

Article

CNN-Based Vehicle Bottom Face Quadrilateral Detection Using Surveillance Cameras for Intelligent Transportation Systems

Gahyun Kim ¹, Ho Gi Jung ² and Jae Kyu Suhr ^{1,*}¹ Department of Intelligent Mechatronics Engineering, Sejong University, Seoul 05006, Republic of Korea² Department of Electronic Engineering, Korea National University of Transportation, Chungju-si 27469, Republic of Korea

* Correspondence: jksuhr@sejong.ac.kr

Abstract: In intelligent transportation systems, it is essential to estimate the vehicle position accurately. To this end, it is preferred to detect vehicles as a bottom face quadrilateral (BFQ) rather than an axis-aligned bounding box. Although there have been some methods for detecting the vehicle BFQ using vehicle-mounted cameras, few studies have been conducted using surveillance cameras. Therefore, this paper conducts a comparative study on various approaches for detecting the vehicle BFQ in surveillance camera environments. Three approaches were selected for comparison, including corner-based, position/size/angle-based, and line-based. For comparison, this paper suggests a way to implement the vehicle BFQ detectors by simply adding extra heads to one of the most widely used real-time object detectors, YOLO. In experiments, it was shown that the vehicle BFQ can be adequately detected by using the suggested implementation, and the three approaches were quantitatively evaluated, compared, and analyzed.

Keywords: vehicle position estimation; bottom face quadrilateral; deep neural network; vehicle-to-infrastructure (V2I); intelligent transportation system; surveillance camera



Citation: Kim, G.; Jung, H.G.; Suhr, J.K. CNN-Based Vehicle Bottom Face Quadrilateral Detection Using Surveillance Cameras for Intelligent Transportation Systems. *Sensors* **2023**, *23*, 6688. <https://doi.org/10.3390/s23156688>

Academic Editors: Wenfeng Li, Xiuwen Fu, Gabriel Lodewijks and Yulian Cao

Received: 26 May 2023

Revised: 15 July 2023

Accepted: 24 July 2023

Published: 26 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Surveillance camera-based vehicle detection has been widely used in various intelligent transportation system (ITS) applications [1,2]. In many of these, an accurate vehicle position is required [3–5]. For instance, autonomous driving based on vehicle-to-infrastructure (V2I) prevents accidents or control vehicles using accurate positions provided by infrastructure surveillance cameras. Various sensors have been used for vehicle detection, including monocular cameras [6–9], lidars [10,11], lidar-camera fusion [12,13], and stereo cameras [14]. Among them, monocular cameras are preferable because they can be installed at a low cost and require less computational load. In addition, widely spread existing monocular surveillance cameras can be reused for ITS applications [8]. Considering these advantages, this paper presents a method for accurately estimating the vehicle position using a monocular surveillance camera.

The most representative way to estimate the vehicle position using a monocular camera is as follows. It first detects the vehicle position in-image coordinates and then transforms it to world coordinates using homography, which expresses the relationship between the ground in-image coordinates and in-world coordinates. This approach has a vital assumption that the detected vehicle position in-image coordinates should be on the ground surface. If this assumption is not satisfied, the vehicle position accuracy in-world coordinates is degraded. Therefore, vehicle positions in-image coordinates should be carefully detected when using monocular cameras.

The simplest method for finding the vehicle position is using the center of the bounding box given by object detectors [15–17]. This method is computationally efficient and easy to implement because it simply reuses conventional object detection results. However, this

method does not follow the aforementioned assumption because the center of the bounding box is not located on the ground surface. Thus, it produces erroneous positioning results, as shown in Figure 1. In this figure, the blue cross in the image indicates the center of the bounding box. If the blue cross is used as the vehicle position in-image coordinates, the vehicle position in-world coordinates are calculated as the blue dot on the ground. This blue dot includes an error because the correct vehicle position on the ground is the red dot on the ground. The amount of this position error depends on the relative position between the vehicle and camera in-world coordinates. To reduce this vehicle position error, the red cross should be detected in the image because it corresponds to the correct vehicle position on the ground (red dot) in Figure 1 [18]. The red cross indicates the bottom face quadrilateral (BFQ) center of the vehicle in the image.

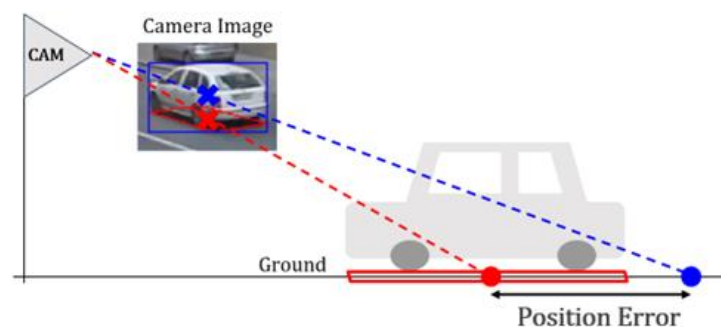


Figure 1. Vehicle position error when using the center of the bounding box. The blue cross is the center of the blue bounding box. The red quadrilateral and cross are the bottom face of the vehicle and its center. The distance between the blue and red dots on the ground indicates the position error.

Therefore, this paper focuses on detecting the BFQ to enhance vehicle position accuracy in visual surveillance environments. Although there have been some methods for detecting the vehicle BFQ using vehicle-mounted cameras, few studies have been conducted using surveillance cameras. For vehicle position, it is enough to estimate only the single point, the BFQ center, but the BFQ with four corners is preferred because it gives additional useful information, such as the size and direction of the vehicle, as shown in Figure 2.

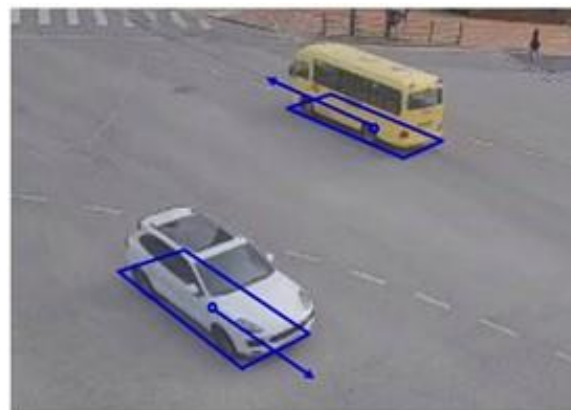


Figure 2. The usefulness of the vehicle BFQ. Quadrilaterals, circles, and arrows indicate BFQs, their centers, and their directions, respectively.

The proposed vehicle BFQ detection and the 3D vehicle detection both provide information on the vehicle's location, direction, and size. Compared to BFQ detection, the only additional information provided by the 3D vehicle detection is the vehicle's height. The vehicle's height is rarely used in V2I-based autonomous driving since this application only requires the vehicle's location, direction, and size. For the 3D vehicle detection, four points of the vehicle top face quadrilateral (TFQ) should be estimated in addition to four points of the BFQ. Therefore, the 3D vehicle detection has the following drawbacks compared to the

proposed vehicle BFQ detection: (1) it requires more effort for labeling to build the dataset, (2) it increases the computational cost and network size, and (3) it degrades the detection performance because the TFQ's shape varies a lot and does not fit the vehicle size, unlike the BFQ.

Single image-based 3D reconstruction can also be used for vehicle positioning in surveillance environments. The most representative approach for this is monocular depth estimation, which is the task of estimating the depth value of each pixel given a single RGB image [19]. This approach has the advantage of obtaining dense distance information with a single image but has the following disadvantages in surveillance camera situations. First, this approach has difficulties in handling various backgrounds and objects that appear in the real world because of the limited dense depth labels. This disadvantage is more prominent in surveillance situations because the camera's intrinsic properties (field of view, distortion, etc.) and extrinsic properties (installation height, angle, etc.) are very diverse. Second, this approach has limitations in handling distant objects captured with small sizes in surveillance camera images. The positions of distant objects are difficult to estimate based on monocular depth estimation because it often ignores small-sized objects or inaccurately estimates their depths. Last, this approach uses a large network with a high computational cost. Since surveillance cameras should use low-end processors when considering their reasonable prices, using a small network with fast inference time is preferable.

Recently, there has been an increase in the convolutional neural network (CNN)-based methods for detecting the BFQ using a monocular camera. Based on the way to represent the quadrilateral, these methods can be categorized into three approaches: corner-based, position/size/angle (PSA)-based, and line-based. The corner-based approach represents the BFQ with its four corners [6–9,13,20–26]. The PSA-based approach represents the BFQ with its position, size, and angle [27–37]. The line-based approach represents the BFQ with four lines that compose its edge [3,38–41]; further details will be explained in the next section. Since most of these methods are for vehicle-mounted cameras, their effectiveness in surveillance environments has not been well addressed. Therefore, this paper conducts a comparative study on various approaches for detecting the vehicle BFQ in surveillance environments.

The proposed method has two main contributions. First, it presents a guideline for developing the vehicle BFQ detection method using a monocular surveillance camera by quantitatively evaluating various approaches. Second, it suggests a way to implement the vehicle BFQ detectors by adding extra heads to one of the most widely used real-time object detectors, YOLO. This enables researchers to develop their own real-time vehicle BFQ detectors more easily by slightly changing their current object detectors. This paper explores all three aforementioned approaches: corner-based, PSA-based, and line-based. The experimental results show that the corner-based approach is the most effective for vehicle BFQ detection. This is because, in this approach, position errors of four corners have less effect upon the shape of the vehicle BFQ. In contrast, in the other two approaches, position errors of PSAs and lines highly degrade the BFQ detection performance.

The rest of this paper is organized as follows. Section 2 reviews previous studies related to vehicle BFQ detection. Section 3 explains the implementation details of various BFQ detection approaches. Section 4 describes the experimental results and discussions. Finally, this paper is concluded with a summary in Section 5.

2. Related Works

Vehicle detection methods can be categorized into four approaches according to the complexity of expressing the vehicle: bounding box-based [42], bottom face quadrilateral (BFQ)-based [5], 3D bounding box-based [33], and wire frame-based [43]. Since this paper deals with the BFQ-based approach that provides the vehicle's position, size, and direction while requiring less computation, the detailed related literature has been reviewed only for this approach. The methods in the BFQ-based approaches can be classified into three

methods: corner-based, position/size/angle (PSA)-based, and line-based. Figure 3 shows brief descriptions of these three approaches. The BFQ, which is the detection target, is shown in red. The blue features represent what each approach finds to detect the BFQ. All three approaches detect the BFQ by finding the four corners, the position/size/angle, and the four lines of the BFQ, respectively.

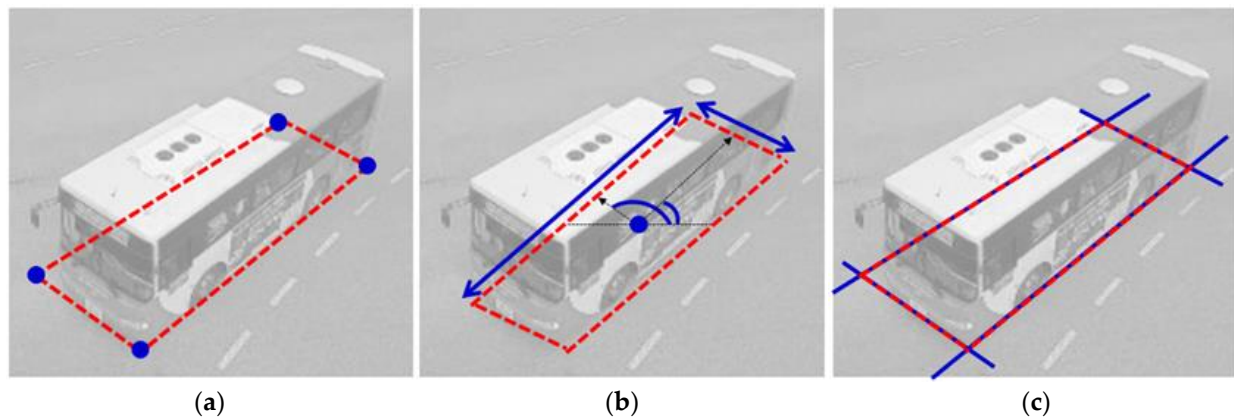


Figure 3. Three approaches for vehicle BFQ detection. The red quadrilaterals represent the vehicle BFQs. (a) The corner-based approach detects the BFQ by finding four corners (blue dots); (b) the PSA-based approach detects the BFQ by finding the position (blue dot), size (blue arrows), and rotation angle (blue arcs); and (c) the line-based approach detects the BFQ by finding four quadrilateral lines (blue lines).

2.1. Corner-Based Approach

The corner-based approach detects the vehicle BFQ by finding the corners of the bounding box, which can be a rectangle or cuboid. The detected corners are used directly or auxiliary to generate the corresponding rectangle or cuboid. In [20,23], eight corners were detected directly by regressing offsets from the reference point to each corner. The reference point was the center of the 2D bounding box, and the offsets were the coordinate differences along the image axes, which are the x and y axes, from the reference point to the corner. The offsets were encoded as ratios of the width and height of the 2D bounding box, respectively. On the other hand, [22] defined the lower left corner of the 2D bounding box as the reference point and encoded the offset as a ratio to the width of the 2D bounding box. In [21], it also directly detects eight corners. They designed a network consisting of four subnetworks of 2D detection, instance depth estimation, projected 3D center estimation, and local corner regression. The eight corners were predicted in local coordinates with the center of the 3D bounding box as the origin. Inspired by CenterNet, [6,24,44] created heatmaps to determine the eight corners and the center of the 3D bounding box. In particular, [6] regressed offsets from the main center to each corner. The main center is the 2D bounding box center. In [9], the corners were located by transforming the ROI from the 2D bounding box to column vectors and passed it through the Softmax layer. In [7,13,26], only four out of eight corners were detected. In addition to four corners of the BFQ, [13,26] found two heights and one height information from the ground to the bottom and top face, respectively. In [7], the Single Shot MultiBox Detector [45] was extended to output 3D bounding shapes using predefined four corners instead of 2D bounding boxes. Three of the four corners form the BFQ, and the remaining corner is used to determine the height. Papers [8,25] differ in using a wire model for a vehicle. In [8], key points were detected that are predefined using a 3D computer-aided design (CAD) model. Vehicles are classified into 5 models, and 14 points were designated for each model. The network uses mask R-CNN [46] as the backbone and directly regresses 14 points in one of the 3 sub-networks. In [25], vehicles were classified into four models and each model was assigned a 3D bounding box with a cuboid shape and a 3D shape defined by several points. The points of the 3D shape were regressed using the network.

2.2. PSA-Based Approach

The PSA-based method detects the vehicle BFQ by finding the position, size, and angle of the bounding box. In [27–37], the position, size, and rotation angle of the bounding box are obtained as the vehicle detection result. In [27], which has two subnetworks, depth-aware convolution was used to extract local features for object detection. The position, size, and angle were estimated by predicting the offset from the anchor. They encoded the offset as a ratio to the width or height of the 2D bounding box for position detection. The size and angle were determined by predicting the residuals and the log-scale offsets from the anchor boxes, which varied in size with the depth. In [31], average sizes were calculated from the dataset and used as anchor boxes. They found the dimension of the vehicle by regressing the residual on the anchor. The angle was decoded into a vectorial representation such as sine and cosine in the network. In [34], the center of the object was found using CenterNet and regressed the log-scale offset on the average size of each class. The angle was regressed on a vectorial representation as in [31]. In [35], the centerness was found, represented with 2D Gaussian distribution, which describes how closer points are to centers and regresses the offset from the center, the size, and the angle. The attributes such as size and angle are regressed from the shared head that consists of convolutional blocks. In [37], they regressed the discretization offset for the position, the log-scale offset for the size, and the vectorial representation for the angle detection. Some methods [28–30,32,33,36] use 2D bounding box information for position detection. Among them, [28,30,32,36] used the center of an edge of the 2D bounding box or the center of the bounding box. In [36], they refined the center position by estimating the disparity for more accurate positioning. In [33], it was assumed that the center of the 2D bounding box coincides with the center of the 3D bounding box. These methods have in common that they use the average size of each class for dimension detection. However, the difference in detail is that [28,36] regress the log-scale offset, while the others regress the residual with the mean size. In [29,30,32,36], they found the angle using the multi-bin method presented in [33]. The multi-bin method divides an angle with continuous values into a finite number of bins and regresses the residual from the center in each bin.

2.3. Line-Based Approach

The line-based approach detects the vehicle BFQ by finding four lines that form it. In [3,40], they found the vehicle blob by segmentation and then computed the tangent lines of the blob. By calculating the intersection of these lines, they obtained the bounding box. Some other methods trained the network to output parameters of the lines. In [38], they introduced a parameter corresponding to the height of the vehicle. The parameter is a ratio that allows it to calculate the location corresponding to the height of the vehicle on the vertical edge of the bounding box. By finding a straight line connecting the point calculated from the parameters and the vanishing point, the vehicle position can be calculated. In [39,41], they considered that some vertices of the 3D bounding box meet the edge of the 2D bounding box. In [39], they defined a local coordinate system, predicted several parameters for regression and classification, and generated the bounding box; unlike [39,41], who generated the bounding box based on the camera calibration. In [47], 3D templates for each class were defined and the bounding box was obtained by using the similarity between vehicles and templates.

3. Implementation Details

This paper implements and compares 11 representative methods for vehicle BFQ detection based on the literature review. These methods are implemented by adding extra heads to one of the most widely used real-time object detectors, YOLO.

3.1. Implementation Using YOLO

YOLO is one of the most widely used one-stage detectors used in various fields such as traffic surveillance [48,49], autonomous driving [50,51], unmanned aerial vehicles [52,53],

and robotics [54]. Among several YOLO variants, YOLOv4 was selected in this paper because it has been proven for a considerable time in a variety of applications and has demonstrated a compromise between detection accuracy and computational cost in various frameworks [55,56]. The vehicle BFQ detectors are implemented by adding extra heads to YOLOv4, as shown in Figure 4. The extra heads for the BFQ detection are represented as red BF blocks, which are added to the Y blocks for vehicle detection. Except for the BF blocks (red blocks), the structure is the same as that of YOLOv4, which consists of the backbone, the neck, and the head. The backbone is CSPDarkNet53 [57] with a spatial pyramid pooling (SPP) module [58] and trained on the COCO dataset [59]. The neck of the network consists of convolution (C), batch normalization (B), leaky ReLU (L), up-sampling, and concatenation modules. Most layers use the first three modules together, but only the last layer of the head uses a convolution. The combination of the three modules (convolution, batch normalization, and leaky ReLU) is denoted as CBL in Figure 4. The up-sampling and concatenation modules are used twice and four times to create multi-scale feature maps. The network takes a color image as an input and produces three feature maps with different scales as outputs. $Y_S + BF_S$, $Y_M + BF_M$, and $Y_L + BF_L$ denote the three output feature maps in Figure 4. Y and BF include information on bounding boxes and BFQs, respectively, and subscripts S, M, and L indicate their scales. The sizes of the output feature maps are 1/8, 1/16, and 1/32 of the input image size, respectively. The output feature map for the bounding box (Y) is the same as that of YOLOv4, including the position and size of the bounding box, the confidence score, and the class. The output feature map for the BFQ (BF) depends on the encoding scheme of the BFQ detection approach. Details are given in the following sections.

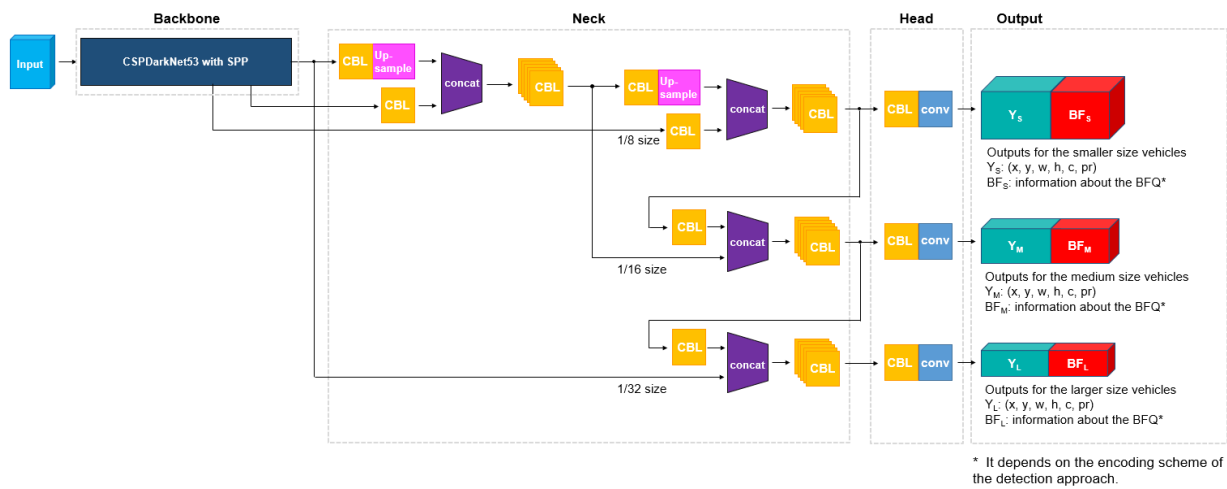


Figure 4. Structure of the proposed vehicle BFQ detector. CBL indicates a combination of convolution, batch normalization, and leaky ReLU. (x, y) and (w, h) are the position and size of the bounding boxes, respectively. c and pr are the confidence score and the class probability, respectively.

3.2. Implementation of Corner-Based Approach

The vehicle BFQ has four corners, and their positions in the image are represented as the sum of an origin coordinate and an offset. Therefore, detecting the corner is equivalent to estimating the offset from the predefined origin to the corner. The origin can be determined in various ways, but in this paper, two methods are applied by referring to the existing methods [20–23]. The offsets are also encoded in two ways. Thus, the total number of combinations of origin and offset is four. Figure 5a,b are cases where the origin (c_x, c_y) is defined differently. In Figure 5, the red quadrilateral represents the vehicle BFQ, and the blue rectangle represents the anchor box of the size $b_w \times b_h$. The anchor boxes are the same as those in YOLOv4, where nine anchor boxes are used (three for each scale). The anchor boxes were selected by applying k-means clustering to the sizes of the objects in the COCO dataset. Figure 5a is the case where the center of the anchor box is defined as the origin,

and Figure 5b is the case where the lower left corner of the anchor box is defined as the origin. As you can see in this figure, a corner coordinate of the BFQ is far from the origin by the offset (d_x, d_y) . The offsets are encoded in two ways using the anchor box size:

$$d_x = \delta_x \cdot b_w, \quad d_y = \delta_y \cdot b_h \quad (1)$$

$$d_x = \delta_x + b_w, \quad d_y = \delta_y + b_h \quad (2)$$

where δ_x and δ_y are the outputs of the added extra heads. δ_x and δ_y in (1) and (2) are the ratios to the anchor box size and the differences from the anchor box size, respectively. Table 1 shows that the four variations of the corner-based approach are created by the combination of origin definitions and offset encoding methods.

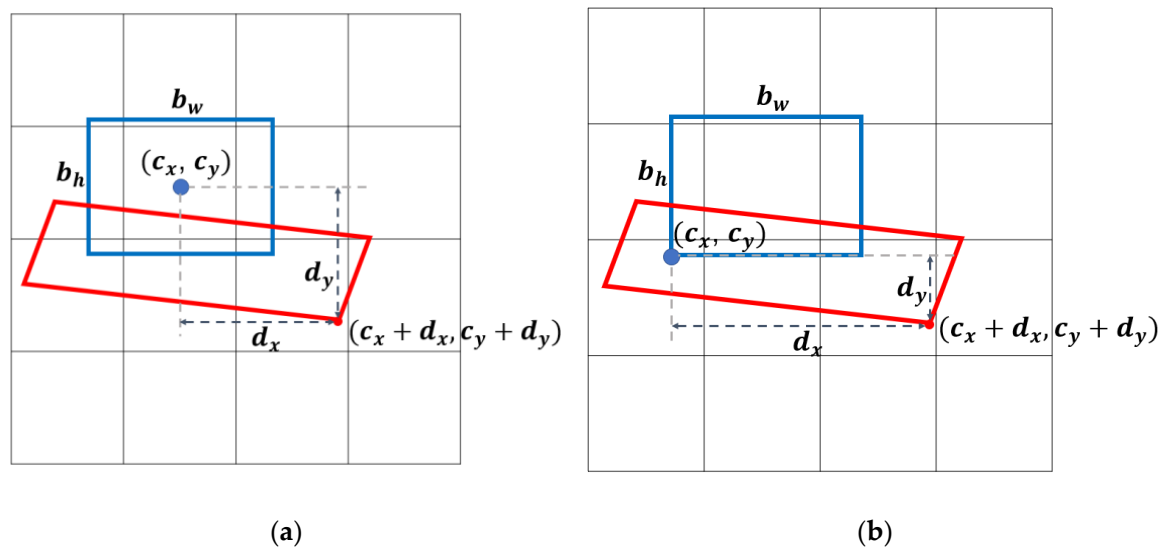


Figure 5. Definition of the origin for the corner-based approach. The blue rectangles are the anchor box with the size $b_w \times b_h$, and the red quadrilaterals are the vehicle BFQ: (a) The origin (c_x, c_y) is defined as the center of the anchor box; (b) The origin (c_x, c_y) is defined as the lower left corner of the anchor box.

Table 1. Four methods of the corner-based approach.

Abbreviation of Variations	Encoding Targets	
	Origin	Offset
C1	center of the anchor	ratio to the anchor size
C2	center of the anchor	residual to the anchor size
C3	bottom-left corner of the anchor	ratio to the anchor size
C4	bottom-left corner of the anchor	residual to the anchor size

3.3. Implementation of PSA-Based Approach

The methods included in the PSA-based approach identify the center, size, and angle of the vehicle for the position detection [27–37]. This approach detects the BFQ center for position detection, the width and height for size detection, and the local and global angles for rotation detection. Here, the local angle indicates how much a rectangle is skewed in order to become a parallelogram, and the global angle indicates how much a parallelogram is rotated. The angles can be used to determine the direction in which the vehicle is moving. Current methods usually establish a reference value, such as the 2D bounding box, the 3D anchor box, or the average value by class, and then detect the offsets from this reference value. In this paper, the reference value is replaced by a 2D anchor box, and the BFQ is assumed to be a parallelogram for simplicity [22,60].

Figure 6 shows a vehicle BFQ represented by the position, size, and angle. In this figure, the red parallelogram of size $l_v \times l_h$ is the BFQ, and the red dot is its center (c_x, c_y) . The center is detected by the C2 method, which is a method of the corner-based approach. The size of the BFQ is equal to the length of the two sides, l_v and l_h , of the parallelogram in Figure 6. The size is encoded using the anchor box size as:

$$l_v = r_x + b_w, \quad l_h = r_y + b_h \quad (3)$$

$$l_v = e^{r_x} \cdot b_w, \quad l_h = e^{r_y} \cdot b_h \quad (4)$$

where r_x and r_y are the outputs of the added extra heads. r_x and r_y in (3) and (4) are the differences from the anchor box size and the log-scaled ratios to the anchor box size, respectively.

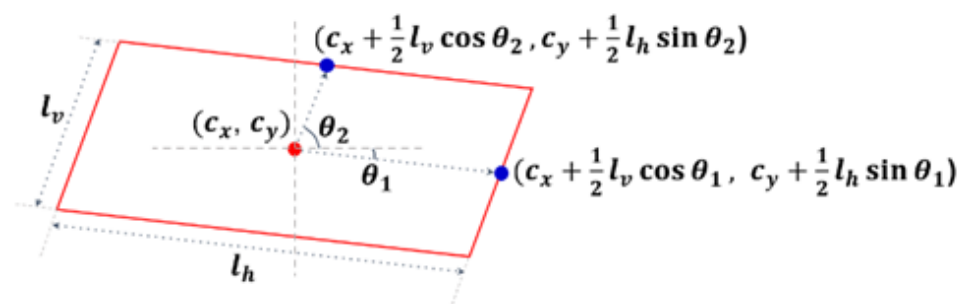


Figure 6. The vehicle BFQ described with its position, size, and angle. The red parallelogram is the vehicle BFQ. The red dot and the blue dot indicate the center of BFQ, and the center of the sides tilted by θ_1 and θ_2 from the image axes, respectively.

The angle of the BFQ can be found by using the vectors from the center (red dot) to the centers of the two sides of the parallelogram (blue dots) in Figure 6. Using the BFQ center, the lengths of the sides, and the angles, the vector representations of the centers of the sides are $(c_x + \frac{1}{2}l_v \cos \theta_1, c_y + \frac{1}{2}l_h \sin \theta_1)$ and $(c_x + \frac{1}{2}l_v \cos \theta_2, c_y + \frac{1}{2}l_h \sin \theta_2)$. In this paper, two methods of angle detection are used: direct estimation of the angle θ and estimation of $(\cos \theta, \sin \theta)$.

In the PSA-based approach, there is a change in the output dimension of the added extra heads for each variation. The output dimensions of the added extra heads for position and size detection are the same. However, the output for angle detection is (θ_1, θ_2) or $(\cos \theta_1, \sin \theta_1, \cos \theta_2, \sin \theta_2)$. According to different encoding methods of size and angle detection, four variations are created. They are shown in Table 2.

Table 2. Four methods of the PSA-based approach.

Abbreviation of Variations	Encoding Targets		
	Position	Size	Angle
PSA1	C2 applied	residual to the anchor size	two angles
PSA2	C2 applied	residual to the anchor size	cosine and sine of two angles
PSA3	C2 applied	log-scale offset	two angles
PSA4	C2 applied	log-scale offset	cosine and sine of two angles

3.4. Implementation of Line-Based Approach

The vehicle BFQ can be detected by estimating four straight lines. This paper presents two methods in the line-based approach: line parameter-based and intersection point-based.

3.4.1. Line Parameter-Based Method

A straight line can be expressed by the normal form of a linear equation. As shown in Figure 7a, if the blue line has a positive angle θ with the x-axis and a normal distance ρ from the origin, its linear equation is $x \cdot \cos \theta + y \cdot \sin \theta - \rho = 0$. In the same way, the four lines of the BFQ can be written as four linear equations. Figure 7b shows how one side of the BFQ is expressed as a linear equation. In this figure, the red rectangle and the black dashed quadrilateral are the bounding box and the BFQ, respectively. The blue line is the extension of one side of the BFQ. If the upper left corner of the bounding box is the same as the origin of the coordinate system, the blue line has the angle θ and the normal distance ρ as the line in Figure 7a. As a result, the equation form of the blue line is also $x \cdot \cos \theta + y \cdot \sin \theta - \rho = 0$. Therefore, the outputs of the added extra heads are four sets of $(\cos \theta, \sin \theta, \rho)$ and the vehicle position is finally detected by calculating the intersection of the four straight lines generated from these sets. This method is referred to as L1 in the following.

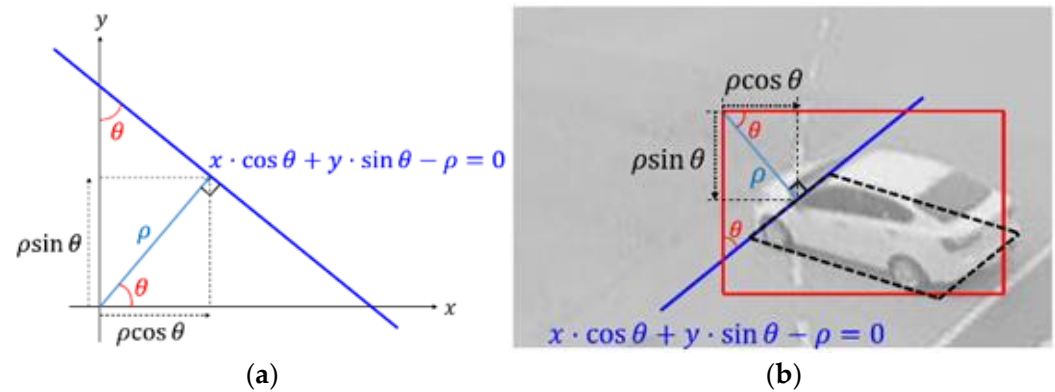


Figure 7. Representation of one side of the BFQ as the normal form of a linear equation: (a) The normal form of a linear equation for the blue line that the angle with the x-axis is θ and the normal distance from the origin is ρ ; (b) the representation of the blue line of the black BFQ as (a).

3.4.2. Intersection Point-Based Method

A straight line obtained by extending a line segment of the vehicle BFQ intersects the edge of the bounding box at two points. Thus, there are eight intersection points as the green circles where the BFQ lines (green lines) meet the red bounding box, as shown in Figure 8. In other words, if the two intersection points can be found on the edges of a bounding box, a straight line of the BFQ is obtained by connecting the two intersection points. The remaining lines of the BFQ are found in the same way. As shown in the figure, two adjacent lines of the BFQ intersect at one point, which is the corner of the BFQ. Therefore, the BFQ can be detected by finding the total number of eight intersection points at the edges of the bounding box.

Intersection point detection can be conducted in two ways. The first one is to find the index of the edge with the intersection point and the offset from a start point to the intersection point. The other one is to find only the offsets from the fixed start point.

The first one uses different start points depending on the intersection point. If the edge index is given from the network, the start point is determined as the point with the smaller coordinate of the two ends of the edge. In Figure 9, the yellow triangles are the start points s_1 and s_2 . The intersection points i_1 and i_2 are at the left and the bottom edges of the bounding box, respectively. If the indices of these edges are obtained from the network, the smaller ends of each edge become the start points. The offset from the start point to the intersection point is the coordinate difference along the x-axis or y-axis:

$$offset = |s_{k,x} - i_{k,x}| + |s_{k,y} - i_{k,y}|, \quad k = 1, \dots, 8, \quad (5)$$

where k is the number of intersection points. The offset is obtained by regressing the ratio r_s of the bounding box size $W \times H$ and the offset which is calculated with the formula:

$$r_s = \frac{|s_{k,x} - i_{k,x}|}{W} + \frac{|s_{k,y} - i_{k,y}|}{H}, \quad k = 1, \dots, 8 \quad (6)$$

and the edge index is obtained by four-class classification. The output of the added extra heads is eight sets of the edge index and the ratio values. This is referred to as L2 in the following.

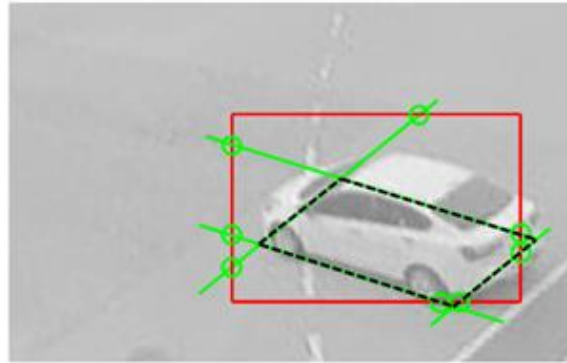


Figure 8. Intersection points of the bounding box and the lines of the BFQ. The red rectangle and the black dashed quadrilateral are the bounding box and the BFQ, respectively. The green lines of the BFQ make the eight intersection points with the red bounding box, as shown by the green circles.

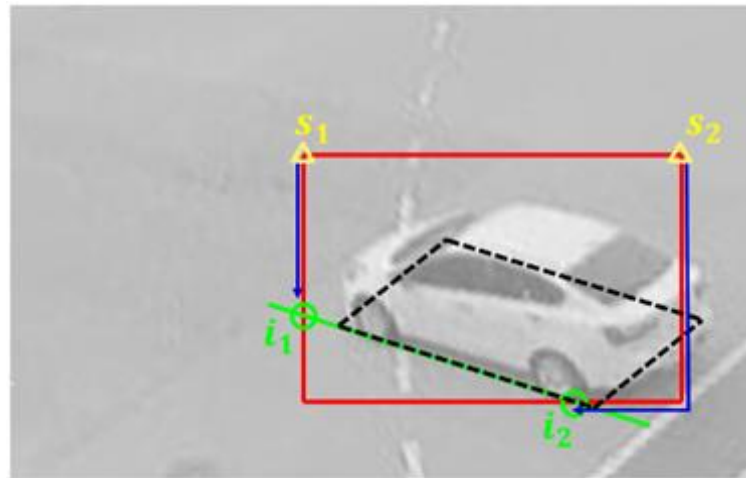


Figure 9. Offset from the corresponding start point to the intersection point. The black dashed quadrilateral indicates the BFQ. The green circles are the intersection points generated by the green line and the red bounding box. The blue arrows depict the offsets from the yellow start points to the corresponding intersection points.

Another way to find an offset is to use the fixed start point as the top left corner of the bounding box, as shown in Figure 10. In this figure, the blue and the light blue lines are the offsets from the start point (yellow triangle) to the intersection points (green circles), respectively. Since the start point is only one, the edge index is not needed; only the offsets are needed. The offset is obtained by regressing the ratio of the offset to the perimeter of the bounding box and the ratio is calculated with the formula:

$$r_s = \frac{|s - i_{k,x}| + |s - i_{k,y}|}{2(W + H)}, \quad k = 1, \dots, 8 \quad (7)$$

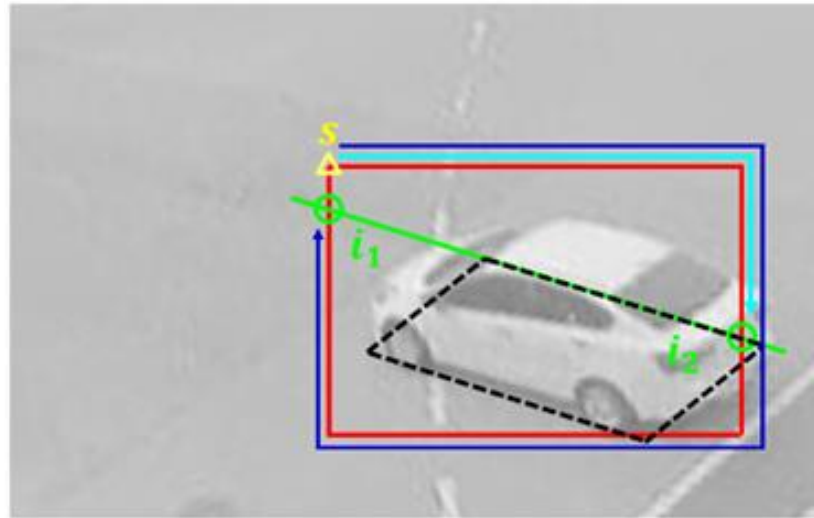


Figure 10. Offset from the fixed start point to the intersection point. The yellow triangle indicates the start point at the top left corner of the bounding box. The green circles are the intersection points of the green line of the BFQ and the red bounding box. The offsets from the start point to the intersection points i_1 and i_2 are indicated by the blue and light blue lines, respectively.

Thus, the added extra heads output eight ratio values. This is referred to as L3 in the following.

4. Experiments

4.1. Dataset and Training

To train and test the presented methods, images captured by surveillance cameras were used. Figure 11 shows example images. It can be noticed that cameras were located at various places with different angles. The images were acquired under various lighting conditions and included multiple types of vehicles, such as sedans, vans, trucks, and buses. The training data include 6591 images with 17,249 vehicles, and the test data consist of 1622 images with 4235 vehicles. Vehicle BFQ labels were manually designated by using camera calibration results.

All networks were optimized using an ADAM optimizer, whose learning rate starts at 10^{-5} and reduces using a cosine annealing schedule. Each network was trained for 200 epochs, and the batch size was set to 16. All the experiments were conducted using TensorFlow and NVIDIA TITAN RTX.

4.2. Evaluation Criteria

For the evaluation of the BFQ detection accuracy, we relied on the Euclidean distance between the ground truth corners and the detected BFQ corners. However, it's worth noting that the vehicle size in the image can vary based on their real-world dimensions or their distance from the camera. For example, Figure 12 shows that the Euclidean errors in one corner of the BFQ remains the same despite the different sizes of the vehicles. The ground truth is represented in red, while the predictions are in green. The Euclidean error of a corner, e_d , can be calculated using:

$$e_d = \sqrt{e_x^2 + e_y^2}, \quad (8)$$

where e_x and e_y are the Euclidean errors along the x -axis and y -axis, respectively.

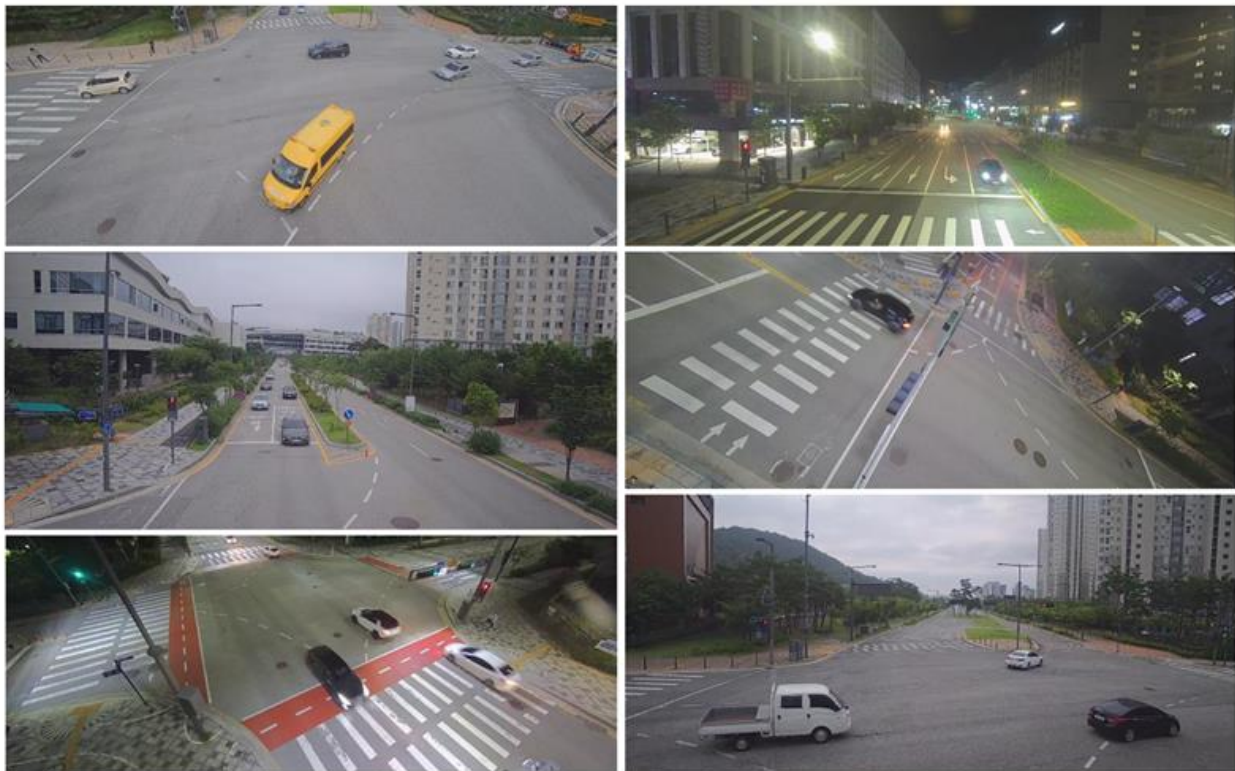


Figure 11. Example images used in the experiment.

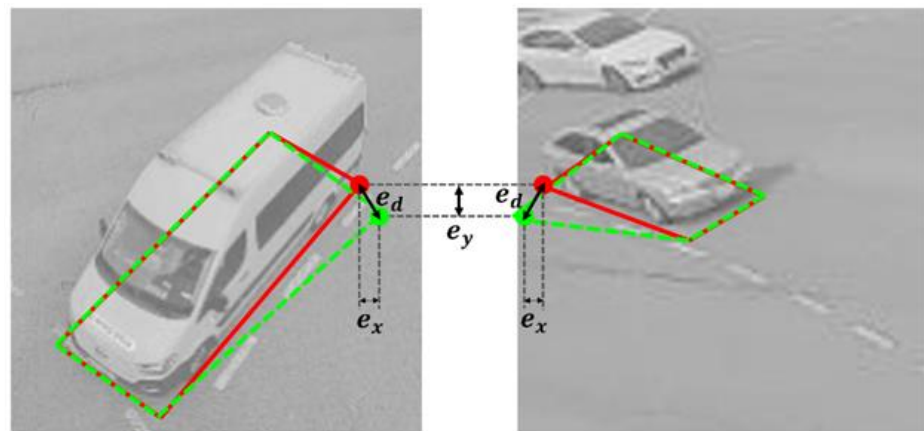


Figure 12. The Euclidean errors affected by the vehicle size in the image. The red and green BFQs are the ground truths and the predictions, respectively. e_x and e_y are the Euclidean errors along the x - and y -axis, respectively. e_d is the Euclidean error for a corner of the BFQ.

Although e_d is the same due to the same e_x and e_y in Figure 12, the detection classification results need to be different. In this figure, two BFQs with non-perfect shapes are detected. On the left image, the difference between the ground truth BFQ and the detected BFQ is not serious. Conversely, on the right image, this difference is considered as being significant because the vehicle size in the image is smaller. Therefore, only the BFQ on the left image should be classified as correctly detected. To overcome this problem, we normalize the Euclidean errors using the vehicle size. The average of the normalized Euclidean errors of the four corners indicates the BFQ detection accuracy of one vehicle

and it is referred to as the position error. The calculation of the average position error over the whole dataset is given by:

$$e = \frac{1}{N} \sum_{i=1}^N \frac{1}{4} \sum_{j=1}^4 \sqrt{\left(\frac{x_{ij} - \hat{x}_{ij}}{W}\right)^2 + \left(\frac{y_{ij} - \hat{y}_{ij}}{H}\right)^2}, \quad (9)$$

where N is the total number of vehicles. (x_{ij}, y_{ij}) and $(\hat{x}_{ij}, \hat{y}_{ij})$ are the coordinates of the ground truth corner and the predicted corner, respectively. e is the average position error when the bounding box size of the detected vehicle is $W \times H$. This means the error of the bounding box normalized to 1×1 pixels.

In addition to the Euclidean distance-based measure, the F1 score is also used to evaluate the BFQ detection performance. The F1 score takes into account the normalized error to determine whether a prediction is correct and combines precision and recall using Equation (10), where precision and recall are computed by Equations (11) and (12), respectively. To be considered a correctly detected BFQ, the average position error must be less than the threshold.

$$F1_{score} = 2 \times \frac{precision \times recall}{precision + recall} \quad (10)$$

$$precision = \frac{\text{the number of correctly detected bottom faces}}{\text{the number of detected bottom faces}} \quad (11)$$

$$recall = \frac{\text{the number of correctly detected bottom faces}}{\text{the number of ground truth bottom faces}} \quad (12)$$

4.3. Evaluation Results

Table 3 shows the average precision (AP) of the vehicle bounding boxes detected by each method. As shown in the table, all methods showed similar levels of bounding box AP between 87% and 91%.

Table 3. Vehicle AP comparison of all BFQ detection methods.

Approach	Method	Vehicle Bounding Box AP (%)
Corner-based	C1	89.42
	C2	88.36
	C3	89.46
	C4	89.34
PSA-based	PSA1	90.28
	PSA2	90.63
	PSA3	90.11
	PSA4	90.36
Line-based	L1	87.98
	L2	87.45
	L3	87.66

Table 4 shows the F1 score, precision, recall, and the average position error evaluated by each method. The F1 score, precision, and recall indicate how many BFQs are detected, and the average position error indicates how accurately they are detected. The final decision for detection is based on two thresholds: 0.2 for strict detection and 0.3 for loose detection. Regardless of the threshold, the L3 method of the line-based approach has the smallest average position error. The C2 of the corner-based approach is in second place.

Table 4. Comparison of BFQ detection results. The detection rate was evaluated through the F1 score, precision, and recall, and detection accuracy was evaluated through the average position error.

Approach	Method	Vehicle BFQ Detection							
		Using Strict Threshold				Using Loose Threshold			
		F1 Score	Precision	Recall	Average Position Error	F1 Score	Precision	Recall	Average Position Error
Corner-based	C1	0.84	0.88	0.81	0.0521	0.92	0.96	0.88	0.0641
	C2	0.86	0.90	0.83	0.0509	0.92	0.96	0.89	0.0614
	C3	0.85	0.88	0.81	0.0515	0.92	0.96	0.89	0.0641
	C4	0.86	0.90	0.83	0.0513	0.92	0.96	0.89	0.0617
PSA-based	PSA1	0.82	0.85	0.78	0.0602	0.89	0.93	0.86	0.0727
	PSA2	0.84	0.88	0.80	0.0605	0.91	0.95	0.87	0.0726
	PSA3	0.82	0.86	0.78	0.0604	0.90	0.95	0.86	0.0739
	PSA4	0.83	0.87	0.79	0.0615	0.91	0.95	0.87	0.0744
Line-based	L1	0.73	0.77	0.68	0.0533	0.83	0.88	0.78	0.0703
	L2	0.75	0.80	0.70	0.0526	0.83	0.89	0.79	0.0670
	L3	0.50	0.54	0.48	0.0389	0.63	0.67	0.60	0.0586

As seen in Table 4, the methods belonging to the same approach show similar performance. However, there is a notable difference in the performance of each approach. The corner-based approach has small average position errors and, at the same time, high F1 scores. The F1 scores of the PSA-based approach are comparable to those of the corner-based approach, but the average position errors are the largest of the three approaches. Finally, L1 and L2 of the line-based approach have lower F1 scores than the other two approaches but have smaller errors than the PSA-based approach. While L3 provides the smallest error of all the methods, it had the lowest F1 score.

4.4. Result Analysis

Figure 13 depicts the detection results of each approach, illustrating the remarkable differences in their performance. To provide a clear comparison, one representative method was selected for each approach, and the detection results of the selected method are shown in this figure. The red and green dashed quadrilaterals represent the ground truth and the detected BFQs by each approach. The images were taken at different times and locations, mostly on roads with intersections and varying driving directions. Despite these differences, the vehicle BFQs were successfully detected in each case. Furthermore, as shown in the right column of Figure 13, the detections were successful even when the vehicle was far away and appeared small.

However, the approaches were not always successful in detecting BFQs, as demonstrated in Figure 14. Figure 14 shows a case where a corner-based approach detects the BFQ with the non-perfect shape due to a significant error at one corner. This approach identifies BFQs by detecting four corners, so a failure to detect even one corner can result in a significant difference between the detected BFQ and the actual one, or worse, the detection may fail altogether.

The PSA-based approach identifies the position, the size, and the angle to detect the BFQ, assuming that the BFQ is a parallelogram. However, in the dataset, the BFQ can have different shapes depending on the vehicle's pose, leading to errors in the parallelogram assumption in most cases. Figure 15 illustrates an example where the ground truth BFQ (red) looks more similar to a rhombus than a parallelogram. Nevertheless, the detected BFQ (green dashed) is a parallelogram due to the assumption. This is an example of the limitations of the parallelogram assumption, which is dependent on the camera angle. In Figure 16, there is a case where the PSA-based approach fails to detect one of the three elements. Although the position and the size are accurately detected, the detected BFQ

shows a large difference from the actual one due to the error in the detected angle. This can lead to incorrect vehicle direction estimation based on the shape of the BFQ.

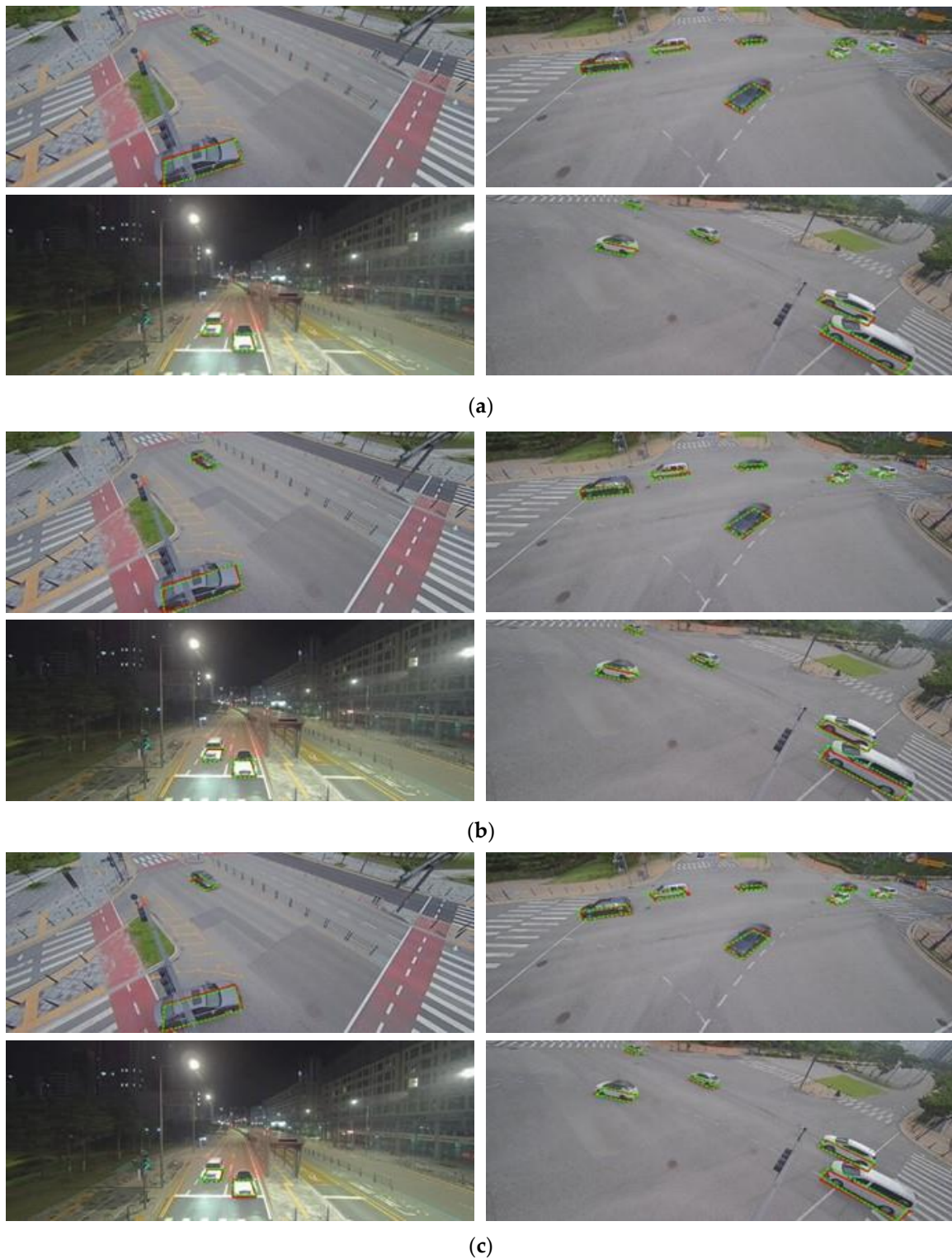


Figure 13. Detection results of the representative methods selected for each approach. The red and the green dashed quadrilaterals represent the ground truth and the detected BFQs, respectively: (a) the detected BFQs using the C2 method; (b) the detected BFQs using the PSA2 method; and (c) the detected BFQs using the L2 method.

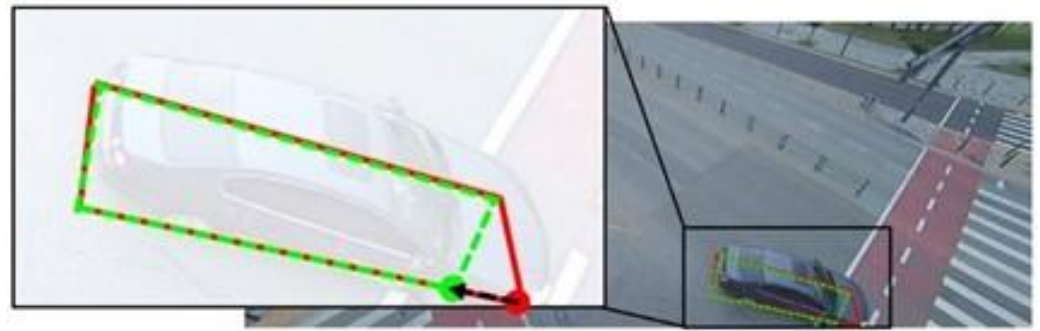


Figure 14. The detected BFQ having a significant error at one corner when using a corner-based approach. The red solid lines and the green dashed lines represent the ground truth and the detected BFQs, respectively. To effectively show the difference between the ground truth and the detected corners, it is indicated by the arrow.



Figure 15. The limitations of the parallelogram assumption of the PSA-based approach. The red and the green dashed quadrilaterals represent the ground truth and the detected BFQs, respectively.

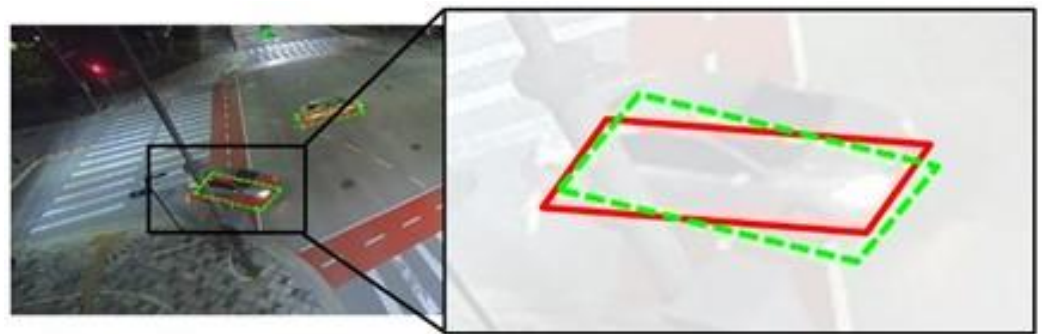


Figure 16. The effect of a single element error on BFQ detection using the PSA-based approach. The red solid lines and the green dashed lines represent the ground truth and the detected BFQs, respectively.

The line-based approach detects the BFQ by finding its four straight lines. The L1 method of this approach identifies the four sets of the line parameters $(\cos \theta, \sin \theta, \rho)$ for a single vehicle. Figure 17 shows the result when one of the 12 parameters fails to be detected. As shown in this figure, a small error in the $\sin \theta$ parameter among $(\cos \theta, \sin \theta, \rho)$ for the left straight line can cause a significant difference in the detected BFQ from the actual one. The remaining methods of the line-based approach identify the intersection points of the bounding box with the lines of the BFQ. Specifically, the L2 method finds the edge index and the ratio for each intersection point. In Figure 18, the detected BFQ resulting from an intersection point with a misidentified edge index (green circle) differs significantly from the actual BFQ. If the edge index is correctly identified, the intersection point would have been at the blue circle, resulting in the blue dashed BFQ. Although the blue BFQ has an error, it can still be considered a correct detection. In contrast, the green BFQ cannot be considered valid because its average position error exceeds the threshold. The importance

of correctly identifying the edge index for each intersection point is illustrated in Figure 18, as it restricts the intersection point's location in this method.

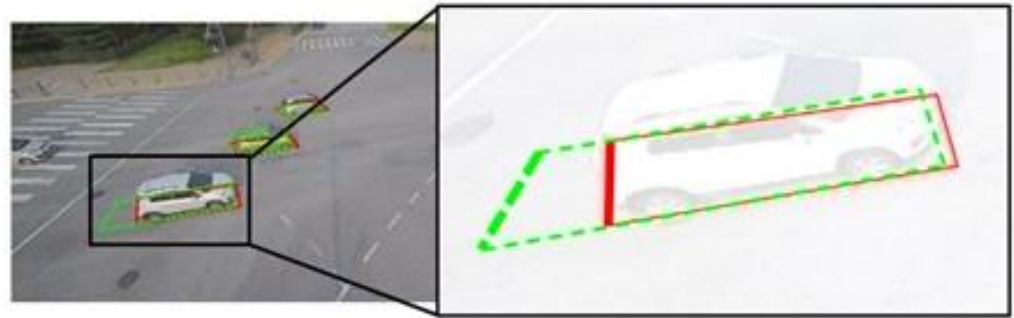


Figure 17. The effect of the small error of a single element when the L1 method is used to detect the BFQ. The red solid and the green dashed quadrilaterals represent the ground truth and the detected BFQs, respectively.

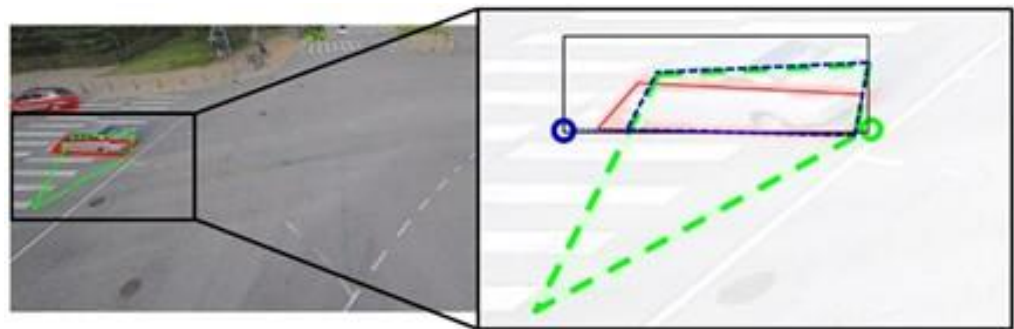


Figure 18. The effect of the misidentified edge index when the L2 method is used to detect the BFQ. The red quadrilateral is the ground truth BFQ, and the black rectangle is the bounding box. The green dashed quadrilateral indicates the BFQ resulting from the intersection point (green circle) where the edge index is misidentified. The blue dashed quadrilateral indicates the BFQ resulting from the intersection point (blue circle) assumed that the edge index is correctly identified.

Figure 19 shows the intersection points caused by the ratio prediction errors and the resulting BFQs, indicated by the circles and quadrilaterals. In Figure 19a, the green BFQ results are from the intersection point (green circle) where the edge index is correctly predicted, but the ratio is incorrect. The intersection point location in the L2 method is limited to the width or the height of the bounding box. On the other hand, the L3 method allows the intersection point to be located at any edge of the bounding box. However, this becomes a limitation for intersection points near the corners of the bounding box, as shown in Figure 19b. The predicted intersection point (green circle) in this figure is near the top right corner of the black bounding box. It should be at the top edge, but it is located at the right edge due to an error in the ratio. This illustrates that the creation of a quadrilateral is easily affected by errors in the predicted ratio. The L3 method has many false detections and a low F1 score. However, the results with significant errors are filtered out, resulting in increased accuracy.



Figure 19. The BFQs resulting from the intersection point with the errors in the predicted ratio. The red circle, line, and quadrilateral are the intersection point, the extended line of an edge, and the BFQ of the ground truth, respectively. The green circle, line, and quadrilateral are the same as the detected BFQ. The bounding box is shown as the black rectangle: (a) result of the L2 method; and (b) result of the L3 method.

5. Conclusions

This paper presents a guideline for accurately detecting the vehicle BFQ in surveillance environments. The proposed method involves implementing BFQ detectors by adding extra heads to YOLO and evaluating three approaches: corner-based, PSA-based, and line-based. The corner-based, PSA-based, and line-based approaches encode the BFQ into four corners, a combination of position/size/angle, and four lines, respectively. The quantitative evaluation shows that the corner-based approach has relatively low errors and high detection rates because the position errors of the BFQ corners have less impact on the overall shape of the BFQ compared to the other approaches. On the other hand, the PSA-based approach showed limitations with potential position errors due to the parallelogram assumption of the BFQ, and the line-based approach showed low detection rates because line estimation errors heavily affect the shapes of the quadrilaterals. Since this paper suggests an approach that detects the BFQ for vehicle position estimation by simply adding extra heads to the existing object detector, it is expected to be useful for those needing practical and accurate vehicle detection systems. In the future, we plan to optimize the proposed network by using channel pruning and quantization-aware training to apply it to a real-time embedded system with a neural processing unit (NPU).

Author Contributions: Conceptualization, G.K., H.G.J. and J.K.S.; software, G.K.; validation, G.K., H.G.J. and J.K.S.; investigation, G.K.; formal analysis, G.K., H.G.J. and J.K.S.; writing—original draft preparation, G.K.; writing—review and editing, H.G.J. and J.K.S.; supervision, H.G.J.; project administration, J.K.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2022R1F1A1074708), and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03038540).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yang, Z.; Pun-Cheng, L.S. Vehicle detection in intelligent transportation systems and its applications under varying environments: A review. *Image Vis. Comput.* **2018**, *69*, 143–154. [[CrossRef](#)]
2. Yu, H.; Luo, Y.; Shu, M.; Huo, Y.; Yang, Z.; Shi, Y.; Guo, Z.; Li, H.; Hu, X.; Yuan, J. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3D object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 21361–21370.
3. Zwemer, M.; Scholte, D.; Wijnhoven, R. 3D Detection of Vehicles from 2D Images in Traffic Surveillance. In Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Online, 6–8 February 2022; pp. 97–106.

4. Chen, Y.; Liu, F.; Pei, K. Monocular Vehicle 3D Bounding Box Estimation Using Homography and Geometry in Traffic Scene. In Proceedings of the ICASSP 2022–2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Singapore, 23–27 May 2022; pp. 1995–1999.
5. Zhu, M.; Zhang, S.; Zhong, Y.; Lu, P.; Peng, H.; Lenneman, J. Monocular 3D vehicle detection using uncalibrated traffic cameras through homography. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 3814–3821.
6. Li, P.; Zhao, H.; Liu, P.; Cao, F. Rtm3d: Real-time monocular 3D detection from object keypoints for autonomous driving. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part III 16, 2020; pp. 644–660.
7. Gähler, N.; Wan, J.-J.; Weber, M.; Zöllner, J.M.; Franke, U.; Denzler, J. Beyond bounding boxes: Using bounding shapes for real-time 3D vehicle detection from monocular rgb images. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 675–682.
8. Barabanau, I.; Artemov, A.; Burnaev, E.; Murashkin, V. Monocular 3D object detection via geometric reasoning on keypoints. *arXiv* **2019**, arXiv:1905.05618.
9. Yang, W.; Li, Z.; Wang, C.; Li, J. A multi-task Faster R-CNN method for 3D vehicle detection based on a single image. *Appl. Soft Comput.* **2020**, *95*, 106533. [[CrossRef](#)]
10. Huang, S.; Cai, G.; Wang, Z.; Xia, Q.; Wang, R. SSA3D: Semantic Segmentation Assisted One-Stage Three-Dimensional Vehicle Object Detection. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 14764–14778. [[CrossRef](#)]
11. Yin, T.; Zhou, X.; Krahenbuhl, P. Center-based 3D object detection and tracking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 11784–11793.
12. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3D object detection network for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1907–1915.
13. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S.L. Joint 3D proposal generation and object detection from view aggregation. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
14. Chen, X.; Kundu, K.; Zhu, Y.; Berneshawi, A.G.; Ma, H.; Fidler, S.; Urtasun, R. 3D object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems (NIPS)*; NIPS: San Diego, CA, USA, 2015; pp. 424–432.
15. Wang, C.; Musaev, A. Preliminary research on vehicle speed detection using traffic cameras. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3820–3823.
16. Giannakeris, P.; Kaltsa, V.; Avgerinakis, K.; Briassoulis, A.; Vrochidis, S.; Kompatsiaris, I. Speed estimation and abnormality detection from surveillance cameras. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 93–99.
17. Gupta, I.; Rangesh, A.; Trivedi, M. 3D Bounding Boxes for Road Vehicles: A One-Stage, Localization Prioritized Approach using Single Monocular Images. In Proceedings of the European Conference on Computer Vision (ECCV) Workshops, Munich, Germany, 8–14 September 2018.
18. Kim, G.; Jung, H.G.; Suhr, J.K. Improvement of vehicle position estimation using CNN-based vehicle bottom face center detection. *Trans. Korean Soc. Automot. Eng.* **2022**, *30*, 599–607. [[CrossRef](#)]
19. Ming, Y.; Meng, X.; Fan, C.; Yu, H. Deep learning for monocular depth estimation: A review. *Neurocomputing* **2021**, *438*, 14–33. [[CrossRef](#)]
20. Qin, Z.; Wang, J.; Lu, Y. Monogrnet: A geometric reasoning network for monocular 3D object localization. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; pp. 8851–8858.
21. Zhang, Y.; Lu, J.; Zhou, J. Objects are different: Flexible monocular 3D object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 3289–3298.
22. Carrillo, J.; Waslander, S. Urbannet: Leveraging urban maps for long range 3D object detection. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; pp. 3799–3806.
23. Rui, Z.; Zongyuan, G.; Simon, D.; Sridha, S.; Clinton, F. Geometry-constrained car recognition using a 3D perspective network. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 1161–1168.
24. Xinyao, T.; Huansheng, S.; Wei, W.; Chunhui, Z. CenterLoc3D: Monocular 3D Vehicle Localization Network for Roadside Surveillance Cameras. *arXiv* **2022**, arXiv:2203.14550.
25. Chabot, F.; Chaouch, M.; Rabarisoa, J.; Teuliere, C.; Chateau, T. 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, Honolulu, HI, USA, 21–26 July 2017; pp. 2040–2049.
26. Wu, Y.; Jiang, X.; Fang, Z.; Gao, Y.; Fujita, H. Multi-modal 3D object detection by 2d-guided precision anchor proposal and multi-layer fusion. *Appl. Soft Comput.* **2021**, *108*, 107405. [[CrossRef](#)]
27. Brazil, G.; Liu, X. M3d-rpn: Monocular 3D region proposal network for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 9287–9296.
28. Li, B.; Ouyang, W.; Sheng, L.; Zeng, X.; Wang, X. Gs3d: An efficient 3D object detection framework for autonomous driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 1019–1028.

29. Mauri, A.; Khemmar, R.; Decoux, B.; Haddad, M.; Boutteau, R. Real-time 3D multi-object detection and localization based on deep learning for road and railway smart mobility. *J. Imaging* **2021**, *7*, 145. [\[CrossRef\]](#)
30. Mauri, A.; Khemmar, R.; Decoux, B.; Haddad, M.; Boutteau, R. Lightweight convolutional neural network for real-time 3D object detection in road and railway environments. *J. Real Time Image Process.* **2022**, *19*, 499–516. [\[CrossRef\]](#)
31. Weber, M.; Fürst, M.; Zöllner, J.M. Direct 3D detection of vehicles in monocular images with a cnn based 3D decoder. In Proceedings of the 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 417–423.
32. Jiaojiao, F.; Linglao, Z.; Guizhong, L. Monocular 3D Detection for Autonomous Vehicles by Cascaded Geometric Constraints and Depurated Using 3D Results. In Proceedings of the 2020 3rd International Conference on Unmanned Systems (ICUS), Harbin, China, 27–28 November 2020; pp. 954–959.
33. Mousavian, A.; Anguelov, D.; Flynn, J.; Kosecka, J. 3D bounding box estimation using deep learning and geometry. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 7074–7082.
34. Liu, Z.; Wu, Z.; Tóth, R. Smoke: Single-stage monocular 3D object detection via keypoint estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 996–997.
35. Wang, T.; Zhu, X.; Pang, J.; Lin, D. Fcos3d: Fully convolutional one-stage monocular 3D object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 913–922.
36. Bao, W.; Xu, B.; Chen, Z. Monofenet: Monocular 3D object detection with feature enhancement networks. *IEEE Trans. Image Process.* **2019**, *29*, 2753–2765. [\[CrossRef\]](#) [\[PubMed\]](#)
37. Wu, Z.; Jiang, X.; Xu, R.; Lu, K.; Zhu, Y.; Wu, M. DST3D: DLA-Swin Transformer for Single-Stage Monocular 3D Object Detection. In Proceedings of the 2022 IEEE Intelligent Vehicles Symposium (IV), Aachen, Germany, 4–9 June 2022; pp. 411–418.
38. Kocur, V.; Ftáčnik, M. Detection of 3D bounding boxes of vehicles using perspective transformation for accurate speed measurement. *Mach. Vis. Appl.* **2020**, *31*, 62. [\[CrossRef\]](#)
39. Gährlert, N.; Wan, J.-J.; Jourdan, N.; Finkbeiner, J.; Franke, U.; Denzler, J. Single-shot 3D detection of vehicles from monocular rgb images via geometrically constrained keypoints in real-time. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 437–444.
40. Zhang, B.; Zhang, J. A traffic surveillance system for obtaining comprehensive information of the passing vehicles based on instance segmentation. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 7040–7055. [\[CrossRef\]](#)
41. Zhu, J.; Li, X.; Zhang, C.; Shi, T. An accurate approach for obtaining spatiotemporal information of vehicle loads on bridges based on 3D bounding box reconstruction with computer vision. *Measurement* **2021**, *181*, 109657. [\[CrossRef\]](#)
42. Sang, J.; Wu, Z.; Guo, P.; Hu, H.; Xiang, H.; Zhang, Q.; Cai, B. An Improved YOLOv2 for Vehicle Detection. *Sensors* **2018**, *18*, 4272. [\[CrossRef\]](#)
43. Reddy, N.D.; Vo, M.; Narasimhan, S.G. Occlusion-net: 2D/3D occluded keypoint localization using graph networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 7326–7335.
44. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as points. *arXiv* **2019**, arXiv:1904.07850 2019.
45. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
46. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2961–2969.
47. Gährlert, N.; Mayer, M.; Schneider, L.; Franke, U.; Denzler, J. Mb-net: Mergeboxes for real-time 3D vehicles detection. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 2117–2124.
48. Zhang, X.; Feng, Y.; Angeloudis, P.; Demiris, Y. Monocular visual traffic surveillance: A review. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 14148–14165. [\[CrossRef\]](#)
49. Bradler, H.; Kretz, A.; Mester, R. Urban Traffic Surveillance (UTS): A fully probabilistic 3D tracking approach based on 2D detections. In Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–17 July 2021; pp. 1198–1205.
50. Ahmed, I.; Jeon, G.; Chehri, A. A Smart IoT Enabled End-to-End 3D Object Detection System for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 1–365. [\[CrossRef\]](#)
51. Arnold, E.; Al-Jarrah, O.Y.; Dianati, M.; Fallah, S.; Oxtoby, D.; Mouzakitis, A. A survey on 3D object detection methods for autonomous driving applications. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3782–3795. [\[CrossRef\]](#)
52. Liu, M.; Wang, X.; Zhou, A.; Fu, X.; Ma, Y.; Piao, C. Uav-yolo: Small object detection on unmanned aerial vehicle perspective. *Sensors* **2020**, *20*, 2238. [\[CrossRef\]](#)
53. Hong, S.-J.; Han, Y.; Kim, S.-Y.; Lee, A.-Y.; Kim, G. Application of deep-learning methods to bird detection using unmanned aerial vehicle imagery. *Sensors* **2019**, *19*, 1651. [\[CrossRef\]](#) [\[PubMed\]](#)
54. Du, G.; Wang, K.; Lian, S.; Zhao, K. Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review. *Artif. Intell. Rev.* **2021**, *54*, 1677–1734. [\[CrossRef\]](#)
55. YOLOv4. Available online: https://docs.nvidia.com/tao/tao-toolkit/text/object_detection/yolo_v4.html (accessed on 6 March 2023).
56. Getting Started with YOLO V4. Available online: <https://kr.mathworks.com/help/vision/ug/getting-started-with-yolo-v4.html> (accessed on 6 March 2023).

57. Wang, C.-Y.; Liao, H.-Y.M.; Wu, Y.-H.; Chen, P.-Y.; Hsieh, J.-W.; Yeh, I.-H. CSPNet: A new backbone that can enhance learning capability of CNN. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, Seattle, WA, USA, 14–19 June 2020; pp. 390–391.
58. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [[CrossRef](#)] [[PubMed](#)]
59. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
60. Ren, X.; Wang, X.; Wang, P. Vehicle Positioning Method of Roadside Monocular Camera. In Proceedings of the 2020 IEEE MTT-S International Wireless Symposium (IWS), Shanghai, China, 20–23 September 2020; pp. 1–3.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.