Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving

Jiwoong Choi[1], Dayoung Chun[1], Hyun Kim[2], and Hyuk-Jae Lee[1]
[1]Seoul National University, [2]Seoul National University of Science and Technology
{jwchoi, jjeonda}@capp.snu.ac.kr, hyunkim@seoultech.ac.kr, hjlee@capp.snu.ac.kr

Abstract

The use of object detection algorithms is becoming increasingly important in autonomous vehicles, and object detection at high accuracy and a fast inference speed is essential for safe autonomous driving. A false positive (FP) from a false localization during autonomous driving can lead to fatal accidents and hinder safe and efficient driving. Therefore, a detection algorithm that can cope with mislocalizations is required in autonomous driving applications. This paper proposes a method for improving the detection accuracy while supporting a real-time operation by modeling the bounding box (bbox) of YOLOv3, which is the most representative of one-stage detectors, with a Gaussian parameter and redesigning the loss function. In addition, this paper proposes a method for predicting the localization uncertainty that indicates the reliability of bbox. By using the predicted localization uncertainty during the detection process, the proposed schemes can significantly reduce the FP and increase the true positive (TP), thereby improving the accuracy. Compared to a conventional YOLOv3, the proposed algorithm, Gaussian YOLOv3, improves the mean average precision (mAP) by 3.09 and 3.5 on the KITTI and Berkeley deep drive (BDD) datasets, respectively. Nevertheless, the proposed algorithm is capable of real-time detection at faster than 42 frames per second (fps) and shows a higher accuracy than previous approaches with a similar fps. Therefore, the proposed algorithm is the most suitable for autonomous driving applications.

1. Introduction

In recent years, deep learning has been actively applied in various fields including computer vision [9], autonomous driving [5], and social network services [15]. The development of sensors and GPU along with deep learning algorithms has accelerated research into autonomous vehicles based on artificial intelligence. An autonomous vehicle with self-driving capability without a driver intervention must accurately detect cars, pedestrians, traffic signs, traffic lights, etc. in real time to ensure safe and correct control decisions [25]. To detect such objects, various sensors such as cameras, light detection and ranging (Lidar), and radio detection and ranging (Radar) are generally used in autonomous vehicles [27]. Among these various types of sensors, a camera sensor can accurately identify the object type based on texture and color features and is more cost-effective [24] than other sensors. In particular, deep-learning based object detection using camera sensors is becoming more important in autonomous vehicles because it achieves a better level of accuracy than humans in terms of object detection, and consequently it has become an essential method [11] in autonomous driving systems.

An object detection algorithm for autonomous vehicles should satisfy the following two conditions. First, a high detection accuracy of the road objects is required. Second, a real-time detection speed is essential for a rapid response of a vehicle controller and a reduced latency. Deep-learning based object detection algorithms, which are indispensable in autonomous vehicles, can be classified into two categories: two-stage and one-stage detectors. Two-stage detectors, *e.g.*, Fast R-CNN [8], Faster R-CNN [22], and R-FCN [4], conduct a first stage of region proposal generation, followed by a second stage of object classification and bbox regression. These methods generally show a high accuracy but have a disadvantage of a slow detection speed and lower efficiency. One-stage detectors, *e.g.*, SSD [17] and YOLO [19], conduct object classification and bbox regression concurrently without a region proposal stage. These methods generally have a fast detection speed and high efficiency but a low accuracy. In recent years, to take advantage of both types of method and to compensate for their respective disadvantages, object detectors combining various schemes have been widely studied [1, 11, 29, 28, 16]. MS-CNN [1], a two-stage detector, improves the detection speed by conducting detection on various intermediate network layers. SINet [11], also a two-stage detector, enables a fast detection using a scale-insensitive network. CFENet [29], a one-stage detector, uses a comprehensive feature enhance-

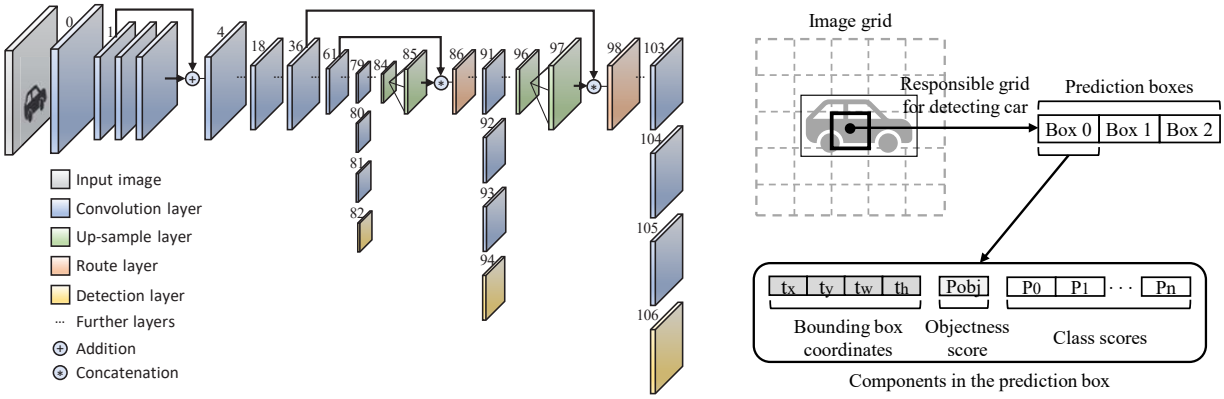
ment module based on SSD to improve the detection accuracy. RefineDet [28], also a one-stage detector, improves the detection accuracy by applying an anchor refinement module and an object detection module. Another one-stage detector, RFBNet [16], applies a receptive field block to improve the accuracy. However, using an input resolution of 512×512 or higher, which is widely applied in object detection algorithms for achieving a high accuracy, previous studies [1, 11, 29, 28] have been unable to meet a real-time detection speed of above 30 fps, which is a prerequisite for self-driving applications. Even if real-time detection is possible in [16], it is difficult to apply to autonomous driving due to a low accuracy. This indicates that these previous schemes are incomplete in terms of a trade-off between accuracy and detection speed, and consequently, have a limitation in their application to self-driving systems.

In addition, one of the most critical problems of most conventional deep-learning based object detection algorithms is that, whereas the bbox coordinates (*i.e.*, localization) of the detected object are known, the uncertainty of the bbox result is not. Thus, conventional object detectors cannot prevent mislocalizations (*i.e.*, FPs) because they output the deterministic results of the bbox without information regarding the uncertainty. In autonomous driving, an FP denotes an incorrect detection result of bbox on an object that is not the ground-truth (GT), or an inaccurate detection result of bbox on the GT, whereas a TP denotes an accurate detection result of bbox on the GT. An FP is extremely dangerous under autonomous driving because it causes excessive reactions such as unexpected braking, which can reduce the stability and efficiency of driving and lead to a fatal accident [18, 23] as well as confusion in the determination of an accurate object detection. In other words, it is extremely important to predict the uncertainty of the detected bboxes and to consider this factor along with the objectness score and class scores for reducing the FP and preventing autonomous driving accidents. For this reason, various studies have been conducted on predicting uncertainty in deep learning. Kendall *et al*. [12] proposed a modeling method for uncertainty prediction using a Bayesian neural network in deep learning. Feng *et al*. [6] proposed a method for predicting uncertainty by applying Kendall *et al*. s scheme [12] to 3D vehicle detection using a Lidar sensor. However, the methods proposed by Kendall *et al*. [12] and Feng *et al*. [6] only predict the level of uncertainty, and do not utilize this factor in actual applications. Choi *et al*. [2] proposed a method for predicting uncertainty in real time using a Gaussian mixture model and applied the method to an autonomous driving application. However, it was applied to the steering angle, and not object detection, and a complicated distribution is therefore modeled, increasing the computational complexity. He *et al*. [10] proposed an approach for predicting uncertainty and utilized it toward object detection. However, because they focused on a two-stage detector, their method cannot support a real-time operation, and remaining a bbox overlap problem, so it is unsuitable for self-driving applications.

To overcome the problems of previous object detection studies, this paper proposes a novel object detection algorithm suitable for autonomous driving based on YOLOv3 [21]. YOLOv3 can detect multiple objects with a single inference, and its detection speed is therefore extremely fast; in addition, by applying a multi-stage detection method, it can complement the low accuracy of YOLO [19] and YOLOv2 [20]. Based on these advantages, YOLOv3 is suitable for autonomous driving applications, but generally achieves a lower accuracy than a two-stage method. It is therefore essential to improve the accuracy while maintaining a real-time object detection capability. To achieve this goal, the present paper proposes a method for improving the detection accuracy by modeling the bbox coordinates of YOLOv3, which only outputs deterministic values, as the Gaussian parameters (*i.e.*, the mean and variance), and redesigning the loss function of bbox. Through this Gaussian modeling, a localization uncertainty for a bbox regression task in YOLOv3 can be estimated. Furthermore, to further improve the detection accuracy, a method for reducing the FP and increasing the TP by utilizing the predicted localization uncertainty of bbox during the detection process is proposed. This study is therefore the first attempt to model the localization uncertainty in YOLOv3 and to utilize this factor in a practical manner. As a result, the proposed Gaussian YOLOv3 can cope with mislocalizations in autonomous driving applications. In addition, because the proposed method is modeled only in bbox of the YOLOv3 detection layer (*i.e.*, the output layer), the additional computation cost is negligible, and the proposed algorithm consequently maintains the real-time detection speed of over 42 fps with an input resolution of 512×512 despite the significant improvements in performance. Compared to the baseline algorithm (*i.e.*, YOLOv3), the proposed Gaussian YOLOv3 improves the mAP by 3.09 and 3.5 on the KITTI [7] and BDD [26] datasets, respectively. In addition, the proposed algorithm reduces the FP by 41.40% and 40.62%, respectively, and increases the TP by 7.26% and 4.3%, respectively, on the KITTI and BDD datasets. As a result, in terms of the trade-off between accuracy and detection speed, the proposed algorithm is suitable for autonomous driving because it significantly improves the detection accuracy and addresses the mislocalization problem while supporting a real-time operation.

2. Background

Instead of the region proposal method used in two-stage detectors, YOLO [19] detects objects by dividing an image into grid units. The feature map of the YOLO output layer is



(a) (b)

Figure 1: (a) Network architecture of YOLOv3 and (b) attributes of its prediction feature map.

designed to output bbox coordinates, the objectness score, and the class scores, and thus YOLO enables the detection of multiple objects with a single inference. Therefore, the detection speed is much faster than that of conventional methods. However, owing to the processing of the grid unit, localization errors are large and the detection accuracy is low, and thus it is unsuitable for autonomous driving applications. To address these problems, YOLOv2 [20] has been proposed. YOLOv2 improves the detection accuracy compared to YOLO by using batch normalization for the convolution layer, and applying an anchor box, multi-scale training, and fine-grained features. However, the detection accuracy is still low for small or dense objects. Therefore, YOLOv2 is unsuitable for autonomous driving applications, where a high accuracy is required for dense road objects and small objects such as traffic signs and lights.

To overcome the disadvantages of YOLOv2, YOLOv3 [21] has been proposed. YOLOv3 consists of convolution layers, as shown in Figure 1a, and is constructed of a deep network for an improved accuracy. YOLOv3 applies a residual skip connection to solve the vanishing gradient problem of deep networks and uses an up-sampling and concatenation method that preserves fine-grained features for small object detection. The most prominent feature is the detection at three different scales in a similar manner as used in a feature pyramid network [13]. This allows YOLOv3 to detect objects with various sizes. In more detail, when an image of three channels of R, G, and B is input into the YOLOv3 network, as shown in Figure 1a, information on the object detection (*i.e.*, bbox coordinates, objectness score, and class scores) is output from three detection layers. The predicted results of the three detection layers are combined and processed using non-maximum suppression. After that, the final detection results are determined. Because YOLOv3 is a fully convolutional network consisting only of small-sized convolution filers of 1×1 and 3×3 like YOLOv2 [20], the detection speed is as fast as YOLO [19] and YOLOv2 [20]. Therefore, in terms of the trade-off between accuracy and speed, YOLOv3 is suitable for autonomous driving applications and is widely used in autonomous driving research [3]. However, in general, it still has a lower accuracy than a two-stage detector using a region proposal stage. To compensate for this drawback, as taking advantage of the smaller complexity of YOLOv3 than that of a two-stage detector, a more efficient detector for an autonomous driving application can be designed by applying the additional method for improving accuracy to YOLOv3 [21]. The Gaussian modeling and loss function reconstruction of YOLOv3 proposed in this paper can improve the accuracy by reducing the influence of noisy data during training and predict the localization uncertainty. In addition, the detection accuracy can be further enhanced by using this predicted localization uncertainty. A detailed description of the above aspects is provided in Section 3.

3. Gaussian YOLOv3

3.1. Gaussian modeling

As shown in Figure 1b, the prediction feature map of YOLOv3 [21] has three prediction boxes per grid, where each prediction box consists of bbox coordinates (*i.e.*, t_x, t_y, t_w, and t_h), the objectness score, and class scores. YOLOv3 outputs the objectness (*i.e.*, whether an object is present or not in the bbox) and class (*i.e.*, the category of the object), as a score of between zero and one. An object is then detected based on the product of these two values. Unlike the objectness and class information, bbox coordinates are output as deterministic coordinate values instead of a score, and thus the confidence of the detected bbox is unknown. Moreover, the objectness score does not reflect the reliability of the bbox well. It therefore does not know

how uncertain the result of bbox is. In contrast, the uncertainty of bbox, which is predicted by the proposed method, serves as the bbox score, and can thus be used as an indicator of how uncertain the bbox is. The results for this are described in Section 4.1.

In YOLOv3, bbox regression is to extract the bbox center information (*i.e.*, t_x and t_y) and bbox size information (*i.e.*, t_w and t_h). Because there is only one correct answer (*i.e.*, the GT) for the bbox of an object, complex modeling is not required for predicting the localization uncertainty. In other words, the uncertainty of bbox can be modeled using each single Gaussian model of t_x, t_y, t_w, and t_h. A single Gaussian model of output y for a given test input x whose output consists of Gaussian parameters is as follows:

$$p(y|x) = N(y; \mu(x), \Sigma(x)), \tag{1}$$

where $\mu(x)$ and $\Sigma(x)$ are the mean and variance functions, respectively.

To predict the uncertainty of bbox, each of the bbox coordinates in the prediction feature map is modeled as the mean (μ) and variance (Σ), as shown in Figure 2. The outputs of bbox are $\hat{\mu}_{t_x}$, $\hat{\Sigma}_{t_x}$, $\hat{\mu}_{t_y}$, $\hat{\Sigma}_{t_y}$, $\hat{\mu}_{t_w}$, $\hat{\Sigma}_{t_w}$, $\hat{\mu}_{t_h}$, and $\hat{\Sigma}_{t_h}$. Considering the structure of the detection layer in YOLOv3, the Gaussian parameters for t_x, t_y, t_w, and t_h are preprocessed as follows:

$$\mu_{t_x} = \sigma(\hat{\mu}_{t_x}),\ \mu_{t_y} = \sigma(\hat{\mu}_{t_y}),\ \mu_{t_w} = \hat{\mu}_{t_w},\ \mu_{t_h} = \hat{\mu}_{t_h} \tag{2}$$

$$\begin{aligned} \Sigma_{t_x} &= \sigma(\hat{\Sigma}_{t_x}),\ \Sigma_{t_y} = \sigma(\hat{\Sigma}_{t_y}), \\ \Sigma_{t_w} &= \sigma(\hat{\Sigma}_{t_w}),\ \Sigma_{t_h} = \sigma(\hat{\Sigma}_{t_h}) \end{aligned} \tag{3}$$

$$\sigma(x) = \frac{1}{(1 + exp(-x))}. \tag{4}$$

The mean value of each coordinate in the detection layer is the predicted coordinate of bbox, and each variance represents the uncertainty of each coordinate. μ_{t_x} and μ_{t_y} in (2) must represent the center coordinates of bbox inside the grid, which are thus processed as values between zero and one with the sigmoid function in (4). The variances of each coordinate in (3) are also processed as values between zero and one with a sigmoid function. In YOLOv3, the $width$ and $height$ information of bbox are processed through t_w, t_h, bbox prior, and exponential functions [21]. In other words, μ_{t_w} and μ_{t_h} in (2), which indicate the t_w and t_h of YOLOv3, are not processed as sigmoid functions because they can have both negative and positive values.

Single Gaussian modeling for predicting the uncertainty of bbox only applies to the bbox coordinates of the YOLOv3 detection layer shown in Figure 1a. Therefore, the overall computational complexity of the algorithm does not increase significantly. In a 512 × 512 input resolution and ten classes, YOLOv3 requires 99 × 10^9 FLOPs; however, after a single Gaussian modeling for bbox, 99.04 × 10^9 FLOPs are required. Thus, the penalty for the detection speed is extremely low because the computation cost increases only by 0.04% as compared with before the modeling. The related results are shown in Section 4.

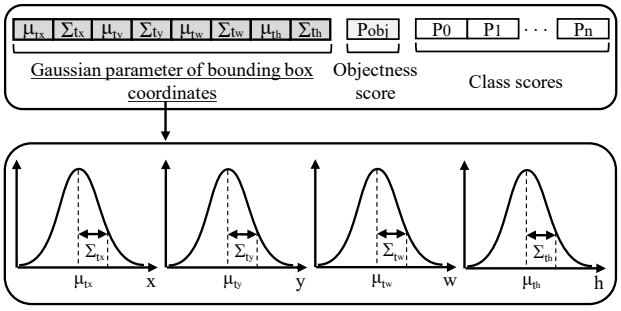


Figure 2: Components in the prediction box of proposed algorithm.

3.2. Reconstruction of loss function

For training, YOLOv3 [21] uses the sum of the squared error loss for bbox, and the binary cross-entropy loss for the objectness and class. Because the bbox coordinates are output as Gaussian parameters through Gaussian modeling, the loss function of bbox is redesigned as a negative log likelihood (NLL) loss, whereas the loss function for objectness and class is not changed. The loss function redesigned for bbox is as follows:

$$\begin{aligned} L_x = -\sum_{i=1}^{W}\sum_{j=1}^{H}\sum_{k=1}^{K} \gamma_{ijk} log(N(x^G_{ijk}|\mu_{t_x}(x_{ijk}), \\ \Sigma_{t_x}(x_{ijk})) + \varepsilon), \end{aligned} \tag{5}$$

where L_x is the NLL loss of t_x coordinate and the others (*i.e.*, L_y, L_w, and L_h) are the same as L_x except for each parameter. W and H are the number of grids of each $width$ and $height$, respectively, and K is the number of anchors. Moreover, $\mu_{t_x}(x_{ijk})$ denotes the t_x coordinates, which is the output of the detection layer of the proposed algorithm, at the k-th anchor in the (i, j) grid. In addition, $\Sigma_{t_x}(x_{ijk})$ is also the output of the detection layer, indicating the uncertainty of t_x coordinate, and x^G_{ijk} is the GT of t_x coordinate. The GT of bbox is then computed as follows:

$$x^G_{ijk} = x^G \times W - i,\ y^G_{ijk} = y^G \times H - j \tag{6}$$

$$w^G_{ijk} = log(\frac{w^G \times IW}{A^w_k}),\ h^G_{ijk} = log(\frac{h^G \times IH}{A^h_k}), \tag{7}$$

where x^G, y^G, w^G, and h^G are the ratios of a GT bbox in an image, IW and IH are the $width$ and $height$ of the resized image, and A^w_k and A^h_k denote the $width$ and $height$ of the k-th anchor box prior, respectively. In YOLOv3, centroid of

bbox is calculated in grid units, and size of bbox is calculated based on an anchor box, and thus the GT is processed accordingly for training.

$$\gamma_{ijk} = \frac{\omega_{scale} \times \delta_{ijk}^{obj}}{2} \tag{8}$$

$$\omega_{scale} = 2 - w^G \times h^G. \tag{9}$$

ω_{scale} in (8) is calculated based on the $width$ and $height$ ratios of the GT bbox in an image, as shown in (9). It provides different weights according to the object size during training. In addition, δ_{ijk}^{obj} in (8) is a parameter applied to include in the loss only if there is an anchor that is most suitable in the current object among the predefined anchors. This parameter is assigned as a value of one when the intersection over union (IOU) of the GT and the *k*-th anchor box in the (*i*, *j*) grid are the largest, and is assigned as a value of zero if there is no appropriate GT. For a numerical stability of the logarithmic function, ε is assigned a value of 10^{-9}.

Because YOLOv3 uses the sum of the squared error loss for bbox, it is unable to cope with noisy data during training. However, the redesigned loss function of bbox can provide a penalty to the loss through the uncertainty for inconsistent data during training. That is, the model can be learned by concentrating on consistent data. Therefore, the redesigned loss function of bbox makes the model more robust to noisy data [12]. Through this loss attenuation [12], it is possible to improve the accuracy of the algorithm.

3.3. Utilization of localization uncertainty

The proposed Gaussian YOLOv3 can obtain the uncertainty of bbox for every detection object in an image. Because it is not an uncertainty for the entire image, it is possible to apply uncertainty to each detection result. YOLOv3 considers only the objectness score and class scores during object detection, and cannot consider the bbox score during the detection process because the score information for the bbox coordinates is unknown. However, Gaussian YOLOv3 can output the localization uncertainty, which is the score of bbox. Therefore, localization uncertainty can be considered along with the objectness score and class scores during the detection process. The proposed algorithm applies localization uncertainty to the detection criteria of YOLOv3 such that bbox with high uncertainty among the predicted results is filtered through the detection process. In this way, predictions with high confidence of objectness, class, and bbox are finally selected. Thus, Gaussian YOLOv3 can reduce the FP and increase the TP, which results in improving the detection accuracy. The proposed detection criterion considering the localization uncertainty is as follows:

$$Cr. = \sigma(Object) \times \sigma(Class_i) \times (1 - Uncertainty_{aver}). \tag{10}$$

$Cr.$ in (10) indicates the detection criterion for Gaussian YOLOv3, $\sigma(Object)$ is the objectness score, and $\sigma(Class_i)$ is the score of the *i*-th class. In addition, $Uncertainty_{aver}$, which is localization uncertainty, indicates the average of the uncertainties of the predicted bbox coordinates. Localization uncertainty has a value between zero and one, such as the objectness score and class scores, and the higher the localization uncertainty, the lower the confidence of the predicted bbox. The results of the proposed Gaussian YOLOv3 are described in Section 4.

4. Experimental Results

In the experiment, the KITTI dataset [7], which is commonly used in autonomous driving research, and the BDD dataset [26], which is the latest published autonomous driving dataset, are used. The KITTI dataset consists of three classes: car, cyclist, and pedestrian, and consists of 7,481 images for training and 7,518 images for testing. Because there is no GT for testing, the training and validation sets are made by randomly splitting the training set in half [25]. The BDD dataset consists of ten classes: bike, bus, car, motor, person, rider, traffic light, traffic sign, train, and truck. The ratio of training, validation, and test set is 7:1:2. In this paper, a test set is utilized for the performance evaluation. In general, the IOU threshold (TH) of the KITTI dataset is set to 0.7 for cars and 0.5 for cyclists and pedestrians [7], whereas the IOU TH of the BDD dataset is 0.75 for all classes [26]. In both YOLOv3 and Gaussian YOLOv3 training, the batch size is 64 and the learning rate is 0.0001. The anchor size is extracted using k-means clustering for each training set of KITTI and BDD. The anchors used in the training and evaluation are shown in Table 1. Other studies are trained using the default settings in the official code of each algorithm. The experiment is conducted on an NVIDIA GTX 1080 Ti with CUDA 8.0 and cuDNN v7.

4.1. Validation in utilizing localization uncertainty

Figure 3 shows the relationship between the IOU and localization uncertainty of bbox for the KITTI and BDD validation sets. These results are plotted for cars, which is the dominant class for all data, and the localization uncertainty is predicted using the proposed algorithm. To show a typical tendency, the IOU is divided increments of 0.1, and the average value of the IOU and the average value of the localization uncertainty are calculated for each range and used as a representative value. As shown in Figure 3, the IOU value tends to increase as the localization uncertainty decreases in both datasets. A larger IOU indicates that the coordinates of the predicted bbox are closer to those of the GT. Based on these results, the localization uncertainty of the proposed algorithm effectively represents the confidence of the predicted bbox. It is therefore possible to cope with mislocalizations and improve the accuracy by utilizing the lo-

	Anchor 0	Anchor 1	Anchor 2
KITTI training set			
First detection layer	(49,240)	(82,170)	(118,206)
Second detection layer	(45,76)	(27,172)	(67,116)
Third detection layer	(13,30)	(23,53)	(17,102)
BDD training set			
First detection layer	(73,175)	(141,178)	(144,291)
Second detection layer	(32,97)	(57,64)	(92,109)
Third detection layer	(7,10)	(14,24)	(27,43)

Table 1: Results of anchor boxes of training sets.

calization uncertainty predicted by the proposed algorithms.

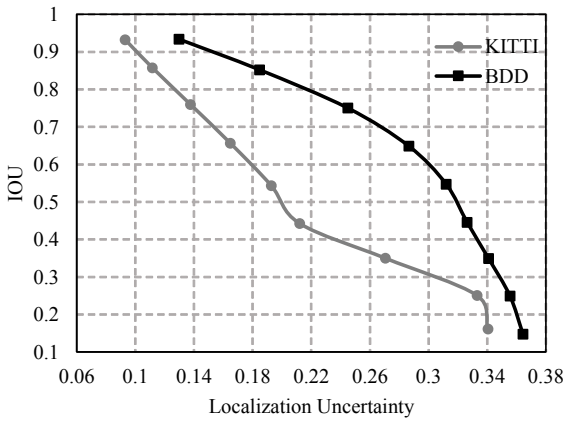


Figure 3: IOU versus localization uncertainty on KITTI and BDD validation sets.

4.2. Performance evaluation of Gaussian YOLOv3

To demonstrate the superiority of the proposed algorithm, its performance (*i.e.*, accuracy and detection speed) is compared with that of other studies [1, 11, 17, 28, 29, 16, 21]. In the experiment on the KITTI validation set, the other studies [1, 11, 17, 28, 16, 21] are trained and evaluated using the official published code of each algorithm. In the case of CFENet [29], the result of the KITTI object detection leader board is used because the official code has not been published. In the experiment on the BDD test data, the results for the BDD test set of SSD [17], CFENet [29], and RefineDet [28] are specified in CFENet [29], and thus the simulation results of these studies are from [29], whereas the remaining comparative studies [1, 11, 16, 21] are trained and evaluated using the official published codes because these studies have not been developed as targets for BDD datasets and therefore have not been evaluated with BDD datasets in previous studies. For a fair comparison of the one-stage detectors, the input resolution is set as in CFENet [29]. The two-stage detector uses the default resolution of each official published code. The official evaluation method of each dataset is used for an accuracy comparison, and IOU TH is set to the value mentioned before. For a comparison of the accuracy, mAP, which has been widely used in previous studies on object detection, is selected.

Table 2 shows the performance of the proposed algorithm and other methods using the KITTI validation set. The mAP of the proposed algorithm, Gaussian YOLOv3, improves by 3.09 compared to that of YOLOv3, and the detection speed is 43.13 fps, which enables real-time detection with a slight difference from YOLOv3. Gaussian YOLOv3 is 3.93 fps faster than that of RFBNet [16], which has the fastest operation speed among the previous studies with the exception of YOLOv3, despite the mAP of Gaussian YOLOv3 outperforming that of RFBNet [16] by more than 10.17. In addition, although the mAP of Gaussian YOLOv3 with a 512 $\times$ 512 resolution is 1.81 lower than that of SINet [11], which has the highest accuracy among the previous methods, it is noteworthy that the fps of the proposed method is 1.8-times better than that of SINet [11]. Because there is a trade-off between the accuracy and detection speed, for a fair comparison, the input resolution of the proposed algorithm is changed and evaluated considering the fps of SINet [11]. The experimental results show that the mAP of Gaussian YOLOv3 with a 704 $\times$ 704 resolution shown in the last row of Table 2 is 86.79 at 24.91 fps, and consequently, Gaussian YOLOv3 outperforms SINet [11] in terms of the accuracy and detection speed.

Table 3 shows the performance of the proposed approach and other methods for the BDD test set. Gaussian YOLOv3 improves the mAP by 3.5 compared with YOLOv3, and the detection speed is 42.5 fps, which is almost the same as YOLOv3. In addition, Gaussian YOLOv3 is 3.5 fps faster than the RFBNet [16], which has the fastest operation speed among the previous studies except for YOLOv3, despite the accuracy of Gaussian YOLOv3 outperforming that of RFBNet [16] by 3.9 mAP. In addition, compared to CFENet [29], which has the highest accuracy among the previous methods, the performance of Gaussian YOLOv3 with a 736 $\times$ 736 input resolution in the last row of Table 3 shows a better mAP of 1.7 and faster operation speed of 1.5 fps, and consequently, Gaussian YOLOv3 outperforms CFENet [29] in terms of the accuracy and detection speed.

Furthermore, on the COCO dataset [14], the AP of Gaussian YOLOv3 is 36.1, which is 3.1 higher than YOLOv3. In particular, the AP_{75} (*i.e.*, strict metric) of Gaussian YOLOv3 is 39.0, which is 4.6 higher than that of YOLOv3. These results indicate that the proposed algorithm outperforms YOLOv3 in general dataset as well as KITTI and BDD.

Based on these experimental results, because the proposed algorithm can significantly improve the accuracy with little penalty in speed compared to YOLOv3, Gaussian YOLOv3 is superior to the previous methods.

Detection algorithm	Average precision (%)									mAP (%)	FPS	Input size
	Car			Pedestrian			Cyclist					
	E	M	H	E	M	H	E	M	H			
MS-CNN [1]	92.54	90.49	79.23	87.46	81.34	72.49	90.13	87.59	81.11	84.71	8.13	1920×576
SINet [11]	99.11	90.59	79.77	88.09	79.22	70.30	94.41	86.61	80.68	85.42	23.98	1920×576
SSD [17]	88.37	87.84	79.15	50.33	48.87	44.97	48.00	52.51	51.52	61.29	28.93	512×512
RefineDet [28]	98.96	90.44	88.82	84.40	77.44	73.52	86.33	80.22	79.15	84.36	27.81	512×512
CFENet [29]	90.33	90.22	84.85	-	-	-	-	-	-	-	0.25	-
RFBNet [16]	87.41	88.35	83.41	65.85	61.30	57.71	74.46	72.73	69.75	73.44	39.20	512×512
YOLOv3 [21]	85.68	76.89	75.89	83.51	78.37	75.16	88.94	80.64	79.62	80.52	43.57	512×512
Gaussian YOLOv3	90.61	90.20	81.19	87.84	79.57	72.30	89.31	81.30	80.20	83.61	43.13	512×512
Gaussian YOLOv3	98.74	90.48	89.47	87.85	79.96	76.81	90.08	86.59	81.09	86.79	24.91	704×704

Table 2: Performance comparison using KITTI validation set. E, M, and H refer to easy, moderate, and hard, respectively.

Detection algorithm	mAP (%)	FPS	Input size
MS-CNN [1]	5.7	6.0	1920×576
SINet [11]	9.0	18.2	1920×576
SSD [17]	14.1	23.1	512×512
RefineDet [28]	17.4	22.3	512×512
CFENet [29]	19.1	21.0	512×512
RFBNet [16]	14.5	39.0	512×512
YOLOv3 [21]	14.9	42.9	512×512
Gaussian YOLOv3	18.4	42.5	512×512
Gaussian YOLOv3	20.8	22.5	736×736

Table 3: Performance comparison using BDD test set.

	YOLOv3	Gaussian YOLOv3	Variation rate (%)
KITTI validation set			
# of FP	1,681	985	-41.40
# of TP	13,575	14,560	+7.26
# of GT	17,607	17,607	0
BDD validation set			
# of FP	86,380	51,296	-40.62
# of TP	57,261	59,724	+4.30
# of GT	185,578	185,578	0

Table 4: Numerical evaluation of FP and TP.

4.3. Visual and numerical evaluation of FP and TP

For a visual evaluation of Gaussian YOLOv3, Figures 4 and 5 show the detection examples of the baseline and Gaussian YOLOv3 for the KITTI validation set and the BDD test set, respectively. The detection TH is 0.5, which is the default test TH of YOLOv3. The results in the first row of Figure 4 and in the first column of Figure 5 show that Gaussian YOLOv3 can detect objects that YOLOv3 cannot find, thereby increasing its TP. These positive results are obtained because the Gaussian modeling and loss function reconstruction of YOLOv3 proposed in this paper can provide a loss attenuation effect in the learning process, so that the learning accuracy for bbox can be improved, which enhances the performance of objectness. Next, the results in the second row of Figure 4 and in the second column of Figure 5 show that Gaussian YOLOv3 can complement incorrect object detection results found by YOLOv3. In addition, the results in the third row of Figure 4 and in the third column of Figure 5 show that Gaussian YOLOv3 can accurately detect bbox of object inaccurately detected by YOLOv3. Based on these results, Gaussian YOLOv3 can significantly reduce the FP and increase the TP, and consequently, the driving stability and efficiency are improved and fatal accidents can be prevented.

For a numerical evaluation of the FP and TP of Gaussian YOLOv3, Table 4 shows the numbers of FPs and TPs for the baseline and Gaussian YOLOv3. The detection TH is the same as the mentioned before. The KITTI and BDD validation sets are used to calculate the FP and TP because the GT is provided in the validation set. For more accurate measurements, the FP and TP of the two datasets are calculated using the official evaluation code of BDD because the KITTI official evaluation method does not count the FP when bbox is within a certain size. For the KITTI and BDD validation sets, Gaussian YOLOv3 reduces the FP by 41.40% and 40.62%, respectively, compared to YOLOv3. In addition, it increases the TP by 7.26% and 4.3%, respectively. It should be noted that the reduction in the FP prevents unnecessary unexpected braking, and the increase in the TP prevents fatal accidents from object detection errors. In conclusion, Gaussian YOLOv3 shows a better performance than YOLOv3 for both the FP and TP related to the safety of autonomous vehicles. Based on the results described in Sections 4.1, 4.2, and 4.3, the proposed algorithm outperforms previous studies and is most suitable for autonomous driving applications.

5. Conclusion

A high accuracy and real-time detection speed of an object detection algorithm are extremely important for the safety and real-time control of autonomous vehicles. Various studies related to camera-based autonomous driving have been conducted, but are unsatisfactory based on a

Figure 4: Detection results of the baseline and proposed algorithms on the KITTI validation set. The first column shows the detection results of YOLOv3, whereas the second column shows the detection results of Gaussian YOLOv3.

Figure 5: Detection results of the baseline and proposed algorithms on the BDD test set. The first and second rows show the detection results of YOLOv3 and Gaussian YOLOv3, respectively, and each color is related to a particular object class.

trade-off between the accuracy and operation speed. For this reason, this paper proposes an object detection algorithm that achieves the best trade-off between accuracy and speed for autonomous driving. Through Gaussian modeling, loss function reconstruction, and the utilization of localization uncertainty, the proposed algorithm improves the accuracy, increases the TP, and significantly reduces the FP, while maintaining the real-time capability. Compared to the baseline, the proposed Gaussian YOLOv3 algorithm improves the mAP by 3.09 and 3.5 for the KITTI and BDD datasets, respectively. Furthermore, because the proposed algorithm has a higher accuracy than the previous studies with a similar fps, the proposed algorithm is excellent in terms of the trade-off between accuracy and detection speed. As a result, the proposed algorithm can significantly improve the camera-based object detection system for autonomous driving, and is consequently expected to contribute significantly to the wide use of autonomous driving applications.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1F1A1057530) and "The Project of Industrial Technology Innovation" through the Ministry of Trade, Industry and Energy (MOTIE) (10082585,2017).

References

[1] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European conference on computer vision*, pages 354–370. Springer, 2016.

[2] Sungjoon Choi, Kyungjae Lee, Sungbin Lim, and Songhwai Oh. Uncertainty-aware learning from demonstration using mixture density networks with sampling-free variance modeling. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6915–6922. IEEE, 2018.

[3] Aleksa Ćorović, Velibor Ilić, Siniša Durić, Malisa Marijan, and Bogdan Pavković. The real-time detection of traffic participants using yolo algorithm. In *2018 26th Telecommunications Forum (TELFOR)*, pages 1–4. IEEE, 2018.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[5] Xuerui Dai. Hybridnet: A fast vehicle detection system for autonomous driving. *Signal Processing: Image Communication*, 70:79–88, 2019.

[6] Di Feng, Lars Rosenbaum, and Klaus Dietmayer. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3266–3273. IEEE, 2018.

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[8] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Yihui He, Xiangyu Zhang, Marios Savvides, and Kris Kitani. Softer-nms: Rethinking bounding box regression for accurate object detection. *arXiv preprint arXiv:1809.08545*, 2018.

[11] Xiaowei Hu, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1010–1019, 2019.

[12] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[15] Feng Liu, Bingquan Liu, Chengjie Sun, Ming Liu, and Xiaolong Wang. Deep learning approaches for link prediction in social network services. In *International Conference on Neural Information Processing*, pages 425–432. Springer, 2013.

[16] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 385–400, 2018.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[18] Aarian Marshall. False positive: Self-driving cars and the agony of knowing what matters. *WIRED Transportation*, 2018.

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[20] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[21] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[23] Young-Woo Seo, Nathan Ratliff, and Chris Urmson. Self-supervised aerial images analysis for extracting parking lot structure. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[24] Junqing Wei, Jarrod M Snider, Junsung Kim, John M Dolan, Raj Rajkumar, and Bakhtiar Litkouhi. Towards a viable autonomous driving research platform. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 763–770. IEEE, 2013.

[25] Bichen Wu, Forrest Iandola, Peter H Jin, and Kurt Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 129–137, 2017.

[26] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687*, 2018.

[27] Chi Zhang, Yuehu Liu, Danchen Zhao, and Yuanqi Su. Roadview: A traffic scene simulator for autonomous vehicle simulation testing. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1160–1165. IEEE, 2014.

[28] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object

detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4203–4212, 2018.

[29] Qijie Zhao, Yongtao Wang, Tao Sheng, and Zhi Tang. Comprehensive feature enhancement module for single-shot object detector. In *Asian conference on computer vision*. Springer, 2018.