## A. Related Work

**Radar-Only Perception:** Learning-based methods have advanced radar detection over traditional model-based approaches [20], benefiting from open large-scale radar point cloud datasets like nuScenes [4], Oxford RobotCar [2], and RADIATE [29]. Image-based and point/voxel-based backbones [14, 30] extract semantic features from radar detection points, generate region proposals, and localize objects. High-resolution heatmaps (e.g., K-Radar [25], HIBER [37], MMVR [26]) and raw ADC data [39] have also been leveraged by previously mentioned RF-Pose [43], RFMask [37], and RETR [40]. CubeLearn [44] replaces Fourier transforms with learnable modules for an end-to-end radar pipeline, while RAMP-CNN [12] enhances range-angle feature extraction via Doppler cues. More recently, diffusion models have been explored for radar applications [8, 11, 24, 36, 42]. Most efforts, e.g., RadarDiffusion [24, 42] and DiffRadar [36], focus on reconstructing LiDAR-like point clouds from low-resolution radar data, while mmDiff [11] estimates and refines pose keypoints from sparse radar points via diffusion process.

**Diffusion-based Object Detection:** Diffusion models [28, 32–34] have shown impressive results in tasks such as image and video generation [3, 18] and multi-view synthesis [6, 41]. For perception tasks, DiffusionDet [7] first reformulates object detection as a generative denoising process and proposes to model the 2D BBoxes as random parameters in the diffusion process. Diffusion-SS3D [16] proposes a diffusion-based detector to enhance the quality of pseudo-labels in semi-supervised 3D object detection by integrating it into a teacher-student framework. CLIFF [22] further leverages language models to enhance diffusion-based models for open-vocabulary object detection. Diffusion models are also considered for 3D object detection in the context of LiDAR-Camera fusion [38] and other tasks such as pose estimation [35] and semantic segmentation [1, 13].

## B. Multi-View Radar Heatmaps

Multi-view radar heatmaps are generated from raw data captured by two radar arrays: a vertical linear array and a horizontal one, as illustrated in Fig. 5. By sampling multiple reflected pulses across the array elements, a 3D raw data cube is constructed for each array, organized along ADC (intra-pulse) samples, pulse (inter-pulse) samples, and array elements. A 3D fast Fourier transform (FFT) converts the data cube into corresponding 3D radar spectra across the range, Doppler velocity, and spatial angle (azimuth for the horizontal array and elevation for the vertical one). To enhance the signal-to-noise ratio (SNR), the 3D radar spectra are integrated along the Doppler domain, generating two 2D radar heatmaps (range-azimuth and range-elevation) in the polar coordinate system. These heatmaps are then transformed into the radar Cartesian coordinate system, where $\boldsymbol{Y}_{\texttt{hor}}(m) \in \mathcal{R}^{W\times D}$ represents the horizontal-depth radar heatmap and $\boldsymbol{Y}_{\texttt{ver}}(m) \in \mathcal{R}^{H\times D}$ the vertical-depth heatmap for the $m$-th frame. To incorporate temporal information, $M$ consecutive radar frames are grouped together as $\boldsymbol{Y}_{\texttt{hor}} \in \mathcal{R}^{M\times W\times D}$ and vertical $\boldsymbol{Y}_{\texttt{ver}} \in \mathcal{R}^{M\times H\times D}$.

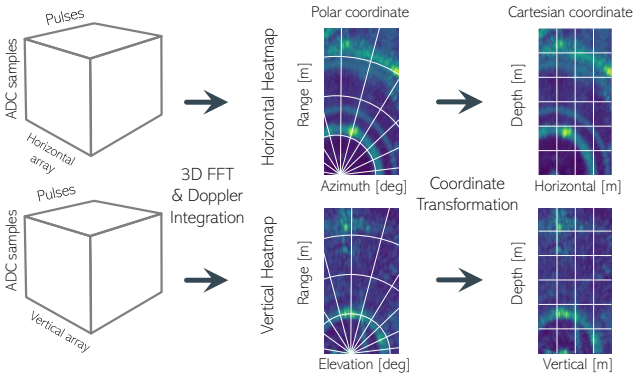


Figure 5. Generation of multi-view heatmaps from raw radar data.

## C. Details of 3D-to-2D Projection and Necessity of the Refinement Module

We present the detailed explanations for 3D-to-2D projection and necessity of the refinement module. Given a 3D BBox which consists of its eight vertices

$$\{\boldsymbol{x}^i_{\texttt{camera}} \in \mathbb{R}^3 \mid i = 1, \ldots, 8\}, \tag{9}$$

where each $\boldsymbol{x}^i_{\texttt{camera}}$ is expressed in the 3D camera coordinate system, our goal is to compute the corresponding 2D BBox $\boldsymbol{b}_{\texttt{init}} \in \mathbb{R}^4$, defined by its center coordinates $(x_c, y_c)$ and its width $w$ and height $h$. To achieve this, we define a projection function with a pinhole camera model as a concrete expression of (5):

$$\texttt{proj}_{\texttt{pinhole}} : \mathbb{R}^3 \to \mathbb{R}^2 : (X, Y, Z) \mapsto (p_x, p_y)\,. \tag{10}$$

In this model, the projection of the point $\boldsymbol{x}^*_{\texttt{camera}} = (X, Y, Z)$ onto the image plane is given by

$$p_x = \frac{f_x X}{Z} + c_x, \quad p_y = \frac{f_y Y}{Z} + c_y, \tag{11}$$

where $f_x$ and $f_y$ are the focal lengths along the $x$ and $y$ axes (in pixels), and $(c_x, c_y)$ represents the coordinates of the principal point in the image. In homogeneous coordinates, this mapping can be expressed as

$$\lambda \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{12}$$

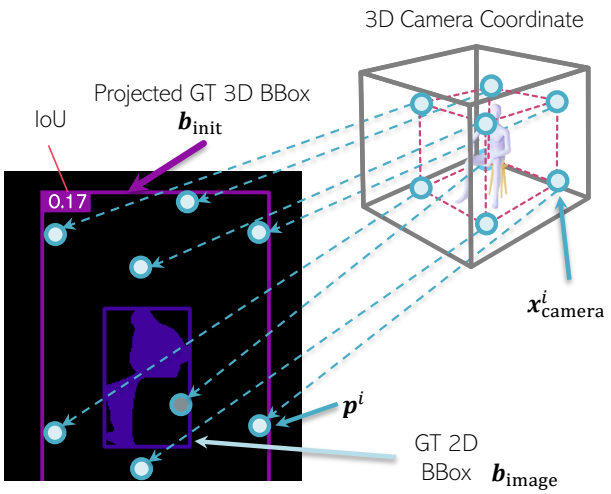


Figure 6. A direct projection of 3D BBoxes to the 2D image plane results in oversized 2D BBoxes. A learnable module is used to refine the projected BBoxes close to the 2D BBox GT.

with the scaling factor $\lambda = Z$. Thus, for each vertex, the projection onto the image plane is given by:

$$\boldsymbol{p}^i = \texttt{proj}_{\texttt{pinhole}}(\boldsymbol{x}^i_{\texttt{camera}}), \quad \text{for } i = 1, \ldots, 8, \tag{13}$$

where $\boldsymbol{p}^i = (p^i_x, p^i_y)$ represents the 2D coordinates of the projected point in the image plane. Once the eight vertices have been projected, the extreme coordinates on the image plane are determined as:

$$u_{\min} = \min_i\{p^i_x\}, \quad u_{\max} = \max_i\{p^i_x\}, \tag{14}$$

$$v_{\min} = \min_i\{p^i_y\}, \quad v_{\max} = \max_i\{p^i_y\}. \tag{15}$$

Using these extremes, the center coordinates, width, and height of the 2D BBox are computed by:

$$x_c = \frac{u_{\min} + u_{\max}}{2}, \quad y_c = \frac{v_{\min} + v_{\max}}{2}, \tag{16}$$

$$w = u_{\max} - u_{\min}, \quad h = v_{\max} - v_{\min}. \tag{17}$$

Thus, the final 2D BBox can be obtained as:

$$\boldsymbol{b}_{\texttt{init}} = (x_c, y_c, w, h)\,. \tag{18}$$

The 2D BBoxes obtained by projection, as shown by the purple box $\boldsymbol{b}_{\texttt{init}}$ in Fig. 6, are often too large. This occurs because projecting the eight vertices $\boldsymbol{x}^i_{\texttt{camera}}$ captures the depth information from the camera, which causes both the near and far parts of the object to be displayed in a 3D manner. As a result, to accurately predict the 2D BBox $\boldsymbol{b}_{\texttt{image}}$ on the image plane, we must use a refinement module. This module reduces the size of the initial BBox, as illustrated by the blue boxes in Fig. 6.

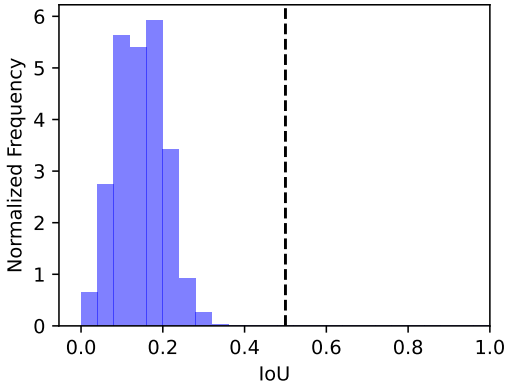


Figure 7. IoU histogram when no image plane supervision. Almost all IoU values are lower than 0.5, resulting in 0 AP.

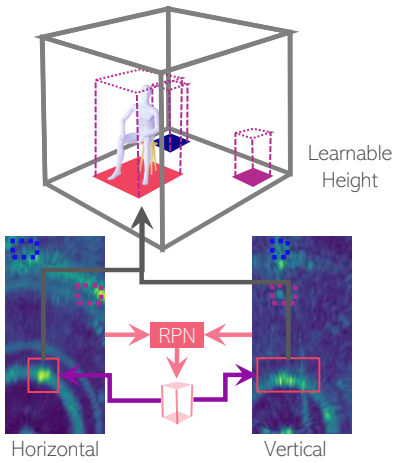


Figure 8. 3D Proposals with RFMask3D.

To better understand the need for refinement, we calculated the Intersection over Union (IoU) between the ground-truth (GT) 3D BBoxes (projected from the 3D space) and the GT 2D BBoxes (defined on the image plane). The histogram of IoU values in Fig. 7 shows a roughly Gaussian distribution with a peak around 0.15, and nearly all IoU values are below 0.5. In fact, in Fig. 6, the IoU is 0.17. This indicates that if we do not apply refinement, even when the 3D BBoxes are correctly predicted in the radar coordinate system, the average precision (AP) on the image plane would be zero. Therefore, our REXO method uses a refinement module.

## D. Baselines

**RFMask, DETR, and RETR** Since RFMask [37] and DETR [5] originally compute the BBox loss only in the 2D horizontal radar plane and the 2D image plane, respectively, we follow the implementation of RETR and enhance both methods with a unified bi-plane BBox loss. Furthermore,

we introduce a DETR variant with a top-$K$ feature selection, allowing it to take features from both horizontal and vertical heatmaps as input. For RETR [40], we set the number of object queries to 10. To ensure a fair comparison, we also set $N_{\texttt{train}} = 10$ for REXO during training.

**RFMask3D** As one of the baselines in our evaluation experiments, we constructed RFMask3D by extending RFMask [37] to 3D. RFMask uses a region proposal network (RPN) to extract regions of interest (RoIs) from a horizontal heatmap based on 2D anchor boxes and predicts 3D BBoxes in the 3D radar coordinate system by combining them with fixed heights. By designing an RPN that uses 3D anchor boxes, we explicitly extract 3D RoIs from both horizontal and vertical heatmaps, as shown in Fig. 8, enabling the estimation of 3D BBoxes. Unlike RFMask, this method allows for the learning of height as well.

## E. Hyperparameters for Performance Evaluation

The hyper-parameters used in our experiments of Section 3 are shown in Table 2. The table is divided into three parts, Data, Model, and Training, each with parameter names, notations, and values for each dataset.

## F. Definition of Metrics

**Mean Intersection over Union:** We adopt average precision on intersection over union (IoU) [10] as an evaluation metric. IoU is the ratio of the overlap to the union of a predicted BBox $A$ and annotated BBox $B$ as:

$$\texttt{IoU}\left(A, B\right) = \frac{|A \bigcap B|}{|A \bigcup B|}. \tag{19}$$

**Average Precision:** Average Precision (AP) can then be defined as the area under the interpolated precision-recall curve, which can be calculated using the following formula:

$$\texttt{AP} = \sum_{i=1}^{n-1} \left(r_{i+1} - r_i\right) p_{\texttt{interp}}\left(r_{i+1}\right), \tag{20}$$

$$p_{\texttt{interp}}\left(r\right) = \max_{r' \geq r} p\left(r'\right), \tag{21}$$

where the interpolated precision $p_{\texttt{interp}}$ at a certain recall level $r$ is defined as the highest precision found for any recall level $r' \geq r$. We present three variants of average precision: $\texttt{AP}_{50}$, $\texttt{AP}_{75}$, and $\texttt{AP}$, where the former two represent the loose and strict constraints of IoU, while $\texttt{AP}$ is the averaged score over 10 different IoU thresholds in $[0.5, 0.95]$ with a step size of $0.05$.

**Average Recall:** Average recall (AR) [19] between 0.5 and 1 of $\texttt{IoU overlap threshold}$ can be computed by averaging over the overlaps of each annotation $\text{gt}_i$ with the closest matched proposal, that is integrating over the $y$ : $\texttt{recall}$ axis of the plot instead of the $x$ : $\texttt{IoU overlap threshold}$ axis. Let $o$ be the $\texttt{IoU}$ overlap and $\texttt{recall}\left(o\right)$ the function. Let $\texttt{IoU}\left(\text{gt}_i\right)$ denote the $\texttt{IoU}$ between the annotation $\text{gt}_i$ and the closest detection proposal:

$$\texttt{AR} = 2 \int_{0.5}^{1} \texttt{recall}(o) \text{d}o \tag{22}$$

$$= \frac{2}{n} \sum_{i=1}^{n} \max\left(\texttt{IoU}\left(\text{gt}_i\right) - 0.5, 0\right). \tag{23}$$

The following are some variations of $\texttt{AR}$:
- $\texttt{AR}_1$: $\texttt{AR}$ given one detection per frame;
- $\texttt{AR}_{10}$: $\texttt{AR}$ given 10 detection per frame;
- $\texttt{AR}_{100}$: $\texttt{AR}$ given 100 detection per frame.

## G. Analysis of Failure Cases:

We provide failure cases in Fig. 9. These are all results of "Unseen," which means the environment that is not included in the training data (d8). As with d8s1 and d8s3, REXO may sometimes predict inaccurate positions, although less frequently than RETR and RFMask. In addition, there are cases where false negatives occur, such as with d8s2, d8s4, d8s5, and d8s6. In particular, it is thought to be difficult to capture the characteristics of individuals that are far away from the radar, such as with d8s2, because the resolution becomes coarse. In addition, REXO frequently gets false positives such as d8s2 - d8s6, so adjusting the threshold is important.

Table 2. Details of hyper-parameters. Fixed height for the HIBER dataset depends on the environment.

| | Name | Notation | Value | | | |
|---|---|---|---|---|---|---|
| | | | P1S1 | P1S2 | P2S1 | P2S2 |
| **Data** | # of training | - | 86579 | 70266 | 190441 | 118280 |
| | # of validation | - | 10538 | 24398 | 23899 | 33841 |
| | # of test | - | 10785 | 13238 | 23458 | 85677 |
| | Input radar heatmap size | $H \times W$ | 256×128 | 256×128 | 256×128 | 256×128 |
| | Segmentation mask size | $H \times W$ | 240×320 | 240×320 | 240×320 | 240×320 |
| | Resolution of range | cm | 11.5 | 11.5 | 11.5 | 11.5 |
| | Resolution of azimuth | deg. | 1.3 | 1.3 | 1.3 | 1.3 |
| | Resolution of elevation | deg. | 1.3 | 1.3 | 1.3 | 1.3 |
| | Scale | - | log | log | log | log |
| **Model** | Backbone | - | ResNet18 | ResNet18 | ResNet18 | ResNet18 |
| | # of input consecutive radar frames | $M$ | 4 | 4 | 4 | 4 |
| | Extracted feature map size | $H/s \times W/s$ | 64×32 | 64×32 | 64×32 | 64×32 |
| | The number of BBoxes | $N_{\texttt{train}}$ | 10 | 10 | 10 | 10 |
| | Threshold for detection | - | 0.5 | 0.5 | 0.5 | 0.5 |
| | Loss weight for `GIoU` on radar coordinate system | $\lambda_{\texttt{GIoU}}$ | 2.0 | 2.0 | 2.0 | 2.0 |
| | Loss weight for `GIoU` on image plane | $\lambda_{\texttt{GIoU}}$ | 2.0 | 2.0 | 2.0 | 2.0 |
| | Loss weight for $\texttt{L}_1$ on radar coordinate system | $\lambda_{\texttt{L}_1}$ | 5.0 | 5.0 | 5.0 | 5.0 |
| | Loss weight for $\texttt{L}_1$ on image plane | $\lambda_{\texttt{L}_1}$ | 5.0 | 5.0 | 5.0 | 5.0 |
| | Loss weight for `radar` | $\lambda_{\texttt{3D}}$ | 1.0 | 1.0 | 1.0 | 1.0 |
| | Loss weight for `image` | $\lambda_{\texttt{2D}}$ | 1.0 | 1.0 | 1.0 | 1.0 |
| **Training** | Batch size | - | 32 | 32 | 32 | 32 |
| | Epoch for detection | - | 100 | 100 | 100 | 100 |
| | Patience for early stopping | - | 5 | 5 | 5 | 5 |
| | Check val every $n$ epoch for early stopping | - | 2 | 2 | 2 | 2 |
| | Optimizer | - | AdamW | AdamW | AdamW | AdamW |
| | Learning rate | - | 1e-4 | 1e-4 | 1e-4 | 1e-4 |
| | Sheduler | - | Cosine | Cosine | Cosine | Cosine |
| | Maximum number of epochs for sheduler | - | 100 | 100 | 100 | 100 |
| | Weight decay | - | 1e-3 | 1e-3 | 1e-3 | 1e-3 |
| | # of workers | - | 8 | 8 | 8 | 8 |
| | GPU (NVIDIA) | - | A40 | A40 | A40 | A40 |
| | # of GPUs | - | 1 | 1 | 1 | 1 |
| | Approximate training time | day | 1 | 1 | 2 | 2 |

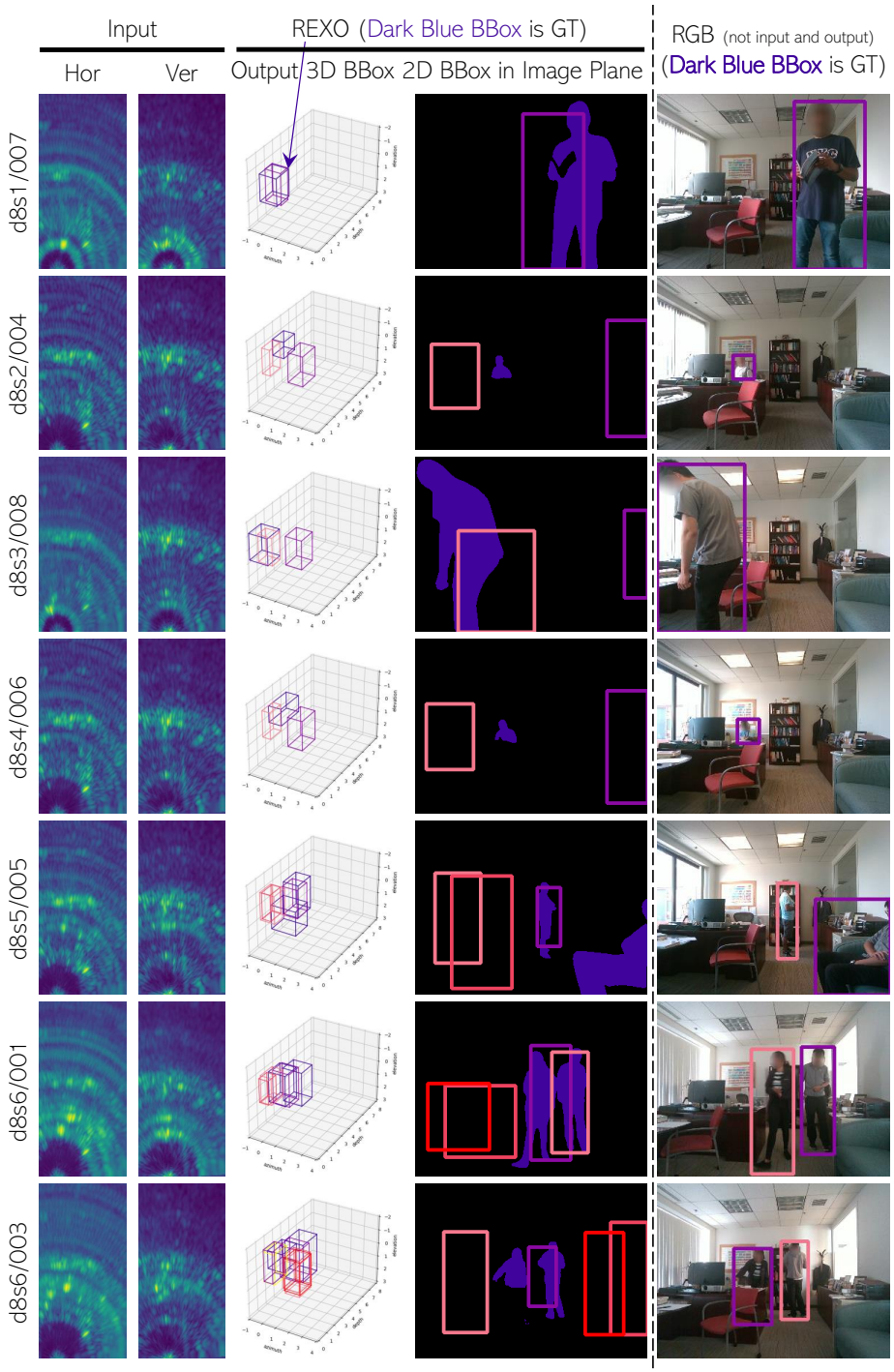


Figure 9. Visualization of failure cases. Each row indicates the segment name used from the P2S2 test dataset.