Vision-Based Parking Assist System with Bird-Eye Surround Vision for Reverse Bay Parking Maneuver Recommendation

Kevin Tirta Wijaya[1], Luqman Yusuf Bharoto[2], Andrey Purwanto[3], Eniman Yunus Syamsuddin[4]
School of Electrical Engineering and Informatics, Bandung Institute of Technology, Bandung, Indonesia
{[1]kevin.tirta18, [2]luqmanyb, [3]andrey327.ap}@gmail.com, [4]eniman@stei.itb.ac.id

***Abstract*— In recent years, the automotive industry has seen rapid growth in autonomous vehicle and advanced driver-assistance system (ADAS) technologies. One of the key aspects of this technology is the parking assist system. The core function of a parking assist system is to help drivers in parking their vehicles safely and efficiently. This paper proposes a method for low-cost real-time semi-autonomous vehicle parking in which a Vision-Based Parking Assist System (VPAS) gives the driver maneuver recommendations for reverse bay parking. The proposed method integrates wide-angle lens correction, bird-eye surround view, and user-guided vision-based parking line detection as ways to calculate recommendations for the driver. VPAS aims for affordability and generality, thus it emphasizes streamlining most of the calculations to compensate for real-time needs while maintaining reliability in a relatively low-cost system that can be run on an entry-level traditional hatchback car. To verify the performance of the proposed methods, experiments on the integrated system are conducted in a controlled environment. From experimental results, we demonstrate the performance of VPAS in terms of real-time and accuracy aspects while running on a 4-cores 3.5 GHz processor powered by the car battery.**

Keywords— autonomous vehicle, ADAS, parking assist system, bird-eye surround view, parking line detection

I. INTRODUCTION

Autonomous vehicle is one of the pinnacles of technology in our time. The idea of driverless vehicles captivates many researchers both in academia and industry. It is commonly predicted that autonomous vehicle technology will increase both people's quality of life and driving safety [1]. However, current technologies are still limited for us to be able to deploy a fully autonomous vehicle. A more practical approach is to use an advanced driver assistance system (ADAS), a semi-autonomous system that helps drivers maneuver their vehicles safely and efficiently.

One major component of ADAS is the parking assist system (PAS) which helps the driver in parking their vehicle. Parking is one of the most common stages of driving where collisions happen. In [2], it is reported that 23% of car insurance claims on the United Kingdom are related to parking collisions, 75% of which are for reverse bay parking. The addition of an onboard parking assist system would help reduce the number of incidents when parking.

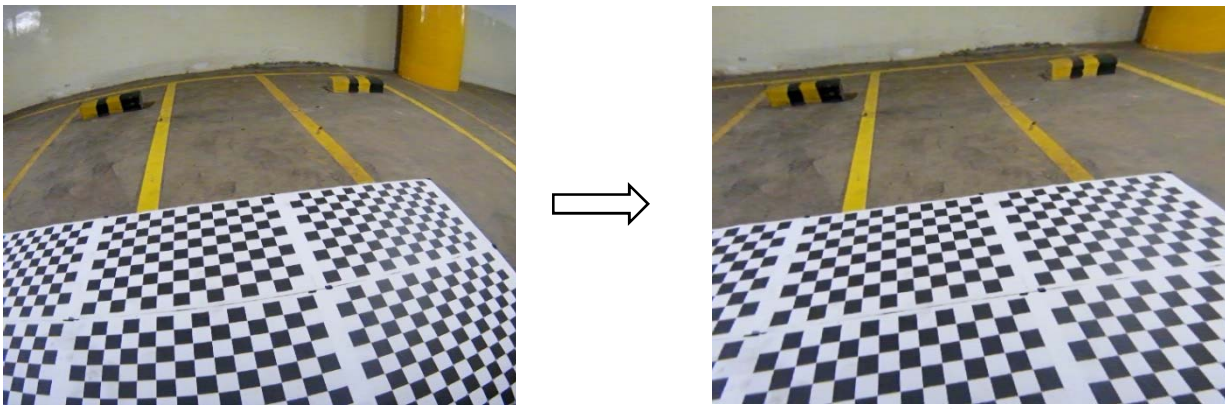
In general, parking assist system is divided into two categories: space-oriented and maneuver-oriented. On a space-oriented system like [3], the system helps the driver in finding free parking spaces for their vehicle. On a maneuver-oriented system like [4, 5, 6], the system helps drivers maneuver their vehicles while parking. On a maneuver-oriented system, the ability to perform decently accurate environmental perception on a real-time basis is crucial.

Previous methods such as [4] use Light Detection and Ranging (LiDAR) or ultrasonic sensors for environmental perception, which gives accurate results on specific cases. Those methods rely on 3-dimensional textures to acquire data and useful for cases where the parking spot is between two objects (e.g. other cars). Unfortunately, when there is no other vehicle present, a parking spot is only marked with 2-dimensional parking lines, thus making LiDAR and ultrasonic sensors unreliable for environmental perception. A vision-based approach is more suitable for 2-dimensional data acquisition, as cameras can give decent information in the form of an image.

This paper proposes a Vision-Based Parking Assist System (VPAS) which helps the driver by giving maneuver recommendations for reverse bay parking. Other method like [5] gives highly accurate results by using around view monitor (AVM), deep learning, and simultaneous localization and mapping (SLAM) to do environmental perception and generate parking path, then automatically control the vehicle to park itself. However, to get the system works in real-time, this method uses three distinct high-power computing units: one for AVM, one for path generation and vehicle control, and one for image segmentation, image recognition, and SLAM. It is too costly to be implemented on most consumer cars, making the solution works only for niche academic research, highly specialized vehicles, or high-end consumer cars. With this consideration, VPAS is aiming for affordability and generality so the system can be run even on entry-level traditional hatchback car. The system has two main submodules, namely: bird-eye surround view generator and parking maneuver recommendation unit.

VPAS utilizes wide-angle cameras to get information about its environment. A bird-eye surround view is generated from the acquired images. Compared with a 3D surround view like [7] and [8], this method produces good performance results for assisting the driver while using less computational power. For parking line detection and tracking, a Haar cascade classifier and centroid tracking are used. Harr classifier is chosen rather than other deep learning-based object recognition methods such as YOLO [9] and SSD [10] because of its simplicity, thus requires less computational power and fewer datasets. To calculate recommendations for the driver, a rules-based system is first established. The rules-based system compares the pose of the vehicle and parking lines to propose a recommendation to the driver.

The remainder of this paper is organized as follows. Section II explains VPAS' system components. Section III introduces the software used. Section IV presents the experimental results. Section V concludes this paper with a brief summary.



(a) (b)

Fig. 1. Camera calibration process – (a) before distortion correction, (b) after distortion correction

II. Hardware Overview

The designed system consists of four wide-angle cameras, a modified personal computer, and a touchscreen LCD. The four cameras are positioned under two rear-view mirrors and near the two tail lamps as seen in Fig. 2, giving a total of 270˚ field of view on the lateral and rear side. While the cameras have 180^0 field of view, the distortion correction algorithm (Section III) reduces some part of the image near the edges, thus making the real field of view less than 180^0. The front side is chosen to be the blind spot as it is the least correlated side when doing a reverse bay parking. A 7" touchscreen LCD is used for communicating with the driver. These components are connected to a 4-cores 3.5 GHz personal computer, drawing power from the car battery.

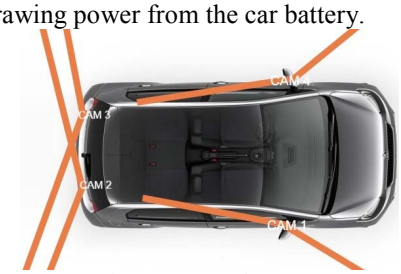


Fig. 2. Camera Placement

III. Software Overview

The flow of our overall algorithms can be found in Fig. 3. The algorithms are divided into two submodules: bird-eye surround view generation and parking maneuver recommendation. In implementing the image processing part, the OpenCV library [11] [12] is extensively used.

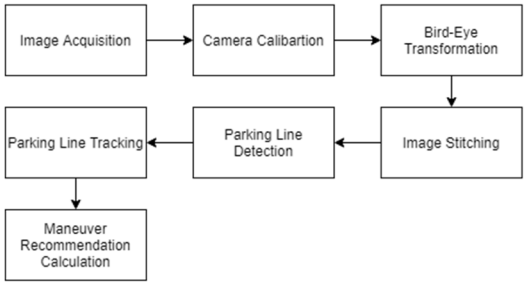


Fig. 3. Framework of VPAS

A. Bird-Eye Surround View Generation

Camera Calibration

Four wide-angle cameras are used to give bird-eye surround vision. Unfortunately, radial distortion is inherent in short focal-length lenses. Since the distortion is affecting the image quality, more so on pixels further from the optical center [13], the images acquired need to be corrected before any other processing is applied.

The radial distortion correction method uses real-world coordinates of a fixed-geometry reference to be transformed into a set of collinear points in pixel coordinates. To automate most of the process, a chessboard pattern is used. The intersection of dark-light squares can be detected using corner detection algorithms like Harris Corner Detector [14] and used as references to be transformed.

The coordinates of a point on real-world reference P inside camera reference frame can be defined in terms of rotation R and translation T matrix

$$P_c = R(P - T)$$

The coordinate of P_c consists of three vectors x, y, and z. Using a pinhole camera model, we can project P_c into 2D coordinates on a flat image plane.

$$x' = \frac{x}{z};\ y' = \frac{y}{z}$$
$$r^2 = x'^2 + y'^2$$

The distorted coordinates on the image plane can be described as [15]

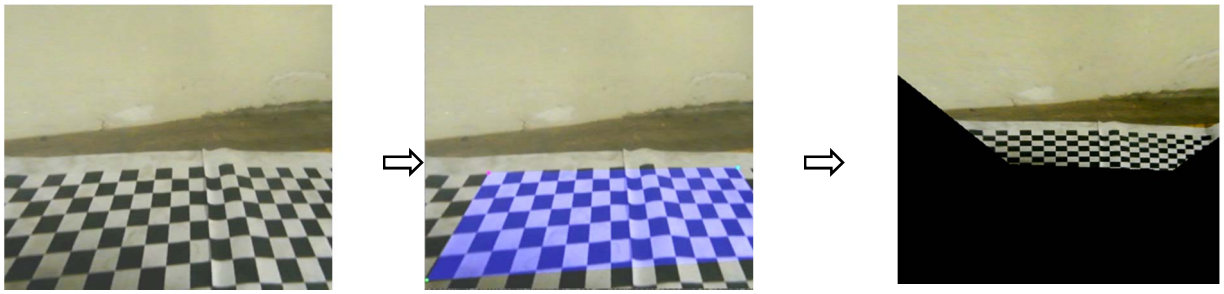
$$x'' = x'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + k_4 r^8)$$
$$y'' = y'(1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + k_4 r^8)$$

where k_n is the n^{th} radial distortion coefficient. The coordinates on the image plane are represented on pixel coordinates $[u, v]$ by

$$u = f_x x'' + c_x$$
$$v = f_y y'' + c_y$$

where f is the focal length of the camera and c is the offset between the center of the imager and the optical axis. Using collinear references from the chessboard corners pattern, OpenCV can estimate f_x, f_y, c_x, c_y, k_1, k_2, k_3, k_4, R, and T, thus enabling us to correct radial distortion.

Figure 1 shows the process of camera calibration. Using a chessboard image, the dark-light intersections are automatically detected and used for calibration. As shown in Fig. 1(b), some edge parts of the image are lost in the calibration process due to inherently insufficient information



(a) (b) (c)

Fig. 4. Bird-eye transformation – (a) initial image, (b) four points selection, (c) transformed image

around the edge parts. This reduces the used cameras' field of view to be less than 180^0.

Bird Eye Surround View Generation

To give the driver a more intuitive and standardize form of images surrounding the vehicle, the images are transformed into bird-eye view before stitching. Bird-eye transformation is done by applying projective transformations using homography matrix obtained from known-form objects [16]. Given corresponding points x'_i and x_i, the homography matrix H is a matrix that satisfies

$$\begin{bmatrix} t_i x'_i \\ t_i y'_i \\ t_i \end{bmatrix} = \boldsymbol{H} \cdot \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

The bird-eye transform homography matrix for each camera is calculated from a priori known-form objects, in this case, chessboard patterns that are laid around the vehicle. Four points on the chessboard patterns (x_i, y_i) are chosen to be transformed into their corresponding points (x'_i, y'_i) which are rectangular in the real world. After the homography matrix H for each camera is found, a perspective transformation is applied to the images acquired by each camera, defined as

$$x_{dst} = \frac{H_{11}x_{src} + H_{12}y_{src} + H_{13}}{H_{31}x_{src} + H_{32}y_{src} + H_{33}}$$

$$y_{dst} = \frac{H_{21}x_{src} + H_{22}y_{src} + H_{23}}{H_{31}x_{src} + H_{32}y_{src} + H_{33}}$$

Figure 4 shows the bird-eye transformation method, in which four points on the chessboard pattern are chosen. The corresponding points locations on the final image are chosen based on real-world geometry of the vehicle and chessboard patterns used.

While bird-eye transformation gives an intuitive and standardized form of images surrounding the vehicle, they are still detached from each other. Image stitching is performed to attach adjacent images. The real-world geometry of the vehicle and chessboard patterns, the relative position of adjacent images, and their overlapping regions are used as references for seam position. The final stitched image of the four bird-eye transformations produces a single-image bird-eye surround vision. Four separate images acquired from each camera are shown in Fig. 5, and an example of a generated bird-eye surround view in a parking lot is shown in Fig 6.

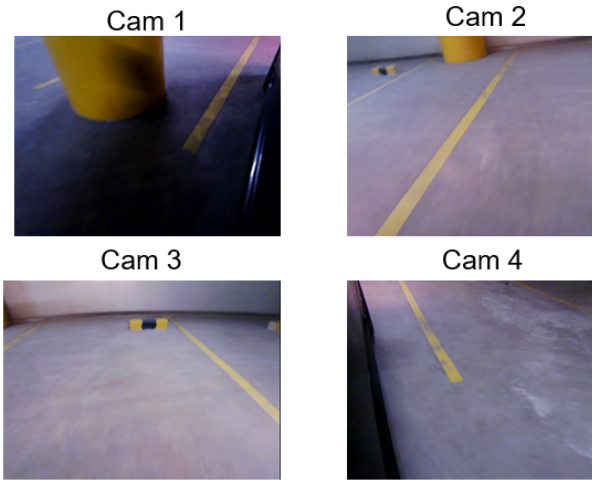


Fig. 5. Image Acquisition from Each Camera

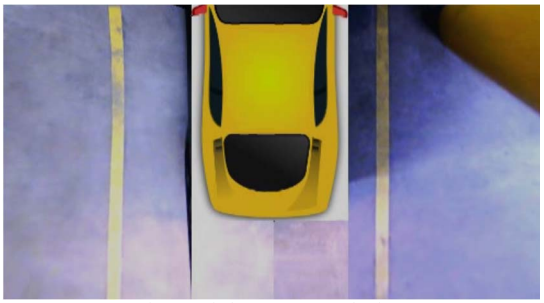


Fig. 6. Bird-eye Surround View

B. Parking Maneuver Recommendation Calculation

Parking Line Detection and Tracking

After a bird-eye surround view is generated, parking line detection can be done from a single image input. Haar-like feature-based cascade classifier using the Viola-Jones algorithm [17] is used for parking line detection. Viola-Jones algorithm is a machine learning-based approach for object detection by training a cascade function.

To train the classifier, a set of Haar-like features as shown in Fig. 7 needs to be defined and used as convolutional kernels. These kernels are then used to generate features all over the input image.

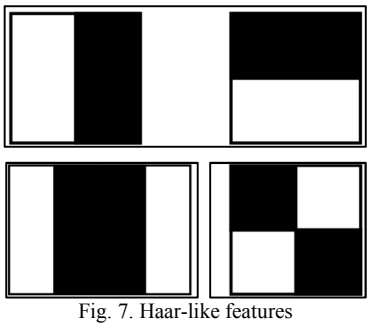


Fig. 7. Haar-like features

The selection of image part that contains Haar-like feature can be achieved by firstly sums up the white pixel intensities *I(x)* and calculates its average value. Secondly, sums up the dark pixel intensities *I(x)* and calculates its average value separated from the white pixel area. Finally, subtracts the average value from the white pixels to the dark pixels. The closer the δ to 1, the more likely a Haar-like feature is found. An example of this can be seen in Fig. 8.

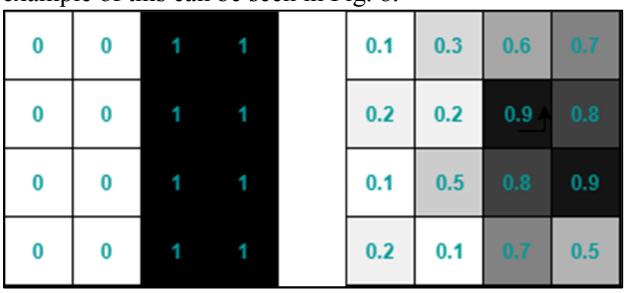


Fig. 8. Haar-like features detection on an image

To optimize the features generation process, an integral image [17] and the AdaBoost algorithm [18] are used. These features are then applied to a window sliding on the input image to determine whether an object is detected or not. To reduce the number of features applied, a cascade of classifiers [17] is used. Each stage uses different features and focuses to discard the non-parking lines object quickly and spend more time on detecting the probable parking line regions on the following stages. The illustration of the cascading classifier is shown in Fig. 9. An example of Haar-like feature-based for parking line detection is shown in Fig. 10.

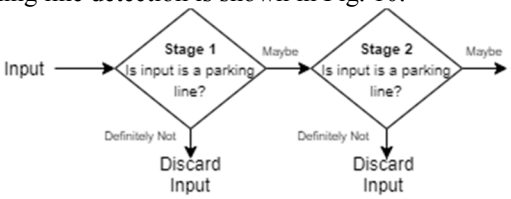


Fig. 9. Illustration of cascading classifier

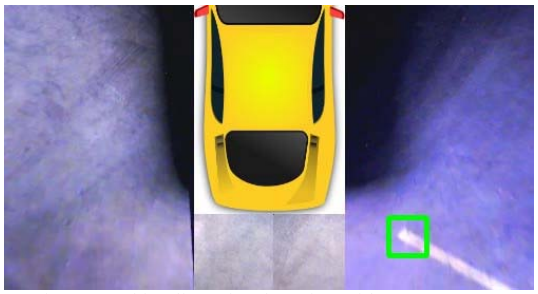


Fig. 10. Parking line edge detection with Haar Detector

While the Haar-like feature-based detector is light weighted, the detection can differ on each frame. A parking line edge can be detected on frame n and not detected on frame n+1 even though both frames show similar images to human eyes. To remedy this, a distance-based tracker is used. The algorithm used for the tracker is shown in Table 1.

Table 1 – Tracker Algorithm

1. Initialize tracker q with detected ROI containing parking line edge from Haar detector
2. Declare a distance threshold δ
3. For each frame after the initialization:
   If new parking line edge is not detected:
   $q_{new} = q_{old}$
   Else:
   Calculate Euclidean distance between q_{old} and q_{new}
   If (dist < δ):
   $q_{new} = q_{new}$
   Else:
   $q_{new} = q_{old}$

$$q_{new} = q_{old}$$

Recommendation Calculation

The recommendation calculation is done by comparing the state of the vehicle to the position of parking lines. Three points are considered for the comparison: two on each side of the vehicle, and one in the middle of the rear-side of the vehicle as shown in Fig. 11. The comparisons are used as inputs for a rule-based system that produces recommendations which will be displayed on an LCD. The algorithm of the rule-based system for right-side reverse-bay parking is given in Table 2.

Fig. 11. Locations of Point of Interests on the Vehicle

Table 2 – Maneuver Recommendation

Initial State: the right side of the vehicle is perpendicular to the parking lines, with two visible ends of parking lines as shown in Fig. 10. On the surround view, let A be the coordinate of upper parking line, B the coordinate lower parking line, P1 the point of interest on right-side, P2 the point of interest on rear-side, and P3 the point of interest on left-side.

Procedure:

1. Move forward until P1.y < A.y
2. Fully turn the steering wheel clockwise
3. Move backward until the distance between P2 and A is greater than a threshold δ
4. Straighten the steering wheel
5. Move forward until P1.y < A.y
6. Fully turn the steering wheel clockwise

7. Move backward until both A.y and B.y values are within a certain range $y \in [y_{min}, y_{max}]$
8. Straighten the steering wheel
9. Move backward

IV. EXPERIMENTAL RESULTS

Experimental Setup

The designed system is tested in a controlled environment. An underground parking lot with appropriate lighting is used for the testing site. The parking slot used has empty adjacent slots. The vehicle used is an entry-level consumer hatchback car with hardware setup as explained in Section II.

Parking Line Detection

The parking line detector and tracker are tested for their performance in detecting parking lines by calculating their precision. In the experiments, a total of 2181 unique frames are used for the detection test. From the experimental results, 1919 frames are successful detection, 208 are partial detection, and 54 are total detection failure. Partial detection occurs when one of the parking lines present in the frame is not detected but doesn't affect the maneuver recommendation algorithm negatively. Based on the results, the accuracy of Haar-like features cascade classifier for full detection is 87.98% and it achieves 97.52% reliability of detecting parking lines crucial for maneuver recommendation algorithm. In comparison, the highest individual class precision that YOLOv2 achieves is 90.6% [9]. Thus, Haar-like features cascade classifier used in VPAS produces competitive accuracy compared to high-performing detectors such as YOLOv2 while being computationally less expensive and without the needs of large datasets.

Integrated Test of the System

Two parameters are tested: response time and accuracy. Response time is defined as how fast the system can update the driver with new information gathered by the camera, measured in frame per second. Accuracy is defined based on $\Delta\delta$ between vehicle's heading final position and an imaginary line positioned on the center of the longitudinal axis of the vehicle and parallel to parking lines shown in Fig. 12, given the driver completely follows recommendations from the system. To measure the $\Delta\delta$, first the length of an imaginary line parallel to parking lines, l_1, and longitudinal axis of vehicle, l_2, are measured from the vehicle's center point. The $\Delta\delta$ is then calculated with the following equation,

$$\Delta\delta = arc\cos\frac{l_1}{l_2}$$

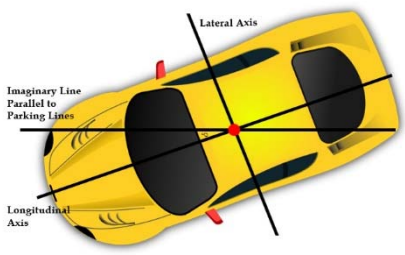


Fig. 12. Illustration for Accuracy Measurement

The driver starts the experiment by turning the system on and starts following the recommendation by the system. After the system gives stopping command, the vehicle is stopped and measurement on accuracy is done. To accurately capture the performance of VPAS, the experiment is conducted 20 times for right-side reverse bay parking. The results of the experiments can be found in Table 3.

Table 3 – Experimental Results for Integrated Test

Parameter	Value		
	Mean	Max.	Min.
FPS	10.42	11.21	5.12
Accuracy (deg)	3.36	5	0

The results show that the overall system has a real-time performance with high accuracy based on the final position of the vehicle. Therefore, VPAS is a robust parking assist system alternative while still being affordable and suitable to be used on general entry-level hatchback car.

V. CONCLUSION

In this paper, a low-cost real-time vision-based parking assist system (VPAS) is proposed. VPAS helps a human driver to park their car in a reverse-bay parking spot by giving them a bird-eye surround view and maneuver recommendations. The system corrected the radial distortion on the four wide-angle cameras and transform them into bird-eye view before stitching all four images into a single surround-view image. This image is then used as the input for parking line detection using Haar-like feature-based cascade classifiers and parking line tracking using a distance-based tracker. A maneuver recommendation is chosen according to a predefined rules-based system. The experimental results show that VPAS achieved real-time performance of 10 fps and high accuracy with mean heading error of 3.36^0, making VPAS suitable as an affordable alternative for parking assist system on general entry-level hatchback car.

REFERENCES

[1] P. Koopman and M. Wagner, "Autonomous Vehicle Safety: An Interdisciplinary Challenge," *IEEE Intelligent Transportation Systems Magazine,* 2017.

[2] M. Avery, "Automated Driving: The Technology and Implications for Insurance," December 2016. [Online]. Available: http://etsc.eu/wp-content/uploads/MatthewAvery.pdf. [Accessed 3 June 2020].

[3] T. Rajabioun, B. Foster and P. Ioannou, "Intelligent Parking Assist," in *21st Mediterranean Conference on Control and Automation, MED 2013*, 2013.

[4] B. Lee, Y. Wei and I. Guo, "Automatic Parking of Self-Driving Car Based on LIDAR," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences,* 2017.

[5] J. Chulhoon, K. Chansoo, L. Seongjin, K. Seokwon, L. Sumyeong and S. Myoungho, "Re-Plannable Automated Parking System With a Standalone Around View Monitor for Narrow Parking Lots," *IEEE Transactions on Intelligent Transportation System,* vol. 21, no. 2, pp. 777-790, 2020.

[6] W. Chunxiang, Z. Hengrun, Y. Ming, W. Xudong, Y. Lei and G. Chunzhao, "2014," *Advances in Mechanical Engineering,* vol. 2014, 2014.

[7] Y. Gao, C. Lin, Y. Zhao, X. Wang, S. Wei and Q. Huang, "3-D Surround View for Advanced Driver Assistance Systems," *IEEE Transactions on Intelligent Transportation Systems,* 2017.

[8] L. Zhang, J. Chen, D. Liu, Y. Shen and S. Zhao, "Seamless 3D Surround View with a Novel Burger Model," 2019.

[9] R. Joseph and F. Ali, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, 2017.

[10] L. Wei, A. Dragomir, E. Dumitru, S. Christian, R. Scott, F. Cheng-Yang and B. Alexander C, "SSD: single shot multibox detector," in *European Conference on Computer Vision*, 2016.

[11] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools,* 2000.

[12] G. Bradski and A. Kaehler, Learning OpenCV, Sebastopol, CA: O'Reilly Media, 2008.

[13] C. Ricolfe-Viala and A.-J. Sánchez-Salmerón, "Lens Distortion Models Evaluation," *Applied Optics,* 2010.

[14] C. G. Harris and M. Stephens, "A Combined Corner and Edge Detection," in *Alvey Vision Conference*, 1998.

[15] D. C. Brown, "Decentering Distortion of Lenses," 1966.

[16] I. S. Kholopov, "Bird's Eye View Transformation Technique in Photogrammetric Problem of Object Size Measuring at Low-altitude Photography," in *Actual Issues of Mechanical Engineering*, 2017.

[17] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001.

[18] Y. Freund and R. E. Schapire, "A Decision-Theoritic Generalization of On-Line Learning and an Application to Boosting," *Journal of Computer and System Sciences,* vol. 55, pp. 119-139, 1997.

[19] H. Tsung-Shiang, "An Improvement Stereo Vision Images Processing for Object Distance Measurement," *International Journal of Automation and Smart Technology,* 2015.

[20] K. Zdenek, M. Krystian and M. Jiri, "Forward-Backward Error: Automatic Detection of Tracking Failures," in *International Conference on Pattern Recognition*, Instanbul, 2010.

[21] L. Bruce D. and K. Takeo, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[22] L. Alan, V. Toma´sˇ, Z. Luka Cˇ ehovin, M. Jirˇ´ı and K. Matej, "Discriminative correlation filter with channel and spatial reliability," in *2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[23] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE computer society conference on computer vision and pattern recognition*, 2010.