## Competition 2: 4D Millimeter-Wave Radar and Monocular Camera Fusion Algorithm

### 1.Competition Background

Traditional 3D millimeter-wave radar point clouds are typically used as auxiliary information for object detection because they are extremely sparse and lack critical height information. The next-generation high-resolution 4D (3D spatial position + 1D velocity) millimeter-wave radar offers higher resolution and the ability to measure pitch/height angles, showing excellent potential for 3D object detection.However, due to its recent market introduction, research on 3D object detection algorithms that fuse 4D radar point clouds with vision data is still limited. Existing 4D radar datasets are scarce and mostly collected for long-distance scenarios.For this competition, a dataset of vehicle-mounted 4D millimeter-wave radar and camera data in road environments is provided. Participating teams are required to develop 3D object detection algorithms based on radar-vision fusion using this dataset. The competition primarily evaluates teams on their mastery and innovation in key techniques, including data augmentation, radar-vision data alignment, and feature fusion.

### 2. Competition Application Scenarios

In the context of the rapid development of intelligent driving and autonomous driving technologies, multi-sensor fusion has become key to enhancing vehicle perception capabilities. 4D millimeter-wave radar and monocular cameras are highly complementary sensors, each performing well under different conditions. Millimeter-wave radar can provide reliable detection in adverse weather conditions such as rain, fog, and snow, or in low-light environments. Monocular cameras, on the other hand, provide high-precision image information in good weather and lighting conditions. However, single sensors have limitations: millimeter-wave radar is relatively weak in resolution and capturing fine details, while cameras may struggle with long-range detection or in complex environments. By leveraging vision fusion algorithms and artificial intelligence, it is possible to achieve accurate detection, classification, and tracking of vehicles, pedestrians, obstacles, and other objects in complex and dynamic environments, thereby enhancing the perception capabilities and safety of autonomous driving systems.

### 3. Competition Tasks

Participants are required to use the high-resolution 4D millimeter-wave radar dense point cloud dataset provided by the organizers to design and implement artificial intelligence algorithms for automatic detection and classification of road objects. The specific tasks include:

**(1) 3D Object Detection:** Accurately locate the 3D positions and bounding boxes of targets.

**(2) Classification**: Determine the category to which each detected target belongs.

**4. Dataset and Data Description**

4.1 Data Source

The data were self-collected on urban roads. The data collection platform is shown in Figure 1. A high-resolution 4D millimeter-wave radar (OCULII-EAGLE) and a camera (Intel RealSense D435i) are mounted at the center on top of the vehicle, approximately 1.85 meters above the ground.To ensure the density of the 4D radar point clouds and generate dense radar point clouds, the short-range mode was selected during data collection. In this mode, as shown in Figure 2, the high-resolution 4D millimeter-wave radar OCULII-EAGLE has a detection range of 0–25 meters, an azimuth angle range of -56.5° to 56.5°, a pitch angle range of -2.5° to 22.5°, and collects 12 frames of point clouds per second.


**Figure 1 Data Collection Platform**

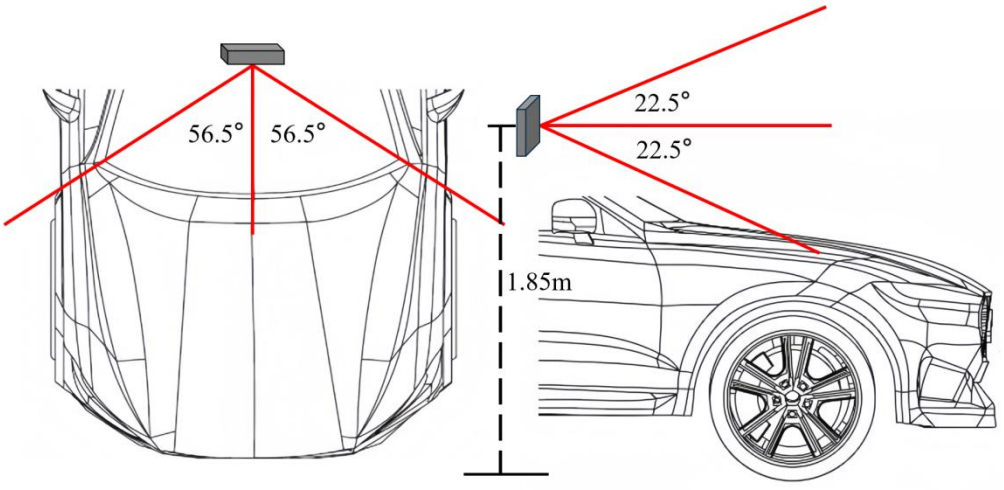


**Figure 2 High-Resolution 4D Millimeter-Ware Radar field of View illustration**

The collected 4D millimeter-wave radar point cloud data are in the format [range, azimuth, elevation, Doppler, power, x, y, z], where “power (dB)” represents the signal-to-noise ratio of the detection signal, “Doppler” indicates the relative velocity between the target and the radar, and “range, azimuth, elevation” are the polar coordinates, while “x, y, z” are the Cartesian coordinates calculated from the polar coordinates. The Intel RealSense D435i camera has a field of view (FOV) of 69° × 42° (H × V), a resolution of 640 × 480 pixels, and captures images at 30 frames per second.

4.2 Parameter Calibration

Sensor parameter calibration is a fundamental requirement to ensure the coordinated operation of multiple sensors in autonomous driving systems. Proper calibration guarantees that data collected from different sensors can be accurately aligned within the same coordinate system, providing a consistent perceptual basis for the perception system. This includes determining the internal mapping relationships of sensors (intrinsic calibration), such as camera focal length and distortion parameters, as well as determining the transformation relationships between sensors and other coordinate systems (extrinsic calibration), for example, the relative position and orientation between the camera and the 4D millimeter-wave radar. The accuracy of calibration directly affects the system’s understanding of the surrounding environment and the reliability of its decision-making.

To ensure stable and precise radar-vision fusion, parameter calibration between the 4D millimeter-wave radar and the camera is required. As shown in Figure 1 of the data collection platform, the 4D radar and the camera use different coordinate systems. In this task, the 4D radar coordinate system is treated as the world coordinate system, with the vehicle’s forward direction as the positive X-axis, the Y-axis pointing to the left of the forward direction, and the Z-axis pointing vertically upward.

Mapping the 4D radar point clouds from the world coordinate system to the camera image’s pixel coordinate system requires three transformations. Ignoring image distortion, the process first involves a rigid-body transformation from the world coordinate system to the camera coordinate system, followed by a perspective projection from the camera coordinate system to the image plane, and finally an affine transformation from the image plane to the pixel coordinate system.

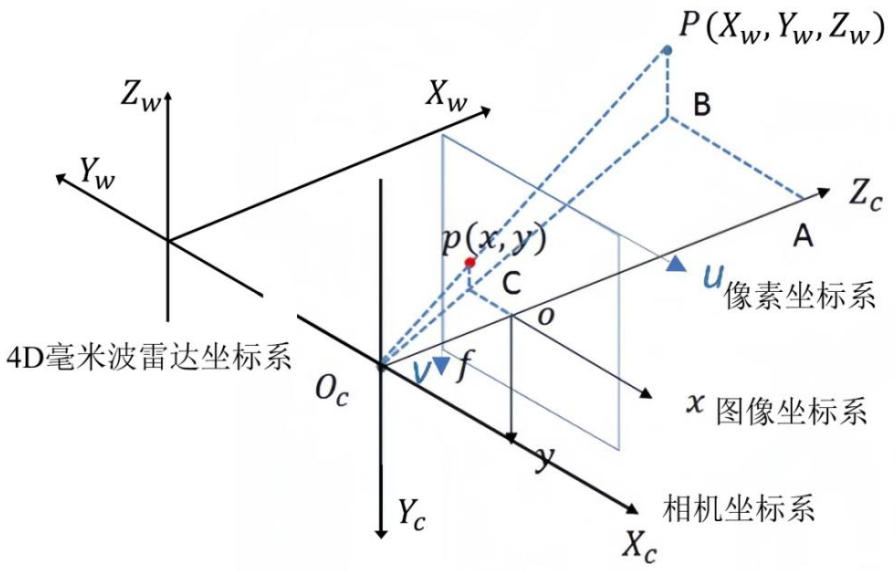


**Figure 3 Muiti-Sensor Coordinate System Relationship Model**

As shown in Figure 3, to associate points in the 4D millimeter-wave radar point cloud (in the world coordinate system) with pixel coordinates in the image, the points must first be transformed into the camera coordinate system using rotation and translation, as expressed in Equation (4.1).

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ X_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.1}$$

Here, R is the rotation matrix and T is the translation matrix. In this paper, we align the camera and the 4D millimeter-wave radar by ensuring they share consistent forward orientation, similar distance, and identical height through installation adjustments. The spatial parameters between the two sensors are obtained through manual measurement. Subsequently, these parameters are manually fine-tuned by comparing the projection of key target point clouds from multiple frames with corresponding image objects. Finally, the calibrated extrinsic parameter matrix is obtained as follows:

$$\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -0.25 \\ 0 & 0 & -1 & 0.4 \\ 1 & 0 & 0 & -0.25 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

After obtaining points $P_c(X_c, Y_c, Z_c)$ in the camera coordinate system, perspective projection is applied. By translating the origin to the top-left corner, the pixel coordinates $p(u, v)$ can be obtained. The matrix formula is as follows:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = KP_c \tag{4.2}$$

Here, $c_x, c_y$is the pixel distance for origin translation,$f_x = f/d_x, f_y = f/d_y$ , and $f$ are the focal lengths, and $d_x, d_y$ is the physical width of a single pixel. $K$ represents the intrinsic matrix. We use the standard intrinsic matrix obtained from the camera's factory calibration as our conversion intrinsic matrix:

$$K = \begin{bmatrix} 605.6403 & 0.0 & 319.2964 & 0.0 \\ 0.0 & 605.6746 & 235.4414 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

Combining formulas (4.1) and (4.2), the conversion relationship between the world coordinate system of the 4D millimeter-wave radar point cloud and the pixel coordinates of the camera image is obtained as follows:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ T_w \\ Z_w \\ 1 \end{bmatrix} \tag{4.3}$$

Based on this, we establish a transformation matrix that relates the 4D millimeter-wave radar to the image. Through this equation, the point cloud can be projected onto the image plane.

4.3 Data Scale

The preliminary round provides a total of 6,551 sample frames, with 5,169 frames in the training set for participants to train their algorithms. The validation set is self-divided by participants for model tuning and performance evaluation, and the test set contains 1,382 frames for final result assessment. The semi-final round will provide a different test set from the preliminary round, while the training set remains the same, and the test set still contains 1,382 frames. Sample data can be accessed at https://pan.baidu.com/s/1X-1rxmZ5SQl5OnYeIPZEEA?pwd=q6px, and the official dataset will be available for download after registration.

4.4 Data Format

A total of over 10,000 radar point cloud frames and a corresponding number of image frames were collected. From these data, multiple consecutive-frame scenes were selected to form the dataset samples, and 7,933 key frames were manually annotated. All objects were labeled with class, 3D bounding box, rotation, and ID, and

the annotations were converted into the KITTI label format, as shown in Figure 4. The annotated objects include Car (small car), Cyclist, and Truck. Pedestrians were not annotated due to the sparse and hard-to-identify point clouds.

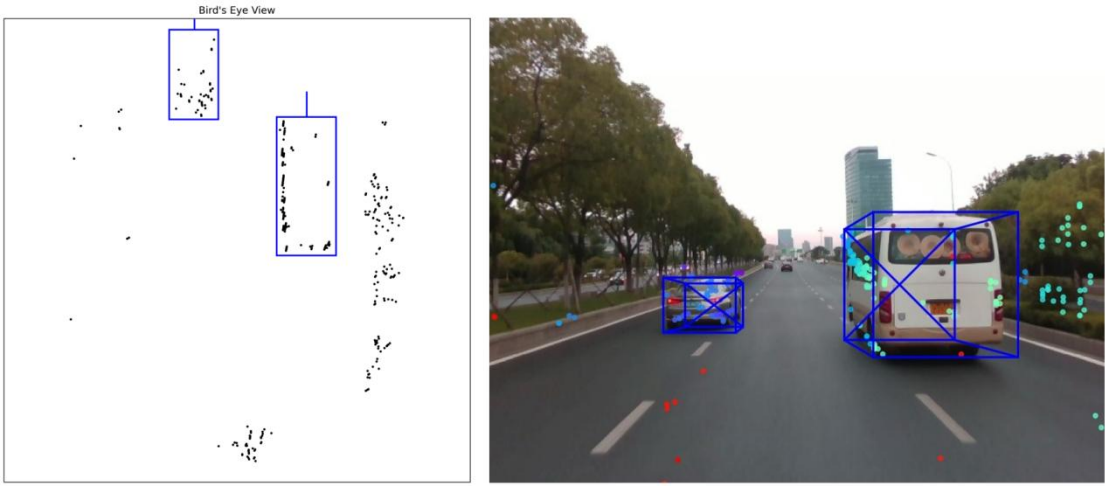


**Figure 4 Data Annotation Illustration**

As shown in Figure 5, the DRadDataset dataset is organized with the ImageSets folder containing the sample splits, the training folder including both training and validation data, and the calib folder holding the calibration files mentioned in Section (II). The image_2 folder contains the captured camera images, label_2 contains the manually annotated labels, and velodyne contains the high-resolution 4D millimeter-wave radar point cloud files in binary .bin format. The image, point cloud, and label files are linked through a unified file name indexing system, ensuring correct correspondence across all modalities.

```
DRadDataset
├── ImageSets
├── testing
├── training
│   ├── calib
│   ├── image_2
│   │   ├──000000.png
│   │   ├──...
│   ├── label_2
│   │   ├──000000.txt
│   │   ├──...
│   ├── velodyne
│   │   ├──000000.bin
│   │   ├──...
```

**Figure 5 Data Organization Type**

Calib Folder Description: The calibration follows the KITTI dataset format. P0-P3 are the camera intrinsic matrices. In this dataset, only one camera is used, so only P0

is valid, while P1-P3 can be ignored. P0 corresponds to the intrinsic matrix K mentioned in Section (II) Parameter Calibration, obtained from the camera factory settings. R0_rect is the rectification matrix of the camera, which is approximately an identity matrix and can be ignored. Tr_velo_to_cam is the extrinsic matrix transforming points from the radar coordinate system to the camera coordinate system, calibrated according to the actual placement of the radar and camera. Tr_imu_to_velo is an identity matrix and can also be ignored.

Label Folder Description: The label files are in .txt format, with each line representing one annotated object in the image. Each object annotation follows the standard KITTI dataset format for 3D object detection labels, specifically:[type, truncated, occluded, alpha, bbox_left, bbox_top, bbox_right, bbox_bottom, height, width, length, x, y, z, rotation_y].

1.type: The object category. Common categories include *Car*, *Pedestrian*, *Cy clist*, etc.

2.truncated: The level of truncation of the object, with values in the range [0, 1], indicating whether the object is partially cut off. *0* means the object is fully visible, while *1* means the object is heavily truncated.

3.occluded: The level of occlusion of the object, with values in the range [0, 3]: *0* means the object is fully visible, *1* means partially occluded, *2* mean s largely occluded, and *3* means fully occluded.

4.alpha: The observation angle of the object, representing the angle between the object and the camera's coordinate system.

5.bbox_left, bbox_top, bbox_right, bbox_bottom: The bounding box coordina tes of the object in the image, corresponding to the left, top, right, and bottom pixel positions.

6.height, width, length: The 3D dimensions of the object (height, width, len gth).

7.x, y, z: The 3D position coordinates of the object in meters.

8.rotation_y: The object's orientation, representing the rotation angle of the object in 3D space, typically around the y-axis, measured in

**5. Algorithm Design Requirements**

5.1 Model Type

To encourage innovation and improve detection accuracy, participants are encouraged to adopt deep learning algorithms for modeling. For example, voxel- or pillar-based 3D object detection networks (such as PointPillars or SECOND),

point-based detection networks (such as PointRCNN), or hybrid methods combining points and voxels (such as PVRCNN) can be employed. For multimodal fusion, participants may integrate mainstream visual object detection models (such as YOLO or Faster R-CNN) and explore cross-modal feature fusion methods based on Transformers. Specifically, Sparse CNNs can be used to extract geometric features from 4D millimeter-wave radar point clouds, while CNNs or Vision Transformers can be applied to extract semantic features from monocular images. Subsequently, efficient multimodal fusion modules can be designed, such as BEV (bird's-eye view) spatial feature alignment and attention-based fusion mechanisms, to achieve accurate detection and classification of various road objects.

5.2 Innovation

Participants are encouraged to propose innovative fusion strategies or improve existing algorithms to enhance detection accuracy and robustness. For example, they may design novel feature fusion modules capable of fully capturing multi-scale contextual information from sparse 4D millimeter-wave radar features, while also performing feature enhancement. Alternatively, effective solutions can be explored for the fusion of dense point cloud features from high-resolution 4D millimeter-wave radar with image features, in order to improve the accuracy of 3D object detection.

5.3 Scalability

The algorithm should possess strong scalability, capable of running efficiently on computing devices with different configurations, while maintaining stable performance when processing large-scale data. For instance, it should operate effectively on both standard workstations and cloud servers, and its performance should not degrade noticeably as the data volume increases.

**6. Performance Metrics Requirements**

6.1 Key Metrics

1.IoU (Intersection over Union): Measures the overlap between the model's detected bounding box and the ground truth. It is calculated as the ratio of the area of intersection to the area of union between the predicted and true boxes. IoU ranges from 0 to 1, with values closer to 1 indicating a higher match between the detection and the actual target. IoU is widely used in evaluating and optimizing object detection algorithms, especially in tasks requiring precise localization, such as autonomous driving and object recognition.

2.AP (Average Precision): An important metric for evaluating object detection performance. It combines Precision and Recall, measuring the area under the

Precision-Recall (P-R) curve to assess the model's detection performance for different classes. A higher AP indicates better performance for a given class. In multi-class detection tasks, AP is calculated for each class, and the mean of all classes' AP values gives the mAP (mean Average Precision).

6.2 Secondary Metrics

Model Size: The size of the trained model file, which reflects the model's complexity and storage requirements. A smaller model size indicates lower complexity, reduced storage cost, and easier deployment, making it more suitable for application across different devices and environments.

**7. Functional Requirements**

7.1 Accuracy

The algorithm must achieve high accuracy in detecting vehicle positions, marking 3D bounding boxes, and performing classification, ensuring precise recognition of targets with sparse point clouds, such as cyclists, while minimizing missed detections and false positives. For vehicle classification, the predicted labels should closely match the ground truth. On the test set, the Intersection over Union (IoU) for 3D object detection should reach at least 0.5.

7.2 Reliability

The algorithm should operate stably and produce reliable results when processing 4D radar point cloud frames collected under varying quality and different scenarios. Even in the presence of noise or other disturbances in the point clouds, the algorithm's performance should not fluctuate significantly, maintaining accurate and consistent 3D object detection and classification for vehicles.

7.3 Interpretability

The algorithm should possess a certain degree of interpretability, providing relevant personnel in the autonomous driving field with explanations for the 3D vehicle detection and classification results. For example, visualization techniques can be used to highlight the key regions the model focuses on when identifying objects, or feature importance analysis can be provided to indicate which features the model relies on for 3D object detection and classification. This helps users understand the algorithm's decision-making process and enhances trust in the results.

7.4 Real-time performance

In autonomous driving scenarios, decision-making latency is critical. The algorithm must meet certain real-time requirements: the time for performing 3D object detection and classification on a single sample frame should be controlled

within [150 milliseconds]. This ensures that the autonomous driving system can quickly perceive and understand road conditions and obstacles, providing key perception data to downstream planning and decision-making modules for rapid and safe actions.

7.5 Robustness

The algorithm should demonstrate strong robustness to anomalies, missing data, and other irregularities. Even when some point cloud data contain minor labeling errors, missing points, or clutter interference, the detection and classification results should remain reliable, ensuring that small imperfections in the data do not cause significant performance degradation.

7.6 Multimodal Fusion Capability

If participants adopt a multimodal data fusion approach (e.g., combining radar point clouds and visual images), the algorithm should effectively integrate different types of data and, after fusion, significantly improve the accuracy and reliability of 3D object detection, demonstrating efficient utilization of information from multiple sources.

**8. Development Environment**

8.1 Programming Language

Python, it is recommended to use Python 3.6 or above due to its rich support for scientific computing libraries and deep learning frameworks.

8.2 Deep Learning Framework

It is recommended to use TensorFlow 2.x or PyTorch 1.x. Both frameworks are widely used in the field of deep learning, offering efficient computation and a rich set of APIs, which facilitate model construction, training, and deployment.

8.3 Computing Resources

Participants can use either local workstations or cloud computing platforms for development and training. Local workstations should be equipped with NVIDIA GPUs (e.g., GTX 10 series or above, or RTX series) to accelerate deep learning computations. Cloud platforms such as Alibaba Cloud Tianchi, Tencent Cloud TI, or Baidu AIStudio offer a variety of computing configurations, allowing participants to flexibly choose resources based on their needs.

8.4 Dependency libraries

Participants are required to install libraries for data processing and visualization such as NumPy, Pandas, and Matplotlib; libraries for point cloud processing such as Open3D and PointNet; as well as relevant libraries corresponding to the chosen deep

learning framework, such as Keras for TensorFlow or TorchVision for PyTorch.

**9. Evaluation Criteria**

9.1 Input Data Format Requirements

Participants' algorithms should be able to correctly read the DRadDataset dataset provided by the organizer, which is organized similarly to the widely used KITTI dataset for autonomous driving algorithm training and evaluation. For the PNG image files, the algorithm must be able to parse and extract the contained image information. For the label files, it should accurately extract all manually annotated object information, including class, 3D bounding box, rotation, and ID. For the calibration parameter text files in the calib folder, the algorithm must accurately extract the transformation matrices describing the relationship between the 4D millimeter-wave radar and the camera images. For the high-resolution 4D millimeter-wave radar point cloud files in the velodyne folder, which are in binary .bin format, the algorithm must correctly parse and extract all contained point cloud information.

9.2 Output Data Format Requirements

The output format should be consistent with the label_2 folder of the training dataset, containing a prediction file (.txt) for each test sample. Each prediction file must have the same file name as its corresponding input data file (for example, 000001.txt corresponds to 000001.png/000001.bin).

Each prediction file should have one line per detected object; if no objects are detected, the file should be empty. Each line contains 16 fields, separated by spaces, in the following fixed order:

| Field Number | Field Name | Data Type | Unit / Range |
|---|---|---|---|
| 1 | Class | String | Car / Cyclist / Truck |
| 2 | Truncation | Float | [0, 1] |
| 3 | Occlusion Level | Integer | 0 / 1 / 2 / 3 |
| 4 | Observation Angle | Float | [-π, π] |
| 5–8 | 2D Bounding Box | Float | Pixel coordinates |
| 9–11 | 3D Dimensions | Float | Meters |
| 12–14 | 3D Center | Float | Meters |
| 15 | Rotation Angle | Float | [-π, π] |
| 16 | Detection Score | Float | [0, 1] |

**Example file content:**

For instance, detecting one car and one cyclist:

```
Car 0.00 0 1.57 712.40 143.00 810.73 307.92 1.65 1.67 3.64 -5.14 2.78 9.12 1.56 0.95
Cyclist 0.30 1 -0.50 512.30 180.40 550.21 220.70 1.75 0.60 0.80 2.40 1.89 15.20 -0.30 0.80
```

9.3 Score Calculation Formula

The scores will be calculated by comparing the algorithm's output results with the ground-truth annotations, using performance evaluation metrics such as IoU and mAP. The final score is obtained by a weighted summation of these metrics.

9.4 Valid Score

A final score above 40 is considered a valid result, with the threshold of 40 ensuring that the object detection algorithm has practical applicability. The number of teams with valid scores serves as the basis for award allocation.

**10. Problem-Solving Approach**

10.1 Data Preprocessing

Perform filtering on the point cloud data to remove background noise points while retaining foreground target points (such as vehicles, pedestrians, cyclists, etc.), reducing interference from irrelevant points and enhancing the point density in target regions. Then, divide the sparse point cloud into regular 3D voxel grids to form a structured representation, where each voxel is a fixed-size cube, reducing computational complexity while preserving spatial geometric information. For images, apply data augmentation techniques such as rotation, scaling, and flipping to expand the training dataset and improve the model's generalization ability. Additionally, image pixel values can be normalized.

10.2 Feature Extraction

Use sparse convolutions to process the voxelized point cloud and extract 3D sparse voxel features. Design combinations of different sparse convolution layers and pooling layers to significantly reduce computational complexity and improve efficiency when handling 3D sparse data. For visual image processing, leverage the powerful feature extraction capability of convolutional neural networks (CNNs) to capture high-level semantic information. Map both point cloud and visual image features into the BEV (Bird's Eye View) space for fusion, generating unified BEV fused features. This approach simplifies height information and reduces computational complexity while preserving scale consistency and 3D geometric information.

10.3 Model Training

Select an appropriate deep learning framework (such as TensorFlow or PyTorch) to build the model, and set reasonable training parameters, including learning rate,

number of iterations, and batch size. During training, use cross-validation to evaluate and fine-tune the model with the validation set data, helping to prevent overfitting.

10.4 Model Fusion and Optimization

You can try fusing multiple models with different architectures or trained at different stages, for example, using voting or weighted averaging to combine their predictions, thereby improving the overall accuracy. At the same time, optimize the models based on performance metrics, such as adjusting the model architecture or increasing the training data size.

## 11.Reference Resources

### 11.1 Books

1.Deep Learning by Ian Goodfellow, Yoshua Bengio, and Aaron Courville — This book systematically introduces the fundamental concepts, model architectures, and training methods of deep learning, providing substantial help in understanding and applying neural networks.

2.Deep Learning with Python by François Chollet — Through numerous code examples, this book explains in detail how to develop deep learning models using Python and the Keras framework, making it suitable for beginners to quickly get started.

### 11.2 Online courses

1.Deep Learning Specialization on Coursera, taught by Andrew Ng — This course covers multiple key areas of deep learning, including neural network fundamentals, convolutional neural networks, and recurrent neural networks, offering rich content with strong practical applications.

2.Introduction to Artificial Intelligence on edX — This course provides foundational knowledge in artificial intelligence and machine learning, including algorithm principles, model training, and application cases, helping participants build a comprehensive knowledge system.

### 11.3 Academic papers

Search academic databases for the latest research papers on 3D object detection from point clouds, such as “PointRCNN: 3D object proposal generation and detection from point cloud”, to understand the cutting-edge technologies and methodologies in this field.

Follow papers published in reputable AI conferences (e.g., AAAI) to keep track of the latest research trends and innovative developments.

## 12.Submission Requirements

**12.1 Algorithm Code**

Submit the complete algorithm code, including all stages such as data preprocessing, model training, and prediction inference. The code should be written in Python and follow the PEP8 coding standards, with clear comments and documentation to facilitate understanding and execution by the reviewers.

**12.2 Technical Report**

Submit a detailed technical report, including the algorithm design rationale, model architecture diagrams, experimental setup (such as training parameters and data augmentation methods), performance analysis (detailed evaluation of primary and secondary metrics), as well as the algorithm's innovations and limitations. The report should be in PDF format and contain no fewer than 3,000 words.

**12.3 Model File**

Submit the trained model files and provide instructions for loading and using the model, including the required runtime environment and dependency libraries. The model files should be able to run correctly in the specified test environment and produce the prediction results.

**13. Competition Process and Award Structure**

13.1 Registration Phase

Participants should complete the registration on the official competition website, submit their personal or team information, and obtain the download link for the preliminary (initial) round dataset.

13.2 Preliminary Round Phase

Participants use the training dataset provided by the organizers to design their algorithm models and utilize the preliminary round test set for validation and debugging of their methods. During the preliminary round, participants can submit results an unlimited number of times per day, but the preliminary leaderboard is updated every hour.

13.3 Semi-Final Round Phase

After the preliminary round ends, the competition enters the semifinal stage, during which the download link for the semifinal dataset is made available. Only teams that submitted valid results in the preliminary round are eligible to participate in the semifinals. During this stage, participants use the semifinal dataset provided by the organizers to fine-tune their algorithm models and submit inference results on the semifinal test data. The semifinal stage lasts for three days, and each team is allowed a maximum of two submissions per day. The semifinal leaderboard is updated every

hour.

13.4 Announcement of Semi-Final Results

The semifinal results are published on the competition's official website. The number of teams participating in the semifinals serves as the base for awarding prizes. First, second, and third prizes for the semifinals are determined according to the provincial competition's prize ratios (certificates are awarded for these prizes). During the evaluation process, any team whose submitted algorithm performance falls below the baseline reference score provided by the organizers will be considered to have an invalid result and will not receive an award. Teams winning first and second prizes in the semifinals advance to the national finals.

13.5 Final Stage

1.Online Evaluation of the Finals: Teams advancing to the finals will be evaluated based on the semifinal leaderboard. Using the number of teams entering the finals as the prize base and following the national competition's prize ratio, the list of candidates for the national first prize and the winners of the national second and third prizes will be determined (certificates will be issued for the second and third prizes).

2.Submission of Final Works: Candidates for the national first prize must submit, within the specified time, their technical documents, algorithm code, model files, demonstration videos, and supplementary materials. No modifications or additions will be accepted after the submission deadline.

3.Fiinal Review Stage: A professional review team will reproduce and verify the results of the submissions from the national first prize candidates. During the review, participants may be asked to provide explanations if any issues arise.

4.Offline Defense of the Finals: Candidates for the national first prize must submit their finalized technical documents, algorithm code, model files, demonstration videos, and supplementary materials within the specified time, and participate in the offline defense of the national finals. The final ranking and determination of the national first prize winners will be based on algorithm performance scores and offline defense scores. Teams that do not participate in the offline defense will be considered to have forfeited the prize. Certificates of honor will be awarded to the national first prize winners.

14. Other Instructions

14.1 Fairness

Any form of cheating is strictly prohibited, including but not limited to data leakage, overlap between pretraining data and test data, plagiarism of others' code, etc.

Once detected, the participant will be immediately disqualified, and relevant responsibilities will be pursued.

14.2 Intellectual Property Rights

Any form of cheating is strictly prohibited, including but not limited to data leakage, overlap between pretraining data and test data, plagiarism of others' code, etc. Once detected, the participant will be immediately disqualified, and relevant responsibilities will be pursued.

15. Contact Information

Competition communication QQ group: 730575503

Email: ymhhmy01@foxmail.com

Registration website: www.aicomp.cn