] M. H. Ng, K. Radia, J. Chen, D. Wang, I. Gog, and J. E. Gonzalez, "Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud," *arXiv preprint arXiv:2006.11436*, 2020.

[6] H. A. Mallot, H. H. Bülthoff, J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological Cybernetics*, vol. 64, no. 3, pp. 177–185, 1991.

[7] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6851–6860.

[8] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8851–8858.

[9] Y. Kim and D. Kum, "Deep learning based vehicle position and orientation estimation via inverse perspective mapping image," in *2019 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2019, pp. 317–323.

[10] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7345–7353.

[11] Z. Liu, Z. Wu, and R. Tóth, "Smoke: single-stage monocular 3d object detection via keypoint estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 996–997.

[12] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O'Dea, M. Uricár, S. Milz, M. Simon, K. Amende *et al.*, "Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9308–9318.

[13] 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*, 2016, pp. 779–788.

[14] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[15] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3d object detection leveraging accurate proposals and shape reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 867–11 876.

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

[17] P. Maddu, W. Doherty, G. Sistu, I. Leang, M. Uricar, S. Chennupati, H. Rashed, J. Horgan, C. Hughes, and S. Yogamani, "Fisheyemultinet: Real-time multi-task learning architecture for surround-view automated parking system," *arXiv preprint arXiv:1912.11066*, 2019.

[18] Y. Liu, Y. Yuan, and M. Liu, "Ground-aware monocular 3d object detection for autonomous driving," *IEEE Robotics and Automation Letters*, 2021.

[19] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[20] M. Toromanoff, E. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde, "End to end vehicle lateral control using a single fisheye camera," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 2018, pp. 3613–3619.

[21] M. Drulea, I. Szakats, A. Vatavu, and S. Nedevschi, "Omnidirectional stereo vision using fisheye lenses," in *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP).* IEEE, 2014, pp. 251–258.

[22] V. R. Kumar, S. A. Hiremath, M. Bach, S. Milz, C. Witt, C. Pinard, S. Yogamani, and P. Mäder, "Fisheyedistancenet: Self-supervised scale-aware distance estimation using monocular fisheye camera for autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2020, pp. 574–581.

[23] M. Bertozzi, L. Castangia, S. Cattani, A. Prioletti, and P. Versari, "360 detection and tracking algorithm of both pedestrian and vehicle using fisheye images," in *2015 IEEE Intelligent Vehicles Symposium (IV).* IEEE, 2015, pp. 132–137.

[24] G. Blott, M. Takami, and C. Heipke, "Semantic segmentation of fisheye images," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 181–196.

[25] V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, "Near-field depth estimation using monocular fisheye camera: A semi-supervised learning approach using sparse lidar data," in *CVPR Workshop*, vol. 7, 2018.

[26] V. R. Kumar, M. Klingner, S. Yogamani, S. Milz, T. Fingscheidt, and P. Mader, "Syndistnet: Self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 61–71.

[27] T. Li, G. Tong, H. Tang, B. Li, and B. Chen, "Fisheyedet: A self-study and contour-based object detector in fisheye images," *IEEE Access*, vol. 8, pp. 71 739–71 751, 2020.

[28] M. Yahiaoui, H. Rashed, L. Mariotti, G. Sistu, I. Clancy, L. Yahiaoui, V. R. Kumar, and S. Yogamani, "Fisheyemodnet: Moving object detection on surround-view cameras for autonomous driving," *arXiv preprint arXiv:1908.11789*, 2019.

[29] Z. Wu, M. Wang, L. Yin, W. Sun, J. Wang, and H. Wu, "Vehicle re-id for surround-view camera system," *arXiv preprint arXiv:2006.16503*, 2020.

[30] B. Arsenali, P. Viswanath, and J. Novosel, "Rotinvmtl: Rotation invariant multinet on fisheye images for autonomous driving applications," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 2373–2382.

[31] E. Plaut, E. B. Yaacov, and B. E. Shlomo, "Monocular 3d object detection in cylindrical images from fisheye cameras," *arXiv preprint arXiv:2003.03759*, 2020.

[32] M. Uricár, J. Ulicny, G. Sistu, H. Rashed, P. Krizek, D. Hurych, A. Vobecky, and S. Yogamani, "Desoiling dataset: Restoring soiled areas on automotive fisheye cameras," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 4273–4279.

[33] V. R. Kumar, S. Yogamani, H. Rashed, G. Sitsu, C. Witt, I. Leang, S. Milz, and P. Mäder, "Omnidet: Surround view cameras based multi-task visual perception network for autonomous driving," *arXiv preprint arXiv:2102.07448*, 2021.

[34] Y. Wu, S. Feng, X. Huang, and Z. Wu, "L4net: An anchor-free generic object detector with attention mechanism for autonomous driving," *IET Computer Vision*, 2021.

[35] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2147–2156.

[36] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2040–2049.

[37] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9287–9296.

[38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[39] G. P. de La Garanderie, A. A. Abarghouei, and T. P. Breckon, "Eliminating the blind spot: Adapting 3d object detection and monocular depth estimation to 360 panoramic imagery," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 789–807.