Progressive Coordinate Transforms for Monocular 3D Object Detection

Anonymous Author(s)
Affiliation
Address
email

Abstract

Recognizing and localizing objects in the 3D space is a crucial ability for an AI agent to perceive its surrounding environment. While significant progress has been achieved with expensive LiDAR point clouds, it poses a great challenge for 3D object detection given only a monocular image. While there exist different alternatives for tackling this problem, it is found that they are either equipped with heavy networks to fuse RGB and depth information or empirically ineffective to process millions of pseudo-LiDAR points. With in-depth examination, we realize that these limitations are rooted in inaccurate object localization. In this paper, we propose a novel and lightweight approach, dubbed *Progressive Coordinate Transforms* (PCT) to facilitate learning coordinate representations. Specifically, a localization boosting mechanism with confidence-aware loss is introduced to progressively refine the localization prediction. In addition, semantic image representation is also exploited to compensate for the usage of patch proposals. Despite being lightweight and simple, our strategy allows us to establish a new state-of-the-art among the monocular 3D detectors on the competitive KITTI benchmark. At the same time, our proposed PCT shows great generalization to most coordinate-based 3D detection frameworks.

1 Introduction

Object detection is a fundamental and challenging task in scene understanding applications. Recently, 3D object detection has received increasing attention and found applications in a wide range of scenarios such as autonomous driving, robotics, visual navigation and mixed reality. Despite the great progress from the area of 2D object detection [31, 42, 34, 16, 3], 3D object detection remains a largely unsolved problem as it aims to predict the object location in the 3D space alongside 3D object dimension and orientation.

Existing prevalent approaches [18, 43, 38, 32, 9] for 3D object detection largely rely on LiDAR sensors, which provide accurate 3D point clouds of the scene. Although these approaches achieve superior performance, the dependence on expensive equipment severely limits their applicability to generic 3D perception. There also exists a cheaper alternative that takes a single-view RGB image as input, termed as monocular 3D object detection. However, its performance is far from satisfactory as itself is an ill-posed problem due to the loss of depth information in 2D image planes. Hence, several recent attempts introduce depth information to help monocular 3D detection. Such attempts can be roughly categorized into two directions, pixel-based and coordinate-based. Pixel-based approaches [8, 33, 29, 35] turn to use estimated depth map as additional input for improved detection performance. But at the same time, this leads to heavy computational burden and large memory footprint since they often operate on the entire image. Coordinated-based approaches [36, 27, 40, 25] pursue the coordinate representations as in LiDAR-based methods. They use the predicted depth map

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: Probing investigation on coordinate-based methods, PatchNet [25] and Pseudo-LiDAR [36]. We examine the potential improvement by replacing the predicted factor with the corresponding ground truth. $*$ indicates our reproduced performance. We can see that coordinate-based methods mostly suffer from inaccurate localization.

Factor	PatchNet* [$\mathrm{AP}_{\mathrm{3D}}/\mathrm{AP}_{\mathrm{BEV}}$]			Pseudo-LiDAR* [$\mathrm{AP}_{\mathrm{3D}}/\mathrm{AP}_{\mathrm{BEV}}$]		
	Mod.	Easy	Hard	Mod.	Easy	Hard
Baseline	26.31/34.14	36.40/46.80	21.07/28.04	23.04/31.06	32.27/42.45	19.67/25.67
dimension	27.26/34.62	40.32/47.24	24.29/28.38	25.88/31.97	36.09/44.35	20.88/26.60
rotation	26.25/34.04	36.09/46.25	23.49/27.99	23.88/31.31	32.42/42.74	19.85/26.07
x	32.80/41.43	45.60/56.22	27.38/34.63	28.36/36.77	39.69/50.78	25.08/29.92
y	30.16/34.14	40.94/46.80	24.58/28.04	25.53/31.06	35.19/42.45	20.69/25.67
z	42.42/53.48	55.42/68.29	35.54/45.60	38.37/50.81	50.04/63.96	32.24/43.32
location(xyz)	**72.58/75.27**	**81.41/85.14**	**57.69/66.10**	**64.36/73.37**	**79.13/83.77**	**55.64/58.27**

to convert the monocular image pixels to 3D coordinate representations, then apply a 3D detector on the converted coordinates. In particular, they are often lightweight since their network inputs are object proposals generated by 2D detectors [27, 25]. Despite being efficient, the performance of coordinated-based methods lags far behind LiDAR-based methods. So we ask, can we identify the bottleneck that holds back the 3D detection accuracy of coordinate-based methods and how can we improve them?

In order to determine the bottleneck, we conduct an investigation on two widely adopted coordinate-based methods, PatchNet [25] and Pseudo-LiDAR [36]. To be specific, for each prediction target, we examine the potential improvement by replacing the predicted value with the corresponding ground truth, and then re-compute the 3D detection accuracy. As shown in Table 1, using ground truth dimension and rotation do not bring significant improvements over the baseline. But using ground truth location (*i.e.*, x, y, z coordinates of the objects) almost triples the 3D detection accuracy. This indicates that coordinate-based methods mostly suffer from inaccurate localization even with the assistance of estimated depth maps.

Based on this observation, we focus on improving the accuracy of 3D center localization. In this work, we propose a lightweight and generalized approach, called Progressive Coordinate Transforms (PCT), to enhance the localization capability for coordinate-based methods. First of all, since the localization regression network in most coordinate-based methods is less accurate but lightweight, we propose to progressively refine its prediction similar to gradient boosting [10, 11]. To be specific, a localization regression network can be seen as a weak learner, and we progressively train multiple consecutive networks such that each network fits the regression residual from the previous networks. These networks share the same lightweight structure so that the computation overhead is negligible. We also predict a confidence score for each network to help stabilize the end-to-end training. We term this progressive refining strategy as confidence-aware localization boosting (CLB). Compared to image-only or pixel-based methods, coordinated-based methods suffer from the problem of missing global context information due to the use of patched input. In order to further improve the localization accuracy, we exploit semantic image representations from 2D detector. We term this module as global context encoding (GCE). We find that GCE can not only improve center localization accuracy, but also contribute to the final 3D box estimation.

Through extensive experiments, our progressive coordinate transforms, consisting of CLB and GCE, is shown to improve popular coordinate-based models [36, 25] by generating more accurate localization. Without bells and whistles, we achieve state-of-the-art monocular 3D detection performance on KITTI [14, 15, 13] with a strong base method [25].

2 Related work

2.1 Monocular 3D object detection

Existing paradigms for monocular 3D object detection can be categorized into two types: image-only methods and depth-assisted methods.

For image-only methods, they often adapt architectures and good practices from popular 2D detectors [31, 42, 34]. However, locating objects in 3D space is much more challenging without depth

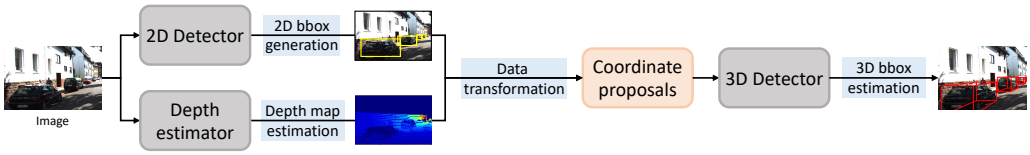

Figure 1: A common pipeline of coordinate-based monocular 3D detectors. It consists of four steps to predict the final 3D boxes: 2D bounding box generation, depth map estimation, data transformation and 3D box estimation. In this work, we focus on improving the last step: 3D box estimation.

information. Hence, several works [28, 1, 23, 5, 42] integrate geometry consistency into the training strategy to constrain the localization prediction. Deep3DBox [28] divides orientation into multi-bins to stably regress them, and combines the 2D-3D box constraint to recover accurate 3D object pose. M3D-RPN [1] utilizes the geometric relationship between 2D and 3D perspectives by sharing the prior anchors and classification targets. MonoPair [5] leverages the spatial relationships between paired objects to improve accuracy on occluded objects.

Depth-assisted methods often estimate a depth map given a input image, and use it in different ways. Some pixel-based approaches [8, 24] directly feed images and estimated depth maps into networks to generate depth-aware features and enhance the 3D detection performance. Some other coordinate-based approaches first transform the pixels of input images to 3D coordinates by leveraging the depth and camera information, then feed the coordinate proposals to a 3D detector. Pioneering work Pseudo-LiDAR [36] imitates the process of LiDAR-based approaches, which uses LiDAR-based 3D detector upon coordinates proposals. AM3D [27] explores the multi-modal input fusion to embed the complementary RGB cue into the network. Recently, PatchNet [25] points out that the efficacy of pseudo-LiDAR representation comes from the coordinate transformation, instead of sophisticated LiDAR-based networks. Hence, they design a simple 2D CNN to perform 3D detection.

In this work, we follow the research of coordinate-based methods [36, 25]. Instead of regressing 3D localization directly with a single lightweight network, we propose to progressively refine the prediction inspired by gradient boosting. We also incorporate RGB image information to complement patch proposals and enhance global context modeling. Different from AM3D [27], we utilize the RGB features from the 2D detector directly which can share the same context, and we do not need to train another RGB network from scratch.

2.2 Gradient boosting

Gradient boosting is a well-known greedy algorithm proposed in [6], which trains a sequence of learners and progressively improves the prediction results. It is a general learning framework, and has been verified to be a formidable force when applied with lightweight learners. Meanwhile, when each learner in the sequence is heavy, the computation cost becomes high and the performance is not beneficial [22]. Early works in 2D detection area [17, 19, 20] also adopt the boosting mechanism following a standard cascade paradigm, and achieve improved performance. Li et al. [19] treat face detection as an image retrieval task and improve it with a boosted exemplar-based face detector. Karianakis et al. [17] and Li et al. [20] feed convolutional features of proposals instead of hand-crafted features to boosted classifiers and distinguish objects from backgrounds.

We are the first to explore the boosting mechanism in coordinate-based methods for 3D object detection. First, instead of the entire 3D detection pipeline, we only progressively boost the localization regression network as its computational cost is insignificant comparing to the entire pipeline. Second, we refine the localization with an additional confidence score in the boosting procedure, such that the loss is balanced. These choices greatly improves the performance with small extra parameters.

3 Background

Before diving into the details, we first revisit recent coordinate-based monocular 3D detection methods and present a visual depiction of its common pipeline in Figure 1. The framework usually consists of four main components: 2D bounding box generation, depth map estimation, data transformation and 3D box estimation. Specifically, given an image I, the process can be described as:

2D bounding box generation. An off-the-shelf 2D object detector F_{2d} such as Faster R-CNN [31] is employed on image I to generate region of interests (RoIs), $\mathcal{R} = F_{2d}(I)$.

Depth map estimation. An off-the-shelf depth estimator F_z such as DORN [12] is applied on image I to predict its depth map, $\mathcal{Z} = F_z(I)$.

Data transformation. To convert a pixel (u, v) within a RoI to 3D space, the associated depth $z = \mathcal{Z}(u, v)$ is used to transform it into its 3D coordinates (c_x, c_y, c_z) by

$$c_x = \frac{(u - u') \times z}{f_u}; \quad c_y = \frac{(v - v') \times z}{f_v}; \quad c_z = z \tag{1}$$

Here, (c_x, c_y, c_z) is a pixel in the generated 3D coordinate patch c. (u', v') is the camera principal point. f_u and f_v are the focal length along horizontal and vertical axis, respectively. u', v', f_u, f_v are usually provided by the datasets.

3D box estimation. Once the 3D coordinates for each RoI are available, the final step is to predict 3D boxes with their center location, rotation and dimension. Different networks F_{3d} such as Frustum PointNet [30] can be employed to conduct 3D box prediction, $\mathcal{B} = F_{3d}(c)$. Here, $\mathcal{B}$ includes the center location (x, y, z), rotation (θ), and dimensions (w, h, l) of the 3D box.

Since the first two steps use off-the-shelf models and the third step can be computed analytically, in this paper, we focus on improving the last step of coordinated-based methods. In particular, our goal is to improve the accuracy of localization prediction motivated by the observation in Table 1.

4 Method

In this section, we present our progressive coordinate transforms (PCT) for improved 3D detection. In order to obtain more accurate localization predictions, we introduce a confidence-aware localization boosting mechanism (CLB) in Sec. 4.1 to progressively refine the prediction. Then in Sec. 4.2, we incorporate RGB image information by a global context encoding (GCE) strategy to compensate for the drawbacks of using patch proposals. In the end, we illustrate the overall framework of PCT in Figure 2 (a) and its implementation details in Sec. 4.3.

4.1 Confidence-aware localization boosting

Following Frustum PointNet [30], most coordinate-based methods [36, 39, 25, 37] divide the last step of 3D box estimation into two major components. The first component is a lightweight 3D localization regressor F, whose input is 3D coordinate proposals generated from data transformation. The second component is a relatively heavy network G used to regress the final 3D box $\mathcal{B}$. Recalling the results in Table 1, 3D localization performance is the weakest point of a coordinate-based model, accounting for up to 50 AP loss when all other modules keep intact. Therefore, can we find an efficient way to improve the accuracy of localization prediction and also generalizes to other coordinated-based methods?

Gradient boosting [10, 11] is a general learning framework that combines multiple weak learners into a single strong one in an iterative fashion. Let $\mathcal{L}(x)$ be the risk of ensemble models, the algorithm devotes to seek an approximation $h(x)$ to minimize $\mathcal{L}(y^*, x) = \Psi(y^*, h(x))$, where y^* is a target value, $\Psi(\cdot)$ is the loss function. $h(x)$ is a linear combination of a set of weak (base) learners $f_t(x)$ from some class $\mathcal{F}$, *i.e.*, $h(x) = \sum_{t=1}^{t=T} \gamma_t f_t(x) + const$. Here, T is the total training iterations and γ_t is the corresponding weight for each weak learner. To minimize the empirical risk, the algorithm starts with a model $h_0(x)$, and then incrementally expands it in a greedy manner. This process manages to fit a new weak learner to the residual errors made by the previous set of learners. Mathematically, the optimization can be formulated as

$$\begin{aligned} h_0(x) &= \arg\min_{\gamma_0} \mathcal{L}_0(y^*, x); \\ &\quad ... \\ h_t(x) &= h_{t-1}(x) + \arg\min_{f_t \in \mathcal{F}} \mathcal{L}_t(y^*, h_{t-1}(x) + f_t(x)). \end{aligned} \tag{2}$$

Inspired by gradient boosting, we imitate its optimization procedure to progressively adapt localization prediction by multiple localization regressors instead of a single one used in previous works [36, 27,

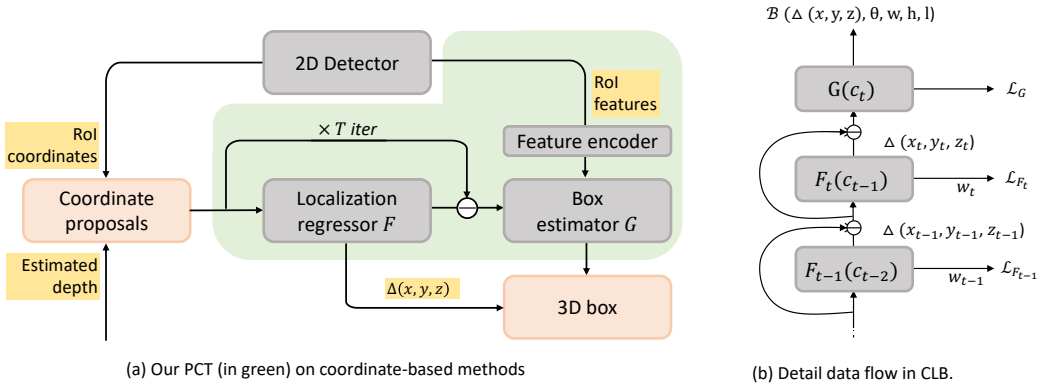

Figure 2: (a) We show Progressive Coordinate Transformation (PCT) marked as green area for coordinate-based methods. (b) The data flow of CLB mechanism in detail.

40, 25]. To be specific, we treat localization network F as a weak learner and stack multiple of them as shown in Figure 2 (b). Given F is a lightweight network, the extra computational cost brought by gradient boosting is insignificant. After the data transformation step, each 2D bounding box obtains its corresponding coordinate patch c. We take the coordinate patch as the input of weak learner F to regress the center localization residual $\Delta(x, y, z)$ based on the prediction from previous stage,

$$\Delta(x_t, y_t, z_t) = F_t(c_{t-1}); \text{ where } c_{t-1} = c_0 - \gamma_0(x_0, y_0, z_0) - \sum_{i=1}^{t-1} \gamma_i \Delta(x_i, y_i, z_i). \tag{3}$$

We denote c_0 and (x_0, y_0, z_0) to be the initial coordinate input patch and object location, respectively. Coordinate input patch c_{t-1} is then transformed according to the localization residual prediction, and fed into the next weak learner. Thus the risk at stage t can be written as,

$$\mathcal{L}_{F_t}((x^*, y^*, z^*), c_{t-1}) = \Psi((x^*, y^*, z^*), \gamma_0(x_0, y_0, z_0) + \sum_{i=1}^{t-1} \gamma_i \Delta(x_i, y_i, z_i)) \tag{4}$$

where (x^*, y^*, z^*) indicates the ground truth location. At this point, we can easily see that Eq 4 is a natural derivation from Eq 2. After T iterations, the final adjusted prediction c_T is fed into the network G to estimate the 3D box $\mathcal{B}$, *i.e.*$\mathcal{B} = G(c_T)$.

Confidence-aware network loss In the case that the target of weak learner f is differentiable, gradient boosting solves the optimization problem in a forward greedy manner as shown in Eq 2. For each iteration, it first fits the weak learner f to the residual error, and then the optimal value of the coefficient weight γ is determined for this weak learner. The optimization procedures train iteratively for T iterations.

However, for our center regression task, we would like to train T weak localization networks $F_t(c_{t-1}), t \in 1, ..., T$ and a 3D box prediction network G in an end-to-end manner instead of bootstrapping. This is challenging in terms of both computational cost and optimization stability, given the simultaneous training of a set of weak learners and their coefficient weights. Therefore, we first simplify the problem by treating all γ_t the same and only focus on optimizing the localization networks. However, the contribution from each weak learner F may not be the same during end-to-end training, which leads to unstable optimization. Hence, we tailor a confidence-aware boosting loss to facilitate network training, by learning confidence score s_t for each localization loss function $\mathcal{L}_{F_t}$. The confidence score s_t is learned from a small decoder and followed a self-balancing formulation closely coupled to the network loss. The overall loss function is defined as

$$\mathcal{L}(\mathcal{B}^*, c_0) = \sum_{t=1}^{T} s_t * \mathcal{L}_{F_t}((x^*, y^*, z^*), c_{t-1}) + \lambda_s \prod_{t=1}^{T}(1 - s_t) + \mathcal{L}_G(\mathcal{B}^*, c_T), \tag{5}$$

where $\mathcal{B}^*$ is the 3D box ground truth, $(x^*, y^*, z^*) \in \mathcal{B}^*$ and λ_s is the balance weight. s_t is the prediction after sigmoid, which represents the confidence of the localization regression loss at t^{th}

stage, and $\prod_{t=1}^{T}(1-s_t)$ is the penalty on the network uncertainty. In other words, if s_t is approaching 1, which means the network is confident about localization refinement at t^{th} stage, then no penalty will be applied. Otherwise, the uncertainty of regression loss is high, thus triggers a higher penalty.

4.2 Global context encoding

Typical coordinate-based methods [36, 39, 25, 37] perform 3D detection based on 2D RoIs, which is similar to two-stage 2D detection frameworks, such as Faster R-CNN [31]. In a two-stage 2D object detection framework, the second stage reuses the features from the first stage via RoIPooling [31] or RoIAlign operators [16] guided by ROI proposals, and then a small decoder is used for localization refinement. However, in 3D detection, only cropped patches with coordinates information are fed to the network for 3D box regression. Neither RGB information nor global context is included.

Considering that the RGB information is a vital visual clue, we explore its aggregation in the last 3D box estimation step. Similar to two-stage 2D detectors, we obtain the RGB information by directly cropping the corresponding features from a 2D detector F_{2d}. Then the input to 3D box estimator G can be formulated as $c = \{[\mathcal{D}(u,v), \mathcal{A}(u,v)], \forall(u,v) \in \mathcal{R}\}$. Here, $\mathcal{D}(\cdot)$ represents the data transformation function and $\mathcal{A}(\cdot)$ represents the RoIAlign operation. Both operations are performed upon the generated regions of interest from $\mathcal{R}$.

After RoIAlign operation, features are of size $C \times K \times K$, where C is the number of channels and $K \times K$ is the corresponding width and height, respectively. A small feature encoder is then used to encode cropped features into vectors with the dimension of C. A feature fusion is followed to integrate coordinate representations with the obtained image representations. Benefiting from the large receptive fields of image feature representations, 3D box estimator can now have access to global context. Besides, directly cropping on RGB features also avoids learning image representations of RoIs from scratch and reduces the overall network parameters.

4.3 Implementation details

Our overall framework of PCT can be visualized in Figure 2. In terms of implementation details, we instantiate our algorithm on two widely adopted coordinate-based methods with public released code [26], PatchNet [25] and Pseudo LiDAR [36]. Bearing efficiency in mind, we use a real-time 2D detector RTM3D [21] with DLA-34 [41] as backbone. For the sake of fair comparison, we adopt depth predictor DORN [12] as in most depth-assisted literature. For the CLB mechanism, we inherit the original localization regression framework in each method. Each F_t shares the same structure. The corresponding confidence is generated following the last convolutional layers of F_t with three linear layers and a Sigmoid function. $T = 3$ and $\lambda_s = 1$ are set for the following experiments except for the ablation study on boosting iterations. For GCE module, we get the corresponding input image features by performing RoIAlign on the features from last convolutional layer of 2D detector. We set the output of RoIAlign as 16×16. As 2D detector use DLA-34 as backbone, the obtained image feature representations have the size of $64 \times 16 \times 16$ and entitle arbitrary sized receptive field theoretically due to the embedded deformable convolution [7]. The structure of feature encoder in global context module is two common $64 \times 3 \times 3$ convolutional layers (stride=2) and a $64 \times 1 \times 1$ convolutional layer (stride=1). Hence, image feature representations are encoded to a vector with 64-dim. The obtained features then concatenate with the coordinate feature vectors from the final global pooling of box prediction network G. With the lightweight structure, we are able to optimize the network end-to-end on a single Nvidia V100 GPU with 16G memory. The training criterion for network F and G and other training settings follow the corresponding base methods for fair comparison.

5 Experiments

5.1 Setup

As mentioned in Sec. 4.3, we perform experiments on two coordinate-based methods, PatchNet [25] and Pseudo-LiDAR [36]. For ablation studies, we only use PatchNet given its current best performance.

Table 2: Ablative analysis on KITTI validation set for $\mathrm{AP}_{\mathrm{3D}}$ and $\mathrm{AP}_{\mathrm{BEV}}$ at IoU = 0.7. Experiment Group (I) is the baseline method. Different experiment settings are explored: (II) applying localization boosting without confidence constraint, (III) performing confidence-aware localization boosting algorithm, (IV) adding global context encoding on Group (II), (V) our full approach.

Group	Localization Boosting	Uncertainty	GCE	$\mathrm{AP}_{\mathrm{3D}}$			$\mathrm{AP}_{\mathrm{BEV}}$		
				Mod.	Easy	Hard	Mod.	Easy	Hard
I	-	-	-	25.88	36.07	20.99	33.34	46.39	27.54
II	✓	-	-	26.78	37.29	24.11	34.39	47.08	28.28
III	✓	✓	-	27.24	38.32	24.39	33.92	46.77	27.98
IV	✓	-	✓	27.12	37.38	24.11	34.46	46.70	28.32
V	✓	✓	✓	**27.53**	**38.39**	**24.44**	**34.65**	**47.16**	**28.47**

Table 3: Comparison of different boosting iteration settings on val split set.

Localization Boost	$\mathrm{AP}_{\mathrm{3D}}/\mathrm{AP}_{\mathrm{BEV}}$		
	Mod.	Easy	Hard
-	25.88/33.34	36.07/46.39	20.99/27.54
$+1$	26.31/34.14	36.40/46.80	21.07/28.04
$+2$	**26.78/34.39**	**37.29/47.08**	**24.11/28.28**
$+3$	26.77/34.21	37.12/47.00	23.48/28.23
$+4$	26.64/34.43	37.24/47.04	23.89/28.27
$+5$	26.49/34.30	37.01/46.99	23.83/28.24

Table 4: Evaluation of different coordinate feature fusion with GCE on validation set. Baseline is the Group (II) in Table 2.

Method	$\mathrm{AP}_{\mathrm{3D}}$		
	Mod.	Easy	Hard
Baseline	26.78	37.29	24.11
F + GCE	27.08	37.33	24.07
G + GCE	**27.12**	37.38	24.11
All + GCE	27.07	**37.43**	**24.18**

Our method is evaluated on the KITTI benchmark [14, 15, 13], which contains 7,481 and 7,518 images for training and testing respectively. We follow [4] to split the 7,481 training images into 3712 for training and 3,769 for validation.

Precision-recall curves are adopted for evaluation, and we report the average precision (AP) results of 3D and Bird's eye view (BEV) object detection on KITTI validation and test set. For fair comparison to previous literature, the 40 recall positions-based metric $AP|_{R40}$ is reported on test set while $AP|_{R11}$ is reported on validation set. Three levels of difficulty are defined in the benchmark according to the 2D bounding box height, occlusion, and truncation degree, namely, "Easy", "Mod.", and "Hard". The KITTI benchmark ranks all methods based on the $\mathrm{AP}_{\mathrm{3D}}$ of "Mod.". In particular, we focus on the "Car" category as in [36, 25], and we adopt IoU = 0.7 as threshold for evaluation.

5.2 Method analysis based on PatchNet

Main ablative analysis In Table 2, we conduct ablation studies to analyze the effectiveness of our contributions: I) Without any localization regression network F, network G generates 3D box prediction directly. II) This configuration only contains localization boosting part without confidence constraint. III) The entire CLB mechanism is included to progressively regress center localization. IV) GCE module is added to the network based on the localization boosting block without confidence constraint since the feature fusion can be performed on either the localization regression networks F or 3D box prediction network G. V) Our full method with all the components.

As depicted in Table 2, we can observe that the performance continues to grow with the addition of every component. From group II, localization boosting brings a noticeable improvement on all settings especially "Hard" set, which confirms its effectiveness in increasing localization accuracy. Group III shows that balancing training loss by adding confidence leads to better and more stable optimization. Group IV reveals that the proposed GCE module can effectively equip RGB information and global context with 3D coordinate representations. In the end, Group V demonstrates the complementarity of the proposed CLB mechanism and GCE module, leading to an improvement from 25.88/36.07/20.99 to 27.53/38.39/24.44 compared with Group I.

Localization boosting iteration settings We explore the effect of different localization boosting iteration settings in this part. For a fair comparison, we do not perform the confidence constraint on regression loss. As illustrated in Table 3, when boosting iteration $T = 3$, we achieve the best

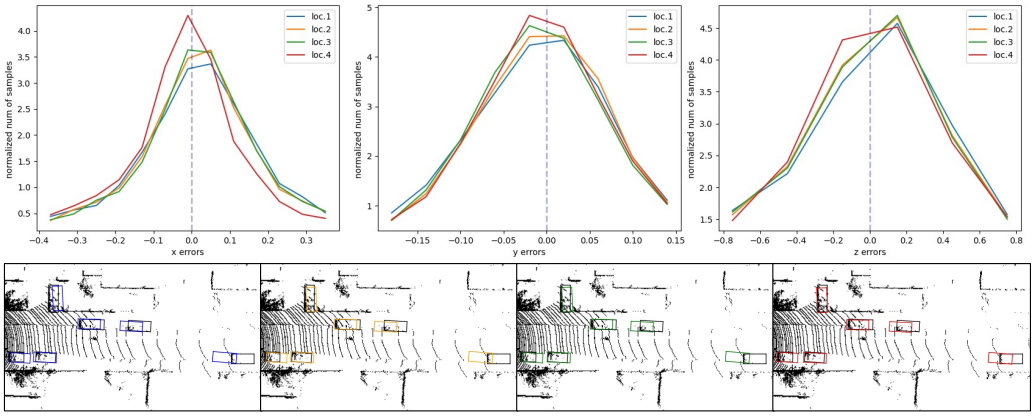

Figure 3: The statistic analysis and comparison on different Localization boosting stage when $T = 3$. The vertical axis of the chart in the first row represents the number of samples after normalization. "loc.1/2/3" denotes the $1/2/3^{th}$ step of localization in F and "loc.4" is the final localization in G. The second row shows the BEV view and ground truth (black) at corresponding stages.

3D detection performance. More iterations of boosting do not bring improvements, which might be caused by overfitting with the increasing of network parameters.

To verify the improvement of each step in boosting procedure, we conduct the comparison of localization errors at iteration $T = 3$ on the specific metrics (location "xyz") of the ground truth. In particular, three stacked localization networks F generate intermediate localization "loc.1/2/3" and G output the final localization "loc.4". As shown in Figure 3 at first row, we can see that the distributions of "x", "y" and "z" errors tend to distributed to zero with localization boosting iterating. For instance, the red line in left chart is narrow and tall near zero along horizontal axis compared with other lines, which means that the corresponding x coordinate is more accurate than others. This further suggests that localization boosting is useful for object localization. We also show the BEV prediction at each step, it can be seen that the predicted box gradually close to the ground truth box (shown in black).

Impact of global context encoding We also explore where the global context representation fusion operates. We take Group II as the baseline, and perform feature fusion on localization regression network F (row one), 3D estimation network G (row two) or on both (final row). RoI features are encoded into a vector with GCE and then concatenate with the feature vectors from network (F or G) global pooling. As shown in Table 4, the operation on G outperforms it on F, which indicates that image representation is more suitable for the overall box prediction rather than only localization as it contains the additional semantic appearance information. Although operation on all networks achieves a lightly higher than it on G on the "Easy" and "Hard" set, introducing parameters is much larger due to operation on stacked localization networks. Hence, we only apply GCE on network G in our approach for a lightweight network and avoid overfitting during training.

5.3 Generally applicable to other coordinate-based algorithm

In this section, we demonstrate the generalization capability of our algorithm to classic coordinate-based methods Pseudo-LiDAR [36]. As shown in Table 5, each component of our algorithm improves the original methods a lot. Specially, our approach improves Pseudo-LiDAR by 1.43/2.06/0.51 while 1.22/1.99/3.36 gains on PatchNet.

Table 5: Comparison of generalization on validation set. $*$ denotes that the method is reproduced by ourselves. With the CLB and GCB get in, the accuracy of method is improved.

Method	AP_{3D}			AP_{BEV}		
	Mod.	Easy	Hard	Mod.	Easy	Hard
Pseudo-LiDAR*[36]	23.04	32.27	19.67	31.06	42.45	25.67
Pseudo-LiDAR + CLB	24.14	34.46	20.16	32.41	44.98	26.82
Pseudo-LiDAR + CLB + GCE	24.43	34.34	20.18	32.50	45.35	26.91

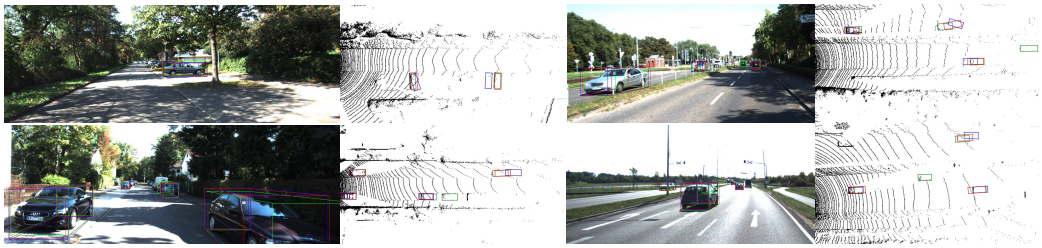

Figure 4: Qualitative comparison of ground truth (green), base method PatchNet (blue), and our method (red) on KITTI val set. The first and second columns show RGB and BEV images respectively.

Table 6: Comparison with SoTA methods on the KITTI test set at IoU = 0.7. Our algorithm achieves new SoTA performance. "Depth" means if the method belongs to depth-assisted methods or not. "Type" indicates the method input pattern, "Pixel" denotes the methods with image as inputs directly while "Coordinate" means the coordinate-based methods with 3D coordinates as inputs.

Method	Depth	Type	AP_{3D}			AP_{BEV}		
			Mod.	Easy	Hard	Mod.	Easy	Hard
RTM3D[21]	no	Pixel	10.34	14.41	8.77	14.20	19.17	11.99
AM3D[27]	yes	Coordinate	10.74	16.50	9.52	17.32	25.30	14.91
PatchNet[25]	yes	Coordinate	11.12	15.68	10.17	16.86	22.97	14.97
D^4LCN[8]	yes	Pixel	11.72	16.65	9.51	16.02	22.51	12.55
Kinematic3D[2]	yes	Pixel	12.72	19.07	9.17	17.52	26.69	13.10
Liu et al. [24]	yes	Pixel	13.25	**21.65**	9.91	17.982	**29.81**	13.08
PCT	yes	Coordinate	**13.37**	21.00	**11.31**	**19.03**	29.65	**15.92**

5.4 Comparison with state-of-the-arts

We build our test detector on the current state-of-the-art coordinate-based method PatchNet, and results are shown in Table 6. Quantitatively, our method achieves the highest performance on "Mod." set with 22 FPS on NvidiaTesla v100 including 2D detector inference time, which is the main setting for ranking on the benchmark. Specially, large margins, 2.25/5.32/1.14 on 3D detection and 2.17/6.68/0.95 on BEV, are observed over the base method PatchNet with only additional 8M parameters. Besides, our methods also outperforms the pixel-based state-of-the-arts methods Liu et al. [24] especially on "Hard" set.

Qualitative comparisons are shown in Figure 4. The ground truth, base method (PatchNet), and our method are colored in green, blue, and red, respectively. For better visualization, the first and second columns show RGB images and BEV images, respectively. Compared with the base method, our algorithm can produce higher-quality 3D bounding boxes in different kinds of scenes.

6 Conclusions

In this paper, we have introduced a novel approach PCT to address the inaccurate localization problem for monocular 3D object detection. This is achieved by iteratively transforming the coordinate representation with a confidence-aware booting mechanism. Meanwhile, global context is introduced to compensate for the missing of semantic image representation in coordinated-based methods. Through extensive experiments, we have shown that our proposed PCT can improve most coordinate-based models by a large margin. We also achieve state-of-the-art monocular 3D detection performance on KITTI with previous best-performing model PatchNet [25].

We have demonstrated that our proposed algorithm yields significant improvements when generalizing to coordinate-based methods. However, there are several limitations that could indicate the possible directions for future work. First, the performance of off-the-shelf 2D detector directly influences the accuracy of coordinate-based methods, hence how to effectively design the 3D box estimation algorithm to fit with existing 2D detectors is important. Second, we only concentrate on the lightweight coordinate-based methods. It requires further exploration to extend our approach to pixel-based methods. Finally, our proposed global context encoding is a simple module. Despite working well, a more tailored feature fusion strategy between coordinate representation and RGB image representation is worth exploring.

References

[1] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *ICCV*, 2019.

[2] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3d object detection in monocular video. *arXiv preprint*, 2020.

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[4] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *NeurIPS*, 2015.

[5] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. In *CVPR*, 2020.

[6] Corinna Cortes, Mehryar Mohri, and Umar Syed. Deep boosting. In *ICML*, 2014.

[7] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[8] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. *arXiv preprint*, 2019.

[9] Liang Du, Xiaoqing Ye, Xiao Tan, Jianfeng Feng, Zhenbo Xu, Errui Ding, and Shilei Wen. Associate-3ddet: Perceptual-to-conceptual association for 3d point cloud object detection. In *CVPR*, 2020.

[10] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 2001.

[11] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 2002.

[12] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. `http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d`. 2013.

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *IJRR*, 2013.

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012.

[16] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[17] Nikolaos Karianakis, Thomas J Fuchs, and Stefano Soatto. Boosting convolutional features for robust object proposals. *arXiv preprint*, 2015.

[18] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 2018.

[19] Haoxiang Li, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Gang Hua. Efficient boosted exemplar-based face detection. In *cvpr*, 2014.

[20] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. In *cvpr*, 2015.

[21] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In *ECCV*, 2020.

[22] Xuchun Li, Lei Wang, and Eric Sung. A study of adaboost with svm based weak learners. In *IJCNN*, volume 1, pages 196–201. IEEE, 2005.

[23] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3d object detection. In *CVPR*, 2019.

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

[25] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In *ECCV*, 2020.

[26] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. `https://github.com/xinzhuma/patchnet`. 2020.

[27] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *ICCV*, 2019.

[28] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *CVPR*, 2017.

[29] Erli Ouyang, Li Zhang, Mohan Chen, Anurag Arnab, and Yanwei Fu. Dynamic depth fusion and transformation for monocular 3d object detection. In *ACCV*, 2020.

[30] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 2018.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[32] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[33] Xuepeng Shi, Zhixiang Chen, and Tae-Kyun Kim. Distance-normalized unified representation for monocular 3d object detection. In *ECCV*, 2020.

[34] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[35] Li Wang, Liang Du, Xiaoqing Ye, Yanwei Fu, Guodong Guo, Xiangyang Xue, Jianfeng Feng, and Li Zhang. Depth-conditioned dynamic message propagation for monocular 3d object detection. 2021.

[36] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *CVPR*, 2019.

[37] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *ICCV workshops*, 2019.

[38] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[39] Xiaoqing Ye, Liang Du, Yifeng Shi, Yingying Li, Xiao Tan, Jianfeng Feng, Errui Ding, and Shilei Wen. Monocular 3d object detection via feature domain adaptation. In *ECCV*, 2020.

[40] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint*, 2019.

[41] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *CVPR*, 2018.

[42] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint*, 2019.

[43] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We have described exactly in abstract section and section 1.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our method focuses on the monocular 3D detection which can be applied in autonomous driving field and assists driving systems. This technology may endanger the employment of human servants and drivers because they may be replaced by autonomous robots and vehicles, which may cause the potential social problems.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We demonstrate the codes in supplementary materials. The dataset is public and URL [13] is also attached in Section 5.1.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.3.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We report experiments results at the fixed seed.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] The type of resources can be seen at Section 4.3. And each GPU can run two experiments, thus, our included experiments (total number of 15) in paper require about 8 Nvidia Tesla v100 GPUs (16G).
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.1.
   (b) Did you mention the license of the assets? [Yes] We conduct experiments on KITTI (license: CC BY-NC-SA 3.0).
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We provide the license of our code in supplementary materials.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We conduct experiments only on the car category.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]