Applied Soft Computing Journal 178 (2025) 113181

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Diffusion-based dynamic super-dense candidate boxes with random center points for 3D object detection☆

Si-Heng He [a], Zi-Jia Wang [a],*, Yuan-Gen Wang [a], Yicong Zhou [b], Sam Kwong [c]

[a] *Computer Science and Cyber Engineering, Guangzhou University, Guangzhou, China*
[b] *Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macao Special Administrative Region of China*
[c] *Department of Computer Science, Lingnan University, Hong Kong, China*

ARTICLE INFO

Keywords:
3D object detection
Diffusion model
Point cloud

ABSTRACT

Diffusion models have achieved promising results in image generation, but their applications in 3D object detection still need further exploration. In this paper, we design a novel model DiffCandiDet based on dense heads with Gaussian distributed center points for 3D object detection, which effectively integrates the anchor-based method and the Gaussian random noise-based method to leverage the powerful denoising and reconstruction capabilities of the diffusion model. To achieve the learning balance for multi-class 3D object detection, we propose a Dynamic Super-dense Candidate Boxes (DSCB) strategy. Notably, DiffCandiDet addresses the issue of traditional models struggling to detect pedestrians walking side by side. In addition to Gaussian distribution, we also propose a DSCB strategy based on discrete uniform distribution (DUCandiDet) and continuous uniform distribution (CUCandiDet), to reduce the runtime consumption and enhance the robustness of the model. Extensive experiments show that DiffCandiDet achieves competitive results on both KITTI and Waymo Open Datasets. **DiffCandiDet ranks 1st** on the KITTI validation set in the Car and Pedestrian detection leaderboard. Code is available at https://github.com/SiHengHeHSH/DiffCandiDet.

1. Introduction

3D object detection based on LiDAR point clouds [1–5] plays a crucial role in autonomous driving. The mainstream approaches for LiDAR-based 3D object detection [6–9] can be broadly categorized into anchor-based [10–16] and anchor-free [17–19] methods. The anchor-based method typically predefines a set of templates (anchors) with fixed sizes and evenly spaced positions and orientations to leverage the common characteristic of real objects with similar sizes. The anchor-free methods can be flexibly applied to diverse views without introducing additional shape priors. However, compared to anchor-based methods that only select high IoU (Intersection over Union) samples, anchor-free methods [18] may select some bad positive samples, leading to inaccurate object predictions. Among anchor-based methods, two-stage anchor-based methods [20,21] have demonstrated significant advantages in detection accuracy, which refine the predicted boxes by introducing a RoI (Region of Interest) network based on the one-stage methods [10,22]. Moreover, multi-modal anchor-based 3D object detection [23,24] involves the fusion of information from images and LiDAR point clouds, further maintaining a leading position in detection performance.

Furthermore, existing 3D object detection methods have recently demonstrated state-of-the-art Car detection performance, e.g. PV-RCNN [12], Voxel-RCNN [25], CasA [21] and LoGoNet [23]. However, their performance on the Pedestrian and Cyclist categories is suboptimal when extended to multi-class 3D object detection [21,23], as shown in Fig. 1. For instance, the traditional anchor-based method CasA-V [21] fails to detect pedestrians walking side by side, as shown in the first and third columns of Fig. 9. Especially in dense scenarios where the Pedestrian and Cyclist are clustered together, mutual occlusion and overlapping may lead to missed detections, which poses a significant safety risk for autonomous driving.

Recently, diffusion models [26–30] have achieved considerable success in image and text generation. However, their application in perception tasks is still an area under exploration. DiffusionDet [31] pioneers the application of diffusion models in 2D object detection [32] tasks, treating 2D candidate boxes (x, y, w, h) as Gaussian noise as shown

☆ This work was supported in part by the National Natural Science Foundations of China (NSFC) under Grants 62106055 and 62176094, in part by the Guangdong Natural Science Foundation, China under Grants 2025A1515010256, and in part by the Guangzhou Science and Technology Planning Project under Grants 2023A04J0388 and 2023A03J0662.

* Corresponding author.
E-mail address: zijiawang@gzhu.edu.cn (Z.-J. Wang).

https://doi.org/10.1016/j.asoc.2025.113181
Received 18 September 2024; Received in revised form 12 March 2025; Accepted 12 April 2025
Available online 21 May 2025

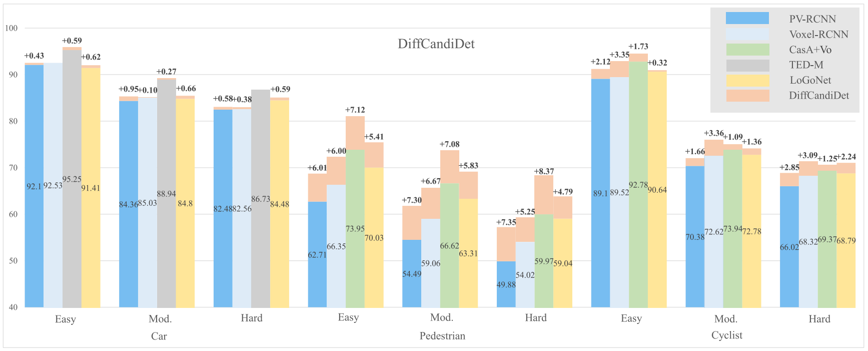

Fig. 1. DiffCandiDet outperforms all the baseline PV-RCNN, Voxel-RCNN, CasA+V, TED-M, and LoGoNet by a large margin on the KITTI val set, especially on the Pedestrian and Cyclist categories with AP calculated by 40 recall positions. The orange parts represent the improvements by adding DiffCandiDet.

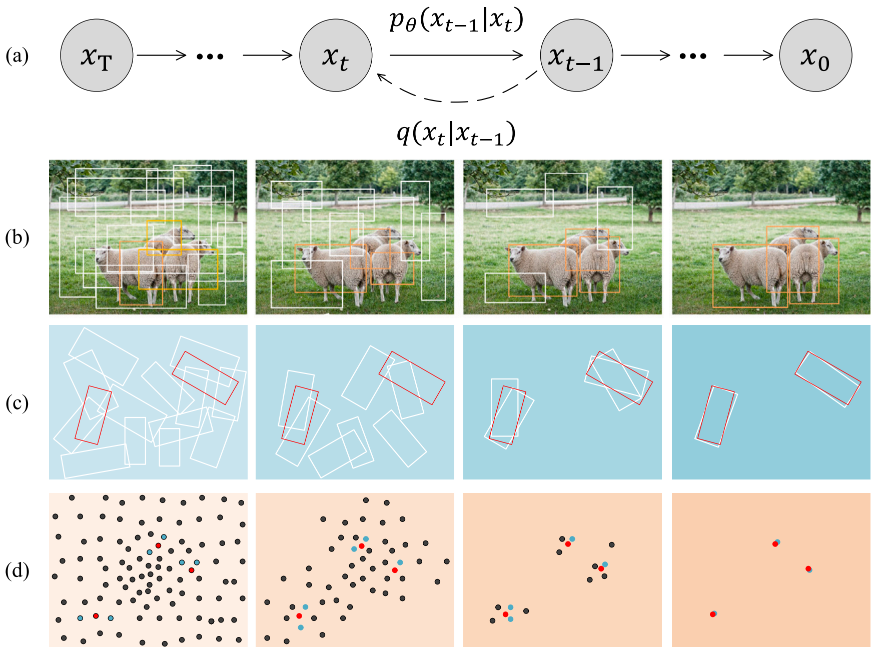

Fig. 2. Diffusion model for 2D and 3D object detection. (a) A diffusion model where q is the forward process and p_θ is the reverse process. (b) Diffusion model for 2D object detection (DiffusionDet). (c) Diffusion model for 3D object detection (Diff3Det). (d) We formulate 3D object detection as a denoising diffusion process based on Dynamic Super-dense Candidate Boxes with Gaussian distributed center points.

in Fig. 2(b). Diff3Det extends DiffusionDet to 3D object detection by projecting 3D detection boxes onto the Bird's Eye View (BEV) to obtain BEV 2D boxes. The five dimensions (x, y, l, w, θ) of these boxes are treated as Gaussian noise within the feature range as shown in Fig. 2(c). Since the BEV features tend to the global representation, simply treating all five degrees of freedom of the projected detection box as Gaussian noise and aggregating BEV features of the corresponding boxes by RoIAlign [33] will increase the complexity of feature extraction.

Based on the above observations, in this paper, we propose a novel model DiffCandiDet based on dense candidate boxes with Gaussian distributed center points (GDCP) to leverage the powerful denoising and reconstruction capabilities of the diffusion model. Instead of fixing the 2D center point coordinates (x and y) of candidate boxes, we model them to follow a Gaussian distribution within the boundary range of the BEV map. DiffCandiDet achieves the dual objective of incorporating the Gaussian randomness of the diffusion model while preserving the height, shape, and orientation characteristics of dense candidate boxes. Compared to traditional anchor-based methods such as Voxel-RCNN [25] and CasA+V [21], DiffCandiDet introduces the iterative denoising process of diffusion models into 3D object detection, incorporating initial randomness. Since the model learns the mapping from randomly initialized bounding boxes to ground truth boxes, each step of the iterative inference becomes more accurate, resulting in enhanced generalization capability and superior detection performance. As illustrated in Fig. 3, the comparative analysis between DiffCandiDet and anchor-based methods (Voxel-RCNN and CasA+V) reveals two critical advantages of our approach:

(1) Error Tolerance via Multi-Step Refinement: The iterative denoising process of the diffusion model progressively rectifies detection inaccuracies through multi-step optimization, significantly enhancing robustness to initialization errors.

(2) Feature Consistency through Geometric Proximity: Anchors initialized closer to ground truth (GT) boxes exhibit higher spatial feature alignment (measured by IoU-driven similarity metrics), which reduces the learning objective's complexity by constraining the solution space.

This dual mechanism of probabilistic refinement and geometry-aware initialization collectively addresses the limitations of conventional anchor-dependent frameworks. Besides, the conventional anchor-based methods need a one-to-one correspondence between the BEV features and the positions of fixed candidate boxes, while GDCP are distributed at arbitrary positions within the voxel grids instead of the central locations of voxel grids. To address this, we employ the

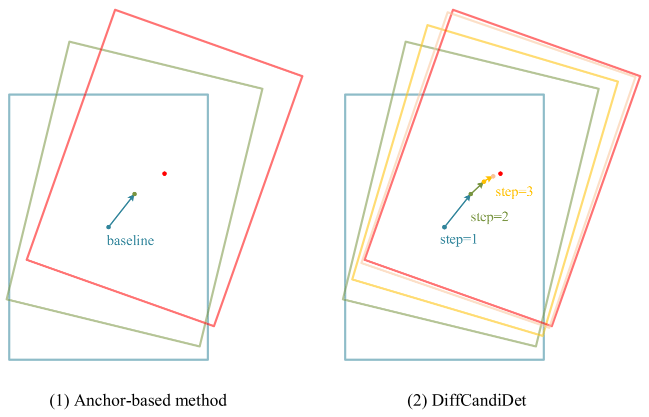

Fig. 3. Compared to conventional anchor-based methods, the iterative denoising mechanism of diffusion models progressively rectifies detection errors through multi-step optimization, thereby achieving superior error tolerance and geometric consistency compared to rigid anchor initialization strategies.

bilinear interpolation [12] to represent the features of center points as a weighted average of the distances from the features of surrounding fixed anchor points.

In the existing threshold-based positive sample selection strategy in the Region Proposal Networks (RPN), each GT has at least one candidate box. However, the IoUs of some GTs and their best-matched candidate boxes are still lower even far lower than the matched threshold, especially for those objects with a significant disparity in size from the candidate boxes or those in turning, thereby increasing the learning difficulty. Consequently, these positive samples that fail to influence the model during the first stage, leading to a smaller probability of being selected as positive samples to enter the subsequent second stage. Therefore, we propose a Super-dense Candidate Boxes (SDCB) strategy to enhance the initial IoU of GT boxes and the matched positive samples. The combining of GDCP and SDCB prevents the issue of insufficient candidate boxes within a small region, which is helpful for the detection of pedestrians walking side by side. Additionally, given the same number of candidate boxes for each category, the number of large objects selected as positive samples is much greater than that of small objects, which easily causes an imbalance in sample quantity and learning difficulty. Hence, we propose a Dynamic Number of Candidate Boxes (DNCB) strategy for different sizes of objects to achieve the learning balance across different categories for multi-class 3D object detection.

In addition to Gaussian distribution, we also propose a Dynamic Super-dense Candidate Boxes (DSCB inherently integrates two core components: the SDCB and DNCB) strategy based on discrete uniform distribution (DUCandiDet) and continuous uniform distribution (CUCandiDet), to reduce runtime and improve robustness.

We have applied diffusion models to 3D object detection and achieved competitive results as shown in Fig. 1, confirming the feasibility and superiority of diffusion models for 3D object detection, and providing a novel paradigm for enhancing the effectiveness of diffusion models in 3D object detection tasks. Our contributions are as follows:

(1) We propose DiffCandiDet for dense candidate boxes with Gaussian distributed center points (GDCP) in the RPN, which effectively integrates the anchor-based method and the Gaussian random noise-based method.

(2) We propose a Super-dense Candidate Boxes (SDCB) strategy to enhance the initial IoU between GT boxes and candidate boxes for objects with significantly different sizes or those undergoing turns. We propose a Dynamic Number of Candidate Boxes (DNCB) strategy to achieve the learning balance across categories of varying sizes.

(3) Dynamic Super-dense Candidate Boxes (DSCB) strategy with center points of discrete uniform distribution (DUCandiDet) and continuous uniform distribution (CUCandiDet) are proposed as an auxiliary rule to reduce the runtime consumption and enhance the robustness of the model.

(4) Our DiffCandiDet, DUCandiDet, and CUCandiDet can be applied to both LiDAR-only and multi-model anchor-based detection models, and we have achieved state-of-the-art results. **DiffCandiDet ranks 1st** on the KITTI validation set in the Car and Pedestrian detection leaderboard.

2. Related work

2.1. Anchor-based 3D object detection

Anchor-based methods [34–37] typically predefine a set of equally spaced 3D rectangular boxes in the RPN, referred to as anchors. Due to the similarity in shape and size of the same category, identical shapes and sizes are assigned to anchors for each category, while variations exist across different categories. Voxel-RCNN [25] fully leverages coarse-grained voxel features to achieve detection accuracy compared to point-based models. PV-RCNN [12] leverages the strengths of point and voxel by Voxel Set Abstraction module to fuse keypoint, voxel, and BEV features. CT3D [38] leverages the high-quality region proposal network and introduces a Channel-wise Transformer architecture. Graph R-CNN [20] utilizes Dynamic Point Aggregation to sample points and uses RoI-graph Pooling to iteratively aggregate and update the features of each node and its neighbors. CasA [21] introduces a cascade attention module that adds the attention mechanism to cascade networks. PDV [14] considers point density as a feature that uses kernel density estimation (KDE) and enhances self-attention with density-aware positional encoding. TED [24] introduces an efficient Transformation-Equivariant 3D Detector, securing the top rank among all submissions on the KITTI 3D Car detection leaderboard while maintaining competitive efficiency. VirConv [16] introduces StVD (Stochastic Voxel Discard) to mitigate computational issues by discarding a large number of nearby redundant voxels and NRConv (Noise-Resistant Submanifold Convolution) to address noise problems by encoding voxel features in 2D images and 3D LiDAR space. However, the fixed anchors suffer from inherent limitations in adapting to structural incompatibilities with initial object morphologies, resulting in geometric discrepancies between candidate regions and target objects.

2.2. Multi-modal 3D object detection

Multi-modal methods alleviate geometric rigidity constraints by fusing LiDAR point clouds with image semantics, where high-resolution image features enhance the detection of distant targets represented by sparse point cloud data. Multi-modal 3D object detection [39–44] typically involves integrating knowledge from images into point clouds, generating pseudo point clouds through depth completion [45], and fusing image and LiDAR features during backbone network, proposal generation, or RoI refinement stages. PointPainting [46] projects LiDAR points into the output of a pure image semantic segmentation network and attaches class scores to each point. EPNet [47] enhances point features with semantic image features on a point-wise basis without any image annotations. SFD [48] utilizes depth completion to generate pseudo-point clouds and employs 3D Grid-wise Attentive Fusion to leverage information from different types of point clouds. Additionally, Color Point Convolution is used to simultaneously explore 2D image features and 3D geometric features of the pseudo-point clouds. LoGoNet [23] projects the proposal grid centers onto images and samples the surrounding image pixels to achieve local fusion and utilizes point centroids for better cross-modal alignment to achieve global fusion.

2.3. Diffusion model for 3D object detection

Compared to prior approaches, diffusion models address geometric rigidity constraints by implicitly learning proposal distributions through a noise-addition and noise-removal process, eliminating the need for predefined anchors. Furthermore, their Markov chain-based optimization enables step-by-step refinement of predictions, effectively mitigating error accumulation. More importantly, the iterative denoising mechanism in diffusion probabilistic models inherently facilitates efficient cross-modal feature fusion through progressive uncertainty reduction. The diffusion model [49–51] has achieved tremendous success in image generation. DDPM [52] pioneers applying the diffusion model to image generation, conceptualizing image generation as a Markov chain. This involves a forward process that introduces noise to GT images gradually and a reverse process that reconstructs clean images from pure Gaussian noise images. DDIM [53] removes the constraint of the inference process being a Markov chain, enabling step-skipping acceleration. DiffusionDet [31] pioneers the application of the diffusion model in 2D object detection tasks, which considers the bounding boxes in the RPN as Gaussian noise boxes. Diff3Det [54] extends DiffusionDet to 3D object detection by treating the candidate boxes as Gaussian noise boxes in the RPN. However, since BEV features inherently capture global spatial representations, naively treating all five degrees of freedom (DoFs) of projected detection boxes as Gaussian noise and aggregating BEV features via RoIAlign introduces significant computational overhead during feature extraction. DiffRef3D [55] considers the residuals of GT and 3D proposal boxes as the noise and introduces conditional diffusion into 3D object detection. However, DiffRef3D learns a mapping from arbitrary boxes to GT boxes, which poses high optimization challenges and yields limited performance gains. DiffBEV [56] utilizes the denoising and recovery capabilities of conditional diffusion models to generate more comprehensive BEV feature representations, aiming to capture fine-grained object details such as precise boundaries for 3D object detection and highly detailed shapes for BEV semantic segmentation. However, its focus on feature-level denoising diverges from DiffCandiDet‘s hybrid detection paradigm.

3. Method

As shown in Fig. 4, we design DiffCandiDet using diffusion probabilistic models based on Dynamic Super-dense Candidate Boxes (DSCB) strategy with Gaussian-distributed center points, which achieve geometry-aware proposal generation and advanced results, confirming the feasibility and superiority of diffusion models in 3D object detection. Furthermore, we introduce DUCandiDet and CUCandiDet as ablation variants of DiffCandiDet, where the center points are sampled from discrete uniform distribution and continuous uniform distribution respectively.

3.1. Preliminaries

3.1.1. Denoising diffusion model

The DDIM [53] includes both forward process q and reverse process p as shown in Fig. 2(a). Assuming the original data follows the distribution $q = q(z_0)$. In the forward process, Gaussian noise is gradually added to the original data z_0, transforming it into z_1, z_2, z_3 until pure Gaussian noise z_t. The forward process is defined as follows:

$$q(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}) := \mathcal{N}(\boldsymbol{z}_t; \sqrt{1-\beta_t}\boldsymbol{z}_{t-1}, \beta_t \boldsymbol{I}) \tag{1}$$

where $\boldsymbol{z}_t$ is noise sample at timestep $t \in \{0, 1, \ldots, T\}$, $\beta_t \in (0, 1)$ is variance schedule of the added noise. Hence, we can directly sample data z_t with noise at an arbitrary timestep t as follows:

$$q\left(\boldsymbol{z}_t \mid \boldsymbol{z}_0\right) := \mathcal{N}\left(\boldsymbol{z}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{z}_0, \left(1-\bar{\alpha}_t\right)\boldsymbol{I}\right) \tag{2}$$

where $\bar{\alpha}_t := \prod_{i=0}^{t} \alpha_i = \prod_{i=0}^{t}(1-\beta_i)$, $\boldsymbol{z}_0$, $\boldsymbol{z}_t$, and β_i represent the sample, pure Gaussian noise sample, and noise variance schedule, respectively.

During training, the progressively denoised pure Gaussian noise inversely reverses the forward process. Specifically, we train a neural network $f_\theta(\boldsymbol{z}_t, t)$ to predict z_0 from z_t by minimizing the training objective with ℓ_2 loss as follows:

$$\mathcal{L}_{train} = \frac{1}{2}\| f_\theta(\boldsymbol{z}_t, t) - \boldsymbol{z}_0\|^2 \tag{3}$$

During inference, we iteratively recover $\boldsymbol{z}_0$ from the arbitrary pure Gaussian noise $\boldsymbol{z}_t$ using the trained model f_θ. The iterative inference process is as follows:

$$\boldsymbol{z}_T \to \boldsymbol{z}_{T\text{-}\Delta} \to \ldots \to \boldsymbol{z}_0 \tag{4}$$

The step-skipping in DDIM enhances the inference speed compared with DDPM.

3.2. Diffusion model with Gaussian distributed center points

Applying the diffusion model to detection tasks essentially involves treating the detection boxes as Gaussian noise. For example, as shown in Fig. 2(b), DiffusionDet [31] considers the center point and size of a 2D anchor (x_a, y_a, w_a, h_a) as Gaussian noise, where $x_a, w_a \in (0, w_I)$ and $y_a, h_a \in (0, h_I)$. w_I and h_I represent the weight and height of the image. As shown in Fig. 2(c), Diff3Det [54] treats the top view of a 3D anchor $(x_a,\ y_a,\ w_a,\ l_a,\ \theta_a)$ as Gaussian noise, where $i_a \in (l_i, u_i)$ and $i \in \{x, y, w, h, \theta\}$. l_x, l_y, u_x and u_y represent the lower and upper bounds of the width and length of a BEV map. l_w, l_h, u_w and u_h denote the lower and upper bounds of the manually set length and width of an anchor. l_θ and u_θ are the lower and upper bounds of the orientation angle of an anchor. Although treating multiple degrees of freedom of a bounding box as Gaussian noise in the RPN is feasible, introducing multiple degrees of freedom in candidate boxes significantly increases uncertainty. Empirically, simultaneous learning with more degrees of freedom tends to increase the complexity and learning difficulty. How to design a reasonable model that can not only utilize the powerful denoising and repair capabilities of the diffusion model but also make good use of the high performance of the existing anchor-based and multi-modal methods is a valuable question.

To solve the above challenges, we innovatively propose a dense candidate boxes strategy with Gaussian distributed center points (GDCP) in the RPN as shown in Fig. 2(c). Specifically, we consider the center points (All the center points in the following text refer to x and y coordinates of center points) of the candidate boxes as Gaussian noise within the boundary range of the BEV map and form dense candidate boxes on this basis. The diffusion model aims to learn complex mappings of center points from a Gaussian distribution to the target distribution. We assume that Gaussian distributions better reflect real-world distributions than uniform distributions for autonomous driving scenarios, particularly considering two key factors:

(1) LiDAR Sensing Characteristics: The point cloud data exhibits a density-distance correlation, where distant objects with excessively sparse point clouds are typically unlabeled as GT and thus excluded from the learning, causing the detected objects to predominantly cluster near the ego vehicle.

(2) Spatial Distribution Patterns: Occlusions caused by static structures (e.g., buildings) and natural traffic patterns restrict LiDAR scanning coverage. Consequently, detectable objects are spatially concentrated around the autonomous vehicle due to these visibility constraints.

The center points of these candidate boxes are not in the same positions while the previous candidate boxes are all located in the center of the grid cell of the BEV feature map as shown in Fig. 4(a) (c). Using nearest-neighbor interpolation [57] is a straightforward method, however, the candidate boxes strategy with GDCP requires one-to-one correspondence between center points and BEV features. Therefore, we apply the bilinear interpolation [12,58] where the features of Gaussian noise points are represented as a weighted average of the distances

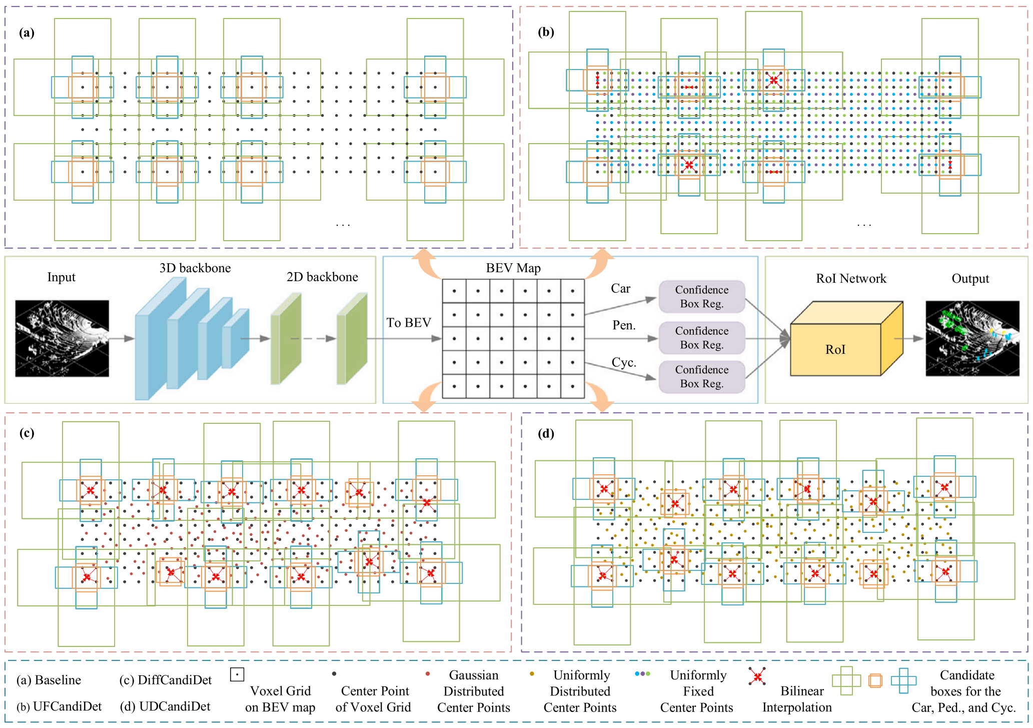

Fig. 4. After voxelization of the input point clouds and subsequent processing through the 3D and 2D backbone, the dense candidate boxes are generated in the different Region Proposal Networks (a), (b), (c), and (d) respectively. (a) previous dense candidate boxes, (b) dynamic super-dense candidate boxes with discrete uniform center points, (c) dynamic super-dense candidate boxes with Gaussian-distributed center points, and (d) dynamic super-dense candidate boxes with continuous uniform center points. Following this, the fully connected (FC) layers are established for confidence and regression, which are separately designed for different categories. After that, the output is obtained from the RoI networks in the second stage. Finally, the denoising diffusion implicit model (DDIM) is utilized to evaluate the distribution of the center points in the next step of DiffCandiDet.

of the center points and its four adjacent fixed grid cell center points separately, as illustrated in Fig. 4(c). In this way, the original data $q = q(z_0)$ represents x_a and y_a of candidate boxes. This allows the diffusion model to be applied to dense anchor-based models. Compared to conventional anchor-based methods, the iterative denoising mechanism of diffusion models progressively rectifies detection errors through multi-step optimization shown in Fig. 3. Notably, the diffusion model with GDCP can effectively integrate the advantage of the anchor-based methods and the Gaussian random noise-based method to leverage the powerful denoising and reconstruction capabilities of the diffusion model. The training and inference process are as follows:

3.2.1. Training

As illustrated in Algorithm 1, we initially extract voxel features using the encoder. Then we incorporate additional center points by padding GT center points with Gaussian distributed center points to a fixed number N_{train} since the number of GT boxes varies in each frame of point clouds. Subsequently, Gaussian noise is added to the padded GT center points, with the noise scale controlled by $\bar{\alpha}_t$ as follows:

$$z_t = \sqrt{\bar{\alpha}_t} z_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon, \quad \text{where} \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{5}$$

The value of $\bar{\alpha}_t$ at different time steps t follows a monotonically decreasing cosine schedule, as proposed in [59]. Following this, dense candidate boxes are generated based on the perturbed center points. We obtain the corresponding features for the candidate boxes through bilinear interpolation [12]. Afterward, RoIs are obtained from the predicted boxes in the RPN by selecting the high-scoring boxes and applying Non-Maximum Suppression (NMS). The refined network extracts RoI features to generate the center points of the final predicted boxes as follows:

$$z_{roi} = f_\theta(z_t, t, f_{bev}, f_{roi}) \tag{6}$$

where f_θ denotes the model of the baseline. f_{bev} and f_{roi} represent the future of BEV map and RoI. The detector is then trained by minimizing

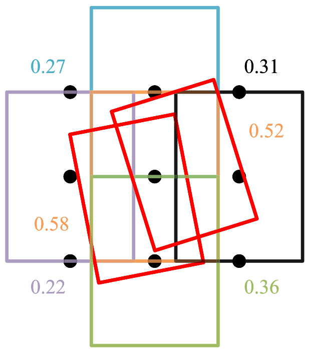

Fig. 5. When two pedestrians are clustered in a small area, their GT boxes often share a single candidate box with the highest IoU.

the residual between the center points of GT boxes and dense candidate boxes in the RPN as follows:

$$\mathcal{L} = \mathcal{L}_{reg}(z_{rpn}, z_0) + \mathcal{L}_{refine}(z_{roi}, z_0) \tag{7}$$

where $\mathcal{L}_{reg}$ represents the regression loss in the RPN. $\mathcal{L}_{refine}$ denotes the loss of the refining stage. z_{rpn} denotes the center points of the predicted boxes in the RPN. The refining stage of the detector follows the baselines.

3.2.2. Inference

As illustrated in Algorithm 2, we generate Gaussian random center points z_t and produce dense candidate boxes based on them. In the iterative inference steps, predicted boxes are obtained, and then we utilize DDIM to evaluate the distribution of center points of predicted boxes in the next step as follows:

$$z_{next} = ddim\_step(z_t, z_{pred}, t, t_{next}) \tag{8}$$

Algorithm 1 Training

```
def train_loss(point_clouds, gt_box):
  # Point_clouds: [B, N_point, 3]
  # Centers of gt_boxes: [B, *, 2]
  # N: number of boxes' centers
  # Encode BEV features
  feats = point_clouds_encoder( point_clouds )
  # Pad centers of gt_boxes to N
  pc = pad_centers(centers of gt_boxes)
  # Padded centers of boxes: [B, N, 2]
  pc = (pc * 2 - 1) * scale               # Signal scaling
  # Corrupt centers of gt_boxes
  t = randint(0, T)                       # time step
  eps = normal(mean=0, std=1)             # noise: [B, N, 2]
  pc_crpt = sqrt(   alpha_cumprod(t)) * pc +
            sqrt(1 - alpha_cumprod(t)) * eps
  pb_crpt = generate_anchor(p_crpt)
  # Predict
  pb_pred = detection_decoder(pb_crpt, feats, t)
  # Set prediction loss
  loss = set_prediction_loss(pb_pred, gt_boxes)
  return loss
```

Algorithm 2 Sampling

```
def infer(point_clouds, steps, T):
  # Point_clouds: [B, N_point, 3]
  # Steps: number of sample steps
  # T: number of time steps
  # Encode BEV features
  feats = point_cloud_encoder(point_clouds)
  # Noisy centers of boxes: [B, N, 2]
  pc_t = normal(mean=0, std=1)          #noisy centers
  pb_t = generate_anchor(pc_t)          #noisy candidate boxes
  # Uniform sample step size
  times = reversed(linespace(-1, T, steps))
  # [(T-1, T-2), (T-2, T-3), ..., (1, 0), (0, -1)]
  time_pairs = list(zip(times[:-1], times[1:])
  for t_now, t_next in zip(time_pairs):
    # Predict pb_0 from pb_t
    pb_pred = detection_decoder(pb_t, feats, t_now)
    # Estimate pb_t at t_next
    pb_t = ddim_step(pb_t, pb_pred, t_now, t_next)
    pc_t = center_renewal(pb_t)          # center renewal
    pb_t = generate_anchor(pc_t)
  return pb_pred
```

where z_{pred} denotes the center points of final predicted boxes. z_{next} represents the center points of the next step t_{next}. The predicted boxes are divided into desired boxes near the bounding boxes of GT with higher scores and undesired boxes at arbitrary positions with relatively lower scores, respectively. Inputting undesired boxes into the next step can lead to performance degradation since their distribution is not constructed from perturbed boxes during training. Therefore, object boxes with scores exceeding a certain threshold μ are retained, while those below μ are discarded to align with training consistency better. The center points of retained object boxes are concatenated with the randomly regenerated center points for iterative inference.

3.3. Super-dense candidate boxes strategy

For most multi-class 3D object detection models, traditional anchor-based methods are uniformly spaced on the BEV plane which struggles to detect pedestrians walking side by side. When multiple small objects cluster together within a confined area, they may share only one candidate box with a high Intersection over Union (IoU) score, ultimately leading to missed detections. Specifically, when two pedestrians walking side by side share the same candidate box selected as a positive sample (above the matched threshold θ_{match}) or ignored sample (between the unmatched threshold $\theta_{unmatch}$ and θ_{match}), while the other candidate boxes are negative samples (below $\theta_{unmatch}$), only one pedestrian can be predicted and another one is missed predicted, as shown in Fig. 5. Therefore, we propose a Super-dense Candidate Boxes (SDCB) strategy. Notably, combining the SDCB strategy and candidate boxes with GDCP can avoid missed detection for pedestrians walking side by side by ensuring multiple candidate boxes within a densely packed region of interest, which reduces the probability of multiple objects sharing a candidate box.

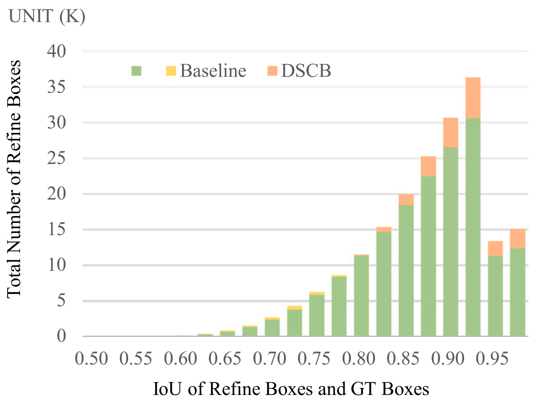

Fig. 6. The proposal quality in the refinement stage improves by a large margin when using DSCB on the KITTI training set. The orange parts represent the improvements by adding DSCB.

Furthermore, the conventional dense candidate box strategy tends to face greater difficulty in learning the residual between candidate boxes and GT boxes since the lower initial IoU between them especially for small objects on average as illustrated in Fig. 7. When the density of candidate boxes is increased to four times that of the previous candidate boxes, the average IoU between GT and the four adjacent candidate boxes rises by a large margin for the Car, Pedestrian, and Cyclist, respectively. Therefore, as the number of candidate boxes becomes denser, the number of candidate boxes selected as positive samples increases. The detailed configuration of SDCB is reflected in the next paragraph.

3.4. Dynamic number of candidate boxes strategy

To balance the proportion of different categories selected as positive samples and reduce the residual between the candidate boxes and the GT at the beginning, we propose a Dynamic Number of Candidate Boxes (DNCB) strategy for multi-class 3D object detection. Specifically, we set the number of candidate boxes positively correlated with the diagonal of the candidate boxes. Since we prefer small objects to have a greater number of candidate boxes empirically. The formula is as follows:

$$Num_{\{car,ped.,cyc.\}} = \{N, N+k, N + \lfloor (k * \frac{L_{cyc.}}{round(L_{ped.})}) \rfloor\} \tag{9}$$

$$L_{c\in\{car,ped.,cyc.\}} = \sqrt{W^2_{c\in\{car,ped.,cyc.\}} + D^2_{c\in\{car,ped.,cyc.\}}} \tag{10}$$

where N denotes the number of candidate boxes for the Car, which is the basis of other categories. k serves as a scale for quantifying the disparity in the number of candidate boxes between the Car and other categories. N and k joint control the DSCB strategy. $round(x)$ represents rounding to the nearest integer. W_c, D_c, and L_c represent the width, depth, and diagonal length of the c category anchor in the BEV map, respectively. The ratio between L_{cyc} and L_{ped} also serves to balance the number of candidate boxes between the Pedestrian and Cyclist.

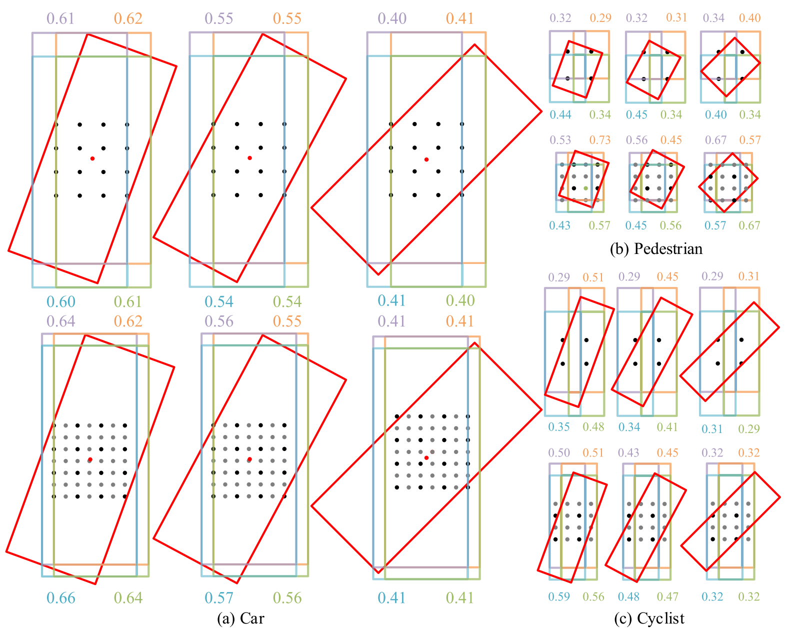

Fig. 7. The IoU between candidate boxes with different center points and orientations and their adjacent four candidate boxes is computed. The average IoU increased from 0.610, 0.545, and 0.405 to 0.640, 0.560, and 0.410 for the Car category. The average IoU increased from 0.348, 0.355, and 0.370 to 0.565, 0.503, and 0.620 for the Pedestrian category. The average IoU increased from 0.410, 0.355, and 0.300 to 0.540, 0.458, and 0.320 for the Cyclist category.

3.5. *Distribution of center points*

To reduce the runtime consumption, we also propose DUCandiDet using DSCB with discrete uniformly distributed center points strategy as shown in Fig. 4(b). DiffCandiDet's candidate boxes need to be updated with changes in the center points each time. However, DUCandiDet allows for the pre-definition of candidate boxes and their center points, thus saving inference time. In addition to Gaussian distribution, a natural progression leads to CUCandiDet using DSCB with continuous uniformly distributed center points strategy which introduces randomness and improves robustness compared to DUCandiDet as shown in Fig. 4(d). Note that DUCandiDet and CUCandiDet are introduced solely as ablation studies to validate the distinct distribution characteristics under different noise assumptions.

3.6. *Diffusion-based 3D object detection*

For DiffRefDet, the forward process involves adding noise to the residuals of GT and 3D proposal boxes (x_r, y_r, z_r, l_r, w_r, h_r, θ_r) as z_0 until Gaussian noise samples z_t. The reverse process entails training a model to recover the residuals from Gaussian noise samples z_t. Therefore, the iterative inference of DiffRefDet includes seven dimensions (x, y, z, l, w, h, θ), resembling DiffusionDet's four dimensions (x, y, w, h) in image and Diff3Det's five dimensions (x, y, l, w, θ) on BEV map. In contrast, the iterations in DiffCandiDet have two degrees of freedom (x and y) which could integrate the 2D BEV features by bilinear interpolation. Note that DiffCandiDet can employ diffusion models in one-stage (SECOND), two-stage (Voxel-RCNN, PV-RCNN, CasA+V), and multi-modal (TED-M and LoGoNet) models, shown its university. In contrast, DiffRefDet requires refinement stages in two-stage models and Diff3Det can be only employed in one-stage models.

3.7. *Loss function*

The losses in the RPN comprise the classification loss and the box regression loss, represented as

$$\mathcal{L}_{rpn} = \beta_1 \cdot \mathcal{L}_{cls} + \beta_2 \cdot \mathcal{L}_{reg} \tag{11}$$

where β_1 and β_2 are the constant factors for the loss terms, respectively. The all loss for two-stage models is represented as

$$\mathcal{L} = \mathcal{L}_{rpn} + \mathcal{L}_{roi} \tag{12}$$

where $\mathcal{L}_{roi}$ represents the loss of RoI head in the refining stage. Furthermore, the loss of diffusion model with GDCP is included in $\mathcal{L}_{reg}$ and $\mathcal{L}_{roi}$, which are the regression of center points x and y.

4. Experimental results

4.1. *KITTI dataset*

The KITTI dataset contains 7481 LiDAR frame samples and 7518 samples for training and testing respectively. Following [12,25], we split the training samples into 3712 frames and 3769 frames as training set and validation set respectively. The primary evaluation metric is the 3D Average Precision (AP) computed at 40 recall thresholds (R40). Each class is categorized into three levels: easy, moderate, and hard. The IoU thresholds in this metric are 0.7, 0.5, and 0.5 for the Car, Pedestrian, and Cyclist, respectively. When submitting test results to the official website, we train the models using 80% of the training data without any Test-Time Augmentation (TTA) and the prediction ensemble from multiple models.

4.2. *Waymo open dataset*

The Waymo open dataset consists of 798 and 202 sequences with 158361 and 40077 LiDAR frames for training and validation, respectively. The evaluation metrics are mean AP (mAP) (L1 and L2), mAPH (L1 and L2), where L1 and L2 denote the detection difficulty level. The mAPH metric takes into account object heading accuracy. The IoU thresholds in this metric are 0.7, 0.5, and 0.5 for the Vehicle, Pedestrian, and Cyclist, respectively.

4.3. *Implementation settings*

4.3.1. *Pre-processing*

For KITTI, the input point cloud range is limited to [0, 70.4] meters on the X axis, [−40, 40] meters on the Y axis, and [−3, 1] meters on

Table 1
The 3D detection results of DiffCandiDet, DUCandiDet, and CUCandiDet on the KITTI val set using SECOND, PV-RCNN, Voxel-RCNN, CasA, and TED as the baseline with AP calculated by 40 recall positions. DiffCandiDet, DUCandiDet, and CUCandiDet achieve the most substantial performance gains in pedestrian detection, with cyclists also showing notable improvements, albeit slightly less pronounced. While car detection exhibits more modest gains, all three methods consistently enhance robustness in hard scenarios, demonstrating their universal effectiveness across object categories. † indicates our reproduced results. ‡ represents the results obtained by running the author's source code. * denotes that the reported runtime is obtained by testing with the original paper's provided weight files on the same GPU configuration used for running DiffCandiDet. The best results are in bold.

Method	Stage	Modality	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)			Runtime
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
SECOND† [10]	one	LiDAR	90.21	81.66	78.72	–	–	–	–	–	–	**60.8**
DiffCandiDet-SE (ours)	one	LiDAR	**90.98**	**82.57**	**79.80**	–	–	–	–	–	–	94.0
CUCandiDet-SE (ours)	one	LiDAR	90.77	81.78	79.14	–	–	–	–	–	–	74.8
Improvement	–	–	+0.77	+0.91	+0.42	–	–	–	–	–	–	-
PV-RCNN† * [12]	two	LiDAR	92.10	84.36	82.48	62.71	54.49	49.88	89.10	70.38	66.02	82.3
DiffCandiDet-PV (ours)	two	LiDAR	92.53	85.31	83.06	**68.72**	61.79	57.23	91.22	72.04	68.87	193.0
DUCandiDet-PV (ours)	two	LiDAR	92.15	84.75	82.55	68.60	**62.62**	**58.32**	91.92	72.39	68.54	**52.8**
CUCandiDet-PV (ours)	two	LiDAR	**92.66**	**85.44**	**83.23**	68.06	60.54	54.29	**92.38**	**72.47**	**69.29**	93.6
Improvement	–	–	+0.65	+1.08	+0.75	+6.01	+8.13	+8.44	+3.28	+2.02	+3.27	-
Voxel-RCNN† [25]	two	LiDAR	92.53	85.03	82.56	66.35	59.06	54.02	89.52	72.62	68.32	**40.0**
DiffCandiDet-Vo (ours)	two	LiDAR	92.49	85.13	**82.94**	**72.35**	**65.73**	59.27	92.87	**75.98**	**71.41**	134.2
DUCandiDet-Vo (ours)	two	LiDAR	92.13	84.79	82.48	71.23	65.64	**61.00**	90.95	72.43	68.03	98.2
CUCandiDet-Vo (ours)	two	LiDAR	**92.65**	**85.46**	82.14	68.51	63.27	56.81	**93.60**	74.11	69.49	60.0
Improvement	–	–	+0.12	+0.43	+0.38	+6.00	+6.66	+6.98	+4.08	+3.36	+3.09	–
CasA+V† * [21]	two	LiDAR	93.21	86.37	83.93	73.95	66.62	59.97	92.78	73.94	69.37	32.6
DiffCandiDet-Ca (ours)	two	LiDAR	92.43	85.54	83.22	**81.07**	**73.70**	**68.34**	94.51	**75.03**	70.62	195.8
DUCandiDet-Ca (ours)	two	LiDAR	93.17	**86.50**	**83.98**	77.59	68.78	61.58	94.19	73.47	69.00	**30.8**
CUCandiDet-Ca (ours)	two	LiDAR	92.98	86.07	83.70	78.27	67.67	60.69	**94.56**	74.36	**71.65**	61.3
Improvement	–	–	−0.04	+0.13	+0.05	+7.12	+7.08	+8.37	+1.78	+1.09	+2.28	–
TED-M† [24]	two	LiDAR+RGB	95.25	88.94	86.73	–	–	–	–	–	–	**166.2**
DiffCandiDet-TEDM (ours)	two	LiDAR+RGB	95.74	**89.21**	86.61	–	–	–	–	–	–	583.5
DUCandiDet-TEDM (ours)	two	LiDAR+RGB	95.61	89.05	86.57	–	–	–	–	–	–	199.1
CUCandiDet-TEDM (ours)	two	LiDAR+RGB	**95.84**	89.03	86.58	–	–	–	–	–	–	279.2
Improvement	–	–	+0.59	+0.27	−0.12	–	–	–	–	–	–	–
LoGoNet‡ [23]	two	LiDAR+RGB	91.41	84.80	84.48	70.03	63.31	59.04	90.64	72.78	68.79	68.4
DiffCandiDet-Lo (ours)	two	LiDAR+RGB	92.03	**85.46**	**85.07**	**75.44**	**69.14**	**63.83**	90.96	**74.14**	**71.03**	58.8
DUCandiDet-Lo (ours)	two	LiDAR+RGB	92.12	85.34	84.94	73.26	67.83	63.56	**91.01**	73.37	69.67	**54.7**
CUCandiDet-Lo (ours)	two	LiDAR+RGB	**92.16**	85.35	84.96	69.79	65.01	60.31	90.29	72.62	70.24	72.8
Improvement	–	–	+0.75	+0.66	+0.59	+5.41	+5.83	+4.79	+0.37	+1.36	+2.24	–

the Z axis. The voxel size is set to [0.05, 0.05, 0.1] meters. For Waymo Open Datasets, the input point cloud range is limited to [−75.2, 75.2] meters on the X axis, [−75.2, 75.2] meters on the Y axis, and [−2, 4] meters on the Z axis. The voxel size is set to [0.1, 0.1, 0.15] meters. We follow all the baselines to perform data augmentation. We have also removed GT Sampling augmentation for the Pedestrian in multi-class 3D object detection only on the KITTI dataset.

4.3.2. *Training details*

To demonstrate the universality and superiority of DiffCandiDet, UFCandiDet, and UDCandiDet, we conducted experiments on six popular baseline detectors: SECOND, PV-RCNN, Voxel-RCNN, CasA, TED and LoGoNet. For each detector on each dataset, we train a single model for three classes (one class in SECOND and TED-M). We train all the detectors on a single 4090 GPU card (a single 3060 GPU card for TED-M) with a batch size of four except for two in PV-RCNN and one in TED-M. The loss functions of DiffCandiDet, DUCandiDet, and CUCandiDet are all consistent to the loss functions in their baseline. DiffCandiDet, DUCandiDet, and CUCandiDet are optimized using the Adam optimizer [60], a division factor of 10, a momentum range of [0.95, 0.85], and a weight decay of 0.01. The maximum learning rate is set to 0.01 for both the KITTI and Waymo datasets. We trained all the models on KITTI and Waymo datasets for 80 and 30 epochs, respectively.

4.4. *Comparison with state-of-the-art methods*

4.4.1. *KITTI validation set*

We compare DiffCandiDet, DUCandiDet, and CUCandiDet with the baselines in Table 1. DiffCandiDet, DUCandiDet, and CUCandiDet all improve PV-RCNN, Voxel-RCNN, CasA, TED, and LoGoNet by a large margin, demonstrating the efficacy of these methods. Specifically, for baseline PV-RCNN, DiffCandiDet-PV improves 3D AP (R40) performance by +6.01%, +7.30%, and +7.35% for the Pedestrian, and +2.12%, +1.66%, and +2.85% for the Cyclist, respectively. For baseline Voxel-RCNN, DiffCandiDet-Vo improves 3D AP (R40) performance by +6.00%, +6.67%, and +5.25% for the Pedestrian, and +3.35%, +3.36%, and +3.09% for the Cyclist, respectively. DiffCandiDet-Ca improves 3D AP(R40) performance by +7.12%, +7.08%, and +8.37% for the Pedestrian and +1.73%, +1.09%, and +1.25% for the Cyclist, respectively, which achieves unprecedented results for the Pedestrian on KITTI val set. The more pronounced performance gains observed in smaller object categories (e.g., Pedestrian and Cyclist) can be primarily attributed to the diffusion framework's iterative refinement mechanism. During the later stages of the multi-step optimization process, candidate proposals initialized closer to GT boxes achieve enhanced BEV feature alignment and spatial consistency, thereby reducing the learning complexity for precise localization compared to static anchor-based initialization strategies. For multi-modal paradigms, DiffCandiDet-TEDM achieves state-of-the-art results for the Car on the KITTI val set based on TED-M. DiffCandiDet-Lo improves 3D AP(R40) performance by +5.41%, +5.83%, and +4.79% for the Pedestrian based on LoGoNet. These performance improvements are fundamentally attributed to the diffusion model's iterative denoising process, which progressively rectifies detection inaccuracies through multi-step optimization, while significantly enhancing robustness to initialization variances compared to conventional anchor-based paradigms. We compare DiffCandiDet with previous methods in Table 2. DiffCandiDet outperforms all the LiDAR-based and multi-modal models. And our DUCandiDet and CUCandiDet also outperform all the baseline on the KITTI val set. Additionally, DUCandiDet achieves both good performance and reduced inference time, making it suitable for real-time 3D object detection. Notably,

Table 2
3D detection results on the KITTI validation set for the car, pedestrian, and cyclist categories with AP calculated by 40 recall positions. DiffCandiDet outperforms all of the previous LiDAR-only, multi-modal and diffusion-based methods. The best results are in bold.

Method	Modality	Car 3D (R40)				Pedestrian 3D (R40)				Cyclist 3D (R40)			
		Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP
PV-RCNN [12]	LiDAR	92.10	84.36	82.48	86.31	62.71	54.49	49.88	55.69	89.10	70.38	66.02	75.17
Voxel-RCNN [25]	LiDAR	92.53	85.03	82.56	86.71	66.35	59.06	54.02	59.81	89.52	72.62	68.32	76.82
PG-RCNN [61]	LiDAR	92.73	85.26	82.83	86.94	68.44	60.63	55.36	61.48	93.84	74.85	70.15	79.61
PDV [14]	LiDAR	92.56	85.29	83.05	86.97	66.90	60.80	55.85	61.18	92.72	74.23	69.60	78.85
CT3D [38]	LiDAR	92.85	85.82	83.46	87.38	65.73	58.56	53.04	59.11	91.99	71.60	67.34	76.98
Graph-Vo [20]	LiDAR	93.27	86.07	83.12	87.49	–	–	–	–	–	–	–	–
SE-SSD [19]	LiDAR	93.19	86.12	83.31	87.54	–	–	–	–	–	–	–	–
BtcDet [62]	LiDAR	93.15	86.28	83.86	87.76	69.39	61.19	55.86	62.15	91.45	74.70	70.08	78.74
CasA+V [21]	LiDAR	93.21	86.37	83.93	87.84	73.95	66.62	59.97	66.85	92.78	73.94	69.37	78.70
Aug-VirConv [63]	LiDAR	92.62	87.76	85.34	88.57	–	–	–	–	–	–	–	–
DiffCandiDet-Vo (ours)	LiDAR	92.49	85.13	82.94	86.86	72.35	65.73	59.27	65.78	92.87	**75.98**	**71.41**	**80.07**
DiffCandiDet-Ca (ours)	LiDAR	92.43	85.54	83.22	87.06	**81.07**	**73.70**	**68.34**	**74.37**	**94.51**	75.03	70.62	80.05
F-PointNet [64]	LiDAR+RGB	83.76	70.92	63.65	72.78	70.00	61.32	53.59	61.64	77.15	56.49	53.37	62.34
EPNet++ [65]	LiDAR+RGB	92.51	83.17	82.27	85.98	73.77	65.42	59.13	66.11	86.23	63.82	60.02	70.02
CLOCs [66]	LiDAR+RGB	92.78	85.94	83.25	87.32	–	–	–	–	–	–	–	–
Graph-VoI [20]	LiDAR+RGB	95.67	86.87	84.09	88.88	–	–	–	–	–	–	–	–
SFD [48]	LiDAR+RGB	95.52	88.27	85.57	89.79	72.94	66.69	61.59	67.07	93.39	72.95	67.26	77.87
ACF-Net [67]	LiDAR+RGB	93.17	88.80	86.53	89.5	–	–	–	–	–	–	–	–
DiffCandiDet-TEDM (Ours)	LiDAR+RGB	**95.74**	**89.21**	**86.61**	**90.52**	–	–	–	–	–	–	–	–

Table 3
3D detection results on the KITTI test set for the car and cyclist categories using Voxel-RCNN as the baseline with AP calculated by 40 recall positions. DiffCandiDet-Vo achieves competitive results. The best results are in bold.

Method	Modality	Car 3D (R40)				Cyclist 3D (R40)			
		Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP
PV-RCNN [12]	LiDAR	90.25	81.43	76.82	82.83	78.60	63.71	57.65	66.65
PV-RCNN++ [68]	LiDAR	90.14	81.88	77.15	83.06	82.22	67.33	60.04	69.86
SE-SSD [19]	LiDAR	91.49	82.54	77.15	83.73	–	–	–	–
STD [69]	LiDAR	87.95	79.71	75.09	80.92	78.69	61.59	55.30	65.19
SVGA-Net [70]	LiDAR	87.33	80.47	75.91	81.24	78.58	62.28	54.88	65.25
PointPainting [71]	LiDAR+RGB	82.11	71.70	67.08	73.63	77.63	63.78	55.89	65.77
F-ConvNet [72]	LiDAR+RGB	87.36	76.39	66.69	76.81	81.98	65.07	56.54	67.86
CLOCs [66]	LiDAR+RGB	88.94	80.67	77.15	82.25	–	–	–	–
EPNet++ [65]	LiDAR+RGB	91.37	81.96	76.71	83.35	76.15	59.71	53.67	63.18
PDV [14]	LiDAR	90.43	81.86	77.36	83.22	83.04	67.81	60.64	70.50
PointRCNN [73]	LiDAR	86.96	75.64	70.70	77.77	74.96	58.82	52.53	62.10
Diff3Det [54]	LiDAR	89.45	80.86	77.41	82.57	–	–	–	–
Voxel-RCNN [25]	LiDAR	90.90	81.62	77.06	83.19	76.42	62.01	55.94	64.79
DiffRef3D [55]	LiDAR	90.45	81.29	76.66	82.80	80.16	66.61	59.98	68.92
DiffCandiDet-Vo (Ours)	LiDAR	91.18	**82.59**	**77.64**	**83.80**	**83.87**	**67.84**	**60.56**	**70.74**

Table 4
3D detection results on the Waymo validation set by training on 20% of the training samples (approximately 32K frames) and validating on the entire validation set. DiffCandiDet-PV outperforms the baseline using the anchor head. The best results are in bold..

Methods	Car(L1)		Car(L2)		Cyclist(L1)		Cyclist(L2)	
	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
PV-RCNN [12]	75.17	74.53	66.55	65.97	67.02	64.67	64.56	62.31
DUCandiDet-PV	**75.91**	**75.28**	**67.34**	**66.76**	**68.80**	**67.29**	**66.23**	**64.78**

DUCandiDet demonstrates faster inference speeds than several baseline models.

4.4.2. KITTI test set

To further demonstrate the advancement of our DiffCandiDet, we trained DiffCandiDet-Vo using 80% of the training data of the KITTI training and validation set. The results on the KITTI test set are summarized in Table 3. DiffCandiDet achieves competitive results.

4.4.3. Waymo open dataset

The results on the Waymo validation set are shown in Tables 4 and 5. DUCandiDet outperforms the baseline PV-RCNN. Compared to the baseline Voxel-RCNN, DiffCandiDet significantly improves the Cyclist detection mAP by +1.92% and +1.90%, and mAPH by +1.61% and +1.56%, respectively. The results further demonstrate the effectiveness of our method.

4.5. Ablation study and runtime analysis

4.5.1. Effectiveness of SDCB and DNCB

The ratio of different models selected as positive samples changes when utilizing the Dynamic Super-dense Candidate Boxes (DSCB) as shown in Fig. 8. The DSCB balances the proposal proportion of different categories in the RPN to prevent the proportion of the Car categories selected as positive samples from becoming too high. When we add the SDCB, as shown in the second and third rows in Table 6, the result is improved by +0.47%, and +0.54% for the Pedestrian and Cyclist on average, respectively. This improvement stems from the SDCB module's ability to increase candidate box density, which substantially raises the

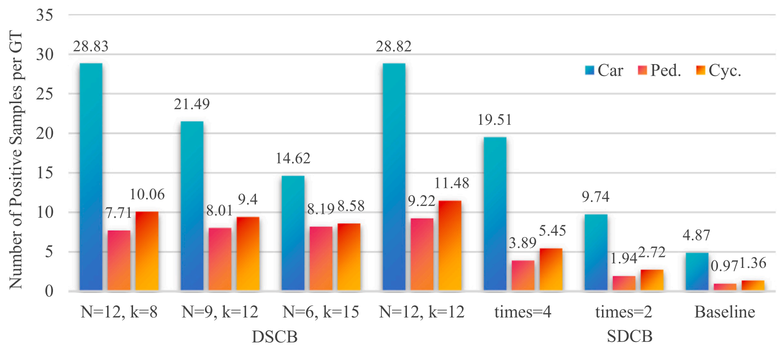

Fig. 8. The positive samples for each GT in the RPN when setting different N and k in DSCB and different times in SDCB on the KITTI training set.

Table 5
3D detection results on the Waymo validation set. All results are obtained by training on 20% of the training samples (approximately 32K frames) and validating on the entire validation set. DiffCandiDet achieves competitive results for Cyclist detection. The best results are in bold.

Methods	Cyclist(L1)		Cyclist(L2)	
	mAP	mAPH	mAP	mAPH
Part-A2-Net [74]	68.60	67.36	66.13	64.93
LIDAR-RCNN [75]	68.60	66.90	66.10	64.40
Voxel-RCNN [25]	68.74	67.56	66.46	65.35
CT3D [38]	69.28	67.88	66.84	65.48
PV-RCNN++ [68]	68.98	67.63	66.48	65.17
CasA+PV [21]	68.19	66.76	65.73	64.33
CasA+V [21]	69.69	68.38	67.07	66.83
DiffCandiDet-Vo (ours)	**70.66**	**69.46**	**68.07**	**66.91**

probability of high-quality candidates (IoU >0.5 for the Car and IoU >0.7 for the Ped. and Cyc.) being selected as positive samples during training and consequently reduces the false negative rate for small objects, especially in highly clustered and dense scenarios for small objects such as pedestrians and cyclists. When we add DNCB, as shown in the fourth and last rows in Table 6, the result is improved by +2.18% and +1.31% for the Pedestrian and Cyclist on average, respectively. And the proposal quality improves in the refinement stage when using DSCB as shown in Fig. 6. The IoU of total positive samples and the GT boxes improves by a large margin. This improvement can be attributed to the balance of DNCB module, which elevates the density capacity for small targets (e.g., Pedestrian and Cyclist), thereby pushing the overall performance limits across all object categories.

4.5.2. *Effectiveness of diffusion model with GDCP*

When we add the diffusion model with GDCP, as shown in the third and fourth rows in Table 6, the result is improved by +0.70% and +1.64% for the Car and Cyclist on average, respectively. The performance enhancements primarily stem from the iterative refinement mechanism inherent to diffusion models, whereby successive denoising stages correct localization errors through multi-stage optimization. This paradigm achieves a reduction in initial proposal variance while demonstrating superior robustness compared to static anchor-based frameworks constrained by rigid geometric priors.

4.5.3. *Inference step*

As shown in Table 7, for DiffCandiDet-Vo and DiffCandiDet-Ca, the optimal inference step is 3 and 4. For most baselines, the performance will increase with the increase of inference steps in the first three steps, which validates the effectiveness of diffusion inference. A more suitable distribution of center points can be explored during iterative inference. It is essential to search for a more appropriate initial distribution of center points in cases where the original fixed candidate boxes have a lower IoU with GT, which is disadvantageous for feature capturing and learning.

4.5.4. *Inference speed*

As shown in Table 7 The consumed time increases with the increasing inference steps, which directly leads to future work. Even so, the increased time can be compensated by the satisfying detection performance. The fastest inference speed occurs when the inference step is 1. It is worth noting that the inference step of 1 takes slightly more time than the baseline but results in a significant performance improvement compared to the baseline. For time-sensitive 3D object detection tasks, a step of 1 is a good choice.

4.5.5. *Effectiveness of removing GT sampling*

When we remove GT Sampling for the Pedestrian category, as shown in the first and second rows in Table 6, the result is improved by +5.04%, +5.74%, and +6.24%, respectively. As shown in Table 8, when removing GT Sampling only for the Pedestrian category for other baselines, Graph-RCNN and CasA both gain amazing enhancement for the Pedestrian without any hyperparameter tuning and contain nearly consistent for the Car and Cyclist. As shown in Table 9, the performance has decreased significantly when we remove GT Sampling only for the Car and Cyclist respectively. This justifies our hypothesis that objects with a tall and sparse point cloud shape, and a relatively low number of points, are often unsuitable for GT sampling data augmentation. As shown in Table 8, GT sampling only for the Pedestrian can improve the Pedestrian detection performance without impacting the detection performance of other categories, and it may even lead to a slight improvement in the Cyclist detection performance. However, removing GT Sampling only for the Car or Cyclist will simultaneously lower the detection performance of other categories. Besides, as the number of GT sampling augmentation decreases, the detection performance of the Pedestrian increases as shown in Table 10. Therefore, GT sampling for the Pedestrian is unnecessary. This finding contradicts the common sense that a certain degree of GT Sampling would benefit Pedestrian detection. Furthermore, removing GT Sampling can easily be applied to other models to enhance detection performance without runtime increasing. Furthermore, as shown in Table 8, 9, and 10, removing GT Sampling across different categories and varying sampling counts in GT Sampling only affect performance without increasing inference time.

4.5.6. *Dynamic N and k for training*

The initial number of candidate boxes is empirically determined to exceed a predefined threshold of 35,200 (calculated as 200 × 172 BEV grids), scaled proportionally across three object categories (Car, Pedestrian, and Cyclist) to maintain consistency with baseline implementations. To ensure robustness against initialization variance, we intentionally set parameter N slightly above this theoretical minimum and systematically evaluated performance through incremental increases. The hyperparameter k governs the adaptive allocation ratio of additional candidate boxes to underrepresented object categories (Pedestrian and Cyclist) relative to the dominant Car category, addressing class imbalance through geometric prioritization. To find the most suitable number of candidate boxes of DiffCandiDet in training, we set

Table 6
3D detection results on the KITTI validation set for the car, pedestrian, and cyclist categories with AP calculated by 40 recall positions by applying different designed components based on Voxel-RCNN. The best results are in bold.

Baseline	Remove GT Sampling	SDCB	Diffusion Model	DNCB	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)		
					Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	Easy.	Mod.	Hard.
✔					92.53	85.03	82.56	66.35	59.06	54.02	89.52	72.62	68.32
✔	✔				92.17	84.93	82.63	71.39	64.80	60.26	89.32	72.42	68.06
✔	✔	✔			92.13	84.79	82.48	71.23	65.64	**61.00**	90.95	72.43	68.03
✔	✔	✔	✔		**92.82**	**85.55**	**83.12**	70.70	63.27	56.84	92.00	74.51	69.82
✔	✔	✔	✔	✔	92.49	85.13	82.94	**72.35**	**65.73**	59.27	**92.87**	**75.98**	**71.41**

Table 7
3D detection results and inference speed of DiffCandiDet with different inference steps on the KITTI validation set for car, pedestrian, and cyclist categories with AP calculated by 40 recall positions using SECOND, PV-RCNN, Voxel-RCNN, CasA, and TED as the baseline. † indicates our reproduced results. The best results are in bold.

DiffCandiDet	Car 3D (R40)				Pedestrian 3D (R40)				Cyclist 3D (R40)				Runtime	Params
step	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	(ms)	(M)
SECOND† [10]	90.21	81.66	78.72	83.86	–	–	–	–	–	–	–	–	60.8	5.3
inference step=1 (ours)	90.81	82.39	79.40	84.20	–	–	–	–	–	–	–	–	78.2	5.3
inference step=2 (ours)	**90.98**	**82.57**	**79.80**	**84.45**	–	–	–	–	–	–	–	–	94.0	5.3
PV-RCNN† [12]	92.10	84.36	82.48	86.31	62.71	54.49	49.88	55.69	89.10	70.38	66.02	75.17	82.3	13.1
inference step=1 (ours)	**92.56**	**85.39**	**83.13**	**87.03**	67.80	59.11	54.86	60.59	90.59	71.58	67.09	76.42	92.8	13.1
inference step=2 (ours)	92.47	85.23	83.01	86.90	**69.21**	61.79	55.76	62.25	90.60	71.62	67.23	76.48	145.6	13.1
inference step=3 (ours)	92.53	85.31	83.06	86.97	68.72	**61.79**	**57.23**	**62.58**	**91.22**	**72.04**	**68.87**	**77.38**	193.0	13.1
Voxel-RCNN† [25]	92.53	85.03	82.56	86.71	66.35	59.06	54.02	59.81	89.52	72.62	68.32	76.82	40.0	7.6
inference step=1 (ours)	92.19	83.34	82.80	86.11	72.56	64.26	57.63	64.82	92.21	74.09	69.39	78.56	68.8	7.6
inference step=2 (ours)	92.30	85.03	82.89	86.74	**72.72**	**66.08**	**59.44**	**66.08**	92.50	74.34	71.03	79.29	103.2	7.6
inference step=3 (ours)	92.49	**85.13**	**82.94**	**86.85**	72.35	65.73	59.27	65.78	**92.87**	**75.98**	**71.41**	**80.09**	134.2	7.6
CasA+V† [21]	93.21	86.37	83.93	87.84	73.95	66.62	59.97	66.85	92.78	73.94	69.37	78.70	32.6	11.3
inference step=1 (ours)	92.66	85.75	83.42	87.28	79.92	69.14	63.87	70.98	95.03	72.72	69.75	79.17	64.1	11.3
inference step=2 (ours)	92.51	83.91	83.39	86.60	79.79	70.79	63.74	71.44	**95.31**	73.35	**70.29**	79.65	112.3	11.3
inference step=3 (ours)	92.71	85.78	83.46	87.32	80.00	71.05	63.93	71.66	95.23	74.58	70.12	79.98	161.1	11.3
inference step=4 (ours)	92.43	85.54	83.22	87.06	**81.07**	**73.70**	**68.34**	**74.37**	94.51	**75.03**	70.62	**80.05**	195.8	11.3
TED-M† [24]	95.25	88.94	86.73	90.31	–	–	–	–	–	–	–	–	166.2	18.9
inference step=1 (ours)	95.74	89.09	86.49	90.44	–	–	–	–	–	–	–	–	254.4	18.9
inference step=2 (ours)	**95.89**	89.13	86.53	90.52	–	–	–	–	–	–	–	–	416.3	18.9
inference step=3 (ours)	95.74	**89.21**	**86.61**	**90.52**	–	–	–	–	–	–	–	–	583.5	18.9

Table 8
3D detection results on the KITTI val set by removing GT Sampling only for the Pedestrian using Voxel-RCNN, Graph-RCNN, and CasA as the baseline with AP calculated by 40 recall positions. † indicates our reproduced results. ‡ represents the results obtained by running the author's source code. The best results are in bold..

Method	Car 3D (R40)				Pedestrian 3D (R40)				Cyclist 3D (R40)				Runtime
	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	(ms)
Voxel-RCNN† [25]	92.53	85.03	82.56	86.71	66.35	59.06	54.02	59.81	89.52	72.62	68.32	76.82	40.0
+Remove GT Sampling for Ped.	92.44	85.13	82.68	86.75	**67.79**	**63.93**	**58.53**	**63.42**	91.94	72.25	67.94	77.38	40.3
Graph-Vo‡ [20]	93.27	86.07	83.12	87.49	67.42	61.76	56.96	62.05	91.46	73.17	68.87	77.83	40.0
+Remove GT Sampling for Ped.	93.19	85.81	82.97	87.32	**72.18**	**64.65**	**59.89**	**65.57**	91.33	72.49	68.24	77.35	37.4
CasA+V† [21]	93.21	86.37	83.93	87.84	73.95	66.62	59.97	66.85	92.78	73.94	69.37	78.70	32.6
+Remove GT Sampling for Ped.	92.98	86.33	83.82	87.71	**76.91**	**69.74**	**62.77**	**69.81**	93.77	74.32	71.36	79.82	38.7

Table 9
3D detection results on the KITTI val set by removing GT Sampling only for the Car and Cyclist categories using Voxel-RCNN and CasA as the baseline with AP calculated by 40 recall positions. The best results are in bold..

Method	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)			Runtime
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
Voxel-RCNN [25]	**92.53**	**85.03**	**82.56**	**66.35**	**59.06**	**54.02**	**89.52**	**72.62**	**68.32**	40.0
+Remove GT Sampling for Car	92.14	82.46	80.09	64.24	58.64	53.92	73.75	56.24	52.02	39.6
+Remove GT Sampling for Cyc.	92.35	84.75	82.41	59.80	53.39	48.82	58.26	44.05	41.66	**39.5**
CasA+V [21]	**93.21**	**86.37**	**83.93**	**73.95**	**66.62**	**59.97**	**92.78**	73.94	69.37	32.6
+Remove GT Sampling for Car	92.85	83.91	83.31	68.28	63.18	57.99	89.13	**74.50**	**70.16**	**31.2**
+Remove GT Sampling for Cyc.	92.52	83.70	83.02	71.52	65.03	59.94	82.49	55.59	51.28	31.9

different numbers of candidate boxes for different categories. Different combinations of N and k in the DNCB strategy are shown in the first row in Table 11, and we choose N=120000 and k=80000. For all the baselines, we set N=120000 and k=80000 (N=60000 and k=80000 for LoGoNet). Within a bounded parameter range, increasing N and k in SDCB can slightly increase inference time. Fortunately, SDCB is necessary for detecting closely adjacent objects.

4.5.7. *Dynamic N and k for testing*

Notably, DiffCandiDet also has Once-for-all properties [31]. For the most suitable N and k selected for training, we test and obtain the best N and k for inference as shown in the second row in Table 12. We also choose N=120000 and k=80000. For all the baselines, we also set N=120000 and k=80000 (N=60000 and k=80000 for LoGoNet). Using

Table 10
3D detection results on the KITTI val set by applying different GT Sampling numbers only for the Pedestrian. using Voxel-RCNN as the baseline with AP calculated by 40 recall positions. The best results are in bold..

Num	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)			Runtime
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
10	92.53	85.03	82.56	66.35	59.06	54.02	89.52	72.62	68.32	40.0
7	92.77	85.06	82.81	67.41	59.73	54.60	89.16	73.15	68.57	42.1
5	92.17	84.79	82.54	66.70	60.81	56.55	91.66	74.43	69.92	41.0
3	92.51	84.96	82.74	**68.82**	62.77	58.20	88.86	70.92	66.53	38.9
0	92.44	85.13	82.68	67.79	**63.93**	**58.53**	91.94	72.25	67.94	40.3

Table 11
3D detection results with different N and k for training in DiffcandiDet with inference step 1 and 3 on the KITTI validation set for the car, pedestrian, and cyclist with AP calculated by 40 recall positions. The best results are in bold.

Step	N	k	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)			Rumtime
			Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	(ms)
Step=1	60 000	80 000	92.53	**85.36**	82.91	65.18	59.67	54.10	92.97	**74.80**	**70.09**	**66.5**
	90 000	80 000	**92.58**	85.26	82.85	67.10	60.11	53.85	92.74	74.48	69.76	68.5
	90 000	120 000	92.51	85.15	**82.99**	71.45	64.07	57.59	**93.20**	73.37	70.03	71.9
	120 000	80 000	92.19	83.34	82.80	**72.56**	**64.26**	**57.63**	92.21	74.09	69.39	68.8
	120 000	120 000	92.31	85.31	82.96	69.96	62.89	56.61	91.13	72.89	68.33	72.2
Step=3	60 000	80 000	92.08	85.07	82.34	69.94	63.78	57.84	92.45	74.35	71.07	132.8
	90 000	80 000	92.47	84.97	82.60	70.78	64.05	57.00	92.67	74.50	71.15	134.2
	90 000	120 000	92.40	85.03	82.85	71.95	65.76	59.03	92.72	75.61	71.26	134.3
	120 000	80 000	92.49	85.13	82.94	72.35	65.73	59.27	92.87	75.98	71.41	134.2
	120 000	120 000	92.45	85.12	83.07	72.25	64.97	59.20	92.95	75.03	71.31	136.1

Table 12
3D detection results with different N and k for testing in DiffcandiDet with inference step 3 (the most suitable N=120000 and k=80000 selected for training) on the KITTI validation set for the car, pedestrian, and cyclist categories with AP R40. The best results are in bold.

N	k	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)			Runtime
		Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	(ms)
60000	80 000	92.41	84.95	82.84	72.53	65.71	59.18	92.82	**76.04**	71.28	131.0
90000	80 000	92.32	84.86	82.80	**72.73**	66.07	59.44	92.59	75.77	71.16	134.8
90000	120 000	92.23	84.86	82.80	72.23	**65.83**	59.09	**92.89**	75.87	71.15	133.9
120000	80 000	**92.49**	**85.13**	**82.94**	72.35	65.73	**59.27**	92.87	75.98	**71.41**	134.2
120000	120 000	92.37	84.88	82.82	72.11	65.61	59.03	92.34	75.68	71.13	135.3

Table 13
3D detection results with different box renewal threshold μ in DiffcandiDet when inference step is 3 on the KITTI validation set for the car, pedestrian, and cyclist categories with AP R40. The best results are in bold.

Renewal Threshold μ	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)		
	Easy.	Mod.	Hard.	Easy.	Mod.	Hard.	Easy.	Mod.	Hard.
0.85	92.30	84.97	82.85	**72.62**	**65.98**	59.38	91.91	75.45	70.79
0.75	**92.49**	**85.13**	**82.94**	72.35	65.73	59.27	92.87	75.98	71.41
0.5	92.27	84.97	82.86	71.94	65.55	59.06	92.58	75.37	70.84
0.3	92.32	84.93	82.91	72.47	65.91	59.30	92.63	75.69	70.95
0.1	92.33	84.94	82.92	72.50	65.92	**59.39**	**94.61**	**75.99**	**71.43**

smaller values for N and k during the inference process is also a good option.

4.5.8. Renewal threshold μ

Retaining the centers of all predicted boxes during diffusion reasoning is not beneficial for inference. This is because, in addition to the GT center point distribution, there is noise in the center point distribution during inference, which is harmful to the recovery of the GT center point distribution. Therefore, we retain the GT distribution of center points from predicted boxes exceeding a renewal threshold μ. We test the different sets of box renewal thresholds μ as shown in the third row in Table 13. We set 0.75 for the box renewal threshold and 0.1 is also a good choice.

4.6. Comparison with diffusion-based method

Compared to other diffusion-based methods [54,55], DiffCandiDet demonstrates significant superiority on both KITTI validation and test set as shown in Tables 3 and 14. The most critical reason is that DiffCandiDet leverages the strengths of anchor-based methods and diffusion models.

4.7. Comparison with multi-class 3D object detection method

As shown in Table 15, we compare DiffCandiDet with existing multi-class 3D object detection methods. Our method still achieves more pronounced advantages in category balance over mainstream multi-class methods. This benefits from the category balance of the DSCB strategy and the iterative optimization of the diffusion probabilistic model.

4.8. Generalization capacity

As shown in Table 1, DiffCandiDet, DUCandiDet, and CUCandiDet can all be integrated into one-stage, two-stage, and multi-modal models, demonstrating excellent performance and generalization capacity in both single-class and multi-class 3D object detection tasks.

Table 14
3D detection results on the KITTI validation set for the Car, Pedestrian, and Cyclist categories with AP calculated by 40 recall positions. DiffCandiDet outperforms all of the previous diffusion-based methods. The best results are in bold.

Method	Car 3D (R40)			Pedestrian 3D (R40)			Cyclist 3D (R40)		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
DIff3Det [54]	89.45	80.86	77.41	–	–	–	–	–	–
DiffRef3D [55]	92.89	84.77	82.64	69.70	63.07	57.70	93.33	74.20	69.63
DiffCandiDet-Vo (ours)	92.49	85.13	82.94	72.35	65.73	59.27	92.87	**75.98**	**71.41**
DiffCandiDet-Ca (ours)	92.43	85.54	83.22	**81.07**	**73.70**	**68.34**	**94.51**	75.03	70.62

Table 15
3D detection results on the KITTI validation set for the car, pedestrian, and cyclist categories with AP calculated by 40 recall positions. DiffCandiDet achieves more significant advantages compared to competitive multi-class 3D object detection methods. The best results are in bold. † indicates our reproduced results. ‡ represents the results obtained by running the author's source code. The best results are in bold..

Method	Car 3D (R40)				Pedestrian 3D (R40)				Cyclist 3D (R40)			
	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP	Easy	Mod.	Hard	mAP
PSA-Det3D† [76]	91.34	80.53	78.09	83.32	68.35	63.09	57.09	62.84	92.23	74.11	69.97	78.77
CTA-Det† [77]	90.12	81.46	79.15	83.58	74.08	66.35	58.92	66.45	87.64	72.82	68.20	76.22
EQ-PVRCNN† [78]	**92.63**	85.41	82.97	87.00	66.78	59.23	54.34	60.12	93.34	75.71	71.11	80.05
LoGoNet† [23]	91.41	84.80	**84.48**	86.90	70.03	63.31	59.04	64.13	90.64	72.78	68.79	77.40
BSAODet† [13]	92.27	85.06	82.75	86.69	71.98	66.00	60.49	66.16	93.23	**76.06**	**72.31**	**80.53**
DiffCandiDet-Vo (ours)	92.49	85.13	82.94	86.85	72.35	65.73	59.27	65.78	92.87	75.98	71.41	80.09
DiffCandiDet-Ca (ours)	92.43	**85.54**	83.22	**87.06**	**81.07**	**73.70**	**68.34**	**74.37**	**94.51**	75.03	70.62	80.05

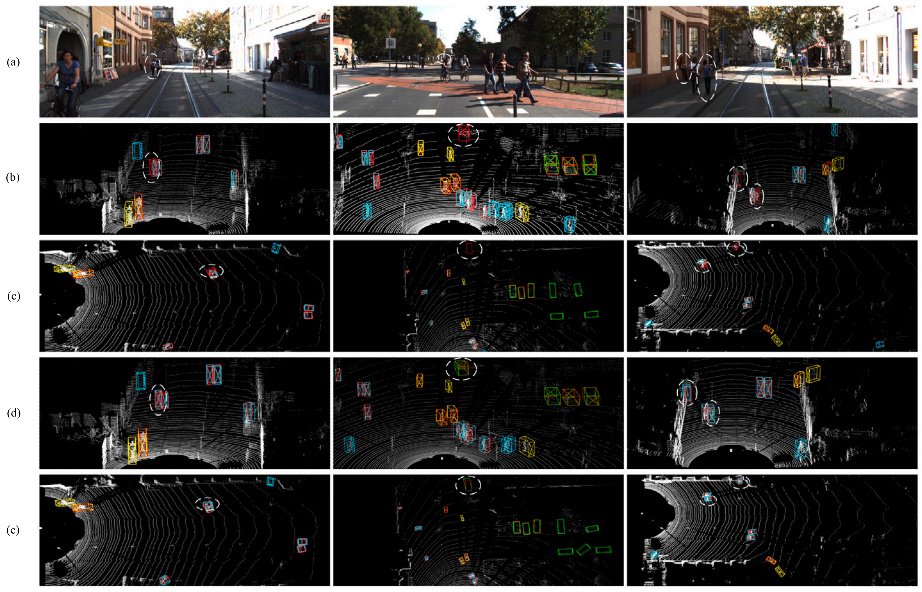

Fig. 9. Visualization of 3D and BEV detection results on the KITTI dataset. (a) Corresponding image frame. (b) 3D detection results by CasA+V [21] using official CasA weights. (c) BEV detection results by CasA+V. (d) 3D detection results by DiffCandiDet-Ca with an inference step of 2. (e) BEV detection results by DiffCandiDet-Ca. In the visualization: The red boxes denote GT bounding boxes. The green boxes indicate the predicted car category. The blue boxes represent the predicted pedestrian category. The yellow boxes denote the predicted cyclist category. In particular, the white dashed ellipses in the image highlight numerous false negative detections predicted by CasA [21]. CasA struggles with accurate detection, particularly pedestrian detection. In contrast, DiffCandiDet-Ca demonstrates superior precision.

4.9. Quantitative analysis

The detection performance of CasA+V [21] is highly advanced and continues to rank near the top on the official KITTI leaderboard. Even so, the comparison between DiffCandiDet-Ca and CasA+V illustrates how DiffCandiDet can surpass CasA in detection accuracy as shown in Fig. 9. The first and third columns demonstrate that DiffCandiDet can detect highly cluttered scenarios like pedestrians walking side by side that CasA misses. This is primarily due to the fact that in traditional anchor-based methods, anchors are uniformly distributed across the spatial domain. During the non-maximum suppression (NMS) process, redundant candidate boxes corresponding to neighboring objects are filtered out prior to being fed into the second stage of the detection pipeline. As a result, only a single prediction is retained from the shared candidate boxes for each group of proximate objects, leading to potential missed detections. However, when the DSCB strategy is employed, a higher density of candidate boxes can coexist within the same localized region simultaneously. This increased density significantly reduces the likelihood of candidate boxes being eliminated by the NMS procedure, thereby enhancing the detection robustness. In the second column, DiffCandiDet detects distant cars that CasA overlooks. Additionally, in the third column, DiffCandiDet identifies a pedestrian leaning against a wall, which validates DiffCandiDet's superior ability to capture detailed features and effectively differentiate closely neighboring objects.

4.10. Complexity analysis

Compared to the baseline, DiffCandiDet maintains identical training procedures as employed by the original algorithm. For two-stage baseline, however, DiffCandiDet diverges from the original algorithm with its complexity showing a positive correlation to the number of inference steps utilized. Specifically, if we consider the baseline algorithm's complexity to be $O(m+n)$, then for a given step k, the computational complexity of DiffCandiDet can be characterized as $O(m + k \times n)$, where the computational complexity of the first-stage backbone is denoted by m, while the computational complexity of the second-stage backbone is represented by n. During the refinement process of the second stage, we will iterate k steps to find more suitable center points of anchors. Therefore, the processing time increases linearly with the number of inference steps k. Designing new technologies to reduce time is a meaningful thing for future work while maintaining similar performance.

4.11. Limitations and future works

DiffCandiDet lies in its performance-latency trade-off: achieving better performance necessitates progressively increasing the number of inference steps, which notably escalates computational overhead [79,80]. This dependency makes it less practical for online detection scenarios where latency constraints prohibit extensive iterative refinement. For future work, we wish to explore adaptive step scheduling or lightweight denoising architectures to mitigate this bottleneck while preserving detection accuracy.

5. Conclusions

In this paper, we propose DiffCandiDet based on the diffusion model with Gaussian distributed center points, which leverages the powerful denoising and reconstruction capabilities of the diffusion model. Moreover, we also propose a Dynamic Super-dense Candidate Boxes strategy to enhance the initial IoU and achieve the balance of learning. In addition to Gaussian distribution, we further propose DUCandiDet and CUCandiDet to reduce the runtime consumption and improve the robustness. DiffCandiDet has achieved state-of-the-art performance on the KITTI validation set, especially in the Car and Pedestrian detection leaderboard.

CRediT authorship contribution statement

Si-Heng He: Writing – original draft, Software, Investigation, Conceptualization. **Zi-Jia Wang:** Writing – review & editing, Supervision, Methodology, Funding acquisition. **Yuan-Gen Wang:** Writing – review & editing, Methodology. **Yicong Zhou:** Writing – review & editing, Validation. **Sam Kwong:** Writing – review & editing, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] H. Zhang, L. Hai, X. Wang, X. Wang, M. Zhou, RCFI-Net: a reliable correspondences evaluation and feature interaction network for fast and accurate point cloud registration, Appl. Soft Comput. (2024) 111839.
[2] H. Xu, J. Bai, SPS-LCNN: A significant point sampling-based lightweight convolutional neural network for point cloud processing, Appl. Soft Comput. 144 (2023) 110498.
[3] J. Zhao, Y. Liu, B. Wu, Multi-scale learnable key-channel attention network for point cloud classification and segmentation, Appl. Soft Comput. 159 (2024) 111622.
[4] M.A. Günen, Adaptive neighborhood size and effective geometric features selection for 3D scattered point cloud classification, Appl. Soft Comput. 115 (2022) 108196.
[5] Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H. Wang, S. Kwong, J. Zhang, Automatic niching differential evolution with contour prediction approach for multimodal optimization problems, IEEE Trans. Evol. Comput. 24 (1) (2019) 114–128.
[6] Q. Tang, M. Yang, Z. Wang, W. Dong, Y. Liu, Boundary points guided 3D object detection for point clouds, Appl. Soft Comput. (2024) 112117.
[7] H. Jiang, Y. Lu, D. Zhang, Y. Shi, J. Wang, Deep learning-based fusion networks with high-order attention mechanism for 3D object detection in autonomous driving scenarios, Appl. Soft Comput. 152 (2024) 111253.
[8] X. Chen, Y. Sun, Q. Zhang, X. Dai, S. Tian, Y. Guo, Three-dimension object detection and forward-looking control strategy for non-destructive grasp of thin-skinned fruits, Appl. Soft Comput. 150 (2024) 111082.
[9] Z.-J. Wang, Z.-H. Zhan, W.-J. Yu, Y. Lin, J. Zhang, T.-L. Gu, J. Zhang, Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling, IEEE Trans. Cybern. 50 (6) (2019) 2715–2729.
[10] Y. Yan, Y. Mao, B. Li, Second: Sparsely embedded convolutional detection, Sensors 18 (10) (2018) 3337.
[11] A.H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, O. Beijbom, Pointpillars: Fast encoders for object detection from point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 12697–12705.
[12] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, H. Li, Pv-rcnn: Point-voxel feature set abstraction for 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10529–10538.
[13] W. Xiao, Y. Peng, C. Liu, J. Gao, Y. Wu, X. Li, Balanced sample assignment and objective for single-model multi-class 3D object detection, IEEE Trans. Circuits Syst. Video Technol. (2023).
[14] J.S. Hu, T. Kuai, S.L. Waslander, Point density-aware voxels for lidar 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8469–8478.
[15] H. Wu, C. Wen, W. Li, X. Li, R. Yang, C. Wang, Transformation-equivariant 3D object detection for autonomous driving, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 2795–2802.
[16] H. Wu, C. Wen, S. Shi, X. Li, C. Wang, Virtual sparse convolution for multimodal 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 21653–21662.
[17] Q. Chen, L. Sun, Z. Wang, K. Jia, A. Yuille, Object as hotspots: An anchor-free 3D object detection approach via firing of hotspots, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16, Springer, 2020, pp. 68–84.
[18] T. Yin, X. Zhou, P. Krahenbuhl, Center-based 3D object detection and tracking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 11784–11793.
[19] W. Zheng, W. Tang, L. Jiang, C.-W. Fu, SE-SSD: Self-ensembling single-stage object detector from point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 14494–14503.
[20] H. Yang, Z. Liu, X. Wu, W. Wang, W. Qian, X. He, D. Cai, Graph R-CNN: Towards accurate 3D object detection with semantic-decorated local graph, in: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VIII, Springer, 2022, pp. 662–679.
[21] H. Wu, J. Deng, C. Wen, X. Li, C. Wang, J. Li, CasA: A cascade attention network for 3-D object detection from LiDAR point clouds, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–11.
[22] Y. Wu, W. Xiao, J. Gao, C. Liu, Y. Qin, Y. Peng, X. Li, Spatial-aware learning in feature embedding and classification for one-stage 3D object detection, IEEE Trans. Geosci. Remote Sens. (2024).
[23] X. Li, T. Ma, Y. Hou, B. Shi, Y. Yang, Y. Liu, X. Wu, Q. Chen, Y. Li, Y. Qiao, et al., LoGoNet: Towards accurate 3D object detection with local-to-global cross-modal fusion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 17524–17534.
[24] H. Wu, C. Wen, W. Li, X. Li, R. Yang, C. Wang, Transformation-equivariant 3D object detection for autonomous driving, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, 2023, pp. 2795–2802.
[25] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, H. Li, Voxel r-cnn: Towards high performance voxel-based 3D object detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, 2021, pp. 1201–1209.
[26] Y. Li, K. Zhou, W.X. Zhao, J.-R. Wen, Diffusion models for non-autoregressive text generation: A survey, 2023, arXiv preprint arXiv:2303.06574.
[27] C. Zhang, C. Zhang, M. Zhang, I.S. Kweon, Text-to-image diffusion model in generative ai: A survey, 2023, arXiv preprint arXiv:2303.07909.
[28] F.-A. Croitoru, V. Hondru, R.T. Ionescu, M. Shah, Diffusion models in vision: A survey, IEEE Trans. Pattern Anal. Mach. Intell. (2023).
[29] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, M.-H. Yang, Diffusion models: A comprehensive survey of methods and applications, ACM Comput. Surv. 56 (4) (2023) 1–39.
[30] Z.-J. Wang, Z.-H. Zhan, S. Kwong, H. Jin, J. Zhang, Adaptive granularity learning distributed particle swarm optimization for large-scale optimization, IEEE Trans. Cybern. 51 (3) (2020) 1175–1188.

[31] S. Chen, P. Sun, Y. Song, P. Luo, Diffusiondet: Diffusion model for object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 19830–19843.

[32] E.T. Mahdi, S.M. Al-Barzinji, W.K. Awad, Object detection using capsule neural network: An overview, Babylon. J. Mach. Learn. 2024 (2024) 157–164.

[33] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.

[34] J. Mao, S. Shi, X. Wang, H. Li, 3D object detection for autonomous driving: A comprehensive survey, Int. J. Comput. Vis. (2023) 1–55.

[35] R. Qian, X. Lai, X. Li, 3D object detection for autonomous driving: A survey, Pattern Recognit. 130 (2022) 108796.

[36] F. Sun, G. Tong, Y. Song, Efficient flexible voxel-based two-stage network for 3D object detection in autonomous driving, Appl. Soft Comput. (2024) 111856.

[37] Z.-J. Wang, Y.-R. Zhou, J. Zhang, Adaptive estimation distribution distributed differential evolution for multimodal optimization problems, IEEE Trans. Cybern. 52 (7) (2020) 6059–6070.

[38] H. Sheng, S. Cai, Y. Liu, B. Deng, J. Huang, X.-S. Hua, M.-J. Zhao, Improving 3D object detection with channel-wise transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 2743–2752.

[39] X. Ma, W. Ouyang, A. Simonelli, E. Ricci, 3D object detection from images for autonomous driving: a survey, IEEE Trans. Pattern Anal. Mach. Intell. (2023).

[40] L. Wang, X. Zhang, Z. Song, J. Bi, G. Zhang, H. Wei, L. Tang, L. Yang, J. Li, C. Jia, et al., Multi-modal 3D object detection in autonomous driving: A survey and taxonomy, IEEE Trans. Intell. Veh. (2023).

[41] Y. Wang, Q. Mao, H. Zhu, J. Deng, Y. Zhang, J. Ji, H. Li, Y. Zhang, Multi-modal 3D object detection in autonomous driving: a survey, Int. J. Comput. Vis. 131 (8) (2023) 2122–2152.

[42] Y. Wu, X. Jiang, Z. Fang, Y. Gao, H. Fujita, Multi-modal 3D object detection by 2D-guided precision anchor proposal and multi-layer fusion, Appl. Soft Comput. 108 (2021) 107405.

[43] R. Hou, G. Chen, Y. Han, Z. Tang, Q. Ru, Multi-modal feature fusion for 3D object detection in the production workshop, Appl. Soft Comput. 115 (2022) 108245.

[44] Z.-J. Wang, Z.-H. Zhan, Y. Lin, W.-J. Yu, H.-Q. Yuan, T.-L. Gu, S. Kwong, J. Zhang, Dual-strategy differential evolution with affinity propagation clustering for multimodal optimization problems, IEEE Trans. Evol. Comput. 22 (6) (2017) 894–908.

[45] M. Hu, S. Wang, B. Li, S. Ning, L. Fan, X. Gong, Penet: Towards precise and efficient image guided depth completion, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2021, pp. 13656–13662.

[46] S. Vora, A.H. Lang, B. Helou, O. Beijbom, Pointpainting: Sequential fusion for 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4604–4612.

[47] T. Huang, Z. Liu, X. Chen, X. Bai, Epnet: Enhancing point features with image semantics for 3D object detection, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16, Springer, 2020, pp. 35–52.

[48] X. Wu, L. Peng, H. Yang, L. Xie, C. Huang, C. Deng, H. Liu, D. Cai, Sparse fuse dense: Towards high quality 3D detection with depth completion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 5418–5427.

[49] R. Po, W. Yifan, V. Golyanik, K. Aberman, J.T. Barron, A.H. Bermano, E.R. Chan, T. Dekel, A. Holynski, A. Kanazawa, et al., State of the art on diffusion models for visual computing, 2023, arXiv preprint arXiv:2310.07204.

[50] H. Cao, C. Tan, Z. Gao, Y. Xu, G. Chen, P.-A. Heng, S.Z. Li, A survey on generative diffusion models, IEEE Trans. Knowl. Data Eng. (2024).

[51] Z.-F. Xue, Z.-J. Wang, Y. Jiang, Z.-H. Zhan, S. Kwong, J. Zhang, Multi-level and multi-segment learning multitask optimization via a niching method, IEEE Trans. Evol. Comput. (2024).

[52] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, Adv. Neural Inf. Process. Syst. 33 (2020) 6840–6851.

[53] J. Song, C. Meng, S. Ermon, Denoising diffusion implicit models, 2020, arXiv preprint arXiv:2010.02502.

[54] X. Zhou, J. Hou, T. Yao, D. Liang, Z. Liu, Z. Zou, X. Ye, J. Cheng, X. Bai, Diffusion-based 3D object detection with random boxes, 2023, arXiv preprint arXiv:2309.02049.

[55] S.-H. Kim, I. Koo, I. Lee, B. Park, C. Kim, DiffRef3D: A diffusion-based proposal refinement framework for 3D object detection, 2023, arXiv preprint arXiv:2310.16349.

[56] J. Zou, K. Tian, Z. Zhu, Y. Ye, X. Wang, Diffbev: Conditional diffusion model for bird's eye view perception, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, 2024, pp. 7846–7854.

[57] O. Rukundo, H. Cao, Nearest neighbor value interpolation, 2012, arXiv preprint arXiv:1211.1768.

[58] M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, Adv. Neural Inf. Process. Syst. 28 (2015).

[59] A.Q. Nichol, P. Dhariwal, Improved denoising diffusion probabilistic models, in: International Conference on Machine Learning, PMLR, 2021, pp. 8162–8171.

[60] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[61] I. Koo, I. Lee, S.-H. Kim, H.-S. Kim, W.-j. Jeon, C. Kim, PG-RCNN: Semantic surface point generation for 3D object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 18142–18151.

[62] Q. Xu, Y. Zhong, U. Neumann, Behind the curtain: Learning occluded shapes for 3D object detection, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 2893–2901.

[63] T. Cortinhal, I. Gouigah, E.E. Aksoy, Semantics-aware LiDAR-only pseudo point cloud generation for 3D object detection, 2023, arXiv preprint arXiv:2309.08932.

[64] C.R. Qi, W. Liu, C. Wu, H. Su, L.J. Guibas, Frustum pointnets for 3D object detection from rgb-d data, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 918–927.

[65] Z. Liu, T. Huang, B. Li, X. Chen, X. Wang, X. Bai, EPNet++: Cascade bi-directional fusion for multi-modal 3D object detection, IEEE Trans. Pattern Anal. Mach. Intell. (2022).

[66] S. Pang, D. Morris, H. Radha, CLOCs: Camera-LiDAR object candidates fusion for 3D object detection, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 10386–10393.

[67] Y. Tian, X. Zhang, X. Wang, J. Xu, J. Wang, R. Ai, W. Gu, W. Ding, ACF-Net: Asymmetric cascade fusion for 3D detection with LiDAR point clouds and images, IEEE Trans. Intell. Veh. (2023).

[68] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, H. Li, PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection, Int. J. Comput. Vis. 131 (2) (2023) 531–551.

[69] Z. Yang, Y. Sun, S. Liu, X. Shen, J. Jia, Std: Sparse-to-dense 3D object detector for point cloud, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1951–1960.

[70] Q. He, Z. Wang, H. Zeng, Y. Zeng, Y. Liu, Svga-net: Sparse voxel-graph attention network for 3D object detection from point clouds, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, 2022, pp. 870–878.

[71] S. Vora, A.H. Lang, B. Helou, O. Beijbom, Pointpainting: Sequential fusion for 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4604–4612.

[72] Z. Wang, K. Jia, Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection, in: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2019, pp. 1742–1749.

[73] S. Shi, X. Wang, H. Li, Pointrcnn: 3D object proposal generation and detection from point cloud, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 770–779.

[74] S. Shi, Z. Wang, X. Wang, H. Li, Part-a^2 net: 3D part-aware and aggregation neural network for object detection from point cloud, 2, (3) 2019, arXiv preprint arXiv:1907.03670.

[75] Z. Li, F. Wang, N. Wang, Lidar r-cnn: An efficient and universal 3D object detector, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 7546–7555.

[76] Z. Huang, Z. Zheng, J. Zhao, H. Hu, Z. Wang, D. Chen, PSA-Det3D: Pillar set abstraction for 3D object detection, Pattern Recognit. Lett. 168 (2023) 138–145.

[77] Y. Zhang, J. Chen, D. Huang, Cat-det: Contrastively augmented transformer for multi-modal 3D object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 908–917.

[78] Z. Yang, L. Jiang, Y. Sun, B. Schiele, J. Jia, A unified query-based paradigm for point cloud understanding, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8541–8551.

[79] Z.-J. Wang, J.-R. Jian, Z.-H. Zhan, Y. Li, S. Kwong, J. Zhang, Gene targeting differential evolution: A simple and efficient method for large-scale optimization, IEEE Trans. Evol. Comput. 27 (4) (2022) 964–979.

[80] Z.-F. Xue, Z.-J. Wang, Z.-H. Zhan, S. Kwong, J. Zhang, Neural network-based knowledge transfer for multitask optimization, IEEE Trans. Cybern. (2024).