

Master's Thesis in Robotics, Cognition, Intelligence

# Active Learning for 3D Object Detection and Labeling

Aktives Lernen zur 3D-Objekterkennung und Beschriftung

<b>Supervisor</b>	Prof. Dr.-Ing. habil. Alois C. Knoll
<b>Advisor</b>	M.Sc. Walter Zimmer, M.Sc. Xavier Diaz
<b>Author</b>	Ahmed Alaaeldin Ghita
<b>Date</b>	December 15, 2023 in Munich



# Disclaimer

I confirm that this Master's Thesis is my own work and I have documented all sources and material used.

Munich, December 15, 2023

---

(Ahmed Alaaeldin Ghita)

## **Acknowledgment**

I want to express my profound gratitude to Walter Zimmer and Xavier Diaz, my research supervisors, for their continuous support, guidance, and encouragement throughout this master's thesis journey. I would also like to thank all the members of the Providentia++ perception team for the weekly constructive discussions and valuable feedback. Their broad perspectives and critical analyses enriched this work tremendously.

To my beloved family, my deepest gratitude for your boundless love, patience, and unwavering belief in me throughout this demanding journey. Your endless encouragement and faith in my abilities gave me strength and resilience, enabling me to navigate and surmount the challenges of this academic endeavor. I am eternally thankful for you.

Lastly, I offer my sincere thanks to everyone who contributed directly or indirectly to this research, whether through academic support or personal encouragement.

## Abstract

3D scene perception in robotics and autonomous driving has become critically reliant on comprehensive labeled 3D datasets. Such datasets contain multiple modalities to provide a robust understanding of the whole scene, cover a wide range of driving scenarios, encompass a diverse set of daytime and weather conditions, and provide annotations of all elements on roads. Yet, the annotation process remains a substantial challenge, especially for 3D point clouds. For instance, benchmark datasets like Waymo [45] have over 12 million LiDAR boxes, for which labeling a precise 3D box takes more than 100 seconds for an annotator [43]. Since deep learning models have a greedy attribute to the data, creating large-scale labeled 3D datasets is an actual bottleneck in developing robust 3D perception models. To alleviate this challenge, active learning aims to reduce the labeling costs by querying labels for a small fraction of the unlabeled data, thus maximizing the model’s performance while annotating the fewest possible samples. Among 3D perception tasks, 3D object detection is one of the most indispensable and fundamental tasks in any automotive or robotic perception system. 3D object detection aims to predict the locations, sizes, and classes of objects in the 3D space. In light of the downstream 3D object detection task, this work investigates the potential of utilizing active learning techniques to maximize the performance gain of 3D detection models while annotating the fewest samples possible. While active learning might significantly reduce annotation costs by selecting only the most informative point cloud frames, it concurrently elevates the computational expense due to the recurrent model training and evaluation cycles. Hence, continual learning methods can potentially reduce the computational burden by facilitating more efficient model updates, thereby maintaining the gains of active learning without encountering prohibitive computational expenses. To this end, we adopt the work of [29] as our baseline active learning framework (*Active3D*) and integrate it with the continuous training methods from [9]. Additionally, we integrate *Active3D* into the *Providentia Annotation (proAnno)* platform to enable AI-assisted features and to minimize manual labeling efforts. Furthermore, our framework ensures that the selected data samples for labeling are of maximum informativeness to our model training. We conduct extensive experiments on the TUMTraf [[tumtraf](#)] dataset, employing the two-stage 3D detector PV-RCNN [38] as our detector model.

## Zusammenfassung

Die 3D-Szenenwahrnehmung ist in der Robotik und beim autonomen Fahren immer mehr abhängig von umfassend beschrifteten 3D-Datensätzen. Solche Datensätze enthalten mehrere Modalitäten, um ein fundiertes Verständnis der gesamten Szene zu ermöglichen, ein breites Spektrum an Fahrscenarien abzudecken, unterschiedliche Tages- und Wetterbedingungen abzudecken und Beschriftungen zu allen Elementen auf Straßen bereitzustellen. Dennoch bleibt der Annotationsprozess eine erhebliche Herausforderung, insbesondere für 3D-Punktwolken. In Benchmark-Datensätzen wie Waymo [45] sind es beispielsweise 12.6 Millionen LiDAR-Boxen, bei denen die Beschriftung einer präzisen 3D-Box mehr als 100 Sekunden dauert [43]. In Anbetracht der Tatsache, dass Deep-Learning Modelle sehr datenhungrig sind, ist die Erstellung umfangreicher beschrifteter 3D-Datensätze ein Engpass bei der Entwicklung von robusten 3D-Wahrnehmungsmodellen. Um diese Herausforderung zu lindern, zielt aktives Lernen darauf ab, die Beschriftung durch eine Abfrage von Labels für einen kleinen Teil der unbeschrifteten Daten, zu reduzieren. Somit werden die Kosten der Modellleistung maximiert, indem nur eine geringe Stichprobe beschriftet wird. Unter den 3D-Wahrnehmungsaufgaben ist die 3D-Objekterkennung eine der unverzichtbarsten Aufgaben in jedem Automobil- oder Roboter-Wahrnehmungssystem. Die 3D-Objekterkennung zielt darauf ab Vorhersagen der Positionen, Größen und Klassen von Objekten im 3D-Raum zu machen. In Anbetracht der 3D-Objekterkennung wird in dieser Arbeit das Potenzial der Nutzung von aktivem Lernen untersucht, um die Performanz der 3D Objekterkennung zu vergrößern, indem möglichst wenige Stichproben beschriftet werden. Während aktives Lernen die Annotationskosten, durch die Auswahl der aussagekräftigsten Datenpunkte, erheblich senken könnte, wird gleichzeitig der Rechenaufwand erhöht, aufgrund der wiederkehrenden Trainings- und Bewertungszyklen des Modells. Daher können Methoden des kontinuierlichen Lernens den Rechenaufwand reduzieren, indem effiziente Modellaktualisierungen genutzt werden, wodurch die Vorteile des aktiven Lernens erhalten bleiben ohne einen hohen Rechenaufwand zu verursachen. Zu diesem Zweck übernehmen wir die Arbeit von [29] als unser Basis-Framework für aktives Lernen (Active3D) und integrieren die Methoden des kontinuierlichen Trainings aus [9]. Darüber hinaus integrieren wir Active3D in die Providentia Annotation (proAnno) Plattform, um eine KI-gestützte Funktion zu ermöglichen und den manuellen Beschriftungsaufwand zu minimieren. Desweiteren stellt unser Framework sicher, dass die ausgewählten Stichproben zum Beschriften von großer Aussagekraft für das Trainieren des Modells sind. Wir führen umfangreiche Experimente mit dem TUMTraf-Datensatz [tumtraf] durch und verwenden dabei den zweistufigen PV-RCNN [38] 3D-Detektor als unser Detektormodell.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	LiDAR-based Object Detection . . . . .	5
2.1.1	Background . . . . .	5
2.1.2	Problem Definition . . . . .	7
2.2	Datasets . . . . .	8
2.2.1	TUMTraffic Highway Dataset . . . . .	8
2.2.2	TUMTraffic Intersection Dataset . . . . .	9
2.2.3	DAIR Dataset . . . . .	10
2.3	3D Labeling . . . . .	11
2.3.1	Coordinate Systems . . . . .	11
2.3.2	Geometries for Labeling . . . . .	12
2.3.3	Spatial Rotation . . . . .	13
<b>3</b>	<b>Related Works</b>	<b>17</b>
3.1	3D Object Detection . . . . .	17
3.1.1	LiDAR-based 3D Object Detection . . . . .	17
3.1.2	Single-stage 3D Object Detection . . . . .	18
3.1.3	Two-stage 3D Object Detection . . . . .	19
3.2	Active Learning . . . . .	20
3.2.1	Active Learning for Object Detection . . . . .	22
3.3	3D Annotations Tools . . . . .	24
3.3.1	3D BAT . . . . .	24
3.3.2	LATTE . . . . .	25
3.3.3	SAnE . . . . .	26
3.3.4	ReBounce . . . . .	27
3.3.5	Xtreme1 . . . . .	28
3.3.6	SUSTech Points . . . . .	29
3.3.7	OpenAnnotate3D . . . . .	30
3.3.8	WebLabel . . . . .	31
3.3.9	PointCloud Lab . . . . .	32
3.3.10	labelCloud . . . . .	32
3.4	Annotation Formats for Object Detection . . . . .	33
3.4.1	KITTI Format . . . . .	34
3.4.2	nuScenes Schema . . . . .	34
3.4.3	ASAM OpenLABEL . . . . .	36
<b>4</b>	<b>Methodology</b>	<b>39</b>
4.1	Continual Training . . . . .	39
4.2	Temporal CRB . . . . .	43

4.3	Active Class Weighting . . . . .	46
4.4	Providentia Annotation 2.0 . . . . .	47
<b>5</b>	<b>Evaluation</b>	<b>53</b>
5.1	Experimental Setup . . . . .	53
5.1.1	Baseline Architecture . . . . .	53
5.1.2	Active Learning Framework . . . . .	54
5.1.3	Dataset . . . . .	55
5.1.4	Evaluation Metrics . . . . .	56
5.2	Quantitative Results . . . . .	57
5.2.1	Continuous Training Strategies . . . . .	57
5.2.2	Active Learning Strategies . . . . .	60
5.3	Ablation Studies . . . . .	65
5.3.1	Temporal Consistency in CRB . . . . .	65
5.3.2	Active Class Weighting in CRB . . . . .	70
5.4	Qualitative Results . . . . .	73
<b>6</b>	<b>Conclusion</b>	<b>79</b>
<b>7</b>	<b>Future Work</b>	<b>81</b>
	<b>Bibliography</b>	<b>83</b>

# List of Figures

1.1	Track sections covered by the Providentia (green) and the extended Providentia++ (blue) project with marked sensor positions . . . . .	2
2.1	Illustration of the LiDAR sensor concept. . . . .	5
2.2	Illustration of the range image. . . . .	6
2.3	Properties of point cloud data: (a) Irregular distribution of dense and sparse areas. (b) Unstructured set of points with no fixed grid structure. (c) Unordered set of points that are permutation invariant. . . . .	7
2.4	Overview of the test bed Providentia++ . . . . .	8
2.5	Two cameras and two LiDARs are used to create the TUMTraffic Intersection dataset. The sensors are mounted on a sign gantry and thus record the central intersection area. Additionally, the coordinate systems for each sensor and the road coordinate system, established at the northern end of the bridge, are illustrated in the figure. . . . .	10
2.6	The DAIR-V2X hardware setup comprises infrastructure sensors and vehicle sensors. . . . .	11
2.7	Right-hand Coordinate System. . . . .	12
2.8	2D Bounding box definition. . . . .	12
2.9	3D Bounding box definition. The highlighted face indicates the forward orientation of the box. . . . .	13
3.1	Single-stage point-voxel 3D object detectors bridge the point features, and voxel features through point-to-voxel and voxel-to-point transforms in the backbone networks. . . . .	19
3.2	Two-stage point-voxel 3D object detection employs a voxel-based detection framework to generate a set of 3D object proposals at the first stage. In the second stage, key points are sampled from the input point cloud, and then the 3D proposals are refined from the key points through point operators . . . . .	19
3.3	Stream-based Selective Sampling AL Scenario. Unlabeled data sampled are presented sequentially in a stream. . . . .	20
3.4	Membership Query Synthesis AL Scenario. The model generates new synthetic data for inquiry based on its current knowledge. . . . .	21
3.5	Pool-based AL Scenario. The learner selects specific data samples according to query criteria. . . . .	21
3.6	An illustrative diagram of the CRB framework [29], which systematically selects point clouds based on unique bounding box labels and distinct gradient and geometric features. This approach helps bridge the gap with the test set while keeping annotation efforts minimal. . . . .	23

3.7	Presents a snapshot of the tool's UI, demonstrating its functional components. 1) Shows an adjustable panoramic camera view that allows users to scroll sideways and adjust the height, offering a 360-degree perspective. 2) Shows the master view consisting of side view (top), front view (middle), and BEV view (bottom). 3) Shows an interactive 3D window where users can annotate objects within the point clouds in both perspective and orthographic views. . .	24
3.8	LATTE main user interface shows the point cloud from BEV perspective, the camera image with 2D masks, and a list of the objects. . . . .	25
3.9	SAnE features a user-friendly interface for semi-automatic annotation, utilizing a one-click annotation framework, further enhanced by a denoising point-wise segmentation technique and an innovative guided-tracking mechanism. .	26
3.10	ReBound's UI shows a multi-modal view with a point cloud viewer (top) and an RGB image viewer (bottom). The editing window provides options for adding, deleting, and adjusting bounding boxes with controls for translation and rotation, as well as other properties such as the annotation type, class, and transformation parameters. . . . .	27
3.11	The Xtreme1 annotation tool main interface with intuitive tools for labeling objects, combining camera images and 3D views to help users annotate and track multiple objects. . . . .	28
3.12	The main user interface of SUSTech POINTS is composed of several elements. It includes a perspective view of the 3D point cloud and a re-sizable photo context that can automatically switch between various camera images. Additionally, Side, Front, and BEV views. There is also a focused photo context, automatically selected based on the object in focus. . . . .	29
3.13	This architecture supports the development of visualization modules and interactive tools for 3D bounding boxes and tracking ID annotation. The platform's data pre-processing stage involves estimating intrinsic and extrinsic sensor parameters, selecting data frames consistent with calibration parameters for algorithm training and verification, and optionally using a set of AI algorithms for detection, tracking, and 2D & 3D fusion to generate initial annotation results.	29
3.14	The OpenAnnotate3D process begins when a user submits a labeling request. Initially, the system analyzes this request using the LLM interpreter and precise, prompt engineering. It's important to note that the interpreter might engage with the promptable vision module in multiple iterations to ensure that the interpreted text aligns with its reasoning capacity. Dense 2D masks are generated, which are then used to create 3D masks through a multi-modal spatial alignment process. To address any imperfections in the 2D masks, a combination of spatio-temporal fusion and correction is implemented to enhance the accuracy of the 3D labels. . . . .	30
3.15	The main user interface of WebLabel . . . . .	31
3.16	Levels of immersion: (a) An interactive non-immersive setup. (b) Semi-immersive setup. (c) Fully-immersive setup incorporating all 6 DoF. . . . .	32
3.17	The diagram showcases the nuScenes schema for capturing and organizing data for autonomous vehicle research. The schema is divided into four main components: Vehicle and Sensor, Extraction, Annotation, and Taxonomy. . . .	35
3.18	ASAM OpenLABEL multi-sensor data labeling concept. . . . .	37
3.19	Header of the JSON schema . . . . .	37
3.20	Content of the JSON schema . . . . .	37
3.21	ASAM OpenLABEL data structure for multi-sensor data labeling. . . . .	38

4.1	Active Continuous Training. Integrating continuous training within the pool-based active learning cycle. . . . .	40
4.2	Sliding window mechanism. . . . .	43
4.3	Conceptual diagram of ProAnno++ workflow. It shows two boxes representing the client and the server sides, which are connected by means of the Python Flask library. . . . .	47
4.4	An illustrative diagram of the Data Manager Class. . . . .	49
4.5	An illustrative diagram of the Data Manager Class. Data History is a Python Dictionary instance. It saves the date of the operation and the data file configurations before and after the operation is executed. The data file configuration is saved as a Python List of file names corresponding to each data split. . . . .	50
4.6	Data file structure on the Active3D end. . . . .	51
4.7	Data file structure on the proAnno end. . . . .	51
4.8	The proAnno++ interface showcases five camera images alongside a point cloud frame, offering a range of GUI options. These include switching between orthographic and perspective views, navigating through sequences and datasets, copying labels to subsequent frames, auto-generating labels, and adjusting individual objects. Additionally, when an object is selected, helper views on the left side of the UI provide visualization from three different angles, reducing the need to switch to a perspective view. This comprehensive approach enhances user interaction and efficiency in object annotation. . . . .	52
5.1	This figure displays the division of TUMTraF Intersection dataset into training, validation, and test splits. It features the distribution of object classes, 3D point distribution, class density per distance, and frame frequency histograms for each set. . . . .	55
5.2	The plot shows the performance of six continuous training strategies in terms of mean Average Precision (mAP) over the number of epochs. The x-axis represents the number of epochs, starting at 30 due to initial pre-training, and the y-axis represents the overall mAP. . . . .	58
5.3	Comparative line plots for three classes – Car, Pedestrian, and Bicycle – illustrating class-specific statistics for the selected frames by AL strategies at the final active training cycle. . . . .	64
5.4	The figures display the mAP scores across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top-left to the bottom right, respectively. The highest mAP score is marked with a red star, and the lowest mAP score is marked with a black circle. . . . .	65
5.5	The figures display the mAP scores for different object classes across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top left to the bottom right, respectively. . . . .	66
5.6	Entropy values for the $K_1$ frames chosen in stage 1 and $K_3$ of CRB and tCRB at the first and the last active training cycles, in the first and second row, respectively. The dashed red and blue lines represent the mean entropy across all frames. Each frame's entropy is calculated as the aggregate of label entropies from the predicted 3D bounding boxes. . . . .	67
5.7	The figure shows the Euclidean distance and Cosine similarity between the weighted mean gradients of CRB and tCRB along the active training process. . . . .	68
5.8	The figure shows each class's point density distribution at stages 1 and 3, comparing the results of CRB and tCRB. Additionally, black dashed lines represent the mean point density for each class at the final active training cycle. . . . .	69

5.9	The figure presents the point density distribution and the number of predicted 3D bounding boxes per frame for CRB and tCRB at the final active training cycle.	70
5.10	The figures display the mAP scores across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top-left to the bottom right, respectively. The highest mAP score is marked with a red star, and the lowest mAP score is marked with a black circle.	71
5.11	The weights of the object classes computed at every active training cycle as the reciprocal of the class count in the current training set.	71
5.12	The figure displays a statistical comparison of the object classes in the selected frames by CRB and tCRB. The statistics are averaged across all active training cycles.	72
5.13	The figure shows the detection results of the PV-RCNN model across three scenarios: Firstly, the first row shows the detection outcomes when employing the CRB query strategy for active training. Secondly, the second row shows the results of the tCRB query strategy for active training. Lastly, the third row shows the detection results of the oracle model, which has been trained on the complete training dataset. It should be noted that the two columns in the figure correspond to two neighboring cameras, which capture different perspectives of the same traffic scene.	73
5.14	The figure shows the detection results of the PV-RCNN model in active training using CRB and tCRB query strategies.	74
5.15	The figure shows the detection results of the PV-RCNN model on the same point cloud frame at different stages in the active training process, employing the tCRB strategy. The first row corresponds to the detections of the model's state at epoch 130 (the 6th selection cycle). The second row corresponds to epoch 150 (the 7th selection cycle). The third row corresponds to epoch 170 (the 8th selection cycle), and finally, the last row corresponds to epoch 190 (the 9th selection cycle).	75
5.16	The figure shows the detection results of the PV-RCNN model on the same point cloud frame at different stages in the active training process, employing the CRB strategy. The first row corresponds to the detections of the model's state at epoch 130 (the 6th selection cycle). The second row corresponds to epoch 150 (the 7th selection cycle). The third row corresponds to epoch 170 (the 8th selection cycle), and finally, the last row corresponds to epoch 190 (the 9th selection cycle).	76
5.17	The figure displays the results of the PV-RCNN inference on a point cloud frame within the proAnno++ interface. This interface includes a view of five different cameras: four cameras mounted on a gantry bridge, and one is mounted on top of a moving vehicle. The visualization presents 3D bounding boxes, rendered both in the default perspective view and in an orthographic view.	77

# List of Tables

2.1	TUMTraffic Highway statistics including the total number of labels, average dimensions in meters, and the average number of 3D LiDAR points among all classes. . . . .	9
2.2	TUMTraffic Intersection statistics including the total number of labels, average dimensions in meters, and the average number of 3D LiDAR points among all classes. . . . .	10
3.1	Comparison of 3D annotation tools. ○ Feature provided ○ Feature unknown ○ Feature not provided . . . . .	33
3.2	Description of the data fields in a KITTI format labels file. . . . .	34
3.3	Description of the data fields in the nuScenes sample data table. . . . .	35
3.4	Description of the data fields in the nuScenes sample annotation table. . . . .	36
4.1	The table shows 6 model update strategies: <b>1.</b> Updating the initial model on the growing set of data. <b>2.</b> Fine-tuning the initial model on the latest acquired set. <b>3.</b> Fine-tuning the initial model on the latest acquired set, and performing the final training cycle on the entire labeled set. <b>4.</b> Updating the latest model using the entire set. <b>5.</b> Fine-tuning the latest model on the latest acquired set. <b>6.</b> Fine-tuning the latest model on the latest acquired set, and performing the final training cycle on the entire labeled set. . . . .	41
4.2	Train and test splits corresponding to the operation commanded by the user. The <i>auto-label</i> , <i>evaluation</i> , and <i>select only</i> commands do not require a training set, and only operate on the test set. While the <i>train only</i> command does not require a test set. . . . .	49
5.1	Mapping the formal strategy names to intuitive experimental names. . . . .	57
5.2	The S-granularity value and mAP score of all training strategies. . . . .	59
5.3	Performance of active learning strategies and the oracle (benchmark) model, detailed across different difficulty levels and metrics. . . . .	60
5.4	The variation in mAP scores across various AL strategies and the oracle model. . . . .	62
5.5	Comparison of active learning strategies across different difficulty levels for the classes Car, Pedestrian, and Bicycle. . . . .	62



# Chapter 1

## Introduction

For several years, the momentum towards automated driving has significantly increased, with numerous automobile manufacturers dedicating substantial resources to advance the field. Despite the progress, contemporary vehicles are not fully autonomous. Instead, it offers a spectrum of advanced driver assistance systems (ADAS) to augment safety and alleviate some driver responsibilities.

SAE International categorizes [33] the progression of vehicle automation into six distinct levels as of 2021:

- Level 0: No Driving Automation - The driver oversees all driving tasks with minimal automated assistance.
- Level 1: Driver Assistance - The vehicle incorporates isolated automated features, such as assisted steering or acceleration, but primarily relies on the driver's control.
- Level 2: Partial Driving Automation - The vehicle can handle a combination of functions like steering and acceleration simultaneously, yet requires the driver to remain actively engaged and monitor the surroundings continuously.
- Level 3: Conditional Driving Automation - The vehicle can conduct most driving tasks, although the driver might need to intervene upon the system's request.
- Level 4: High Driving Automation - The vehicle can autonomously perform all driving tasks within specified conditions.
- Level 5: Full Driving Automation - The vehicle can handle all driving operations under any condition, requiring no human intervention.

As the National Highway Traffic Safety Administration (NHTSA) reports that 94% of significant accidents are attributable to human errors [48], escalating to more advanced levels of autonomous driving can potentially diminish these incidents significantly. In this context, developing precise and reliable 3D perception models becomes a critical endeavor, forming the backbone of autonomous systems capable of navigating complex environments safely and efficiently. This technical evolution not only reduces accident rates but also revolutionizes the approach to vehicle safety and navigation.

The Providentia++ project is the extension of the Providentia project, an ITS research project that aims to represent the current traffic on the German highway A9 near Munich as a digital twin using sensors placed on gantries. The extension, the Providentia++ project, also expands to the urban area, including high-traffic intersections, to cover a total road distance of 3 km. In order to cover this route optimally, sensors are placed into the environment at seven



**Figure 1.1:** Track sections covered by the Providentia (green) and the extended Providentia++ (blue) project with marked sensor positions

different locations. The locations and the monitored area are depicted in Figure 1.1. Building on this foundation, perception plays a pivotal role in the progression of autonomous driving technologies. Perception systems in autonomous vehicles are responsible for accurately interpreting and understanding the vehicle’s surroundings, a task crucial for safe navigation and decision-making. These systems rely heavily on advanced sensors and algorithms to detect and classify objects, anticipate potential hazards, and make informed decisions in real-time. Among the various components of perception systems, 3D object detection emerges as a critical task. It involves identifying and locating objects in the environment in three dimensions, essential for creating a comprehensive understanding of the vehicle’s surroundings. Effective 3D object detection enables autonomous vehicles to accurately measure distances, understand the spatial relationship of objects, and predict their movements. This ability is fundamental in ensuring the safety and reliability of autonomous driving systems, making it a key area of focus in the development and advancement of autonomous vehicle technology. This thesis is dedicated to the 3D object detection task, explicitly focusing on LiDAR-based 3D object detection in the context of autonomous driving.

LiDAR is an essential and widely adopted sensor for autonomous vehicles to provide precise distance measurements of the 3D surroundings and extract high-level information about the underlying scenery. LiDAR-based 3D object detection is a crucial yet challenging task for all autonomous systems. The development of accurate 3D detection models relies on the availability of large-scale labeled point clouds, where 7 degrees of freedom 3D bounding boxes - consisting of a position, size, and orientation information - for each object are annotated. In benchmark datasets like Waymo [45], there are 12.6 million LiDAR boxes, for which labeling a precise 3D box takes more than 100 seconds for an annotator [43]. This pre-requisite for the performance gain makes it challenging to apply models to the wild, especially when the annotation budget is limited. To overcome this limitation, active learning provides a promising framework to reduce labeling costs by querying labels for only a small subset of the unlabeled data pool. A criterion-based query selection process iteratively selects the most informative samples for the subsequent model training until the labeling budget is exhausted. We adopt [29] as our baseline active learning framework, deploying a hierarchical, cost-effective point cloud acquisition criteria at the 3D bounding box level, termed CRB. The iterative process of active learning is time- and resource-consuming, as the 3D detector model is trained from scratch every time new data is selected and added to the training set. This approach also dis-

cards the knowledge the model has gained in previous active training iterations. To address this challenge, we explore various continuous training methods and integrate the most efficient method regarding computational demand and detection performance. This approach aligns with our hypothesis that active learning can achieve high performance with reduced data utilization without placing excessive demands on computational resources.

In a parallel line of work, the swift advancement of 3D sensors and LiDAR-based object detection technologies is driving an increased need for open-source and intuitive 3D annotation platforms. These platforms are crucial for enabling both researchers and individuals without specific domain expertise to re-annotate and create their own datasets. The task of manually annotating 3D datasets presents unique challenges. Annotating 3D bounding boxes directly on LiDAR point clouds is complex and often requires specialized knowledge due to the data sparsity and irregularity. Conversely, using 2D RGB images for 3D bounding box annotation is inefficient, primarily due to issues with occlusions and depth ambiguity. Despite the necessity of integrating 2D images with 3D point clouds for high-quality and efficient annotation, the process remains intricate and demands meticulous annotator training, stringent quality assurance, and ongoing enhancements in tool usability. In response to these challenges, we build upon the work of [31], an advanced web-based 3D annotation platform, itself developed from the 3D-BAT [65] tool. We have further integrated this platform with an active learning pipeline, Active3D, to incorporate AI-assisted features that enhance the efficiency of the annotation process. Through active learning, the tool not only reduces manual efforts but also augments annotation capabilities, ensuring that the data used for training machine learning models is of the highest informativeness and contributes significantly to model performance. The combination of human expertise and AI-assisted features forms the foundation of the tool's methodological advancements. In addition, we have resolved various usability issues and introduced new features to improve the overall workflow.

The structure of the thesis is outlined as follows:

- Chapter 2: The background chapter concisely establishes the foundational knowledge required for readers to understand and engage with the work presented in this thesis.
- Chapter 3: The related works chapter explores literature and previous studies relevant to our research area. It emphasizes the novelty and contributions of significant works in the field, covering topics such as LiDAR-based 3D object detection, active learning applications in object detection, 3D annotation tools, and various annotation formats and standards.
- Chapter 4: In the methodology chapter, we introduce our advancements in the active learning pipeline (Active3D) and the 3D annotation tool (proAnno). These include the development of temporal CRB, active class weighting in CRB, continuous training techniques, the integration of proAnno with Active3D, and improvements in proAnno's usability features. The last two contributions introduce the latest version of our 3D annotation tool, proAnno++.
- Chapter 5: The evaluation chapter is dedicated to evaluating and analyzing the outcomes of our research. This includes comparing different query strategies in Active3D, including our newly developed temporal CRB, besides investigating the effects of various continuous training methods on the active learning cycle and conducting an extensive ablation study on how the temporal consistency constraint influences the CRB acquisition function in Active3D. We also provide qualitative results to visually demonstrate our findings and observations.

- Chapter 6: In the conclusion, we summarize and reflect on our findings and contributions, highlighting both the benefits and drawbacks of our work, along with emphasizing our key results.
- Chapter 7: Finally, we offer various insights and suggestions for future research, drawing from the outcomes and experiences of our work.

In summary, the key contributions of this thesis are as follows:

- We utilize an active learning framework for 3D object detection, named *Active3D*, for our datasets, achieving the performance of a fully supervised 3D object detector using only 50% of the available point cloud data.
- We integrate multiple continuous training methods, alleviating the computational demand of active learning, achieving the fully supervised performance with 72% of the computational requirements of the fully supervised 3D detector.
- We introduce temporal consistency to *Active3D*, enabling *Active3D* to select the most informative and sequential point cloud frames.
- We integrate *Active3D* into *proAnno*, enabling annotators to focus on the most informative point cloud frames, hence reducing annotation time and costs.
- We enhanced the usability of *proAnno*, adding and improving multiple features including zooming into images, showing Front, Side, and BEV views for a selected bounding box, copying labels to consecutive frames, switching between dataset and sequences within a dataset. Collectively contributing to a user-friendly and intuitive 3D annotation process.

# Chapter 2

## Background

In this chapter, we lay the groundwork for our discussions in the next chapters, offering a focused and technical overview of essential topics. These include LiDAR point clouds and LiDAR-based 3D object detection, 3D datasets, and basics of 3D labeling techniques.

### 2.1 LiDAR-based Object Detection

#### 2.1.1 Background

LiDAR sensors are integral to autonomous driving, enabling precise, reliable, and fast decision-making. These sensors are critical for object detection within the perception system, helping to interpret the driving environment. While 2D object detection has advanced significantly in the era of deep learning, its inability to capture depth information limits a comprehensive understanding of the environment and the accurate localization of objects. In contrast, 3D sensors like LiDAR provide depth and spatial information, enhancing 3D perception systems.

Light Detection And Ranging (LiDAR), shown in Figure 2.1, functions on the time-of-flight measurement principle. The device consists of a source that emits laser pulses and a detector for capturing the reflected pulses from objects in the environment. The source sends a laser beam toward an object, and the detector measures the times it takes for the reflected pulse to return. This time interval is then used to calculate the distance to the object. The output of the entire process is a sequence of range images. These images provide a 3D representation of the sensor's surroundings, capturing the contours and distances of various objects relative to the LiDAR's position.

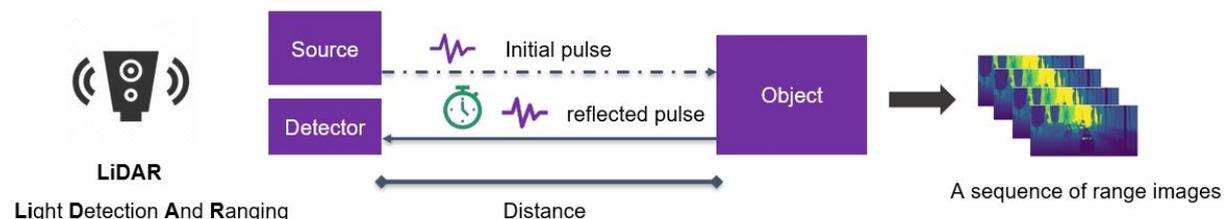


Figure 2.1: Illustration of the LiDAR sensor concept.

A LiDAR sensor with  $m$  beams performing  $n$  measurements in a single cycle, generates an  $m \times n$  matrix called a *range image*. Each column of the range image corresponds to a fixed

azimuth, representing the horizontal angle relative to the sensor's origin, while each row corresponds to a constant inclination, denoting the vertical angle. The pixel values in the range image encode the distance (range) to the point on the object where the laser pulse was reflected and the intensity of the reflected laser pulse, and they may include additional information. A single pixel in the range image contains three geometric information: the range  $r$ , azimuth  $\theta$ , and inclination  $\phi$ . These values construct a spherical coordinate system, as shown in Figure.

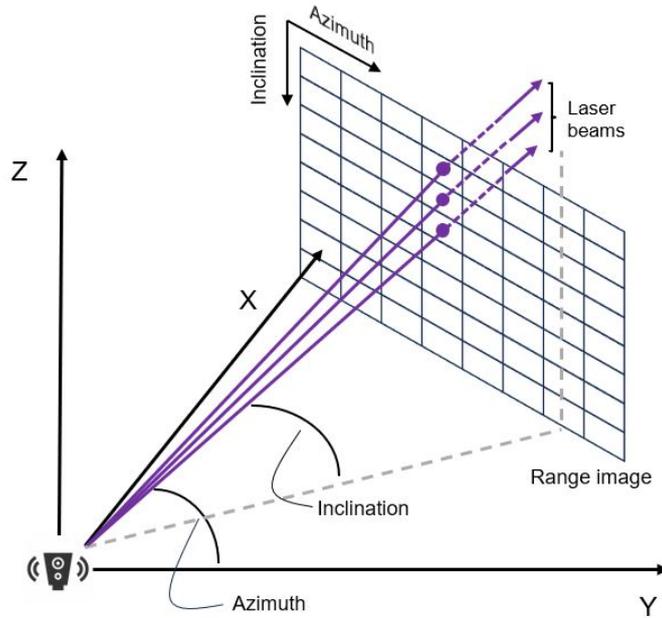


Figure 2.2: Illustration of the range image.

The standard point cloud represented in Cartesian coordinates is derived from the spherical coordinate system:

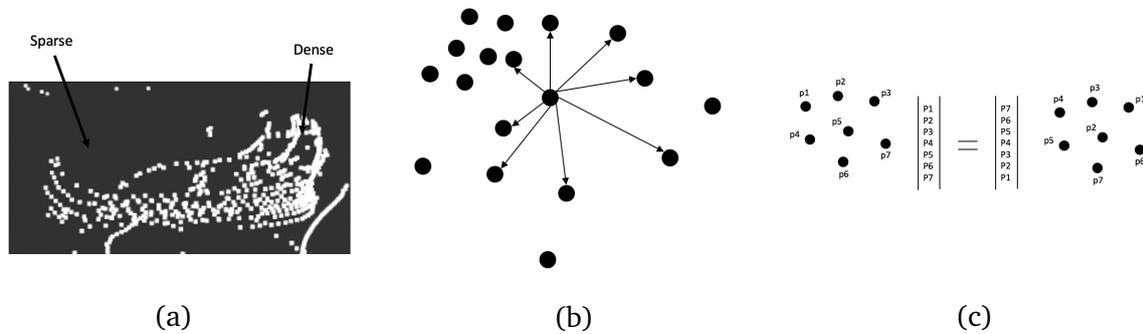
$$x = r \cos(\theta) \cos(\phi), \quad y = r \cos(\theta) \sin(\phi), \quad z = r \sin(\theta); \quad (2.1)$$

Point clouds are more commonly used because they provide a direct representation of the spatial structure, unlike range images, which require interpretation of pixel values to get depth and reflectance. Point clouds offer a fully 3D reconstruction of the scene, with rich geometric, shape, and scale information. This enables the extraction of meaningful features that boost the detection performance. However, point clouds have significant challenges due to their characteristics and how they are processed. For object detection using deep learning, data is usually required to be in a grid format, like images, which is not fulfilled by point clouds. They are typically irregular, unstructured, and not ordered.

- **Irregular** means that the points of a point cloud are not evenly sampled across the scene. This results in certain areas having a higher concentration of points than others. Particularly, distant objects tend to be sparsely represented, due to the current sensors' limited recording range.
- **Unstructured** indicates that the points in a point cloud do not align to a regular grid. As a result, the distances between adjacent points can differ. This contrasts with image pixels, which always have a fixed position relative to their neighboring pixels.

- **Unordered** means that a point cloud is simply a collection of points whose overall structure remains unchanged regardless of how its points are arranged. Particularly, the sequence in which the points are stored does not change the representation of the scene. This characteristic is known as ‘permutation invariance.’

Figure 2.3 illustrates the point cloud data characteristics.



**Figure 2.3:** Properties of point cloud data: (a) Irregular distribution of dense and sparse areas. (b) Unstructured set of points with no fixed grid structure. (c) Unordered set of points that are permutation invariant.

To ensure a correct processing of the input data by 3D object detectors, the input point cloud data must be available in a suitable representation. Here, we focus on 3D representations, namely point-wise representations and grid-based representations.

- **Point-wise Representation:** Point-wise representations scatter points sparsely over a spatial structure, capturing the visible surface with precise location details. PointNet [32] processes this by grouping nearby points and extracting compressed features from each set’s low-dimensional point features, enabling a raw point-based approach for 3D object detection models. Point-based representations generally retain more information than voxel- or projection-based methods. However, they become inefficient with large numbers of points. Reducing point clouds, as in cascading fusion methods, leads to information loss. Currently, no representation method perfectly balances efficiency and performance.
- **Grid-wise Representation:** CNNs face challenges with point cloud data due to the lack of a structured grid for convolution operations. To utilize deep learning on point clouds, they must be transformed into a grid-like structure. This can be done by quantizing them into 3D volumetric grids like in [44] and [62], or discretizing into multi-view projections as in [25] and [7]. Volumetric representation involves dividing the point cloud into uniform grid cells or voxels. The point cloud is converted into a 3D voxel grid with fixed dimensions (x, y, z). Voxels may contain raw points or features like point density or intensity. Networks using voxel-based representations are more computationally efficient and use less memory. They extract features from clusters of points (voxels) instead of individual points.

### 2.1.2 Problem Definition

3D object detection in driving environments aims to identify the characteristics of 3D objects using sensor inputs. The fundamental formula for this process is represented as:

$$B = \mathbf{f}_{\text{det}}(\mathcal{I}_{\text{sensor}}) \quad (2.2)$$

Here,  $B$  is the set of detected 3D objects in a scene,  $f_{\text{det}}$  is the detection model, and  $\mathcal{I}_{\text{sensor}}$  denotes sensory inputs. In mathematical terms, we consider a LiDAR point cloud  $P = \{x, y, z, e\}$ , with each point having 3D coordinates  $(x, y, z)$  and reflectance  $e$ . The detection model aims to locate these objects as a set of 3D bounding boxes  $B = \{b_k\}_{k=1}^{N_B}$ , where  $N_B$  is the number of detected boxes, and assign class labels  $Y = \{y_k\}_{k=1}^{N_B}$  from  $C$  possible classes. Each bounding box  $b$  indicates the object's central position  $(p_x, p_y, p_z)$ , dimensions  $(l, w, h)$ , and orientation angle  $\theta$ . The detectors process point clouds  $P$  to extract features  $x \in \mathbb{R}^{W \times L \times F}$ , using methods like point-wise analysis or voxelization. The features  $x$  are then fed into a classifier  $f(\cdot; w_f)$  and regression heads  $g(\cdot; w_g)$ . The output includes the identified bounding boxes  $\hat{B} = \{\hat{b}_k\}$  and the corresponding labels  $\hat{Y} = \{\hat{y}_k\}$ , determined from predefined anchored areas.

## 2.2 Datasets

Developing comprehensive driving datasets has significantly contributed to 3D object detection research, providing multi-modal sensory data and 3D annotations. The growth of these large-scale datasets is a hallmark of the data-driven era, continually enriching the research community. Four key aspects are critical when assessing a dataset: 1) The scale of the data, including the number of point clouds, images, and annotations. 2) The data diversity encompassing diverse daytime and weather conditions and driving scenarios. 3) The breadth of annotation categories covering various road elements. 4) The data modality, including LiDAR point clouds, RGB images, Radar images, thermal images, and more. Furthermore, integrating vehicle and infrastructure sensors offers substantial benefits, such as a global perspective beyond the immediate surroundings of the ego vehicle and covering blind spots. This integration has been enhanced by advancements in communication technologies like V2X (Vehicle to Everything), enabling the use of data from infrastructure sensors for a comprehensive perception of the driving situation. This section presents multiple relevant datasets to our research, namely TUMTraffic Highway [8], TUMTraffic Intersection [64], and DAIR [60] datasets.

### 2.2.1 TUMTraffic Highway Dataset



**Figure 2.4:** Overview of the test bed Providentia++

The TUMTraffic Highway dataset, recorded from the sensor equipment of the Providentia++ project near Munich, Germany, features data recordings using sensors installed on stationary

structures like gantry bridges and masts. These sensors offer an extensive view of the roadway. This dataset provides annotated images and LiDAR point clouds from various segments and perspectives of the road. Encompassing diverse road environments such as the autobahn A9, rural paths, and urban settings, it includes many traffic scenarios, ranging from freeways and highways to roundabouts and intersections in urban areas, complete with pedestrian and bicycle traffic.

The TUMTraffic Highway dataset includes 14,459 labeled instances across a range of vehicle and road user classes, as detailed in Table 2.1. The dataset predominantly features daytime scenarios with good visibility, covering a variety of classes, ranging from typical vehicles like cars, trucks, and trailers to other categories such as vans, motorcycles, buses, pedestrians, and bicycles, among others. It’s important to note the under-representation of certain classes like motorcycles and bicycles. The dataset also includes 1,567 3D LiDAR points, providing the highway environment analysis depth. While the current version focuses on favorable lighting conditions, future iterations will expand to include diverse visibility situations.

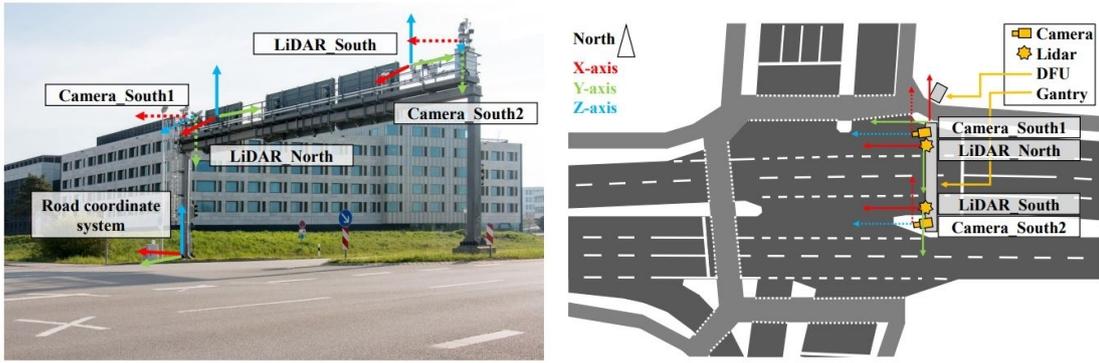
Class	# Labels	$\phi$ Length	$\phi$ Width	$\phi$ Height	$\phi$ Points
Car	9,310	4.58	1.94	1.48	90
Truck	1,633	5.80	2.51	3.55	143
Trailer	1,828	12.49	2.82	3.79	1097
Van	1,366	5.68	2.09	2.22	195
Motorcycle	33	2.32	0.75	1.64	-
Bus	59	13.58	2.61	3.35	-
Pedestrian	220	0.55	0.49	1.68	42
Bicycle	1	1.42	0.48	1.24	-
Other	9	7.28	2.36	2.87	-
Total	14,459	-	-	-	1,567

**Table 2.1:** TUMTraffic Highway statistics including the total number of labels, average dimensions in meters, and the average number of 3D LiDAR points among all classes.

## 2.2.2 TUMTraffic Intersection Dataset

Building upon the TUMTraffic Highway dataset, the TUMTraffic Intersection dataset encompasses more complex traffic scenarios at a busy intersection, featuring a broader range of road users. This dataset was captured using the hardware infrastructure of the Providentia++ project, which utilizes sensors positioned on a gantry at an intersection in Munich, Germany. The data collection setup includes two cameras and two LiDARs mounted alongside each other, as depicted in Figure 2.5. Positioned at a height of 7 meters, these sensors effectively monitor the dynamic traffic at the intersection.

Overall, the TUMTraffic Intersection dataset comprises 4,800 labeled LiDAR point cloud frames drawn from four distinct sequences. Within these frames, 57,406 3D objects (comprising 506 unique objects) have been marked with 273,861 attributes. Following the label fusion from both LiDAR systems, we have obtained a dataset of 38,045 registered 3D objects



**Figure 2.5:** Two cameras and two LiDARs are used to create the TUMTraffic Intersection dataset. The sensors are mounted on a sign gantry and thus record the central intersection area. Additionally, the coordinate systems for each sensor and the road coordinate system, established at the northern end of the bridge, are illustrated in the figure.

(encompassing 482 unique objects), annotated with 171,045 attributes. The statistics presented below are based on this fused dataset, which includes the training, validation, and test sets. Table 2.2 provides a statistical overview of the 3D box labels.

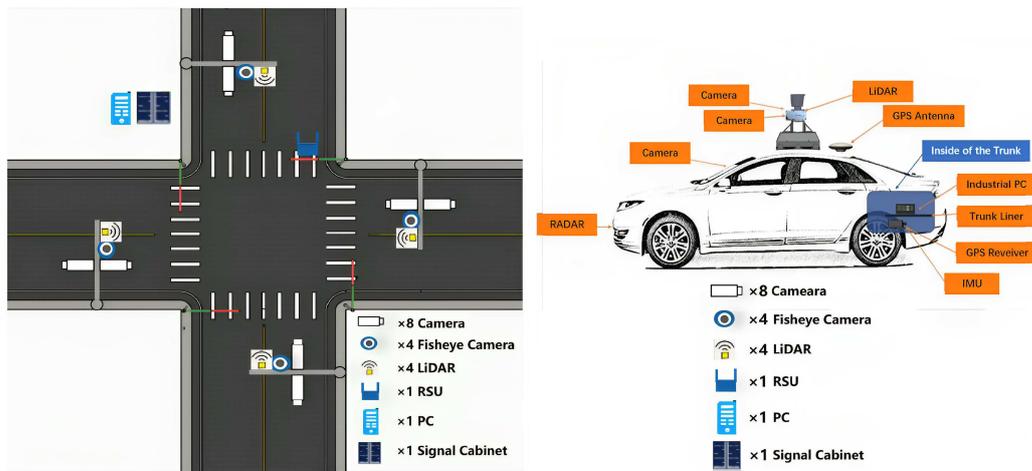
Class	# Labels	$\phi$ Length	$\phi$ Width	$\phi$ Height	$\phi$ Points
Car	22,773	4.27	1.91	1.59	34.03
Truck	2,704	3.11	2.90	3.43	116.87
Trailer	3,177	10.19	3.12	3.65	328.36
Van	4,353	6.35	2.52	2.47	86.11
Motorcycle	734	1.90	0.83	1.60	21.23
Bus	908	12.65	2.95	3.27	222.36
Pedestrian	2,507	0.80	0.73	1.72	14.98
Bicycle	663	1.57	0.74	1.72	20.95
Emergency Vehicle	142	6.72	2.35	2.35	58.95
Other	84	5.28	1.92	1.90	128.17
Total	38,045	-	-	-	103.20

**Table 2.2:** TUMTraffic Intersection statistics including the total number of labels, average dimensions in meters, and the average number of 3D LiDAR points among all classes.

### 2.2.3 DAIR Dataset

The DAIR-V2X Dataset stands as the first comprehensive, multi-modality and multi-view dataset specifically designed for Vehicle-Infrastructure Cooperative Autonomous Driving (VI-CAD). This dataset features 71,254 LiDAR and camera frames, all captured in intersection settings equipped with vehicle-mounted and infrastructure-based sensors. The frames are split such that 40% are sourced from infrastructure sensors, while the remaining 60% are from vehicle sensors. Expert annotators have precisely labeled each frame in the dataset. It encompasses a broad range of driving environments, including 10 km of urban roads, 10

km of highways, 28 intersections, and a total driving area of 38 km<sup>2</sup>, offering a variety of weather and lighting conditions. Data collection equipment consists of two types: infrastruc-



**Figure 2.6:** The DAIR-V2X hardware setup comprises infrastructure sensors and vehicle sensors.

ture sensors and vehicle sensors. a) For infrastructure sensors, each of the 28 intersections within the Beijing High-level Autonomous Driving Demonstration Area is equipped with four sets of LiDAR sensors and high-resolution cameras. However, the DAIR-V2X dataset utilizes only one set from each location. b) Regarding vehicle sensors, autonomous vehicles are fitted with a single LiDAR sensor and a high-quality forward-facing camera mounted on the top of the vehicle.

## 2.3 3D Labeling

In this section, we explain the concepts of object labeling methods according to the ASAM OpenLABEL standard [2].

### 2.3.1 Coordinate Systems

A data stream in the OpenLABEL format can include sensor data from various sensors and diverse types of label information linked to this sensor data. It's essential to establish clear connections: how data from different sensors is interconnected, the relationship between label data and sensor data, and the relationship between sensor data and the real-world context. To accomplish this, multiple coordinate systems are utilized, with transformations between them to maintain these relationships. For example, a data stream may contain images from two cameras and point clouds from two different LiDAR sensors that are mechanically mounted on a moving vehicle or in a traffic infrastructure.

The key concepts here are:

- **Coordinate system:** A way to specify the location of points in some space. For example, the 2D position of a pixel within an image or the 3D position of a LiDAR point relative to a pre-defined global origin point that can be the vehicle's center-of-gravity (CoG) or rear axle or simply the LiDAR sensor's position. Coordinate systems are generally 3D right-handed Cartesian systems, as shown in Figure 2.7.

- **Transformation:** A transformation allows the conversion of coordinates in one coordinate system to another coordinate system such that they represent the same point in space. Two key types of transforms are relevant: the *camera-transform*, a projective transformation detailing the projection of a real-world point onto a camera sensor's pixel. This typically involves various components, including intrinsics (like focal length and optical center), distortion coefficients (correcting lens distortion), and extrinsic (describing the camera's position and orientation in space). The other is the *cartesian-transform*, which converts from a 3D Cartesian coordinate system to an ellipsoidal GNSS-style coordinate system.

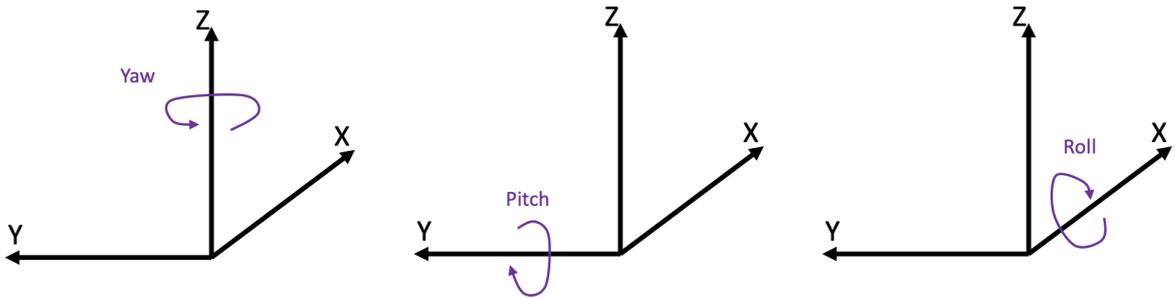


Figure 2.7: Right-hand Coordinate System.

### 2.3.2 Geometries for Labeling

Labeling objects within data streams requires varying geometries determined by the type of sensor stream being used. This necessitates either 2D or 3D geometrical approaches. The OpenLABEL Standard offers a collection of primitives suitable for labeling objects and regions in these sensor streams. Our focus here is specifically on 2D and 3D bounding boxes.

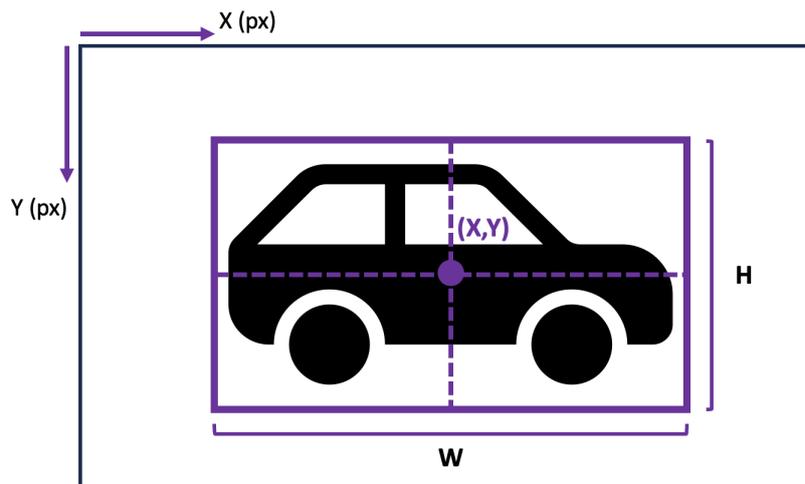


Figure 2.8: 2D Bounding box definition.

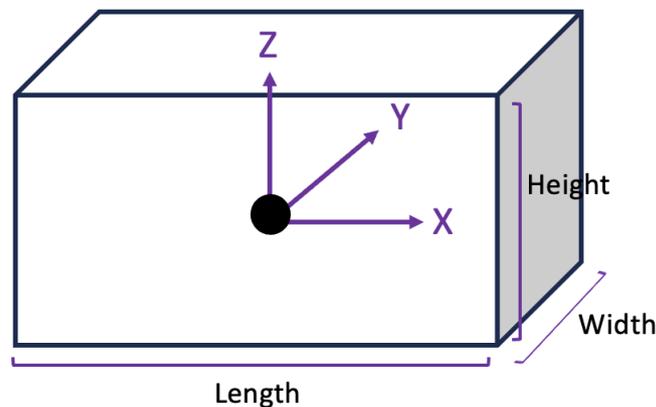
A 2D bounding box, as shown in Figure 2.9, is defined as a rectangle by an array of four or five floating point numbers depending on whether rotation is included or not:

- **X**: In pixel units — Specify the x-coordinate of the center of the rectangle.
- **Y**: In pixel units — Specify the y-coordinate of the center of the rectangle.
- **W**: In pixel units — Specify the width of the rectangle in the x/y-coordinate system.
- **H**: In pixel units — Specify the height of the rectangle in the x/y-coordinate system.
- (optional) **Alpha**: In radians — Specifies the bounding box's rotation as a right-handed rotation, implying a positive rotation from the x-axis to the y-axis. The rotation's point of origin is situated at the center of the bounding box.

A 3D bounding box represents a cuboid shape within 3D Euclidean space, characterized by its position, rotation, and dimensions. 3-dimensional vectors specify the position and dimensions. There are two possible representations for rotation: a 4-vector quaternion or a 3-vector Euler notation, which should be applied in the ZYX sequence, corresponding to yaw-pitch-roll order.

- **Position**: In meters — Specifies the x, y, and z coordinates of the 3D position of the center of the cuboid.
- **Rotation**: Specify the quaternion in non-unit form (x, y, z, and w) as in the SciPy convention.
- **Dimension**: In meters — Specifies the x, y, and z dimension of the cuboid or the x-coordinate.

Another option is to use the Euler angles for rotation. In this case, we use rz, ry, and rx to specify the Euler angles yaw, pitch, and roll, respectively.



**Figure 2.9:** 3D Bounding box definition. The highlighted face indicates the forward orientation of the box.

### 2.3.3 Spatial Rotation

Multiple methods exist for representing spatial rotations, each with its own set of advantages and disadvantages. These include rotation matrices, Euler angles, and quaternions. A

rotation matrix is a 3x3 matrix that consists of orthogonal unit vectors. The multiplication of rotation matrices equals the concatenation of rotations and thus yields rotation matrices. Euler angles describe rotation in three-dimensional space through three angles, each representing a rotation around one of the principal axes (X, Y, Z). This method involves a sequence of three rotations about the axes of a coordinate system, typically referred to as roll, pitch, and yaw. Euler angles are intuitively simple and straightforward to use. However, they pose computational challenges, requiring conversion to rotation matrices or quaternions for practical application. They are prone to gimbal lock, which reduces the rotation's degrees of freedom from three to two. On the upside, unlike quaternions and rotation matrices, Euler angles do not require normalization.

Quaternions use a similar mathematical concept as complex numbers, where complex numbers are defined by the inclusion of the imaginary unit  $i$ , where  $i^2 = -1$ . A complex number  $a + bi$  can be represented as a point in a two-dimensional coordinate system.

A quaternion  $q = w + xi + yj + zk$  is normalized if its norm  $n(q)$  equals 1, where

$$n(q) = \sqrt{w^2 + x^2 + y^2 + z^2}. \quad (2.3)$$

The conjugate of  $q$  lets the rotation axis point in the opposite direction.

$$\bar{q} = w - xi - yj - zk. \quad (2.4)$$

Quaternions form an algebraic field, thus, addition, subtraction, multiplication, and division are defined. The neutral element regarding multiplication is  $1 = 1 + 0i + 0j + 0k$ . The inverse regarding multiplication is:

$$q^{-1} = \frac{\bar{q}}{n(q)^2}. \quad (2.5)$$

A vector  $p = (p_1, p_2, p_3)$  is rotated by  $q$  via

$$q' = q * (0 + p_1i + p_2j + p_3k) * \bar{q}. \quad (2.6)$$

The unit quaternion, which performs the rotation around a unit vector considered as axis  $u = (u_1, u_2, u_3)$  with an angle  $a$  is constructed via

$$q = \cos\left(\frac{a}{2}\right) + \sin\left(\frac{a}{2}\right)u_1i + \sin\left(\frac{a}{2}\right)u_2j + \sin\left(\frac{a}{2}\right)u_3k. \quad (2.7)$$

For the transformation between rotation representations:

- **Quaternion to rotation matrix:** Given a normalized quaternion  $q = w + xi + yj + zk$ , the corresponding rotation matrix  $R(q)$  is given by:

$$R(q) = \begin{bmatrix} 1 - 2y^2 - 2z^2 & 2xy - 2zw & 2xz + 2yw \\ 2xy + 2zw & 1 - 2x^2 - 2z^2 & 2yz - 2xw \\ 2xz - 2yw & 2yz + 2xw & 1 - 2x^2 - 2y^2 \end{bmatrix}. \quad (2.8)$$

- **Rotation matrix to quaternion:** A rotation matrix  $m$  can be converted to a quaternion  $Q(m)$ :

$$Q(m) = \frac{1}{\sqrt{1 + \text{tr}(m)}} \left( \sqrt{1 + \text{tr}(m)}, (m_{21} - m_{12}), (m_{02} - m_{20}), (m_{10} - m_{01}) \right), \quad (2.9)$$

where  $\text{tr}(m) = m_{00} + m_{11} + m_{22}$  is the trace of the matrix.

- **Quaternion to Euler angles:** The Euler angles  $\gamma$  (roll),  $\beta$  (pitch), and  $\alpha$  (yaw) can be derived from a quaternion  $q$  as follows:

$$\alpha(q) = \text{atan2}(2(q_y q_w - q_x q_z), 1 - 2(q_y^2 + q_z^2)), \quad (2.10)$$

$$\beta(q) = \text{asin}(2(q_x q_y + q_z q_w)), \quad (2.11)$$

$$\gamma(q) = \text{atan2}(2(q_x q_w - q_y q_z), 1 - 2(q_x^2 + q_z^2)). \quad (2.12)$$

However, at the poles, there are singularities. If  $q_x q_y + q_z q_w = 0.5$ , then:

$$\alpha(q) = 2 \cdot \text{atan2}(q_x, q_w), \quad (2.13)$$

$$\beta(q) = \frac{\pi}{2}, \quad (2.14)$$

$$\gamma(q) = 0. \quad (2.15)$$

Conversely, if  $q_x q_y + q_z q_w = -0.5$ , then:

$$\alpha(q) = -2 \cdot \text{atan2}(q_x, q_w), \quad (2.16)$$

$$\beta(q) = -\frac{\pi}{2}, \quad (2.17)$$

$$\gamma(q) = 0. \quad (2.18)$$

- **Euler angles to quaternion:** The order of the Euler angles application affects the resulting rotation. Assuming the angles are applied in the order of yaw ( $\gamma$ ), pitch ( $\beta$ ), and roll ( $\alpha$ ), the quaternion  $Q(\alpha, \beta, \gamma)$  is:

$$\begin{aligned} Q(\alpha, \beta, \gamma) = & \cos\left(\frac{\gamma}{2}\right) \cos\left(\frac{\beta}{2}\right) \cos\left(\frac{\alpha}{2}\right) - \sin\left(\frac{\gamma}{2}\right) \sin\left(\frac{\beta}{2}\right) \sin\left(\frac{\alpha}{2}\right) + \\ & \left( \sin\left(\frac{\gamma}{2}\right) \sin\left(\frac{\beta}{2}\right) \cos\left(\frac{\alpha}{2}\right) + \cos\left(\frac{\gamma}{2}\right) \cos\left(\frac{\beta}{2}\right) \sin\left(\frac{\alpha}{2}\right) \right) i + \\ & \left( \sin\left(\frac{\gamma}{2}\right) \cos\left(\frac{\beta}{2}\right) \cos\left(\frac{\alpha}{2}\right) + \cos\left(\frac{\gamma}{2}\right) \sin\left(\frac{\beta}{2}\right) \sin\left(\frac{\alpha}{2}\right) \right) j + \\ & \left( \cos\left(\frac{\gamma}{2}\right) \sin\left(\frac{\beta}{2}\right) \cos\left(\frac{\alpha}{2}\right) + \sin\left(\frac{\gamma}{2}\right) \cos\left(\frac{\beta}{2}\right) \sin\left(\frac{\alpha}{2}\right) \right) k. \quad (2.19) \end{aligned}$$



# Chapter 3

## Related Works

### 3.1 3D Object Detection

3D perception is a prerequisite in autonomous driving, automated railway operations, and robotics. A comprehensive understanding of the ego vehicle's environment will facilitate downstream components to react accordingly. To fully grasp the dynamics of driving environments, various vision tasks like object detection and tracking, lane detection, semantic and instance segmentation, *etc.* are integrated into perception systems. Central to these tasks is 3D object detection, a fundamental component in perception setups. The goal of 3D object detection is to predict the location, size, orientation, and class of objects of interest *e.g.* pedestrians, cars, cyclists, buses, trucks, *etc.* in the 3D space. In contrast to 2D object detection, which confines object detection to image planes and overlooks the exact distances between objects and the ego-vehicle, 3D object detection focuses on the recognition and localization of objects in the real-world 3D coordinate system. The geometric information predicted by 3D detectors is directly utilized to measure the distances between the ego-vehicle and other objects, further plan driving routes and avoid collisions.

3D object detection methods can be categorized based on the sensory input, whether it's RGB camera, LiDAR, radar, or multi-modal sensory data. This section presents a comprehensive review of LiDAR-based 3D object detection, which will be known as 3D object detection from now on.

#### 3.1.1 LiDAR-based 3D Object Detection

RGB cameras and LiDAR (Light Detection and Ranging) sensors are the two most widely adopted sensors for 3D object detection. Cameras produce images  $\mathcal{I}_{cam} \in \mathcal{R}^{W \times H \times 3}$  for 3D object detection, where  $W$ ,  $H$  are the width and height of an image, and each pixel has 3 RGB channels. Although cameras are cost-effective, readily available, and capture rich semantic information, they possess inherent limitations when employed for 3D object detection:

1. Cameras capture only texture information and are not able to directly retrieve 3D structural information from a scene.
2. Cameras are vulnerable to variations in weather and lighting conditions, making the identification of objects during night-time or foggy weather notably challenging compared to clear, sunny conditions.

LiDAR provides an alternative solution. LiDAR sensors can obtain fine-grained 3D structural information of a scene by emitting multiple laser beams and measuring their reflective information. A LiDAR sensor that emits  $m$  beams and conducts measurements for  $n$  times in

one scan cycle can produce a range image  $\mathcal{I}_{range} \in \mathcal{R}^{m \times n \times 3}$ , where each pixel of the range image has three channels: range  $r$ , azimuth  $\alpha$ , and elevation  $\phi$  in the spherical coordinate system. Range images are the raw data format produced by LiDAR sensors and can be further converted to a more generic structure - point clouds - by transforming spherical coordinates into Cartesian coordinates.

A point cloud can be represented as  $\mathcal{I}_{point} \in \mathcal{R}^{N \times 3}$  where  $N$  denotes the number of points in a scene, and each point has XYZ coordinates. Additional features, such as reflective intensity, can be attached to each point. The point cloud offers a fully 3-dimensional reconstruction of the scene, providing rich geometric, shape, and scale information. This enables the extraction of meaningful features that boost the detection performance. Nevertheless, point clouds face severe challenges which are based on their nature and processability. In particular, point clouds exhibit irregular, unstructured, and unordered data characteristics. On the other hand, detection in autonomous driving scenarios generally has a requirement for real-time inference. Therefore, developing a 3D detector model that can effectively handle point cloud data and maintain a high efficiency is an open challenge in the research community, and recent years have seen tremendous advancements in addressing this challenge.

LiDAR-based 3D object detection methods are usually categorized based on the data representations *e.g.*, point-based, voxel-based, point voxel-based, and range-based representations. We narrowed down our focus to point-voxel-based 3D object detection. Point-voxel-based approaches utilize hybrid architectures that integrate the merits of both voxel-based and point-based approaches. Point-voxel-based approaches can be further divided into two categories: single-stage (Sec. 3.1.2) and two-stage detection frameworks (Sec. 3.1.3).

### 3.1.2 Single-stage 3D Object Detection

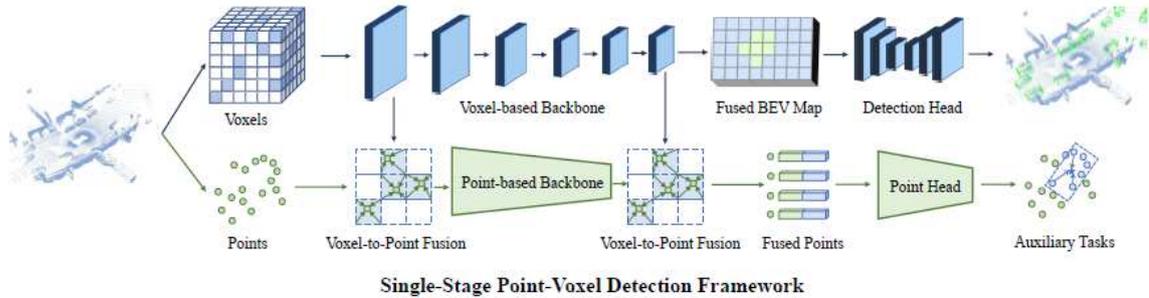
Single-stage point-voxel-based 3D object detectors try to bridge the features of points and voxels with the point-to-voxel, and voxel-to-point transforms in the backbone networks. Voxel-based methods [24, 55, 58, 63] is computationally effective, but the desertion of fine-grained patterns degrades further refinement. While point-based methods [40, 41, 56, 57] have relatively high computational costs but preserve the irregularity and locality of point clouds. Combining points and voxels in the feature extraction stage naturally benefits from both representations. Figure 3.1 provides a general illustration of single-stage detection frameworks.

SPVNAS [46] introduced the novel sparse point-voxel convolution (SPVConv) module to tackle the difficulty of recognizing small object instances (*e.g.*, pedestrians, cyclists) as a result of low-voxelization resolution and aggressive downsampling in voxel-based methods. To capture the fine details, the new 3D module adds a low-cost, high-resolution point-based branch to the vanilla sparse convolution.

In SA-SSD [17], the authors propose to exploit the fine-grained structure information to improve the localization accuracy and single-stage 3D detectors. They design an auxiliary network that guides the backbone network to learn more discriminative features with point-level supervision. The auxiliary network converts the backbone features back to point-wise representation and performs two auxiliary tasks: foreground segmentation and point-wise center estimation. Foreground segmentation increases the sensitivity of the backbone to the boundaries of the objects. At the same time, center estimation enhances the intra-object relationships by learning to estimate object-specific centers from point-specific features.

PVGNet [30] aims to tackle the loss of information during voxelization by integrating point

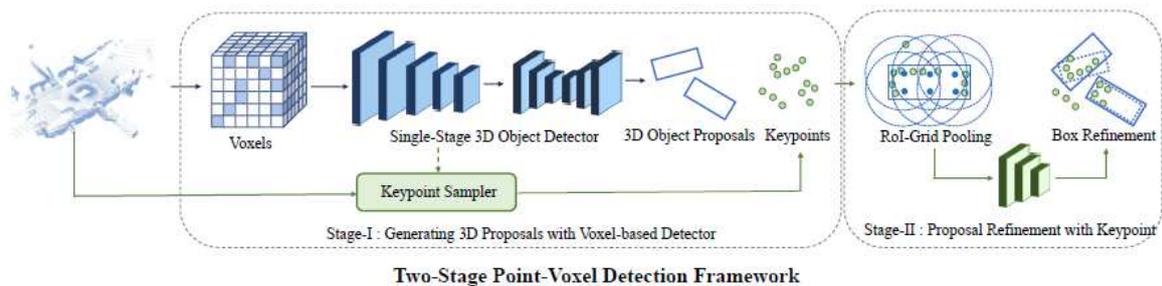
features, voxel features, and BEV grid features in a unified backbone and applying a point-wise supervision signal on the point-level fusion feature. The proposed PVGNet is a bottom-up 3D detector with point-level supervision that segments point clouds and generates bounding boxes simultaneously in a one-stage manner.



**Figure 3.1:** Single-stage point-voxel 3D object detectors bridge the point features, and voxel features through point-to-voxel and voxel-to-point transforms in the backbone networks.

### 3.1.3 Two-stage 3D Object Detection

As its name indicates, two-stage point-voxel-based 3D object detectors perform object detection in two stages. They rely on different data representations for different detection stages. In the first stage, a voxel-based detection framework generates 3D object proposals. In the second stage, key points are first sampled from the input point cloud, and then the 3D proposals are further refined from the key points through novel point operators.



**Figure 3.2:** Two-stage point-voxel 3D object detection employs a voxel-based detection framework to generate a set of 3D object proposals at the first stage. In the second stage, key points are sampled from the input point cloud, and then the 3D proposals are refined from the key points through point operators

LiDAR R-CNN [27] is a novel two-stage 3D object detector based on PointNet, designed to refine existing proposals in 3D space. The authors uniquely define and address the size ambiguity problem inherent in point-based detectors by proposing two effective solutions: boundary offset, which appends boundary distance to the point features, and virtual points, which augment spacing with evenly distributed grid points, providing the RCNN model with a clearer perception of the proposal size.

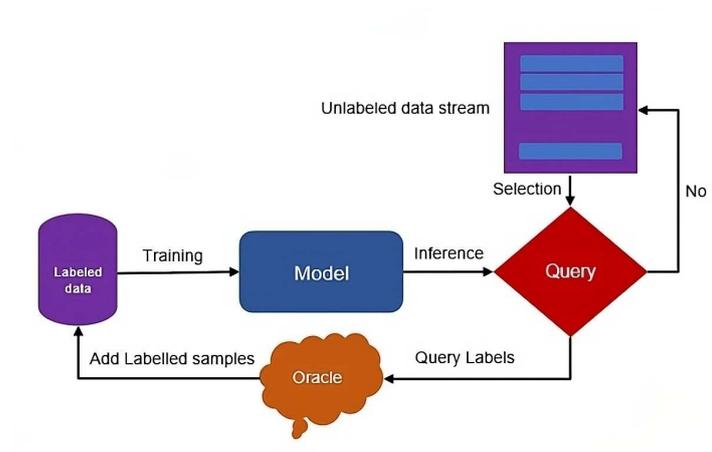
PV-RCNN [38] addresses the challenge of effectively fusing the voxel-based and point-based

features to enhance detection accuracy. The authors propose a voxel-to-keypoint scene encoding to encode scene features into key points by capturing essential multi-scale voxel features. This is followed by a keypoint-to-grid RoI feature abstraction step for box proposals, aggregating features from key points with multi-scale context. This approach effectively leverages the strengths of both feature representations to deliver improved 3D detection performance.

PV-RCNN++ [39] builds upon PV-RCNN and introduces two key improvements. First, a sectorized proposal-centric keypoint sampling strategy for more effective scene encoding was introduced. Second, a VectorPool aggregation module was proposed for efficient local feature encoding in point clouds. PV-RCNN++ optimizes resource consumption and significantly increases inference speed, making it suitable for resource-constrained applications.

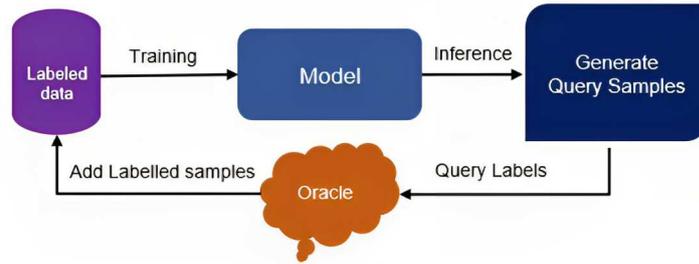
## 3.2 Active Learning

Active Learning (AL) is a methodological approach designed to optimize the efficiency of machine learning systems. It primarily focuses on enhancing performance gains while minimizing the need for extensive manual data labeling. The core principle of AL involves selecting the most informative data samples from an unlabeled pool of data and handing it over to the human annotator for labeling to reduce the labeling costs while still maintaining performance gains. AL approaches can be categorized into three main types based on their function in different scenarios: Stream-based selective sampling, membership query synthesis, and pool-based.



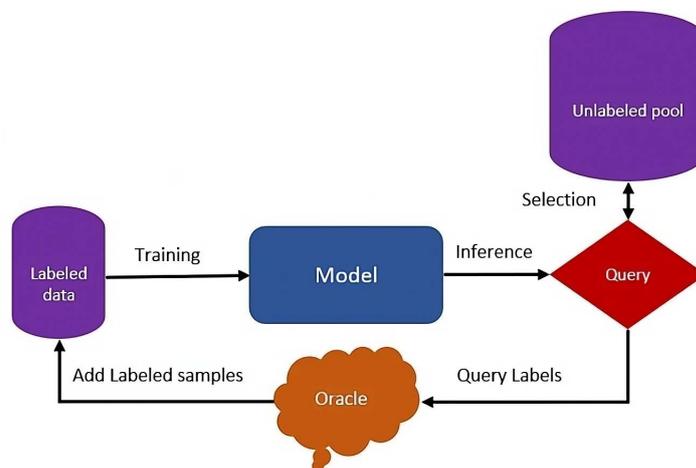
**Figure 3.3:** Stream-based Selective Sampling AL Scenario. Unlabeled data sampled are presented sequentially in a stream.

In the stream-based selective sampling scenario, Figure 3.3, unlabeled data samples are presented sequentially in a stream. The model must determine in real-time whether to request the label for each data sample based on a predefined set of criteria. If a data sample meets these criteria, it's selected for querying from the oracle; if not, it's disregarded, and the algorithm evaluates the next sample in the stream. In the membership query synthesis scenario, Figure 3.4, the model generates new synthetic data samples to inquire about based on its current knowledge. The generated samples are typically near the decision boundary. This approach can quickly produce many queries but may not work well for complex tasks like 3D detection or natural language processing (NLP) because the synthetic samples might not make sense to human experts. [52] proposed to tackle this issue by combining membership



**Figure 3.4:** Membership Query Synthesis AL Scenario. The model generates new synthetic data for inquiry based on its current knowledge.

query synthesis with pool-based AL. It generates queries near the decision boundary but only asks about real, existing samples similar to the synthetic ones. Thus benefiting from efficient query generation while ensuring the queries remain explainable to human experts. In the



**Figure 3.5:** Pool-based AL Scenario. The learner selects specific data samples according to query criteria.

more common pool-based AL scenario, Figure 3.5, the model begins with access to a large pool of unlabeled data samples. From this pool, the learner selects specific samples to inquire about based on a hypothesis model built from initially labeled samples. Each selected sample, once labeled by an oracle, is added to the growing set of labeled data, and the model is re-trained. This approach is ideal for scenarios with robust computational and storage capabilities, allowing for an iterative cycle of training, querying, and updating the model. The process continues, enriching the labeled dataset until the allocated resources for labeling are exhausted or a specific performance target is met. For the rest of this section, we will focus on pool-based AL approaches. Pool-based AL can be divided into three categories based on the sample selection criteria, typically called the query strategy or the acquisition function.

The first category includes uncertainty-based methods, where the focus is on selecting data samples that exhibit high levels of predictive uncertainty (uncertainty of predictions). Predictive uncertainty can be divided into aleatoric and epistemic uncertainties. Aleatoric uncertainty arises from the inherent randomness in the data generation process, reflecting the natural variability in the data itself. While epistemic uncertainty arises from the learning processes, reflecting the knowledge gaps within the model [15]. Uncertainty-based methods utilize predictive uncertainty to inform the query strategy. AL-DL [51] utilizes the class label information entropy as a metric, which considers all class label probabilities for measuring

uncertainty. It focuses on the classification head of a detector and selects the top N-ranked samples based on the entropy of the samples. Bayesian Active Learning by Disagreement (BALD) [14] utilizes Bayesian neural networks, particularly performs approximate variational inference to choose data points that are expected to maximize the mutual information between predictions and model parameters, i.e., data points expected to maximize the information gained by the model parameters. Learning Loss for Active Learning (LLAL) [59] is a task-agnostic method that attaches a small auxiliary network to a deep network, which learns to predict the losses for the unlabeled data input and enables querying samples for which the deep network is likely to produce incorrect predictions.

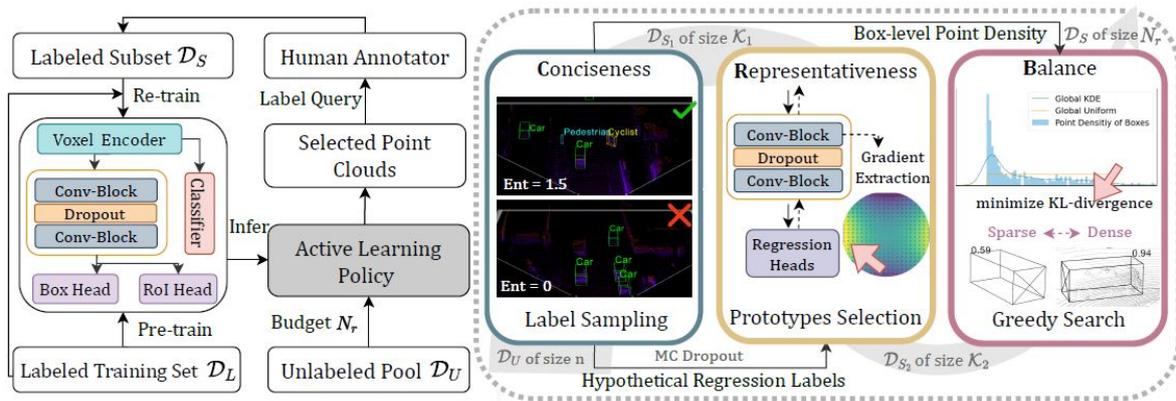
The second category involves diversity-based query methods, which aim to select a batch of samples representative of the unlabeled data that, once labeled, can act as a surrogate of the entire dataset. CoreSet [37] defines active learning as a core-set selection problem that aims to find a small subset that can be used as a proxy for the entire dataset. CoreSet employs the greedy furthest-first search on labeled and unlabeled embeddings in each selection round. Variational Adversarial Active Learning (VAAL) [42] is a task-agnostic method that utilizes a Variational Autoencoder (VAE) and an adversarial network to learn a low-dimensional latent space and sample selection from labeled and unlabeled data. The network is trained by min-max optimization, with the adversarial network predicting whether a point belongs to the unlabeled data. The intuition behind VAAL is that after training the network, the probabilities associated with the adversarial network's predictions reflect how representative each sample is from the pool it belongs to.

The third category comprises hybrid approaches. Hybrid approaches argue that query strategies that rely solely on model uncertainty or batch diversity often face challenges, as they might not consistently perform well across different model architectures, batch sizes, or datasets. In addition, they can lead to inefficiencies, such as selecting nearly identical data samples or samples that add little new information. Moreover, in real-world AL scenarios, hyper-parameter sweeps can increase labeling costs. Therefore, hybrid strategies aim to achieve a trade-off between uncertainty and diversity in query selection. Batch Active Learning by Diverse Gradient Embeddings (BADGE) [3] captures a batch of data samples that are most diverse and where the current model is most uncertain. BADGE measures uncertainty and diversity by the gradient magnitude and orientation w.r.t the model parameters in the output layer, respectively.

### 3.2.1 Active Learning for Object Detection

Training a deep object detector for effective and robust classification and localization of 2D/3D bounding boxes requires massive and comprehensive labeled data. While collecting data via onboard sensors such as LiDAR or camera is relatively simple, annotating data is very costly and time-consuming. Active learning has the potential to minimize the data annotation efforts while maximizing the object detector's performance. Despite the limited literature on active 3D object detection, mainly because of the challenges of determining the informativeness of object-location hypotheses, this field has recently been attracting significant attention from the research community [18] [19] [20] [21]. One of the earliest methods, [23], implements an uncertainty-based AL method, which explicitly considers the localization of detected objects by proposing two new metrics. Localization Tightness (LT) measures the variation of IoU between box proposals in the Region Proposal Network (RPN) and the final predicted bounding box. Localization Stability (LS) measures the variation in

box predictions when Gaussian noise is added to the image. Images are selected for annotation based on the discrepancy between their box classification and both LT and LS. [36] utilizes uncertainty-based AL by introducing the consensus score, a new metric for measuring the uncertainty of 2D and 3D ensemble networks by calculating the variation ratio of the minimum IoU value for each RoI-match of 2D/3D boxes. In addition, we propose active class weighting in the loss function to alleviate the common class imbalance in autonomous driving (AV) datasets. [12] employs an uncertainty-based AL method, which utilizes Monte Carlo dropout associated with mutual information from 2D image frustums and 3D point clouds to determine the uncertainty of point clouds, and hence proposes to reduce the 3D labeling efforts by allowing the human annotator to label 3D point clouds within frustums. In [29], the authors introduce the CRB framework, which incorporates three criteria for point cloud selection: label conciseness, feature representativeness, and geometric balance. The method hierarchically filters the point clouds, selecting those with non-redundant labels, discriminative features, and balanced geometric properties. The CRB selection process contributes to minimizing the generalization error by aligning the distribution of the selected samples with the prior distribution of the test set, promoting the model's robustness to the data diversity and further reducing the likelihood of overfitting.



**Figure 3.6:** An illustrative diagram of the CRB framework [29], which systematically selects point clouds based on unique bounding box labels and distinct gradient and geometric features. This approach helps bridge the gap with the test set while keeping annotation efforts minimal.

In the following, we provide an overview of the three stages of the CRB approach.

1. **Label Conciseness:** In the first stage, the goal is to address label redundancy and class imbalance. This is achieved by computing the entropy of bounding box predictions. The point clouds with the highest entropy, indicating a diverse set of labels, are selected for further analysis. This step ensures that the frames chosen for annotation are the most informative and do not contain redundant label information.
2. **Feature Representativeness:** In the second stage, the goal is to discover the uniqueness of the point clouds encoded in the unlabeled data and not in the labeled data. This is achieved by utilizing the magnitude and orientation of the gradient vectors of the point clouds, representing uncertainty and diversity, respectively.
3. **Geometric Balance:** The final stage aims to maintain a balance in the geometric properties of the selected point clouds. This involves minimizing the KL-divergence between the marginal distributions of point cloud density of each predicted bounding box. By doing so, the framework ensures that the model is trained on geometrically diverse point clouds, including both sparse and dense objects.

### 3.3 3D Annotations Tools

The rapid development of 3D sensors and LiDAR-based object detection methods fuels a growing demand for open-source and user-friendly 3D annotation platforms that enable researchers and non-domain experts to re-annotate and create their own datasets. Manually annotating 3D datasets is particularly challenging. On the one hand, annotating 3D bounding boxes on LiDAR point clouds is difficult and requires a certain level of expertise due to the inherent sparsity and irregularity of LiDAR point clouds. On the other hand, annotating 3D bounding boxes using 2D RGB images is inefficient due to occlusions and depth ambiguity. Although it is essential to combine 2D images and 3D point clouds for a high-quality and efficient 3D bounding box annotation, the annotation process remains challenging and demands a careful procedure of annotator training, quality assurance, and continuous improvement of the tool's usability. In this section, I review some of the latest advanced and open-source 3D annotations tools and highlight their key features and contributions to an effective and user-friendly annotation process.

#### 3.3.1 3D BAT

3D Bounding Box Annotation Tool (3D BAT) [65] is an open-source, web-based application designed for efficient and accurate 3D annotation, visualization, and tracking of objects. Developed for use in applications related to autonomous and partially automated vehicles, this tool supports obtaining 2D and 3D labels as well as track IDs for objects on the road. Its key features include semi-automatic labeling of tracks using interpolation, which is crucial for motion prediction and planning. The semi-automatic labeling feature streamlines the process of labeling object trajectories by interpolating the movements of objects between frames. Besides projecting annotations for 3D point clouds to 2D images, an additional Master-View feature provides Bird's Eye View (BEV), side, and front views for observing objects from different perspectives.



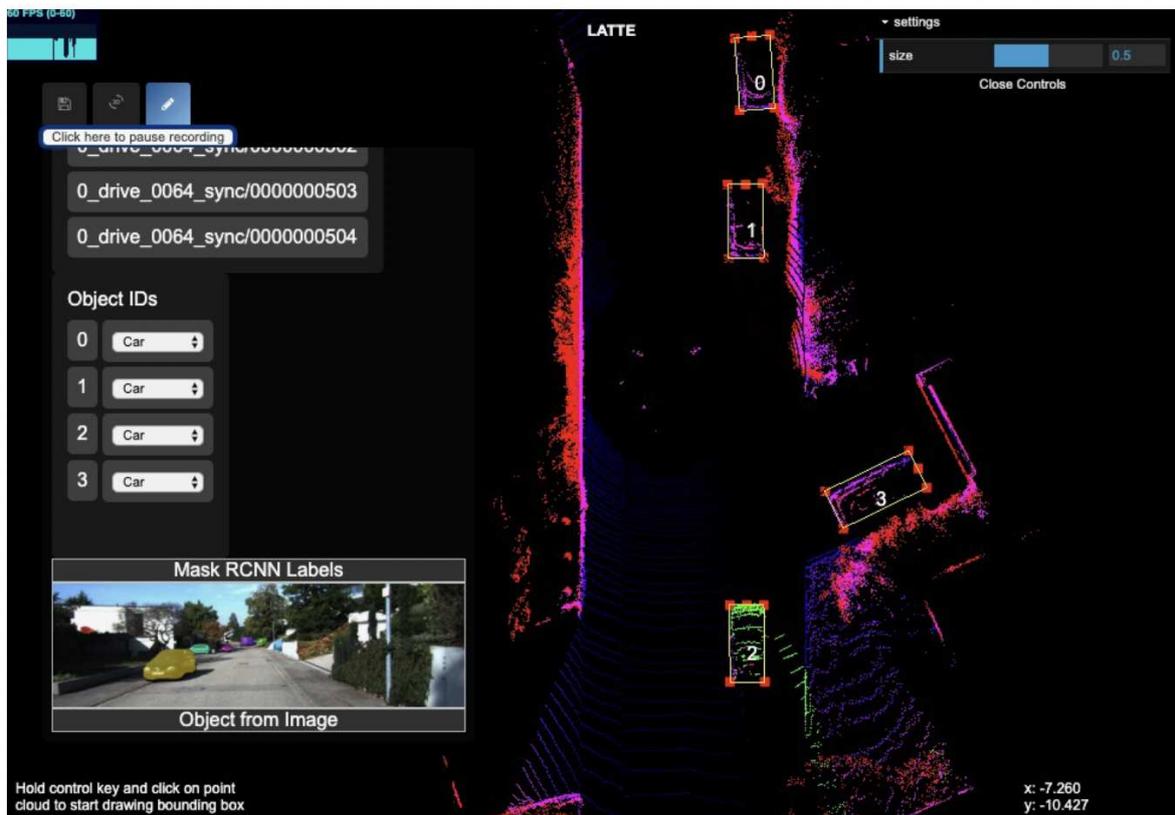
**Figure 3.7:** Presents a snapshot of the tool's UI, demonstrating its functional components. 1) Shows an adjustable panoramic camera view that allows users to scroll sideways and adjust the height, offering a 360-degree perspective. 2) Shows the master view consisting of side view (top), front view (middle), and BEV view (bottom). 3) Shows an interactive 3D window where users can annotate objects within the point clouds in both perspective and orthographic views.

### 3.3.2 LATTE

LATTE [50] is an open-source annotation tool designed to accelerate the annotation of LiDAR point clouds. The authors describe the main challenges of annotating LiDAR point clouds as point cloud sparsity, low resolution, difficulty drawing accurate 3D bounding boxes, and the sequential nature of LiDAR data, which often lead to repetitive annotations. LATTE introduces multiple features to address these challenges:

- **Sensor Fusion:** This feature leverages image-based detection algorithms to pre-annotate objects in an image, then the labels are projected to the point cloud. The feature helps annotators recognize objects more easily in sparse or low-resolution point clouds.
- **One-click annotation:** To simplify the annotation process, the annotator clicks once on a point of the target object in the point cloud. Using the DBSCAN [11] clustering algorithm, The tool automatically generates a top-view 2D bounding box around the object.
- **Tracking:** Integrating Kalman filtering [53] for tracking into an annotation process allows for the transfer of labels from one frame to a subsequent one, significantly reducing repetitive annotations.

In conclusion, LATTE represents a significant advancement in LiDAR point cloud annotation, offering a more efficient and accurate tool for researchers and developers in LiDAR-based object detection.

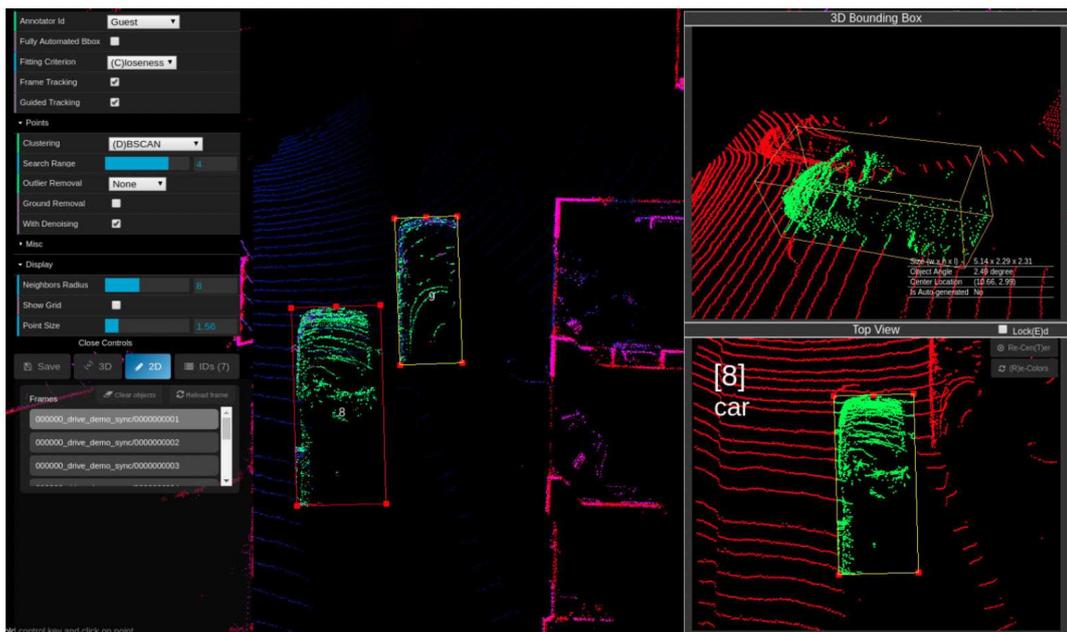


**Figure 3.8:** LATTE main user interface shows the point cloud from BEV perspective, the camera image with 2D masks, and a list of the objects.

### 3.3.3 SAnE

SAnE (Smart Annotation and Evaluation) [1] is an open-source semi-automatic annotation tool designed for accelerating the annotation process of LiDAR point clouds, with a particular focus on efficiency and ease of use. The main features and innovations of SAnE include:

- **Denoising Point-wise Segmentation:** This unique feature of SAnE enhances the one-click annotation by providing nearly noise-free semantic segmentation. This is achieved by using a novel denoising technique that removes the need for a ground removal algorithm, which is often required in other annotation tools. This segmentation strategy significantly improves the accuracy and reliability of one-click annotations.
- **Guided Tracking:** SAnE enhances the frame-to-frame annotation process by utilizing a guided tracking algorithm that builds upon traditional motion model techniques by incorporating heuristic approaches like greedy search and backtracking, allowing for minimal adjustments by the annotator.
- **Improved Annotation Flow:** The tool includes user interface improvements, such as keyboard-only annotations, multi-user environments, user-adjusted parameters, and 3D bounding box estimation, all contributing to a more efficient and user-friendly annotation experience.



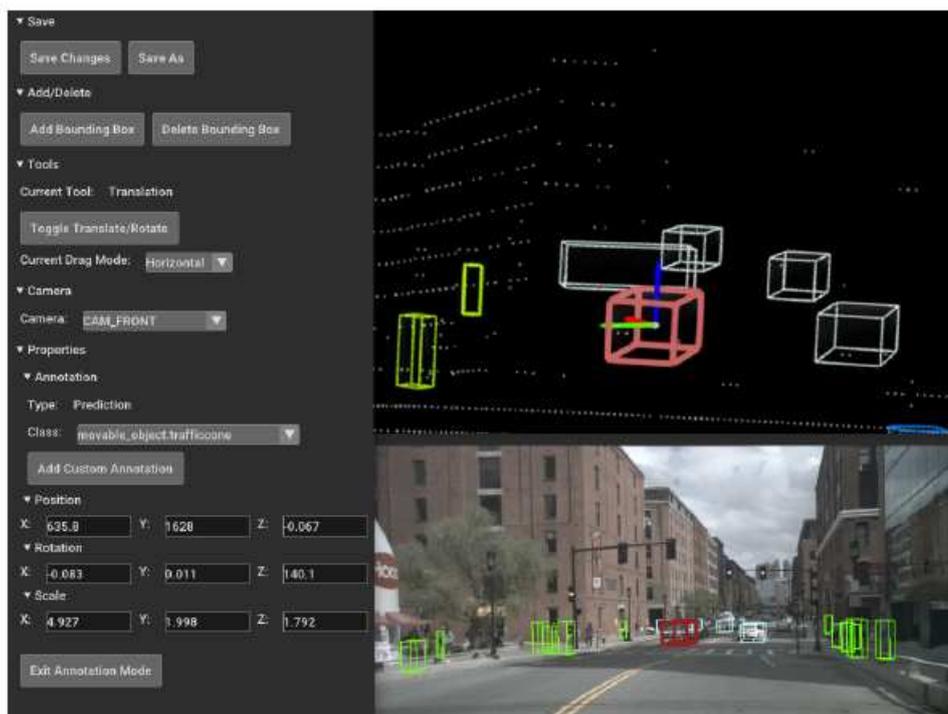
**Figure 3.9:** SAnE features a user-friendly interface for semi-automatic annotation, utilizing a one-click annotation framework, further enhanced by a denoising point-wise segmentation technique and an innovative guided-tracking mechanism.

In summary, SAnE represents a step forward in point cloud data annotation. Its innovative features, such as denoising point-wise segmentation and guided tracking, combined with an improved annotation flow, make it a valuable tool for expert annotators and crowd-sourcing contexts.

### 3.3.4 ReBounce

ReBounce [5] is an open-source 3D bounding box annotation tool designed to utilize active learning in the context of autonomous vehicle (AV) research and 3D object detection. Developed to address the limitations of existing datasets, which often have incomplete annotations, ReBounce allows for the re-annotation of various datasets. This tool is effective for both data exploration and active learning, offering functionalities like correcting bounding box predictions, creating custom labels, analyzing model predictions, and exporting annotations to dataset-specific formats. The tool's interface is user-friendly and intuitive, consisting of three primary windows: a control window, a point cloud viewer, and an RGB image viewer. In addition, users can navigate through data, filter annotations, and interact with the 3D point cloud using Open3D for rendering, editing, or deleting annotations.

ReBounce supports nuScenes [4], Waymo Open Dataset [45], and Argoverse 2.0 [54], using a Python command line tool to convert their annotations into a unified ReBounce format. This generic format, focusing on essential 3D bounding box details, allows easy adaptation to other datasets. Additionally, ReBounce enables users to export modified annotations back to their original formats. This streamlined process, which doesn't alter LiDAR or RGB images, enhances export speed and allows for cross-dataset annotation conversions, aiding in model evaluation and research in model robustness and domain adaptation. Users can extend the tool's functionality to accommodate new datasets



**Figure 3.10:** ReBounce's UI shows a multi-modal view with a point cloud viewer (top) and an RGB image viewer (bottom). The editing window provides options for adding, deleting, and adjusting bounding boxes with controls for translation and rotation, as well as other properties such as the annotation type, class, and transformation parameters.

### 3.3.5 Xtreme1

The Xtreme1 LiDAR Annotation Tool [13], part of the BasicAI Cloud platform, is designed for efficient and precise annotation of LiDAR data, crucial for developing advanced driver-assistance systems (ADAS) and autonomous vehicles. Key features of this tool include:

- **Cuboid Creation:** Annotator can create new cuboids on a 3D canvas, which is integral for identifying and marking objects within LiDAR
- **AI-assisted Annotation:** The tool offers AI-assisted capabilities to automatically fit cuboids accurately on the objects within the point cloud data. This feature significantly enhances the efficiency and accuracy of the annotation process.
- **Model Pre-annotation:** The Xtreme1 platform includes present models for ADAS scenarios. These models can auto-generate annotation results with a simple button click, streamlining the annotation workflow. Additionally, there's an option to integrate other models into the platform for more versatility.
- **Data Information Display:** The tool provides functionality to display critical data information like coordinates and points of the selected scenes, aiding in a more comprehensive understanding of the LiDAR data.

Combined with BasicAI Cloud's data-centric MLOps platform, which covers the full AI lifecycle from data management to model training and inference, this annotation tool enhances the efficiency of AI model development, particularly for complex, multi-modal data like LiDAR.

The architecture of BasicAI Cloud is designed following modern cloud-based standards. It uses containerized methods for setting up its software, with Kubernetes managing these containers for better performance and flexibility. The platform is organized into various independent services for user interface and server-side operations, which can be upgraded as needed. These services are built to adjust automatically based on user demand. BasicAI Cloud also uses widely-accepted, cloud-compatible software for its databases and other key operations, ensuring it can efficiently manage large amounts of data. The entire process, from initial code development to final deployment, is automated using GitLab CI, making it easier for developers to focus on writing code.



**Figure 3.11:** The Xtreme1 annotation tool main interface with intuitive tools for labeling objects, combining camera images and 3D views to help users annotate and track multiple objects.

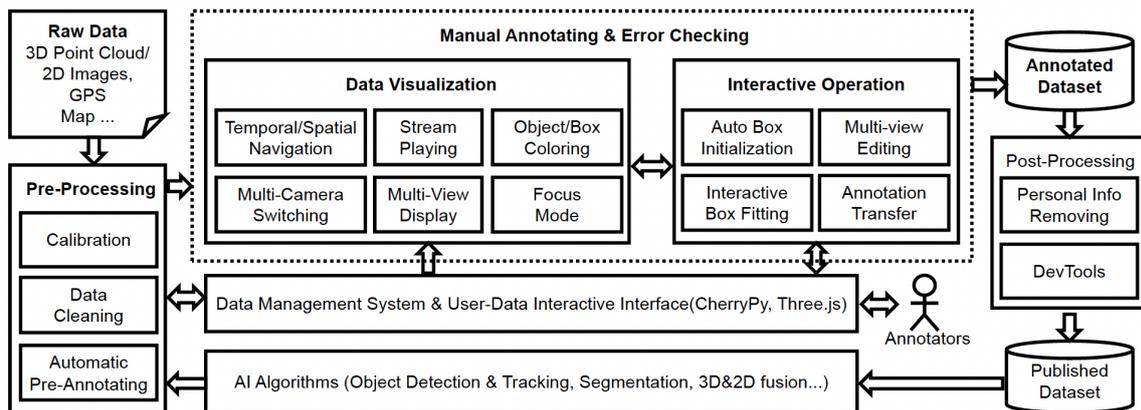
### 3.3.6 SUSTech Points

SUSTech Points [26] is a web-based interactive 3D point cloud annotation platform, Figure 3.12, designed with a particular focus on portability and user-friendliness. The tool presents several key features and innovations. Firstly, the platform offers an intuitive and interactive 3D interface for efficient navigation and annotation of complex point clouds. Its portability and compatibility across different hardware platforms and operating systems further extend its utility. The tool includes advanced tools for rapid bounding box generation, easy label assignment, and quick editing, streamlining the annotation workflow. Additionally, it provides advanced visualization features to assist users in accurately distinguishing and annotating different elements within the point cloud.

**Figure 3.12:** The main user interface of SUSTech POINTS is composed of several elements. It includes a perspective view of the 3D point cloud and a re-sizable photo context that can automatically switch between various camera images. Additionally, Side, Front, and BEV views. There is also a focused photo context, automatically selected based on the object in focus.



The system architecture of SUSTech POINTS, Figure 3.13, is built around a web server managed with the CherryPy web application development framework and a front-end providing module functionalities programmed with the WebGL library Three.js.

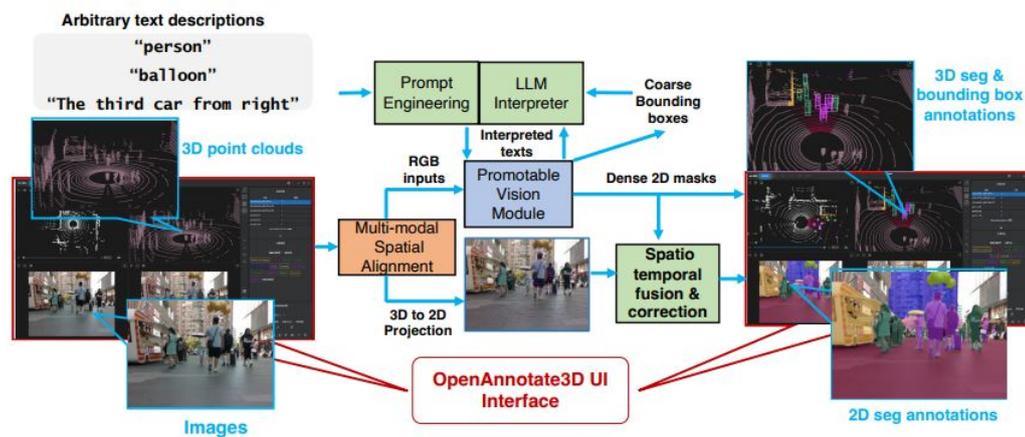


**Figure 3.13:** This architecture supports the development of visualization modules and interactive tools for 3D bounding boxes and tracking ID annotation. The platform's data pre-processing stage involves estimating intrinsic and extrinsic sensor parameters, selecting data frames consistent with calibration parameters for algorithm training and verification, and optionally using a set of AI algorithms for detection, tracking, and 2D & 3D fusion to generate initial annotation results.

### 3.3.7 OpenAnnotate3D

OpenAnnotate3D [61] is a cutting-edge, open-source system designed for labeling multi-modal 3D data. It's particularly notable for its capability to create 2D and 3D masks as well as 3D bounding box annotations automatically. This tool simplifies the typically complex process of annotating 3D data, differentiating itself from traditional tools that require manual, direct interaction with 3D data. OpenAnnotate3D utilizes Large Language Models (LLMs) and a vision module that can respond to prompts, making the annotation process more efficient and user-friendly. It starts by interpreting a user's labeling request through LLMs, potentially involving multiple interactions with the vision module for precise execution. This approach reduces manual effort and makes the tool adaptable to various data types.

This approach is notably different from conventional tools, which usually require hands-on



**Figure 3.14:** The OpenAnnotate3D process begins when a user submits a labeling request. Initially, the system analyzes this request using the LLM interpreter and precise, prompt engineering. It's important to note that the interpreter might engage with the promptable vision module in multiple iterations to ensure that the interpreted text aligns with its reasoning capacity. Dense 2D masks are generated, which are then used to create 3D masks through a multi-modal spatial alignment process. To address any imperfections in the 2D masks, a combination of spatio-temporal fusion and correction is implemented to enhance the accuracy of the 3D labels.

navigation and annotation within complex 3D data. OpenAnnotate3D automates the initial reasoning and processing stages, significantly reducing manual effort. It starts by creating 2D masks, then aligns them to generate 3D masks, and finishes with a spatio-temporal fusion and correction step. This automated and intelligent processing simplifies the annotation process and enables open-vocabulary annotations, making it suitable for a broader range of data and annotation needs.

The pipeline for iterative text interpretation in OpenAnnotate3D plays a crucial role in this process. The Large Language Model (LLM) initially interprets the user's prompt, identifies the main objectives, and performs an initial query to the vision module. This prompt is continually refined by the LLM based on feedback from the vision module, which analyzes the scene. This iterative process between the LLM interpreter and the vision module significantly boosts the system's reasoning abilities and segmentation precision. The spatio-temporal and correction feature is important for video data annotation. It offers an interpolation method that automatically labels frames between a user-defined start and end, enhancing efficiency but potentially lacking accuracy for intermediate frames. Additionally, it offers a frame-by-frame auto-labeling feature that uses spatial and temporal data to accurately track and correct the movement of objects.

### 3.3.8 WebLabel

WebLabel [49] is a comprehensive 3D labeling tool designed to handle complex multi-sensor data in compliance with the OpenLABEL standard. Key features include a flexible user interface supporting various data formats and use cases and an architecture integrating with the VCD Toolkit for efficient translation into OpenLABEL format. WebLabel is unique in its capacity to handle a wide range of labeling tasks, such as image labeling, multi-view video object annotation, point-cloud view-based labeling for 3D geometries, and action recognition. The tool is particularly notable for its application in the automotive industry, aiding in developing and testing autonomous driving systems. WebLabel's versatility and compliance with OpenLABEL make it a valuable resource for advanced, multi-dimensional data annotation tasks.

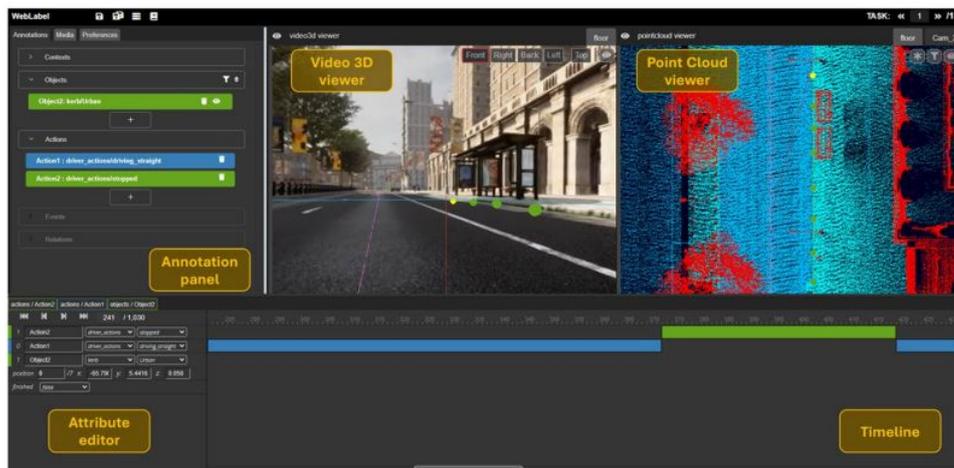


Figure 3.15: The main user interface of WebLabel

The architecture of WebLabel is structured to support its multipurpose, web-based application nature, fully compatible with the OpenLABEL standard. It encompasses six main parts: Data Tools for managing annotations, Content Tools for processing multimedia content, Viewers for functionality and display, and Timeline for annotation duration analysis and modification. The Attribute Editor defines classes and attributes of elements, and the entire application is orchestrated with AngularJS, a framework for developing single-page applications. The Data Tools use the VCD library, ensuring full compliance with OpenLABEL and facilitating loading, saving, and managing annotation data.

### 3.3.9 PointCloud Lab

PointCloudLab [10] is a non-conventional and innovative platform for 3D point cloud annotation. This tool introduces varied levels of immersion, as shown in Figure 3.16, allowing annotators to engage with the data in a manner that best suits their preferences and the task requirements. It provides different visual aids like distance-based coloring, region-based coloring, and object-based highlighting to assist users in identifying and annotating objects within the point clouds.



**Figure 3.16:** Levels of immersion: (a) An interactive non-immersive setup. (b) Semi-immersive setup. (c) Fully-immersive setup incorporating all 6 DoF.

The tool's visual aids complement the different levels of immersion. Distance-based coloring helps annotators comprehend the depth and relation of objects. In contrast, region and object-based coloring clearly distinguish areas and items, reducing the chances of mislabeling. These visual aids are crucial in reducing the cognitive load on annotators, especially when working within more immersive environments where spatial awareness is vital. By integrating these visual supports with the flexibility of immersion, PointCloudLab ensures a customizable and user-centric annotation experience that aims to optimize both the speed and quality of the annotating process.

### 3.3.10 labelCloud

labelCloud [34] is a domain-agnostic, lightweight tool designed specifically for 3D object detection within point clouds. Structured according to the Model-View-Controller (MVC) design paradigm and developed with Python, which allows the integration of useful modules such as Open3D and NumPy.

In labelCloud, an intuitive labeling process involves detecting objects, creating bounding boxes, and correcting box parameters. Besides, it offers two labeling modes, namely picking and spanning. The picking mode is for objects with known sizes, allowing for quick adjustment of bounding boxes. The spanning mode simplifies labeling by reducing the process to four clicks and defining the box dimensions and orientation, which is particularly efficient for objects on flat surfaces. Moreover, labelCloud provides a feature to align point clouds with the floor, requiring the user to select three points on the floor plane to compute a transformation matrix that re-aligns the point cloud to the world coordinate system.

Table 3.1 provides a compact overview that summarizes the previous discussions about 3D annotation tools. It presents a comparative analysis of the tools, highlighting the availability and absence of key features across different platforms. This table serves as a quick reference to assess the capabilities of each tool, such as camera and LiDAR tracking, HD maps integration, support for ASAM OpenLABEL standard, 3D navigation, and other features. The presence of a feature is highlighted by a checkmark and a green background, while the absence of a feature is highlighted by a red background, and a question mark denotes that it's unclear whether the tool includes the corresponding feature or not.

**Table 3.1:** Comparison of 3D annotation tools. ○ Feature provided ○ Feature unknown ○ Feature not provided

Tool	3D BAT [65]	LATTE [50]	SAnE [1]	SUSTech POINTS [26]	Label Cloud [35]	Re Bound [6]	Point Cloud Lab[10]	Xtreme1 [13]	pro Anno
Year	2019	2020	2020	2020	2021	2023	2023	2023	2023
Support V2X	-	-	-	-	-	-	-	✓	✓
Camera+LiDAR	✓	✓	-	✓	✓	✓	-	✓	✓
Tracking	✓	✓	✓	✓	-	-	✓	✓	✓
HD Maps	-	-	-	✓	-	-	-	✓	✓
OpenLABEL support	-	-	-	-	-	-	-	-	✓
Web-based	✓	-	-	✓	-	-	-	✓	✓
3D Navigation	✓	✓	✓	✓	-	✓	✓	✓	✓
3D transform controls	✓	✓	✓	✓	-	✓	✓	✓	✓
Side views	✓	-	✓	✓	-	✓	-	✓	✓
One-click Annotation	✓	✓	✓	✓	-	✓	✓	✓	✓
Label attributes	-	-	-	✓	-	✓	(?)	✓	✓
Batch-mode editing	-	-	-	✓	-	-	-	✓	✓
Perspective view editing	✓	✓	✓	✓	✓	✓	✓	✓	✓
Orthographic view editing	✓	✓	✓	-	-	✓	✓	✓	✓
Support multiple cameras	✓	-	-	✓	-	✓	-	✓	✓
Support JPG/PNG files	-	(?)	(?)	✓	-	-	-	(?)	✓
Object coloring	✓	-	✓	✓	-	✓	✓	✓	✓
Focus mode	-	-	-	✓	-	-	✓	✓	✓
Open-source	✓	✓	✓	✓	✓	✓	-	✓	✓
Github stars	531	374	62	670	461	20	-	542	0
Citations	46	36	16	33	16	0	2	0	0
License	MIT	Apach. 2.0	Apach. 2.0	GPL 3.0	GPL 3.0	Apach. 2.0	-	Apach. 2.0	MIT

### 3.4 Annotation Formats for Object Detection

The development of automated driving heavily relies on Machine Learning (ML) for perception and prediction tasks, requiring huge amounts of annotated training data. Without a standardized format, the academic and industrial communities face challenges in reusing, maintaining, and sharing datasets, leading to reduced annotation quality. This issue wasn't fully recognized in the earlier stages of machine learning research. However, as datasets have expanded in size and complexity to fulfill the requirements of autonomous vehicle (AV) research and development, the need for a structured and comprehensive annotation format has become increasingly critical. In this subsection, we review three different annotation formats that are widely used in 3D object detection datasets: KITTI format, nuScenes format, and ASAM OpenLABEL.

### 3.4.1 KITTI Format

The KITTI [16] label convention is one of the earliest adopted conventions for 3D object detection datasets in the context of autonomous vehicles. It's a simple, easy-to-read, and easy-to-use TXT format. The label files are structured in row-major, where each row contains 15 columns specifying the information of one bounding box:

<b>Type</b>	Describes the type of object.
<b>Truncated</b>	Float from 0 (non-truncated) to 1 (truncated), where truncated refers to the object leaving image boundaries.
<b>Occluded</b>	Integer (0,1,2,3) indicating occlusion state: 0 = fully visible, 1 = partly occluded 2 = largely occluded, 3 = unknown
<b>Alpha</b>	Observation angle of object, ranging $[-\pi, \pi]$
<b>Bbox</b>	2D bounding box of the object in the image (0-based index): contains left, top, right, bottom pixel coordinates.
<b>Dimensions</b>	3D object dimensions: height, width, length (in meters)
<b>Location</b>	3D object location $x,y,z$ in camera coordinates (in meters)
<b>Rotation<sub>y</sub></b>	Rotation around Y-axis in camera coordinates $[-\pi, \pi]$
<b>Score</b>	Only for results: Float, indicating confidence in detection, needed for p/r curves; Higher is better.

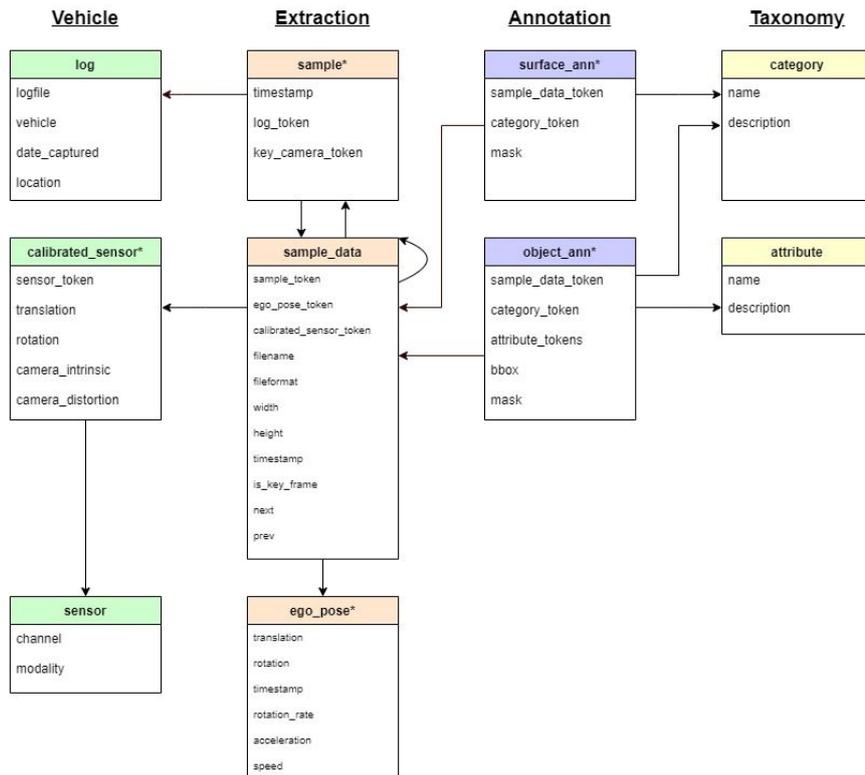
**Table 3.2:** Description of the data fields in a KITTI format labels file.

Intrinsic calibration parameters and LiDAR-to-Camera projection matrices are provided in a separate calibration file. While the KITTI format is easy to use, it has some limitations: it scatters various details about the same frame across multiple files, making it more complex to manage. Additionally, it lacks extensive data about the frame, including metadata and synchronization details between LiDAR and camera streams.

### 3.4.2 nuScenes Schema

Another widely used annotation format is the nuScenes annotation schema, which formats the autonomous vehicle (AV) research data into a relational database, as shown in Figure 3.17. The annotation schema comprises several key components:

- **Vehicle and Sensor:** Information about the data-collection vehicles and their various sensors, detailing their types and calibration parameters.
- **Extraction:** Records of driving sequences (scenes) and specific moments (samples) within them, each uniquely timestamped.
- **Annotation:** Details about the identification and description of object instances within each scene, along with their attributes.
- **Taxonomy:** The system categorizes objects and breaks them down into main groups and finer divisions, also covering how visible they are and other attributes.



**Figure 3.17:** The diagram showcases the nuScenes schema for capturing and organizing data for autonomous vehicle research. The schema is divided into four main components: Vehicle and Sensor, Extraction, Annotation, and Taxonomy.

In the following, we show two examples of important data tables utilized in the nuScenes format.

**Sample Data:** Sensor data, like images, point clouds, or radar feedback, are classified based on timestamps.

token	<str> – Unique record identifier.
sample_token	<str> – This points to a sample NOT a sample_data.
ego_pose_token	<str>.
calibrated_sensor_token	<str> [n].
filename	<str> – Relative path to data blob on disk.
fileformat	<str> – Data file format.
width	<int> – The image width in pixels.
height	<int> – The image height in pixels.
timestamp	<int> – Unix time stamp.
is_key_frame	<bool> – True if sample_data is part of key_frame, else False.
next	<str> – Sample data from the same sensor that follows this.
prev	<str> – Sample data from the same sensor that precedes this.

**Table 3.3:** Description of the data fields in the nuScenes sample data table.

**Sample Annotation:** A bounding box marks the location of an object observed in a sample. The location information for this bounding box is provided in terms of the global coordinate system.

token	<str> – Unique record identifier.
sample_token	<str>. NOTE: this points to a sample NOT a sample_data.
instance_token	<str> – Which object instance is this annotating.
attribute_tokens	<str> [n] – List of attributes for this annotation.
visibility_token	<str> - Visibility may also change over time. If no visibility is annotated, the token is an empty string.
translation	<float> [3] – Bounding box location in meters.
size	<float> [3] – Bounding box size in meters as width, length, height.
rotation	<float> [4] – Bounding box orientation as quaternions: w, x, y, z.
num_lidar_pts	<int> – Number of lidar points in this box.
num_radar_pts	<int> – Number of radar points in this box.
next	<str> – Sample annotation from the same object instance that follows this.
prev	<str> – Sample annotation from the same object instance that precedes this.

**Table 3.4:** Description of the data fields in the nuScenes sample annotation table.

### 3.4.3 ASAM OpenLABEL

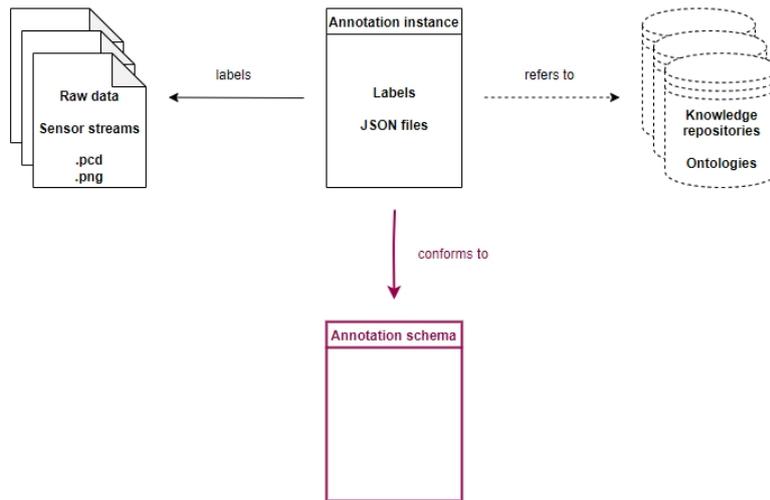
The ASAM OpenLABEL standard [2] simplifies multi-sensor data annotation and scenario tagging by offering a unified JSON format for annotations, reducing time and resource expenditure. It supports various labeling techniques, including 2D bounding boxes for images and 3D bounding boxes for point clouds, and enables straightforward integration with tools for parsing. OpenLABEL also allows for efficient scenario categorization and retrieval through tagging, providing essential details like scenario origin and data collection setup, streamlining the arrangement of multi-sensor data, and test scenarios for increasing efficiency.

ASAM OpenLABEL serves key groups such as ML and perception engineers, research scientists, data annotation professionals, and test engineers, streamlining the development and validation of automated driving functions. It presents an annotation standard with the following main characteristics:

- Supporting efficient data sharing among industries and academic circles.
- Enhancing annotation quality with a unified data structure.
- Boosting maintainability and re-usability of datasets.

The annotation schema is structured as JSON schema and conforms to the JSON schema Draft 7 specification [22], which outlines how annotations should be formatted and understood. Key aspects relevant to multi-sensor data labeling include:

- Labeling different spatio-temporal objects.
- Temporal persistent identities of objects.



**Figure 3.18:** ASAM OpenLABEL multi-sensor data labeling concept.

- Capturing geometric, non-geometric, nested, and custom attributes.
- Multi-sensor object annotations
- Managing different coordinate systems, transformations, and sensor configurations.
- Different types of elements: objects, actions, events, and contexts.

The ASAM OpenLABEL annotation schema is structured as a dictionary and can be described from top to bottom. Any ASAM OpenLABEL JSON data has a root key named *openlabel*, as shown in Figures 3.19 and 3.20. Its value is a dictionary containing the rest of the structure.

JSON example

```

1  {
2    "openlabel": {
3      "metadata": {
4        "schema_version": "1.0.0"
5      }
6    }
7  }
```

JSON example

```

1  {
2    "openlabel": {
3      "objects": { ... },
4      "actions": { ... },
5      "events": { ... },
6      "contexts": { ... },
7      "relations": { ... },
8      "frames": { ... },
9      "frame_intervals": { ... },
10     "metadata": { ... },
11     "ontologies": { ... },
12     "resources": { ... },
13     "coordinate_systems": { ... },
14     "streams": { ... }
15   }
16 }
```

**Figure 3.19:** Header of the JSON schema

**Figure 3.20:** Content of the JSON schema

The ASAM OpenLABEL format is designed to organize data from various sensors using dictionaries. These dictionaries categorize elements like objects and activities, their interactions and the scenarios they are part of, as shown in Figure 3.21. Each element is listed as a pair: a unique key that identifies it and a value that gives detailed, unchanging information. There are also additional components to the structure:

- Ontologies to categorize and define data
- External resources that link the data to additional information
- Defined coordinate systems for clear data conversion instructions.
- Data streams that provide details about the labeled data, such as the calibration parameters of a camera.

When it's essential to include timing, like in video labeling, 'frames' hold a dictionary that organizes data by individual frames. 'Frame intervals' compile the specific frames that carry information relevant to the ASAM OpenLABEL annotation file. This comprehensive structure ensures that every piece of data has its place and relation to the whole, making it a robust system for labeling and analyzing multi-sensor data.

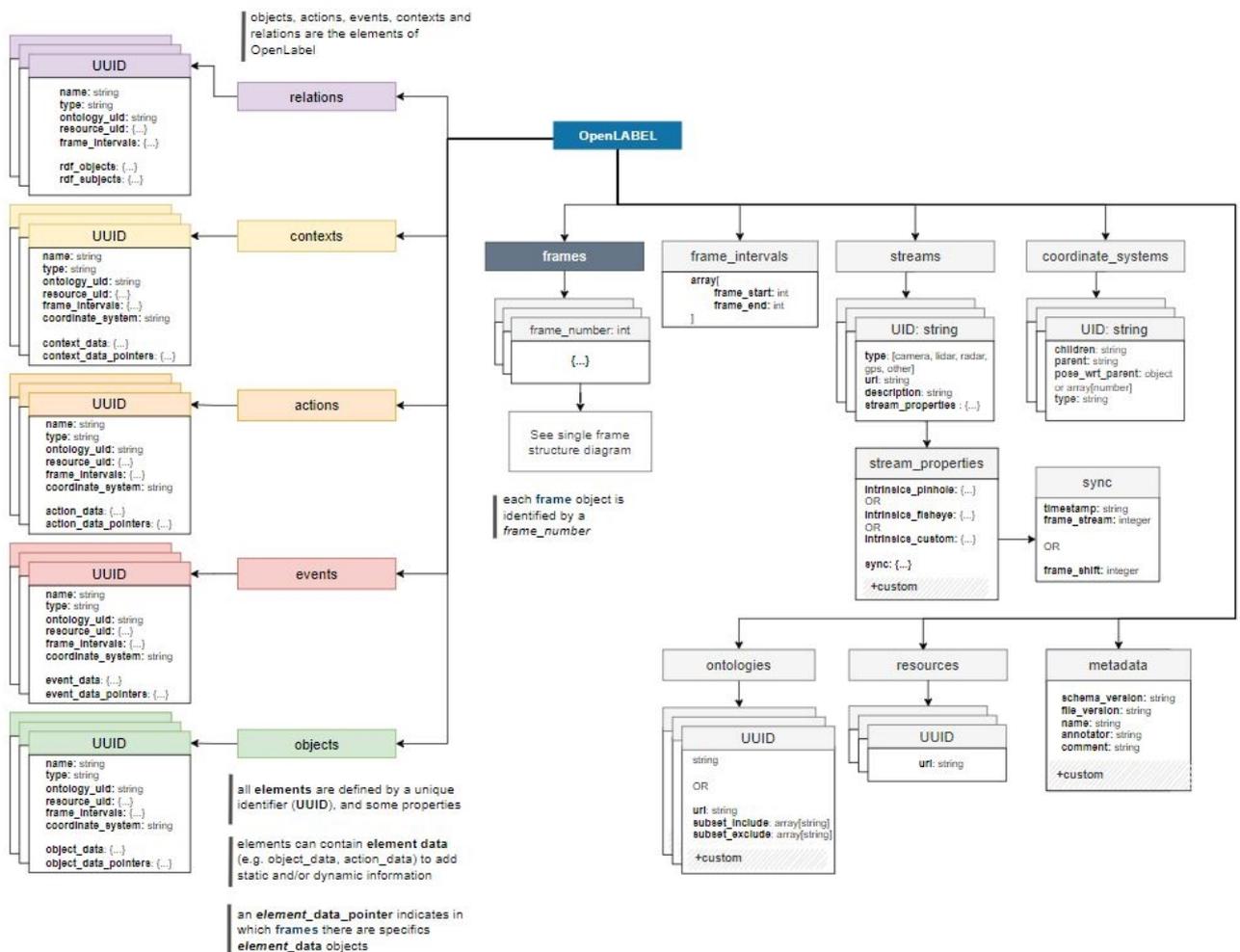


Figure 3.21: ASAM OpenLABEL data structure for multi-sensor data labeling.

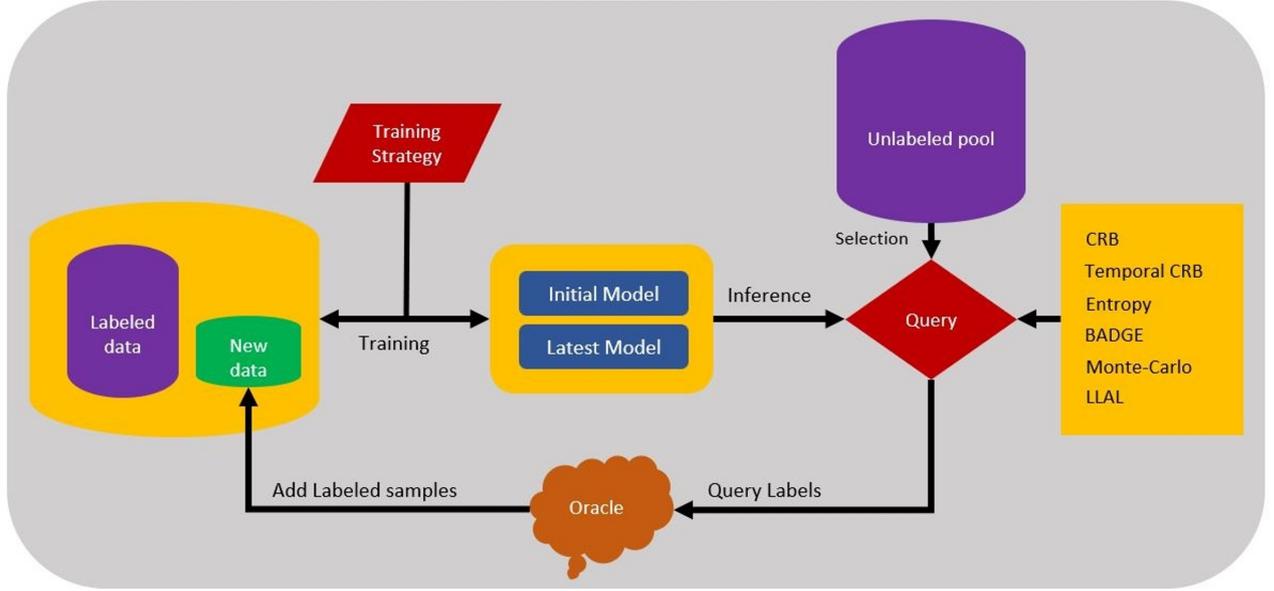
# Chapter 4

## Methodology

In this chapter, we present the improvements and modifications made to the framework of active learning (AL) in the context of 3D object detection for autonomous vehicles (AV) and mobile robots. Our methodology encompasses three key contributions. Firstly, we tackle the class imbalance prevalent in 3D datasets by introducing *active class weighting* in the CRB framework. This involves adjusting the entropy calculation to incorporate weights inversely proportional to class frequencies in the training data, promoting a more balanced sample selection. Next, we propose *temporal CRB (tCRB)*, an extension of the hierarchical CRB strategy that incorporates a sequential sample selection mechanism. tCRB is tailored to the requirements of 3D point cloud annotation, where an annotator may prefer a sequence of frames rather than a random selection. Thirdly, we introduce *continuous training* strategies to the active learning pipeline. Acknowledging the drawbacks of re-training models from scratch in the active learning cycle, we present multiple training strategies that retrain and build upon the knowledge from previous cycles, making the model’s development more resource-efficient. Lastly, we enhance the *Providentia Annotation* platform, a web-based 3D and 2D annotation tool, by integrating it with the Active3D pipeline. This integration brings AI-assisted features such as Active Learning for selecting the most informative frames to annotate and Auto-labeling by a pre-trained 3D object detector, minimizing manual annotation efforts while maximizing data informativeness and model performance. Our approach not only streamlines user interaction but also utilizes the active learning process. Each of these contributions is designed to refine the active learning process, ensuring more effective and efficient model training and deployment in real-world AV and robotics applications.

### 4.1 Continual Training

Considering the pool-based active learning scenario in the context of 3D object detection, where an initial 3D detector is pre-trained on the initially labeled dataset. After a certain amount of time (*episode*), the acquisition function determines the most informative point clouds and queries the labels from the oracle. The initial model is then updated using the set of acquired point clouds. This is a typical scenario, particularly in applications of Autonomous Vehicles (AV) and mobile robotics, where the vehicle or the robot *episodically* encounters new data, and the model is updated accordingly. However, we find this process time- and resource-consuming as the model is retrained from scratch every *episode* on the growing set of labeled data. Besides that, the model disregards the knowledge acquired in the previous *episodes*. To address this issue, we augment the active learning cycle with multiple yet simple continuous training strategies.



**Figure 4.1:** Active Continuous Training. Integrating continuous training within the pool-based active learning cycle.

As shown in Figure 4.1, the process begins with an initial model, a small labeled set, and a large pool of unlabeled data. The model performs inference on the unlabeled data, and based on its predictions, a predefined query strategy selects the most informative point clouds and queries their labels from the oracle (*i.e.*, the human expert). The following episodes of training, model update, and sample selection depend on the predefined training strategy. The procedural details of the active continuous training approach are presented in Algorithms 1 and 2, offering a clear representation of the process.

We use the notation  $\mathcal{M}_i$  to represent a model obtained after episode  $i$  and write

$$\mathcal{M}_i = \mathcal{M}_{i-1} \oplus \{D_i\} \quad (4.1)$$

to indicate fine-tuning the model  $\mathcal{M}_{i-1}$  with the new dataset  $\{D_i\}$  selected at episode  $i$ , to obtain  $\mathcal{M}_i$ . Moreover, we implement multiple strategies for model update and for the final training episode, as shown in Table 4.1

In the context of continuous training, a critical aspect of performance evaluation is the quantification of training intensity. Two key metrics that provide insight into the computational effort are the total number of training epochs  $\mathcal{E}_{\text{total}}$  and the total number of training step  $\mathcal{N}_{\text{total}}$ .

The total number of training epochs represents the cumulative cycles of training that the model undergoes across all episodes of the active learning process. It is a measure of how many times the model is exposed to the entire dataset. We name it as the **E-granularity** (*i.e.*, the total number of epochs), and it can be expressed as:

$$\mathcal{E}_{\text{total}} = \mathcal{E}_{\text{pre}} + \left(\mathcal{E}_{\text{int}} \cdot \frac{B}{N_r}\right) \quad (4.2)$$

On the other hand, the total number of training steps  $\mathcal{N}_{\text{total}}$ , considers the granularity of the training process. It reflects the total number of individual batches the model is trained on. We name it as the **S-granularity** (*i.e.*, the total number of training steps), and how it's

Strategy	Model Update	Final Training
Initial Model Update	$\mathcal{M}_i = \mathcal{M}_0 \oplus \bigcup_{i=1}^k D_i$	$\mathcal{M}_f = \mathcal{M}_0 \oplus \bigcup_{i=1}^k D_i$
Initial Model Fine-tuning	$\mathcal{M}_i = \mathcal{M}_0 \oplus \{D_i\}$	$\mathcal{M}_f = \mathcal{M}_0 \oplus \{D_k\}$
Initial Model Fine-tuning - Final Training All	$\mathcal{M}_i = \mathcal{M}_0 \oplus \{D_i\}$	$\mathcal{M}_f = \mathcal{M}_0 \oplus \bigcup_{i=1}^k D_i$
Incremental Update	$\mathcal{M}_i = \mathcal{M}_{i-1} \oplus \bigcup_{i=1}^k D_i$	$\mathcal{M}_f = \mathcal{M}_k \oplus \bigcup_{i=1}^k D_i$
Incremental Fine-tuning	$\mathcal{M}_i = \mathcal{M}_{i-1} \oplus \{D_i\}$	$\mathcal{M}_f = \mathcal{M}_k \oplus \{D_k\}$
Incremental Fine-tuning - Final Training All	$\mathcal{M}_i = \mathcal{M}_{i-1} \oplus \{D_i\}$	$\mathcal{M}_f = \mathcal{M}_0 \oplus \bigcup_{i=1}^k D_i$

**Table 4.1:** The table shows 6 model update strategies: **1.** Updating the initial model on the growing set of data. **2.** Fine-tuning the initial model on the latest acquired set. **3.** Fine-tuning the initial model on the latest acquired set, and performing the final training cycle on the entire labeled set. **4.** Updating the latest model using the entire set. **5.** Fine-tuning the latest model on the latest acquired set. **6.** Fine-tuning the latest model on the latest acquired set, and performing the final training cycle on the entire labeled set.

---

#### Algorithm 1 Continuous Training

---

**Require:**  $\mathcal{M}_0$  ▷ Initial model  
**Require:**  $\mathcal{M}_{i-1}$  ▷ Latest model  
**Require:**  $\mathcal{D}_L$  ▷ Labeled set  
**Require:**  $\mathcal{D}_S$  ▷ Selected set

**function** TRAINSTRATEGY( $\mathcal{M}_0, \mathcal{M}_{i-1}, \mathcal{D}_L, \mathcal{D}_S$ )

**if** strategy is Initial Model Update **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_0 \oplus \bigcup_{j=1}^k \mathcal{D}_j$  where  $D_j \in \mathcal{D}_L \cup \mathcal{D}_S$

**else if** strategy is Initial Model Fine-tuning **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_0 \oplus \mathcal{D}_S$

**else if** strategy is Initial Model Fine-tuning and Final Training on All Data **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_0 \oplus \mathcal{D}_S, \quad \mathcal{M}_k \oplus \bigcup_{i=1}^k D_i$

**else if** strategy is Incremental Update **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \oplus \bigcup_{j=1}^k D_j$  where  $D_j \in \mathcal{D}_L$

**else if** strategy is Incremental Fine-tuning **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \oplus \{D_s\}, \quad \mathcal{M}_k \oplus \{D_k\}$

**else if** strategy is Incremental Fine-tuning and Final Training on All Data **then**

$\mathcal{M}_i \leftarrow \mathcal{M}_{i-1} \oplus \{D_s\}, \quad \mathcal{M}_k \oplus \bigcup_{i=1}^k D_i$  ▷ Final training

**end if**

$\mathcal{D}_L \leftarrow \mathcal{D}_L \cup \mathcal{D}_S$  ▷ Update datasets

**return**  $\mathcal{M}_i, \mathcal{D}_L$

**end function**

---

**Algorithm 2** Active Continuous Training**Require:** Query function QUERYFUNCTION**Require:** Training strategy TRAINSTRATEGYInitialize unlabeled dataset  $\mathcal{D}_U$ Initialize labeled dataset  $\mathcal{D}_L$ Initialize new data  $\mathcal{D}_S = \emptyset$ 

▷ Start with an empty new data set

 $\mathcal{M}_0 \leftarrow \mathcal{M}_0 \oplus \mathcal{D}_L$ ▷ Model initial training  $\mathcal{M}_0$ **repeat** $\mathcal{D}_S \leftarrow \text{QUERYFUNCTION}(\mathcal{M}_i, \mathcal{D}_U)$  $\mathcal{D}_S \leftarrow \text{ORACLE}(\mathcal{D}_S)$ 

▷ Query labels from the oracle

 $\mathcal{M}_{i+1}, \mathcal{D}_L \leftarrow \text{TRAINSTRATEGY}(\mathcal{M}_0, \mathcal{M}_i, \mathcal{D}_L, \mathcal{D}_S)$ 

▷ Model Update

**until** stopping criterion is met

expressed depends on the training strategy. For initial or fine-tuned model update, it can be expressed as:

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}} \left( \frac{|D_{\text{pre}}|}{B_s} + i \cdot \frac{N_r}{B_s} \right) \quad (4.3)$$

In case of initial model fine-tuning or incremental fine-tuning (where the final training is performed on the latest acquired data set):

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}} \left( \frac{N_r}{B_s} \right) \quad (4.4)$$

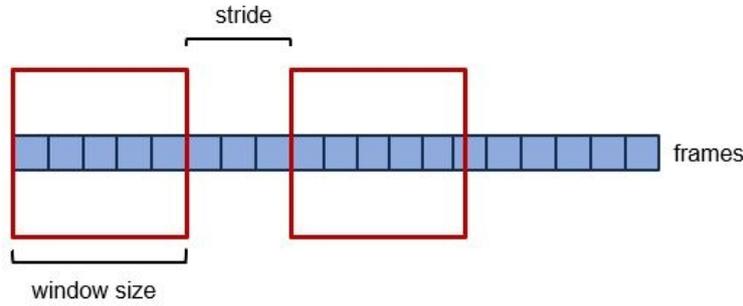
In case of initial model fine-tuning or incremental fine-tuning, where the final training is performed on all the available data:

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}-1} \left( \frac{N_r}{B_s} \right) + E_{\text{int}} \cdot \frac{|D_{\text{pre}}| + B}{B_s} \quad (4.5)$$

- $\mathcal{E}_{\text{pre}}$ : Number of pre-training epochs.
- $\mathcal{E}_{\text{int}}$ : Number of epochs between consecutive sample selections.
- $B$ : Selection Budget
- $B_s$ : Batch size
- $N_r$ : Number of samples queried in each active learning episode.
- $\mathcal{D}_{\text{pre}}$ : The dataset used for pre-training.

## 4.2 Temporal CRB

In [29], the hierarchical query strategy CRB selects the most informative samples from the unlabeled data pool according to the label conciseness, feature representativeness, and geometric balance criteria. However, CRB lacks the ability to select samples that are in a sequential order. This is a case that is particularly useful in 3D point cloud annotation. The human annotator may want to select the next sequence of frames to annotate rather than a random sequence of frames. In this section, we present temporal CRB (tCRB), a modification of the original CRB query strategy that enables sequential sample selection while maintaining sample informativeness. Specifically, we adhere to the hierarchical formulation of CRB and augment each stage separately with a sliding window mechanism to select the most informative sequence of frames.



**Figure 4.2:** Sliding window mechanism.

In the first stage, Concise Label Sampling (CLS), the goal is to select a subset  $\mathcal{D}_{S_1}^*$  of size  $\mathcal{K}_1$  from the unlabeled data pool  $\mathcal{D}_U$ , that minimizes the Kullback-Leiber (KL) divergence between the predicted label probability distribution  $P_{Y_S}$  and the prior label uniform distribution  $P_{Y_U}$ . The KL-divergence is formulated using the Shannon entropy  $H(\cdot)$ .

To this end, we obtain the predictive labels from the detector  $\{\hat{y}_i\}_{i=1}^{N_B}$  for  $N_B$  bounding boxes. The label entropy of the  $i$ -th point cloud can be formulated as:

$$H_{Y_{S_1}}^i = - \sum_{c=1}^C p_{i,c} \log p_{i,c}, \quad p_{i,c} = \frac{e^{y_i^c - C}}{\sum_{c=1}^C e^{y_i^c - C/N_B}}. \quad (4.6)$$

Given this, we first compute the label entropy of all frames in  $H(Y_U)$ . Afterwards, we sort the label entropy values based on the frame timestamp (*i.e.*, temporal sort).

$$H_{Y_U}^{sorted} = \text{sort}(H_{Y_U}, \text{key} = \text{Timestamp}) \quad (4.7)$$

Then applying the sliding window mechanism (see Figure 4.2) to get the sequential  $\mathcal{D}_{S_1}^*$  with the largest sum of label entropy values:

$$D_{S_1}^* = \arg \max_{i \in \{1, 2, 3, \dots, N_w\}} \left( \sum_{j=i}^{i+w-1} H_{Y_S}^j \right) \quad (4.8)$$

Where  $w$ ,  $s$ , and  $N_w$  are the window size, stride, and number of sliding windows, respectively. The number of sliding windows in the current stage can be calculated as:

$$N_w = \left\lfloor \frac{|\mathcal{D}_U| - w}{s} \right\rfloor + 1 \quad (4.9)$$

In the second stage, Representative Prototype Selection (RPS), the goal is to select a subset  $\mathcal{D}_{S_2}^*$  with size  $\mathcal{K}_2$  that covers the unique knowledge encoded only in  $\mathcal{D}_U$  and not in  $\mathcal{D}_L$ . This is achieved by utilizing the feature representativeness of the gradient vectors, where magnitude and orientation represent the uncertainty and diversity in the gradient space, respectively. After obtaining the gradient maps  $\mathcal{G}_{S_1}$  from the fully connected layers, we sort them based on the timestamps, similarly to label entropy values at stage 1.

$$\mathcal{G}_{S_1}^{sorted} = \text{sort}(\mathcal{G}_{S_1}, \text{key} = \text{Timestamp}) \quad (4.10)$$

Re-formulating the problem using the sliding window mechanism. For each window  $i$ , we compute the weighted mean gradient  $\mathbf{M}_i$  to represent the dominant learning direction of the window, where the norm of gradient  $\mathbf{G}_i$  is considered as its weight:

$$\mathbf{M}_i = \frac{\sum_{j=1}^n \|\mathbf{G}_j\| \mathbf{G}_j}{\sum_{j=1}^n \|\mathbf{G}_j\|}, \quad \text{where } n \in [1, w] \quad (4.11)$$

Afterwards, we compute the Euclidean distance  $\mathbf{E}_i$  and the cosine similarity  $\mathbf{S}_i$  of each gradient  $\mathbf{G}_i$  in the window to  $\mathbf{M}_i$ :

$$\mathbf{E}_i = \sum_{j=1}^n \|\mathbf{G}_j - \mathbf{M}_i\|, \quad \mathbf{C}_{simi} = \sum_{j=1}^n \frac{\mathbf{G}_j \cdot \mathbf{M}_i}{\|\mathbf{G}_j\| \|\mathbf{M}_i\|}, \quad \text{where } n \in [1, w] \quad (4.12)$$

We then formulate a combined score  $\mathcal{CS}_i$  for gradient informativeness of window  $i$ , as the difference between  $\mathbf{E}_i$  and  $\mathbf{C}_{simi}$ , and select the window with largest combined score:

$$D_{S_2}^* = \arg \max_{i \in \{1, 2, 3, \dots, N_w\}} (\mathbf{E}_i - \mathbf{C}_{simi}) \quad (4.13)$$

Where the number of sliding windows in the current stage is:

$$N_w = \left\lfloor \frac{\mathcal{K}_1 - w}{s} \right\rfloor + 1 \quad (4.14)$$

Algorithm 3 provides an algorithmic illustration of the temporal modifications at stage 2. Algorithm 3 has a time complexity of  $O(N_w \cdot W)$  and space complexity of  $(O(\mathcal{K}_1 + N_w))$ . Where  $W$  and  $N_w$  are the window size and number of windows, respectively.

The rationale for using Euclidean distance and cosine similarity is to capture the uncertainty and diversity of the gradient information, respectively.

- The Euclidean distance between each gradient and the weighted mean gradient quantifies the magnitude of gradient deviation from the mean. Larger distances indicate gradients significantly different from the average learning trend, highlighting areas where the model is more uncertain.
- The cosine similarity measures the angular difference between two vectors, representing the orientation of a gradient with respect to the average learning direction. Lower cosine similarities (closer to -1) suggest various features or patterns that the model is learning.
- The combined score considers both magnitude and orientation by subtracting the cosine similarity from the Euclidean distance as we seek high magnitudes (higher uncertainties) and low cosine similarities (diverse gradient orientations).
- The window with the largest combined score is selected, as it most likely represents the region with the most informative and diverse patterns for the model to learn.

In the third stage, Greedy Point Density Balancing (GPDS), the goal is to select a subset  $\mathcal{D}_{S_{final}}^*$  with size  $\mathcal{K}_3$  from the latest selected set  $\mathcal{D}_{S_2}^*$ , that best aligns its geometric characteristics with the geometric characteristics of the unlabeled data pool  $\mathcal{D}_U$ . The point density within each bounding box is utilized to preserve the geometric characteristics of an object in 3D point clouds. The point density distribution of the unlabeled pool is not observed and is assumed to follow a uniform distribution. Besides, the point density distribution of  $\mathcal{D}_{S_2}^*$  is not provided and is to be estimated from the bounding box predictions of stage 1 using Kernel Density Estimation (KDE). The problem is then formulated as the KL-divergence between the estimated point density distribution and the uniform distribution, with the objective of selecting the subset that minimizes the KL-divergence.

To this end, after estimating the point density distribution  $\mathcal{P}_{\mathcal{D}_{S_2}}$  for all point clouds in  $\mathcal{D}_{S_2}$ , we sort it by the timestamp, then apply the sliding window mechanism to select the window that has the least sum of KL-divergence values for its point clouds:

$$\mathcal{D}_{S_{final}}^* = \underset{i \in \{1, 2, 3, \dots, N_w\}}{\operatorname{arg\,min}} \left( \sum_{j=i}^{i+w-1} \mathcal{P}_{\mathcal{D}_{S_2}}^j \right) \quad \text{where} \quad N_w = \left\lfloor \frac{\mathcal{K}_2 - w}{s} \right\rfloor + 1 \quad (4.15)$$

---

**Algorithm 3** Temporal RPS
 

---

**Require:** Window size  $W$

**Require:** Stride  $S$

**Require:** Sorted gradient map  $\mathcal{G}_{sorted}$

Initialize the number of windows  $N_w \leftarrow \left\lfloor \frac{\mathcal{K}_1 - W}{S} \right\rfloor + 1$  ▷ Total number of windows

Initialize the best score  $\mathcal{CS}^* \leftarrow -\infty$

Initialize the best window index  $i^* \leftarrow -1$

**for**  $i \leftarrow 1$  to  $N_w$  **do**

Compute the weighted mean gradient  $M_i$  for window  $i$

Compute the Euclidean distance sum  $E_i$  and cosine similarity sum  $S_i$  for window  $i$

Compute the combined score  $\mathcal{CS}_i \leftarrow E_i - C_{sim_i}$

**if**  $\mathcal{CS}_i > \mathcal{CS}^*$  **then**

$\mathcal{CS}^* \leftarrow \mathcal{CS}_i, \quad i^* \leftarrow i$

**end if**

**end for**

**return** Best window starting index  $i^*$  and score  $\mathcal{CS}^*$

---

### 4.3 Active Class Weighting

Class imbalance is a common issue in many datasets, particularly in 3D datasets in the context of autonomous vehicles (AV) and mobile robots, where classes usually exhibit a long-tailed distribution. Training on such datasets can bias the model towards over-represented classes rather than under-represented ones. Active Learning (AL) is one promising approach to counter the class imbalance issue by querying samples that are expected to balance the training data.

In the first stage of the CRB framework [29], Concise Label Sampling, the goal is to select several samples that best align the predicted label distribution with the prior uniform distribution of the unlabeled pool. By assuming a uniform label distribution of the unlabeled data, the query strategy is expected to select data samples that balance the overall training data. However, there are multiple reasons why this might not work as expected:

- Mismatch between the prior assumption and the actual data distribution.
- When the selection process is based on the prior uniform assumption, it might end up unintentionally favoring frames that contain more common classes. Because of the long-tailed nature of the dataset, the chance of randomly encountering a point cloud that tends to have a uniform label distribution is high if it includes a high number of common classes.

We propose a simple modification to address this observation. We re-formulate the entropy as the weighted entropy, where the weights are the inverse of class counts in the current training dataset.

The weighted Shannon entropy computation:

$$H_w(Y_{S_1}) = - \sum_{c=1}^C w_c \cdot p_{i,c} \log p_{i,c}, \quad \text{where} \quad p_{i,c} = \frac{e^{|y_i=c|/N_B}}{\sum_{c'=1}^C e^{|y_i=c'|/N_B}} \quad (4.16)$$

Where  $w_c$  is the weight of class  $c$ :  $w_c = \frac{\frac{1}{f_c}}{\sum_{j=1}^C \frac{1}{f_j}}$

The class weights are updated in every active learning episode to reflect the class counts in the growing set of labeled data.

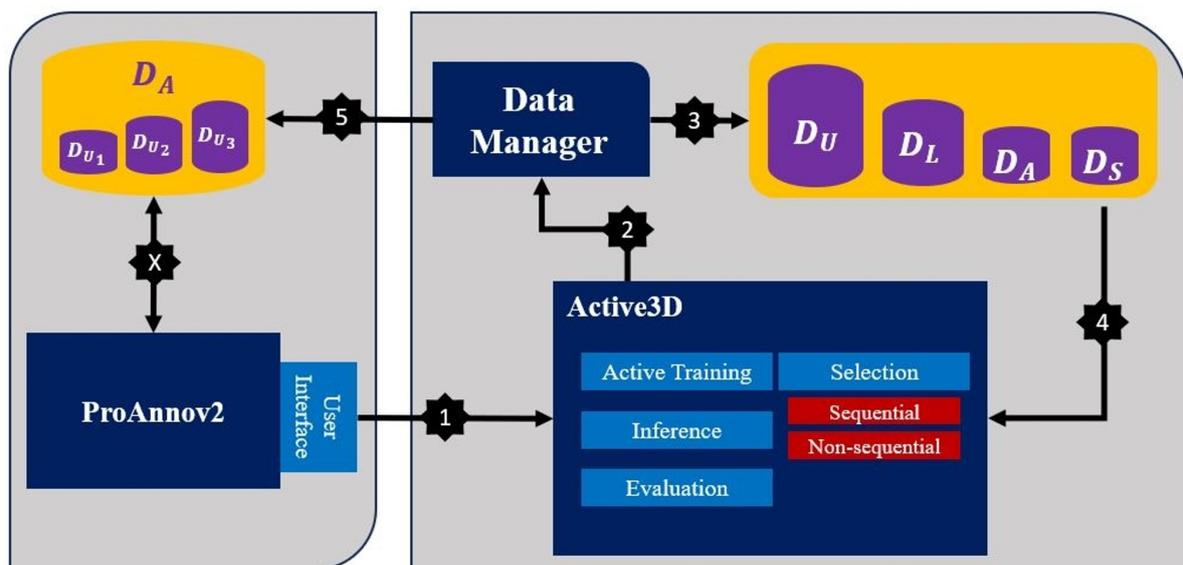
Normalization of class weights is critical. Without normalization, class weights could disproportionately influence the entropy computation.

By multiplying the entropy by the updated class weights, we assume to adjust the selection process to favor frames that not only approximate a uniform label distribution but also emphasize under-represented classes and can, therefore lead to a more balanced training data.

## 4.4 Providentia Annotation 2.0

Providentia Annotation (*ProAnno*), an advanced web-based 3D annotation platform, developed on the foundation of the 3D-BAT [65] tool, extending its functionalities and capabilities for efficiently annotating 3D point clouds and 2D images in the context of autonomous vehicles (AV) and mobility infrastructure. We further integrate it with an active learning pipeline, Active3D, to address the need for AI-assisted features to increase the efficiency of the annotation process. By leveraging active learning, the tool reduces the manual workload and enhances the annotation capabilities, ensuring that the data used to train machine learning models is of the highest informativeness and contributes the most to the models' performance gains. The synergy between human expertise and AI-guided learning is the core of the tool's methodological improvements. Additionally, we addressed multiple usability issues and developed other features for an overall enhanced workflow.

In our setup, the *ProAnno* tool resides on one machine and interfaces with Active3D, which is located on a separate server. The inter-machine communication follows a client-server architecture which is established through Flask, the backbone of RESTful API. *ProAnno* acts as the client, while Active3D acts as the server. *ProAnno* initiates operations and requests results from Active3D. The client-server model is a fundamental paradigm in networked systems, allowing distributed components to work together smoothly.



**Figure 4.3:** Conceptual diagram of ProAnno++ workflow. It shows two boxes representing the client and the server sides, which are connected by means of the Python Flask library.

The upgraded tool, proAnno++, incorporates the following key features:

1. **Active Training and Sample Selection:** In the Active3D pipeline, a PV-RCNN [38] model is integrated for active training. This involves training the model with already labeled data and then using it to select new samples for labeling. Active3D allows for different training strategies, as detailed in section 4.1. Furthermore, users have the option to choose between labeling sequential or non-sequential sets of frames. In the case of non-sequential frames, Active3D offers multiple options for the query strategy: CRB [29], Entropy [51], BADGE [3], LLAL [59], CoreSet [37], Confidence sampling,

Monte-Carlo sampling, and Random sampling. In the case of sequential frames, Active3D offers only tCRB sampling.

2. **Auto-Labeling:** To generate 3D bounding boxes, the PV-RCNN model is utilized to infer the frames currently annotated. The predicted bounding boxes are further processed to remove redundant and overlapping boxes and keep the best ones. Users can choose between auto-labeling a single frame or multiple frames by specifying the range of frames to auto-label.
3. **Comparative Evaluation:** Another simple yet beneficial feature is comparative evaluation, where Active3D evaluates the model’s generated labels against the ground-truth labels created by the human expert for the same set of frames. We believe this helps validate the effectiveness of AI-assisted annotation and provides insights into the learning process of the 3D detector model. We assume that with increasing active learning and labeling cycles, the model’s performance will converge to an accepted threshold regarding the quality of the generated 3D bounding boxes.
4. **Enhanced Usability:** *proAnno++* now includes advanced usability features that facilitate a more detailed and user-friendly annotation process. Users can zoom into images for finer detail work, which helps the annotator identify and locate distance objects that are not clearly recognizable in the point cloud. Additional helper views, such as Bird’s Eye View (BEV) and side and front views of the selected bounding box, provide more context when annotating objects. The tool also offers a navigation functionality, allowing users to easily switch between different datasets and sequences within a dataset for a more streamlined workflow. Furthermore, *proAnno++* supports the ASAM OpenLABEL annotation standard, ensuring compatibility with a broad range of datasets.

In the following, we provide a detailed explanation of the workflow and user guidelines for an efficient annotation process.

We first lay down the necessary notations, where  $D_U$  is the unlabeled set,  $D_L$  is the labeled set,  $D_A$  is the set currently undergoing annotation,  $D_S$  is the latest selected set by the AL query strategy. Shown in Figure 4.3 is a high-level, conceptual representation of the workflow, which can be broken down into the following components:

- **User Command Interpretation:** Initially, the user (*i.e.*, human annotator) decides on a specific annotation action. The Flask application then bridges the *proAnno* tool with the Active3D pipeline, which receives the user’s command and adjusts the necessary settings for the operation. The available user options in this context are:
  1. Auto-label the current frame
  2. Auto-label a range of frames
  3. Auto-label all frames
  4. Evaluate the current frame
  5. Evaluate a range of frames
  6. Evaluate all frames
  7. Active train on all labeled data
  8. Active train on latest labeled data
  9. Select only
  10. Train only
- **Data Preparation:** Upon receiving instructions, the Data Manager preserves the current data structure in a Pickle file. Then it constructs a new data set as required by the user’s operation. For instance, if the user’s command is *Active train on all labeled data*, then the data manager will construct a ‘training’ set composed of  $D_L$  and  $D_A$ . If the command is *active train on latest labeled data*, then the constructed ‘train’ set will be composed of  $D_A$  only.

- **Operation Execution:** Active3D then initiates the commanded operation, whether it's active training, inference (*i.e.*, auto-labeling), or evaluation.
- **Post-Processing:** Following the successful execution of the operation, the Data Manager performs post-processing tasks to finalize the data structure and send the necessary information to proAnno for synchronization. For example, after the execution of *Active train* operation, the Data Manager re-structures the  $D_U$  such that the selected samples are excluded (*i.e.*,  $D_U = D_U - D_S$ ) and construct  $D_S$ , which is then set to proAnno for labeling.

The Data Manager Class, shown in Figure 4.4, is a critical component of the workflow. It manages the data splits, converts the data from OpenLABEL to KITTI format, keeps track of the data split history, and performs post-processing.

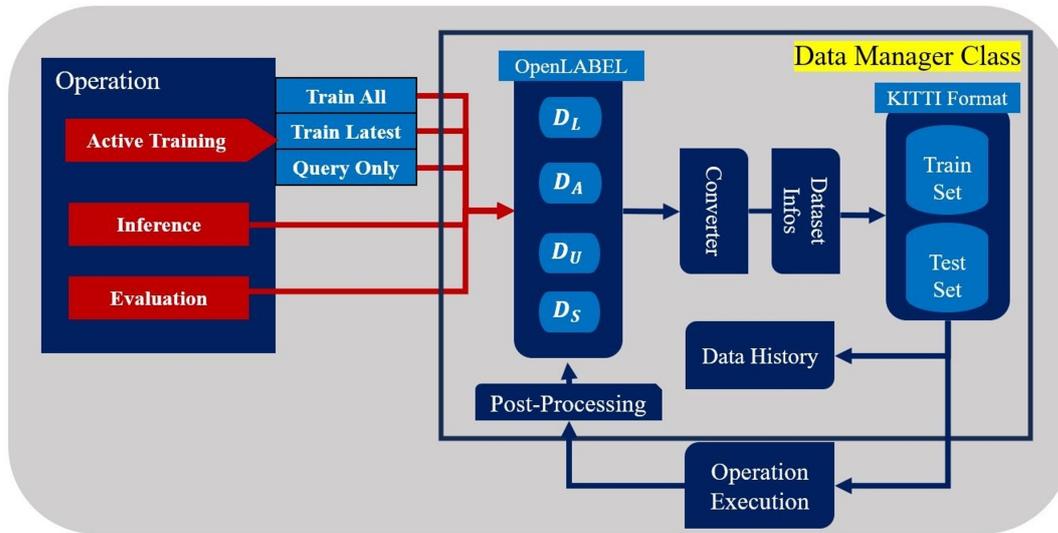


Figure 4.4: An illustrative diagram of the Data Manager Class.

The following Table 4.2 shows the user command options and their corresponding data splits:

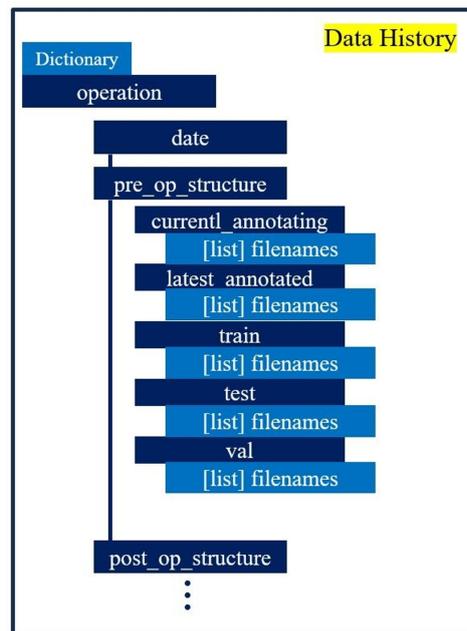
User Command	Train Split	Test Split
Auto-Label	—	$D_U$
Evaluation	—	$D_U$
Active train all data	$D_L = D_L \cup D_A$	$D_U$
Active train latest data	$D_L = D_A$	$D_U$
Select Only	—	$D_U$
Train only	$D_L$	—

**Table 4.2:** Train and test splits corresponding to the operation commanded by the user. The *auto-label*, *evaluation*, and *select only* commands do not require a training set, and only operate on the test set. While the *train only* command does not require a test set.

Post-processing is necessary, particularly in scenarios where the efficiency and smoothness of the workflow are of high priority. In the proAnno++ workflow, different operations require different post-processing functionalities:

- **Active Training:** After active training, the Data Manager Class updates the unlabeled data  $D_U$  by splitting the set of selected samples from it. It also saves the model checkpoint to benefit from the training process and use the updated model for the next training cycle.
- **Auto-Labeling:** The PV-RCNN model performs inference on the test split, generating predicted 3D bounding boxes. The bounding boxes then go through a refining process where overlapping boxes are excluded and the highest scoring boxes only are selected. Besides, boxes containing a number of points below a pre-defined threshold are also excluded, possibly reducing the number of false positives.
- **Evaluation:** The mAP metric is used to evaluate the generated auto-labels versus the human adjustments of these labels. Afterward, the Data Manager Class saves the evaluation files and logs for further investigations.

The data history component within the Data Manager Class acts as a tracking mechanism. It systematically logs each operation along with the relevant data splits, ensuring a robust and recoverable process. This functionality is key for instances where an operation might need to be reversed or a data split requires restoration due to an error. It provides a record of the data manipulation through different stages or operations such as labeling, training, testing or evaluation.



**Figure 4.5:** An illustrative diagram of the Data Manager Class. Data History is a Python Dictionary instance. It saves the date of the operation and the data file configurations before and after the operation is executed. The data file configuration is saved as a Python List of file names corresponding to each data split.

Currently, our system operates on a rigid setup where the data file structure must be identical on both the proAnno and Active3D ends, as shown in Figures 4.6 and 4.6. The hard-coded

configuration requires a uniform file organization across both machines for smooth operation and data exchange. This approach, however, lacks flexibility and is not optimal for the evolving requirements of the annotation process.

It is important to mention that the point cloud and annotation file names should be based only on the timestamps, as shown in Figure 4.7. e.g., 1688626890\_040199717.json, and 1688626890\_040199717.pcd

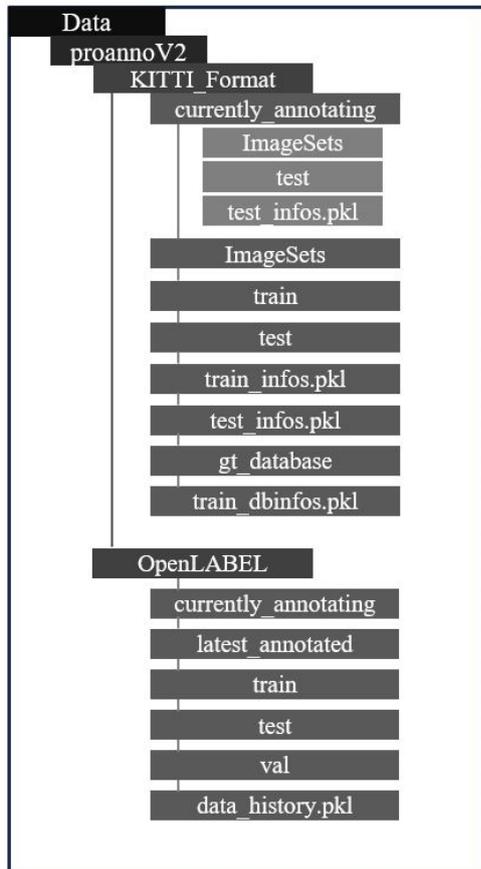


Figure 4.6: Data file structure on the Active3D end.

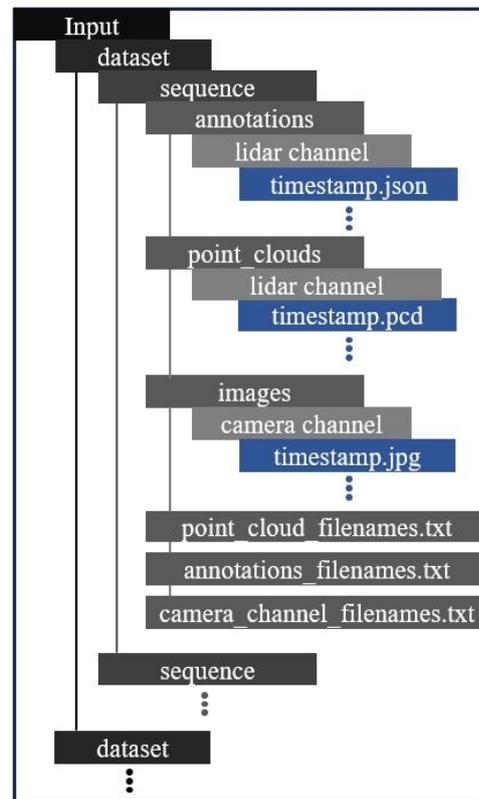
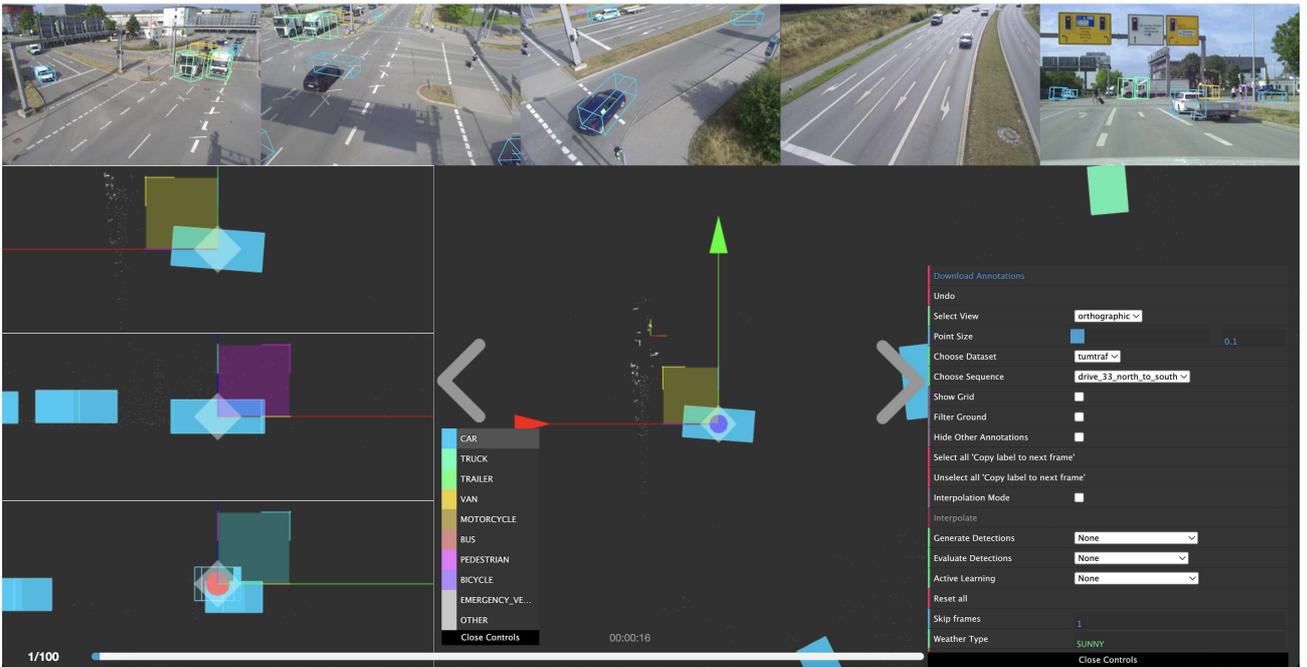


Figure 4.7: Data file structure on the proAnno end.

Finally, proAnno++ showcases multiple modifications of usability features, each designed to streamline the annotation process and enrich the user experience. Figure 4.8 shows these upgrades, which are detailed as follows:

1. **Dynamic Image Display:** The tool dynamically adjusts to the number of camera channels available in the current sequence. It computes the size of the image display windows by dividing the available display width by the number of camera channels, ensuring that each image is given optimal screen space.
2. **Zoom Functionality:** A zoom feature is activated at the cursor's location, allowing annotators to closely inspect images. This function is vital for recognizing objects too distant to be recognized in point clouds.
3. **Helper Views:** Although already implemented in the older 3D-BAT tool, the helper views functionality was not correctly implemented in proAnno. We implement and integrate it well, reducing the need to constantly switch to perspective view mode by showing multiple angles of the selected bounding box.

4. **Label Copying Fix:** A previously encountered issue where a label would not be copied to the subsequent frame without moving two steps (either forward or forward and backward). We fix this issue and ensure users can smoothly copy labels to subsequent frames.
5. **Dataset/Sequence Transition Fix:** Previously implemented features for switching between datasets or data sequences were not functioning properly. We fix this and ensure a smooth transition and data file loading between different data sets.



**Figure 4.8:** The proAnno++ interface showcases five camera images alongside a point cloud frame, offering a range of GUI options. These include switching between orthographic and perspective views, navigating through sequences and datasets, copying labels to subsequent frames, auto-generating labels, and adjusting individual objects. Additionally, when an object is selected, helper views on the left side of the UI provide visualization from three different angles, reducing the need to switch to a perspective view. This comprehensive approach enhances user interaction and efficiency in object annotation.

# Chapter 5

## Evaluation

In this chapter, we focus on evaluating and analyzing the results of our study. We have applied the methods described earlier to collect evaluation data, which is essential for testing our hypotheses and adding to the existing knowledge in our field. Specifically, we compare various query strategies in active training, including our newly developed tCRB strategy and a modified version of CRB that incorporates active class weighting. Additionally, we examine different continuous training methods and their impact on the active training process, identifying the most efficient continuous training method in terms of computational demand and detection performance. Furthermore, we conduct ablation studies to compare CRB with tCRB and CRB with CRB plus dynamic class weighting. Moreover, we present several qualitative results to showcase our findings with representative visualizations and enrich our discussion. In summary, this chapter systematically presents our findings and observations, providing a foundation for our conclusions and suggestions and aiming to offer valuable insights that could inspire more research and practical applications.

### 5.1 Experimental Setup

To evaluate the performance of the proposed methods, the experiments were primarily conducted using the following standardized configurations.

#### 5.1.1 Baseline Architecture

For the baseline architecture in our experiments, we employed the PV-RCNN [38] model as implemented in the open-source library OpenPCDet [47].

- **Point Cloud Range:**

$$X\text{-axis} \in [-75.2, 75.2], \quad Y\text{-axis} \in [-75.2, 75.2], \quad Z\text{-axis} \in [-8, 0] \quad (5.1)$$

- **Point Cloud Features:** The utilized features are the X, Y, and Z coordinates, along with the reflectance intensity.
- **Voxel and Grid Size:**

$$V_x = 0.1, \quad V_y = 0.1, \quad V_z = 0.15 \quad (5.2)$$

$$H = \left\lceil \frac{(y_{\max} - y_{\min})}{V_y} \right\rceil, \quad W = \left\lceil \frac{(x_{\max} - x_{\min})}{V_x} \right\rceil, \quad D = \left\lceil \frac{(z_{\max} - z_{\min})}{V_z} \right\rceil \quad (5.3)$$

- **Voxel Feature Encoder (VFE):** The mean feature of all points within each voxel is calculated as the representative feature of the voxel.
- **3D Backbone:** We utilize the *VoxelBackBone8x* module, which is a 3D sparse convolutional network featuring a down-sampling factor of 8.
- **Mapping to BEV:** We utilize the *HeightCompression* module to map the voxel-based 3D features into a 2D BEV grid with 256 features.
- **Point Feature Encoding (PFE):** We utilize the *VoxelSetAbstraction* module with 2048 key points to sample, *Furthest-Point Sampling* as the sampling method, 128 output features, and the feature sources are the BEV map, all the 3D convolutional layers, and raw points.
- **2D Backbone:** We utilize the *BaseBEVBackbone* module for 2D feature learning from the BEV map.
- **Dense Head:** We utilize the *AnchorHeadSingle* module as the detection head module.
- **Point Head** We employ the *PointHeadSimple* module for key-point classification, with *smooth-l1* as the regression loss.
- **RoI Head:** We employ the *PVRCNNHead* module for the refinement of predictions.
- **Post-Processing:** We define three Recall thresholds: 0.3, 0.5, 0.7. A score threshold of 0.1 and post-NMS size of 500 boxes.
- **Optimization:** We utilize the Adam optimizer [28], with a learning rate of 0.01, weight decay rate of 0.01, and momentum of 0.9. Besides, we set the batch size to 4 for both train and test splits.
- **Hardware:** All experiments are conducted on a single RTX 4090 GPU.

### 5.1.2 Active Learning Framework

For active learning and training experiments, we adopt the active learning for the 3D object detection pipeline [29], which utilizes the OpenPCDet framework. Our experiments are conducted on the TUMTraF *train* split consisting of 1920 samples. We randomly sample 5% - 100 samples to create the initial labeled set  $D_{pre}$ . We pre-train the PV-RCNN model on  $D_{pre}$  for 30 epochs. During the pre-training phase, we save a model checkpoint every 10 epochs and select the last checkpoint for the active training cycle.

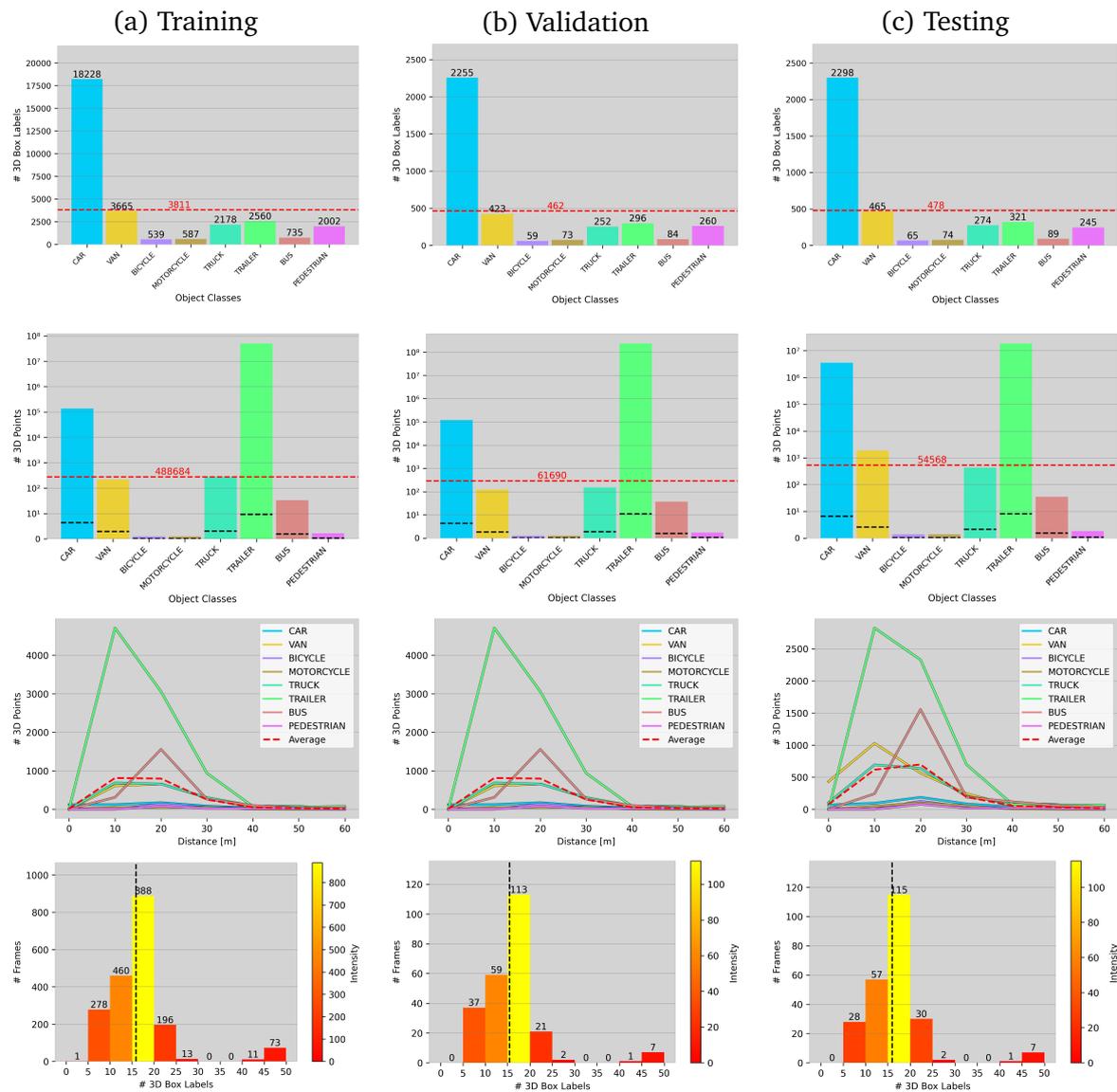
In the active training cycle, the rest of the *train* split is considered the unlabeled data pool for evaluating the informativeness of the data samples. The labels of the unlabeled data pool are assumed to be unknown, and no prior assumption is made on the informativeness of the data samples. For robust evaluation, we standardize the active training hyper-parameters across all experiments. We set the pre-training duration for 30 epochs, and the model is trained for 20 epochs between subsequent cycles of sample selection. We set the selection budget to 50% of the *train* split after splitting  $D_{pre}$ . This means a selection budget of 900 samples; we select 100 samples every active training cycle. Continuous training within the active learning cycle considers two key decisions: Firstly, the choice of the model for the upcoming active training cycle, whether to continue with the tuned model from the most recent cycle or revert to the initially pre-trained model. Secondly, the choice of the data set for the next cycle, considering either the most recently selected data only or the growing set of data, including all

previously and currently selected data as well as the initial labeled set. For the query strategy, we compare between CRB, tCRB (*i.e.*, temporal CRB), wCRB (*i.e.*, weighted CRB), BADGE, CoreSet, Monte-Carlo, Confidence sampling, Entropy sampling, and Random Sampling.

To set a standard for comparison, the PV-RCNN model is trained on the complete training split for a duration of 80 epochs, using a batch size of 4. This model is referred to as the "oracle" model. Its performance is a benchmark against which we evaluate the effectiveness of models trained through active learning methods.

### 5.1.3 Dataset

Our experiments are carried out using the TUMTraF Intersection dataset, which comprises a total of 2,400 labeled frames. These samples are divided into training, validation, and testing splits, constituting 80%, 10%, and 10% of the original data, respectively.



**Figure 5.1:** This figure displays the division of TUMTraF Intersection dataset into training, validation, and test splits. It features the distribution of object classes, 3D point distribution, class density per distance, and frame frequency histograms for each set.

### 5.1.4 Evaluation Metrics

- **Mean Average Precision - mAP:** In 3D object detection, precision and recall are two fundamental metrics used to evaluate the performance of a model. Precision measures the proportion of correctly identified positive instances among all instances that the model classified as positive, while Recall measures the proportion of correctly identified positive instances out of all ground-truth positive instances in the dataset:

$$\text{Precision (P)} = \frac{TP}{TP + FP}, \quad \text{Recall (R)} = \frac{TP}{TP + FN} \quad (5.4)$$

Mean Average Precision (mAP) extends these concepts by computing the average precision across all classes and/or over different levels of recall. The precision-recall curve is plotted for each class by computing precision and recall at various threshold levels. The average precision is then calculated as the area under the precision-recall curve. The mAP is the mean of these AP values over all classes. Formally, having  $C$  classes, then:

$$mAP = \frac{1}{C} \sum_{c=1}^C AP_c \quad (5.5)$$

We calculate the mAP across different difficulty levels. These levels are stratified based on the distance of the detected objects from the sensor. *Easy* detections within - to 30 meters, *Moderate* detections from 30 to 50 meters, and *Hard* detections beyond 50 meters. Then, the overall mAP is essentially the weighted average of the mAP at all difficulty levels, where the weights are the number of samples per difficulty level:

$$mAP_{\text{overall}} = \frac{N_{\text{easy}} \cdot mAP_{\text{easy}} + N_{\text{moderate}} \cdot mAP_{\text{moderate}} + N_{\text{hard}} \cdot mAP_{\text{hard}}}{N_{\text{total}}} \quad (5.6)$$

- **E- and S-granularity:** The E-granularity metric represents the total number of training epochs the model undergoes throughout the active learning process:

$$\mathcal{E}_{\text{total}} = \mathcal{E}_{\text{pre}} + \left( \mathcal{E}_{\text{int}} \cdot \frac{B}{N_r} \right) \quad (5.7)$$

The S-granularity metric represents the total count of training steps, indicating how many individual batches the model has been trained on. The S-granularity has 3 versions, as discussed in Section 4.1. For incremental and initial model updates with final training on all data:

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}} \left( \frac{|D_{\text{pre}}|}{B_s} + i \cdot \frac{N_r}{B_s} \right) \quad (5.8)$$

For incremental and initial model fine-tuning:

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}} \left( \frac{N_r}{B_s} \right) \quad (5.9)$$

Finally, for incremental and initial model fine-tuning with final training on all data:

$$\mathcal{N}_{\text{tot}} = \frac{|D_{\text{pre}}|}{B_s} \cdot E_{\text{pre}} + E_{\text{int}} \sum_{i=1}^{\frac{B}{N_r}-1} \left( \frac{N_r}{B} \right) + E_{\text{int}} \cdot \frac{|D_{\text{pre}}| + B}{B_s} \quad (5.10)$$

In active learning scenarios, E-granularity and S-granularity metrics provide essential insight for evaluating the training strategy’s computational efficiency and impact. These metrics provide insights into the model’s utilization of resources and the potential need for optimization, particularly if the model requires extensive training to achieve acceptable performance gain. By comparing these metrics with the mAP, it’s possible to find an equilibrium between achieving high model performance and maintaining reasonable computational resources and training times.

- **3D Bounding Box Labeling Efficiency Metric:** This metric is the total number of user clicks required to create one valid 3D bounding box on the point cloud. A *click* is considered any intentional user action contributing to creating, adjusting, or finalizing a single 3D bounding box. This includes initiating the bounding box, adjusting its dimensions, location or orientation, and setting its attributes. The efficiency is inversely proportional to the number of clicks: fewer clicks indicate higher efficiency in the labeling process.

## 5.2 Quantitative Results

### 5.2.1 Continuous Training Strategies

We start our experimentation with the integration of continuous training methodologies within the active learning cycle, focusing exclusively on the CRB acquisition function, as it serves as the baseline for our experiments. To facilitate a more understandable experimental framework, we have translated the names of the training strategies as listed in Table 4.1 into more intuitive names:

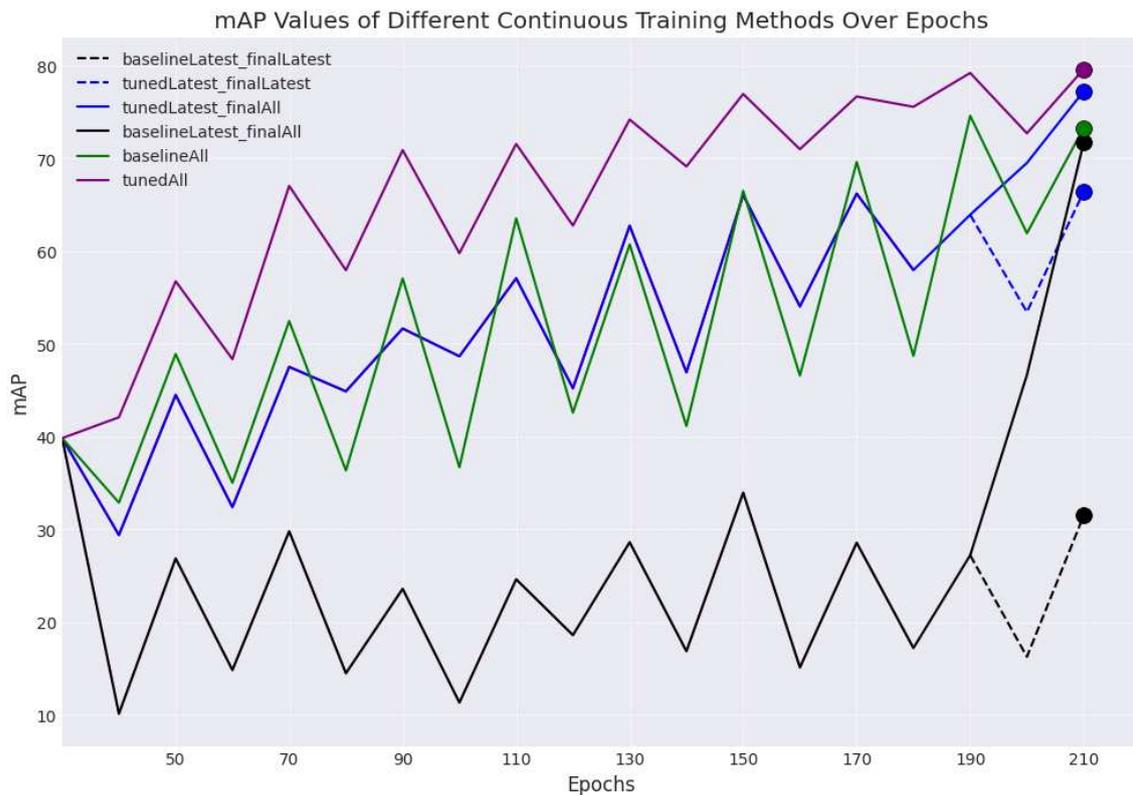
Strategy Name	Experimental Name
Initial Model Update	baselineAll
Initial Model Fine-tuning	baselineLatest_finalLatest
Initial Model Fine-tuning - Final Training All	baselineLatest_finalAll
Incremental Update	tunedAll
Incremental Fine-tuning	tunedLatest_finalLatest
Incremental Fine-tuning - Final Training All	tunedLatest_finalAll

**Table 5.1:** Mapping the formal strategy names to intuitive experimental names.

There are two key objectives of this experiment. Firstly, to identify which training strategy achieves the highest mAP over the same number of epochs, which correspond to the same selection budget. Given a fixed computational budget, this enables us to determine the most effective strategy for model training. Secondly, it provides insights into the evolution of the model’s learning process across different model updates and data compositions. By observing how the mAP changes over epochs, we can infer how each training strategy copes with introducing new data, reflecting the model’s ability to learn and generalize from the active learning queries.

## 1. Model Update:

- Strategies that use the initial model for updates tend to show a pattern where performance drops significantly by the beginning of every active learning cycle. This pattern is clear in the **baselineAll**, **baselineLatest\_finalAll**, and **baselineLatest\_finalLatest** strategies. The severe oscillations in these strategies' learning processes highlight that the model is not leveraging the iterative learning process. This is reasonable due to using the initial model for model updating. Nonetheless, the data composition and the final training cycle can lead to a good mAP performance at the end of the process, as evident in **baselineAll** and **baselineLatest\_finalAll**.
- On the other hand, strategies that fine-tune the latest model, such as **tunedAll**, **tunedLatest\_finalAll** and **tunedLatest\_finalLatest**, still experience strong oscillations in performance, especially in the earlier training cycles. This could happen because, in the initial cycles, the model learns from smaller data sets and is then exposed to new knowledge in later cycles. It might also be due to the model forgetting some of the knowledge it learned in earlier cycles, a phenomenon known as *catastrophic forgetting*. However, regardless of the data composition, these strategies tend to converge to a high mAP score. Additionally, performing the final training cycle with all the available data further improves the mAP score.



**Figure 5.2:** The plot shows the performance of six continuous training strategies in terms of mean Average Precision (mAP) over the number of epochs. The x-axis represents the number of epochs, starting at 30 due to initial pre-training, and the y-axis represents the overall mAP.

## 2. Data Composition:

- Strategies that incorporate all the available data for training, such as **baselineAll**

and **tunedAll** consistently show a higher mAP score compared to other strategies, regardless of how the model is updated. This suggests that the model’s ability to generalize improves with an increase in training data. A concept that is commonly agreed upon in the research community.

- On the other hand, strategies that incorporate only the most recently selected data for training, such as **baselineLatest** and **tunedLatest**, typically fail to achieve a high mAP score and need to be exposed to the entire data set in the final training stage for a performance boost. This could be because the latest selected data is not comprehensive enough for the model to develop a robust generalization ability. Coupling this observation with the phenomenon of catastrophic forgetting—in the case of **tunedLatest**— eventually will not lead to an acceptable generalization ability.
3. **Final Training Cycle:** The impact of conducting the final training cycle on all available data is evident in the **tunedLatest** and **baselineLatest** strategies. This suggests that allowing the model to learn from the entire dataset at the end can make up for any shortcomings from earlier training cycles. Essentially, this comprehensive exposure to data helps the model correct errors it may have made previously, leading to an overall improvement in performance.

In conclusion, the combination of fine-tuning approaches and comprehensive data utilization, especially when applied in the final training phase, appears to be the most effective in achieving high model performance. The **tunedAll** strategy stands out, as it benefits from continual fine-tuning and comprehensive data exposure, leading to high final mAP and low variability throughout the training epochs.

Table 5.2 presents the S-granularity values of different training strategies alongside their corresponding mAP scores. S-granularity offers insight into the computational effort involved in each strategy, which helps evaluate the trade-off between the employed computational resources and the model performance.

Strategy	S-granularity	overall mAP
baselineAll	27,750	73.31
baselineLatest_finalLatest	5,250	31.47
baselineLatest_finalAll	9,750	71.72
tunedAll	27,750	79.66
tunedLatest_finalLatest	5,250	66.43
tunedLatest_finalAll	9,750	77.28
oracle	38,400	83.50

**Table 5.2:** The S-granularity value and mAP score of all training strategies.

We include the **oracle** model in this table to benchmark the computational demand and performance of various training strategies. Notably, the **tunedAll** strategy, which is the best-performing continuous training strategy, has a lower S-granularity of 27,750 compared to the oracle’s 38,400. This clearly shows that **tunedAll** is less computationally intensive while still

achieving a mAP score (79.65) quite close to the oracle’s score of 83.50. This finding challenges the initial assumption that active learning, with its iterative nature, might be more computationally demanding. It suggests that active learning can be efficient while maintaining high performance.

Furthermore, the **tunedLatest\_finalAll** strategy stands out for its significantly lower S-granularity of 9,750, significantly lower than that of other strategies and the oracle. This highlights the strategy’s efficiency, achieving a notable mAP score with considerably fewer training steps than the oracle and other strategies. This could be particularly relevant in scenarios where computational resources are limited, yet a high mAP score is still required. These observations highlight the potential of active learning strategies, primarily **tunedAll**, to balance computational efficiency and model effectiveness.

## 5.2.2 Active Learning Strategies

Next, we conduct experiments with various active learning approaches, including the standard CRB baseline and our newly developed version, tCRB. We provide a comparative evaluation of these active learning approaches against the benchmark oracle performance. Building on insights from the previous experiment, we adopt the **tunedAll** continuous training strategy to train the PV-RCNN model within the active learning cycle. Table 5.3 presents the mAP scores of the active learning approaches at different levels based on the distance of the objects to the LiDAR sensor. Distances from 0 to 30 meters are classified as *easy*, distances from 30 to 50 meters are classified as *moderate*, and distances further than 50 meters are classified as *hard*. The key objective of the experiment is to highlight the best performances of the active learning approaches versus that of the oracle model across the three difficulty levels.

Strategy	Easy	Moderate	Hard	Overall
CRB	79.347	84.843	45.186	79.657
tCRB	68.870	80.464	33.077	72.653
CoreSet	68.014	80.971	36.5	74.425
BADGE	76.689	84.789	48.09	79.144
Entropy	76.287	83.195	42.883	77.252
<b>Confidence</b>	<b>77.876</b>	<b>85.687</b>	<b>42.44</b>	<b>79.742</b>
Monte-Carlo	78.861	82.644	47.955	78.703
Random	81.428	85.289	44.572	79.09
Oracle	82.135	88.048	49.754	83.505

**Table 5.3:** Performance of active learning strategies and the oracle (benchmark) model, detailed across different difficulty levels and metrics.

We summarize our analysis and interpretations in the following:

1. **CRB and Confidence Sampling:** CRB achieves notable overall performance, which can be attributed to its hierarchical and hybrid nature combining uncertainty-based criteria, diversity-based criteria, and geometric characteristics, enabling the query strategy to balance between diverse detection scenarios as defined by the distance categories. On the other hand, confidence sampling, which focuses on the uncertainty of objectness in

frames, achieves the highest mAP performance. This can be attributed to its narrower focus, allowing it to select frames that help the model learn about various objects, unlike Entropy sampling, which targets the uncertainty of object classification, which might be biased towards certain object classes. This bias could limit the ability to select a diverse range of frames, potentially impacting the overall learning performance.

## 2. Performance by Difficulty Levels:

- **Hard:** The hybrid-based BADGE strategy shows effectiveness in hard scenarios where data is sparse. Its hybrid nature possibly helps. On the other hand, Monte-Carlo sampling shows effectiveness in hard scenarios despite being an uncertainty-based method. This success might be due to its use of MC dropout, which introduces randomness into the model during training. By repeatedly dropping out different sets of neurons, the model learns to make predictions less dependent on any specific set of features.
- **Moderate:** Random sampling and, again, Confidence sampling show effective performance at moderate difficulty scenarios. Random sampling, with its unselective and indiscriminating approach, serves as a robust baseline, as it does not bias the selection towards any specific characteristics of the data. Hence, in scenarios with moderate-range objects where the data is at a balance between density and sparsity, Random sampling can ensure a diverse and representative sample of the scenario.
- **Easy:** CRB shows strong performance in Easy scenarios, where objects are dense and more predictable. This performance is partly due to CRB including a geometric-based criterion, which focuses on selecting frames with uniformly distributed points. Such a criterion is especially beneficial for detecting objects at close range, where point density tends to be consistent and uniform.

3. **Temporal Consistency of tCRB:** Although tCRB applies the CRB strategy on a temporal window, it does not achieve a competitive performance. This could be attributed to its temporal constraint. By focusing on selecting frames within specific time windows that collectively have a high score of uncertainty, diversity, and geometry, tCRB might overlook more informative frames outside of the temporal window. In dynamic environments, the most informative frames for model training might not necessarily align with temporally consistent frames. This leads to a mismatch between the selected frames for active training and the more informative frames, eventually limiting the model's generalization ability.

In conclusion, the dominance of the Oracle model across all ranges is evident, setting a high benchmark in performance. Notable strategies like CRB, Random sampling, and Confidence sampling have their strength in different detection ranges, suggesting the importance of strategy selection based on specific range requirements in active learning settings. In addition, the analysis shows that the strategy **tCRB** could be beneficial when it's required to query a temporal sequence for labeling. Still, its overall effectiveness might be limited compared to other strategies.

Table 5.4 provides an insightful look at the variation in mAP scores across the AL strategies and the Oracle model, tracking their progress from the minimum mAP score to the maximum, as well as the final mAP score. A key observation is that for all AL strategies, in addition to the oracle, the final mAP score aligns with the maximum mAP. This indicates a consistent upward trajectory in model performance over time, suggesting that the active learning process effectively contributes to progressive improvements in the detection performance. The fact that

no AL strategy reached its peak performance before the last epoch shows that these strategies continuously enhanced their performance until the final stages of active training. Moreover, the difference in the minimum and maximum mAP range between the AL strategies and the oracle is attributed to their different training methodologies. Active learning strategies start with a smaller initial training set and incrementally build up by selectively adding new data. This approach leads to a wider range of performance as the model gradually learns from a growing dataset. In contrast, the Oracle model is trained on the full dataset from the beginning, offering a more stable and consistent learning process.

Strategy	Overall		
	mAP <sub>final</sub>	mAP <sub>min</sub>	mAP <sub>max</sub>
<b>CRB</b>	<b>79.657</b>	<b>39.831</b>	<b>79.657</b>
tCRB	72.653	34.104	72.653
CoreSet	74.425	36.061	74.425
BADGE	79.144	39.831	79.144
Entropy	77.252	35.167	77.252
<b>Confidence</b>	<b>79.742</b>	<b>38.921</b>	<b>79.742</b>
Monte-Carlo	78.703	39.509	78.703
Random	79.090	37.396	79.090
Oracle	83.505	70.169	83.505

**Table 5.4:** The variation in mAP scores across various AL strategies and the oracle model.

Strategy	Car			Pedestrian			Bicycle		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
CRB	82.39	94.06	45.235	94.1	81.0	62.345	80	67.37	24.471
tCRB	84.6	94.39	37.468	91.73	70.62	34.07	65.45	51.43	0
CoreSet	78.89	93.36	41.6	79.267	75.226	43.067	71.55	63.08	0
BADGE	85.05	94.49	47.01	93.6	77.893	76.214	100	73.67	8
Entropy	82.82	93.9	48.51	95.11	79.103	65.187	100	68.958	8
Confidence	84.34	94.257	50.07	97.09	80.97	61.182	100	69.94	8
Monte-Carlo	83.51	94.37	48.02	91.067	79.56	65.307	100	59.74	32
Random	83.77	94.56	45.96	92.77	80.88	73.04	100	74.92	0
Oracle	89.731	94.70	41.80	82.07	86.41	86.8	100	74.453	6.667

**Table 5.5:** Comparison of active learning strategies across different difficulty levels for the classes Car, Pedestrian, and Bicycle.

To provide a more detailed understanding of how active learning strategies perform across different classes of road users, we adopt a common convention in the research community by focusing on three pivotal classes: car, pedestrian, and bicycle. This selection reflects their notable presence in traffic scenarios. Table 5.5 shows the mAP scores at the varying difficulty levels—Easy, Moderate, and Hard—for the active learning strategies alongside the

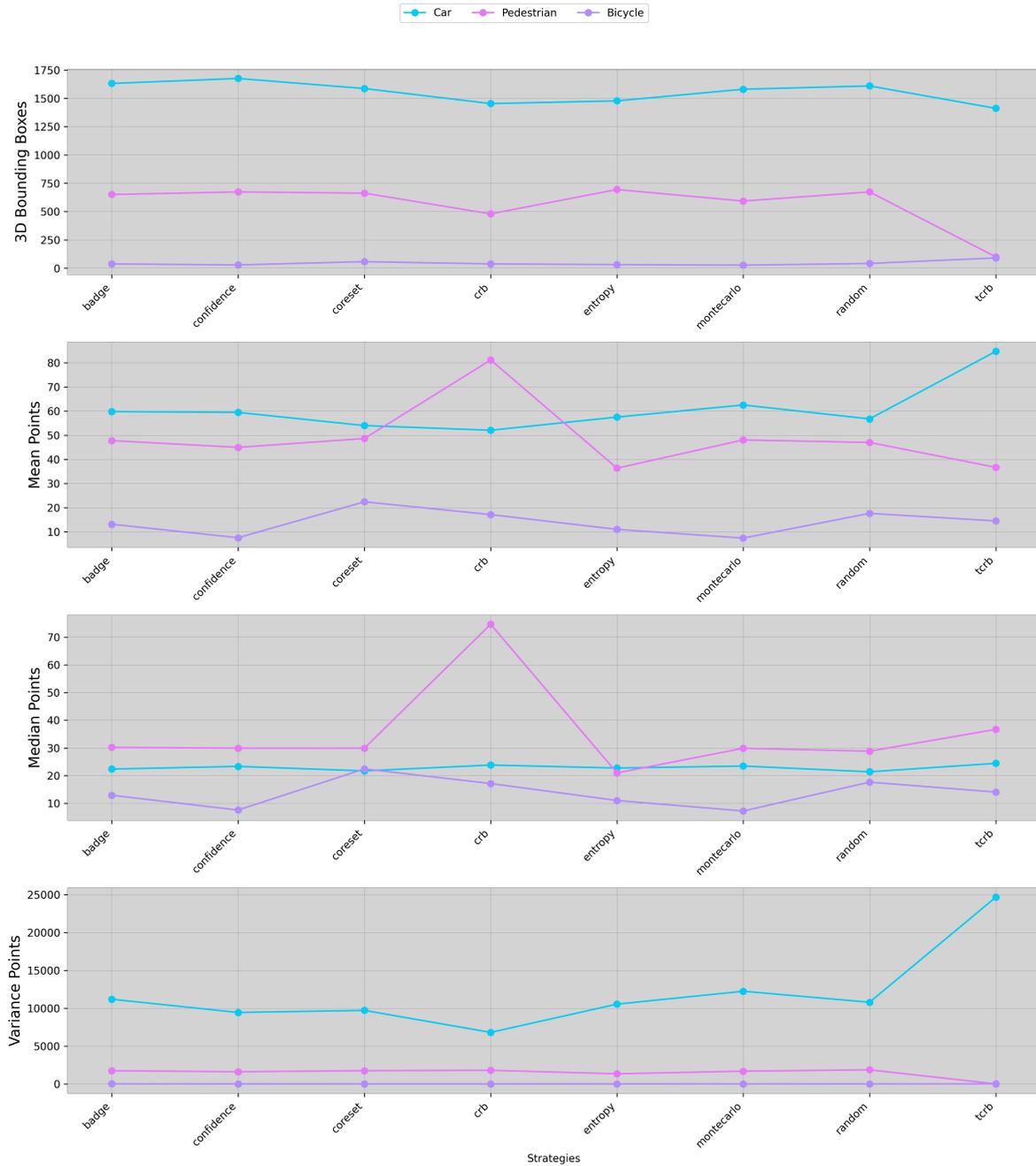
Oracle model. The objective of presenting these results is to evaluate the performance of each strategy with different object classes in varied complexities, which helps inform the choice of the optimal active learning strategy for real-world scenarios.

We summarize our observations and analysis in the following points:

- CRB demonstrates robust performance across most classes and difficulties, particularly in the Easy and Moderate categories for cars and pedestrians. However, tCRB, while effective, shows a notable decline in Hard scenarios, suggesting its temporal consistency approach might be less effective in complex detection environments. The difference in performance between CRB and tCRB, especially in challenging situations, indicates that temporal elements in tCRB could impact its adaptability to rapidly changing or difficult detection contexts.
- CRB stands out with its relatively higher performance in the Hard difficulty for bicycles compared to most other strategies, except Monte-Carlo. This could result from CRB's effective handling of complex, sparse data at greater distances. In contrast, tCRB shows a marked decrease in performance, aligning with most other strategies. Generally, the lower performance of detecting bicycles at longer ranges can be linked to their smaller size besides their limited number of instances at that range. Interestingly, the Easy difficulty for bicycles sees relatively better detection rates, likely due to closer proximity and less occlusion, enhancing detection accuracy.
- An interesting observation is the higher mAP scores for pedestrians and bicycles in Easy scenarios compared to cars. This could be due to the more distinct movement patterns and profiles of pedestrians and bicycles, which might be easier for the models to learn and recognize at closer distances. In contrast, cars, despite their larger size, might present more homogeneous features that are less distinct at close range, leading to a slightly reduced detection rate in Easy scenarios.
- Another interesting observation is the higher mAP scores for pedestrians in Hard scenarios compared to cars, although cars have bigger sizes than pedestrians. This can be linked to the higher occurrence of pedestrian instances in Hard scenarios, providing more learning opportunities for the models. In addition, one might also argue that pedestrians exhibit a variety of movements and postures, making their overall visual profile more diverse and easier for the models to learn compared to the uniform visual shape of cars.
- Across all strategies, the data reveals a trend where the detection of objects becomes increasingly challenging as the difficulty level rises, with a more noticeable effect in smaller or more complex object classes like bicycles. The superior performance of the Oracle model across all classes and difficulties serves as a benchmark, highlighting the potential of comprehensive training data in enhancing detection accuracy. However, the effectiveness of active learning strategies, especially CRB, in scenarios with limited data is noteworthy, suggesting their viability as practical solutions in real-world applications.

Figure 5.3 provides multiple statistics of the selected frames for cars, pedestrians, and bicycles at the final cycle of the active learning process. It breaks down four important statistics, including the number of predicted 3D bounding boxes, the mean and median number of points within the bounding boxes, and the variance in point counts per box. This figure reflects the outcomes of the sample selection performed by various active learning strategies, which utilize the model detections to assess frames' informativeness and select a subset of frames from the unlabeled data pool. This comparison not only reveals the selection frequency of

each object class by different strategies but also provides an indirect evaluation of the detection quality through the density and consistency of LiDAR points within the selected frame samples. Additionally, analyzing how these frame selection statistics correlate with the active learning strategies offers insights into the effectiveness of each strategy in utilizing data for subsequent learning phases.



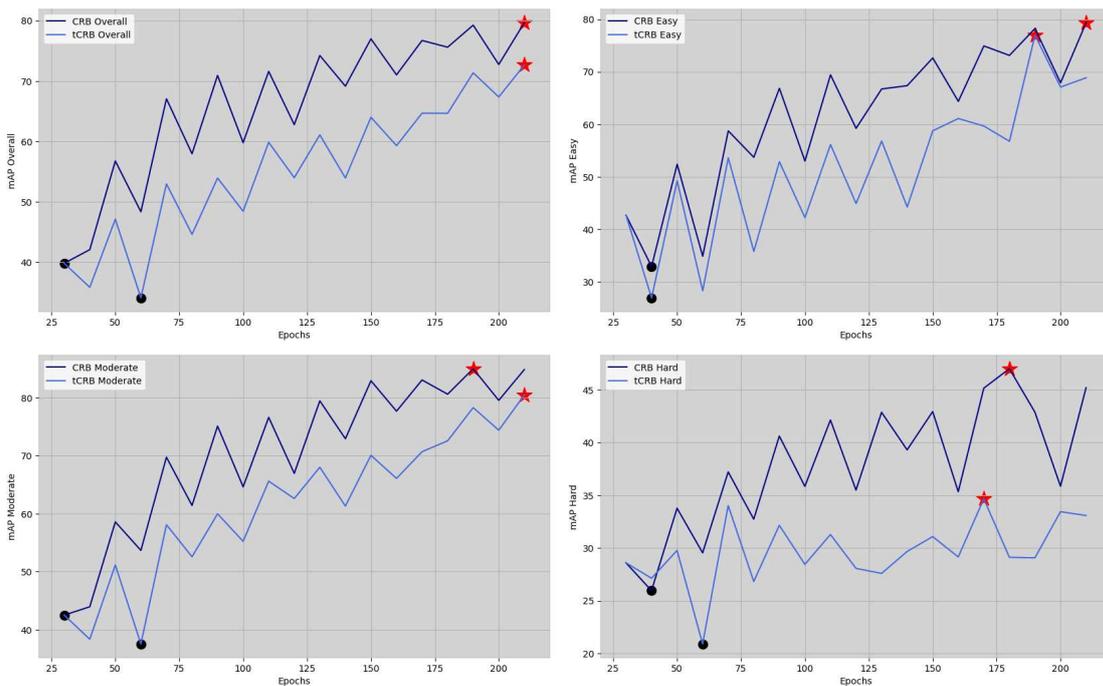
**Figure 5.3:** Comparative line plots for three classes – Car, Pedestrian, and Bicycle – illustrating class-specific statistics for the selected frames by AL strategies at the final active training cycle.

## 5.3 Ablation Studies

In this chapter, our focus is on assessing the changes and improvements we introduced to the standard approach. We offer an in-depth analysis of the role of temporal consistency within the CRB query strategy as used in Active3D, particularly its impact on the three stages of CRB. Additionally, we explore how the introduction of weighted entropy computations in CRB for active class balancing affects the class distribution in the selected frames.

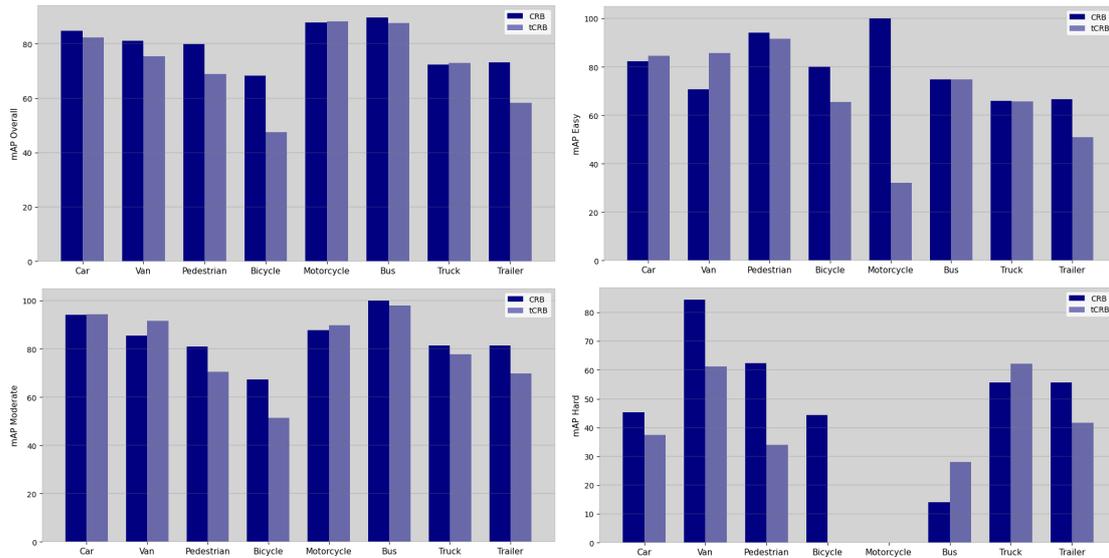
### 5.3.1 Temporal Consistency in CRB

In Section 4.2, we discussed how temporal CRB (tCRB) accounts for the preference for sequential data samples in the annotation. This involved utilizing a sliding window method to evaluate the informativeness of a series of point cloud frames rather than assessing each frame individually. Introducing this temporal constraint could potentially enhance or degrade the overall performance of the active training process. To understand its impact without any prior assumptions, we provide a thorough comparative analysis of the computational aspects of tCRB and that of the original CRB, highlighting any differences that might influence the overall active training performance.



**Figure 5.4:** The figures display the mAP scores across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top-left to the bottom right, respectively. The highest mAP score is marked with a red star, and the lowest mAP score is marked with a black circle.

Regarding the performance in active training, it appears that the original CRB outperforms tCRB across various difficulty levels. As illustrated in Figure 5.4, the learning curves exhibit a zigzag pattern, with the highest and lowest mAP scores typically occurring at the end and start of the learning process, respectively. This suggests a progressive improvement from one cycle to the next. Notably, in the *Hard* difficulty level, the learning curves show extreme fluctuations, reflecting the challenges the models face in accurately detecting distant objects, even with additional learning cycles.



**Figure 5.5:** The figures display the mAP scores for different object classes across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top left to the bottom right, respectively.

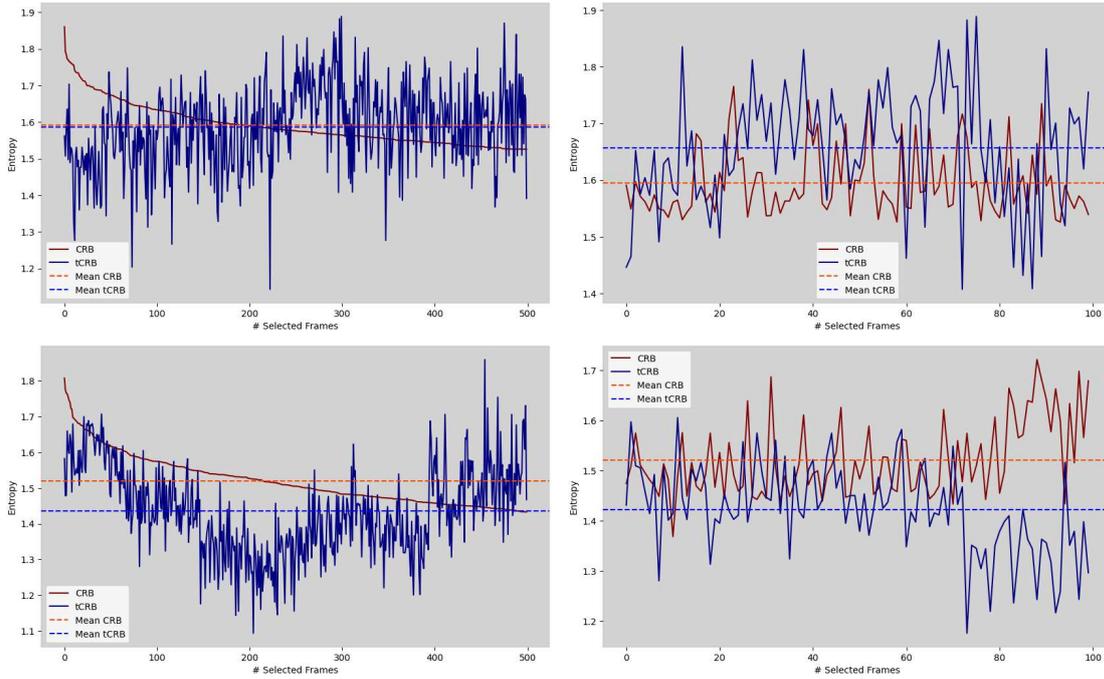
Figure 5.5 details the mAP scores for each object class across the three difficulty levels and the overall mAP score to assess class-specific performance. It's notable that tCRB closely matches or occasionally exceeds CRB in most object classes. For example, in the *Easy* level, tCRB outperforms CRB in the *Car* and *Van* classes. However, tCRB seems to struggle with the *Motorcycle* class, particularly at the *Hard* level. This observation aligns with the findings in Figure 5.4, which helps explain the lower overall performance of tCRB compared to CRB in the *Hard* level.

Next, we investigate the three stages of CRB, namely the entropy computation in stage 1, the gradient clustering in stage 2, and the point density estimation in stage 3, to understand the impact of introducing the temporal consistency constraint.

### Entropy Computation

Figure 5.6 illustrates the selected frames' entropy values during the first and third stages of CRB and tCRB. We summarize our analysis as follows:

- In stage 1, the entropy values of CRB exhibit a descending order, in contrast to those of tCRB, reflecting the different sorting priorities in each strategy. CRB sorts frames by the descending order of the overall entropy, while tCRB sorts frames in chronological order (by timestamp).
- In stage 3, the entropy values of CRB are no longer in descending order. This is due to the computational changes in stages 2 and 3, which focus on criteria other than entropy, resulting in a final frame selection that doesn't necessarily prioritize the highest uncertainty or entropy order.
- Early in the active training process, the average entropy values for CRB and tCRB are similar at stage 1, reflecting the model's initial uncertainty when learning from a small training set. At this point, whether frames are sorted by entropy or time seems to have



**Figure 5.6:** Entropy values for the  $K_1$  frames chosen in stage 1 and  $K_3$  of CRB and tCRB at the first and the last active training cycles, in the first and second row, respectively. The dashed red and blue lines represent the mean entropy across all frames. Each frame’s entropy is calculated as the aggregate of label entropies from the predicted 3D bounding boxes.

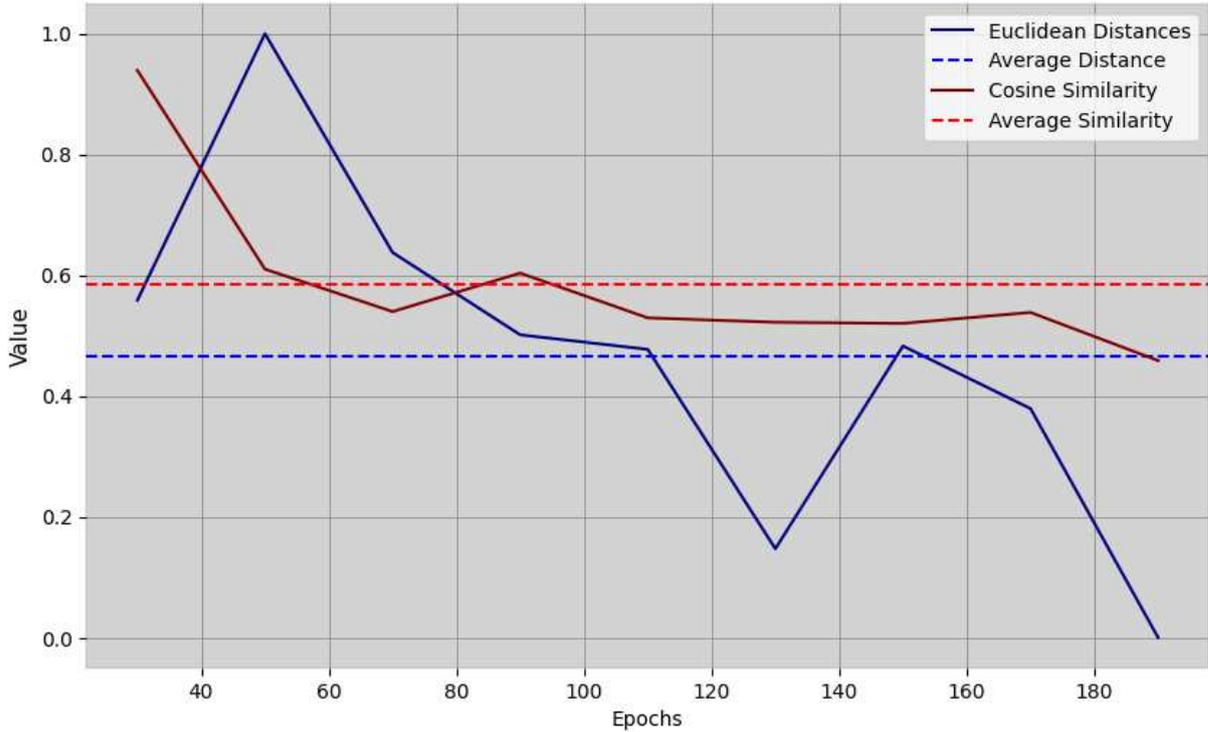
a minimal impact. However, by stage 3, the average entropy of tCRB is higher, as the temporal constraint maintains the high initial variation in uncertainty across sequential frames.

- Towards the end of the training process, the entropy values of sequential frames for tCRB still show fluctuations, but the variations are more minor compared to the earlier cycles. This observation indicates that the model has acquired knowledge over time, smoothing out the uncertainty fluctuations as learning progresses.

### Gradient Clustering

Recall our discussion in Section 4.2, in stage 2, tCRB selects a  $K_2$  subset of frames based on the representativeness of its gradient embeddings. The representativeness of gradients is measured by their magnitude and orientation w.r.t to the weighted mean gradient. In contrast to CRB, which applies K-Means++ clustering on the gradient embeddings to form  $K_2$  cluster and return their centroids as the most representative set of gradients, whose frames are then selected. We further compute the weighted mean gradient of these clusters, where the gradient weights are their norms. For meaningful comparative analysis, we compute the Euclidean distance and cosine similarity between the weighted mean gradients of CRB and tCRB to provide insight into the impact of the temporal constraint introduced by tCRB on the learning process. Figure 5.7 shows the Euclidean distance and Cosine similarity calculations. The following points summarize our interpretations and analysis:

- Earlier in the active training cycle, a high Cosine similarity and Euclidean distance indicate that both CRB and tCRB are identifying similar informative features to improve the model. However, tCRB might be selecting frames that are informative in a sequential context but not necessarily the most informative overall. This can explain why, despite a



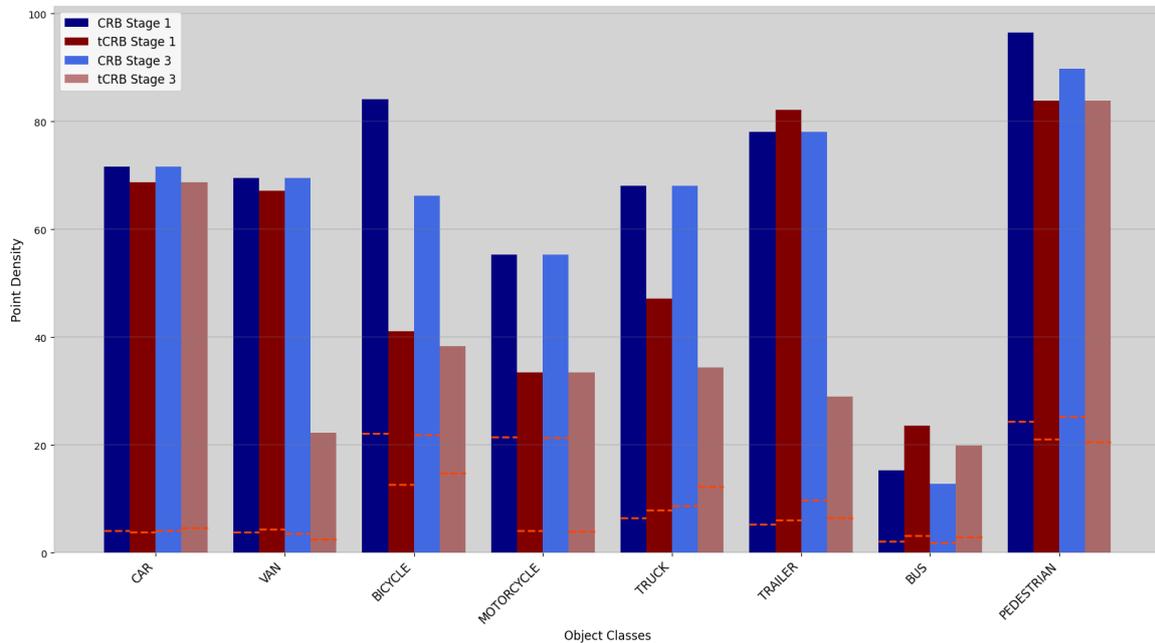
**Figure 5.7:** The figure shows the Euclidean distance and Cosine similarity between the weighted mean gradients of CRB and tCRB along the active training process.

similar direction of optimization (high Cosine similarity), there is still a large difference in the magnitude of optimization (high Euclidean distance). CRB might make larger updates by selecting the most informative frames regardless of their order, leading to a higher impact on the learning process, as evident in Figure 5.4.

- As active training progresses, the Cosine similarity converges to an intermediate level while the Euclidean distance keeps decreasing. The temporal constraint in tCRB may start to limit the diversity of the training data because it focuses on sequences rather than individual informativeness. This can lead to a convergence in the direction of optimization but necessarily the optimal direction, hence the intermediate level of Cosine similarity. On the other hand, the decreased Euclidean distance indicates that both strategies' selections are leading to more minor updates in the model parameters over time, which is typical as the model converges. However, CRB might be continuously fine-tuning the model more effectively by considering a more diverse set of frames, thus leading to better performance overall.
- Essentially, while the temporal constraint may add relevant contextual information, it could potentially restrict the model from exploring a broader range of informative features in the data, especially earlier in the active training process. This is consistent with the performance results that CRB outperforms tCRB as it allows the models to access a more diverse data set.

### Point Density Estimation

At the final stage, the  $K_2$  selected frames from stage 2 are further analyzed based on the point distribution of the 3D bounding boxes, and the set of frame that have the closest point density distribution to the uniform distribution is finally selected. Figures 5.8 and 5.9 provide insight into the behavior of tCRB versus CRB. Our observations are summarized as follows:

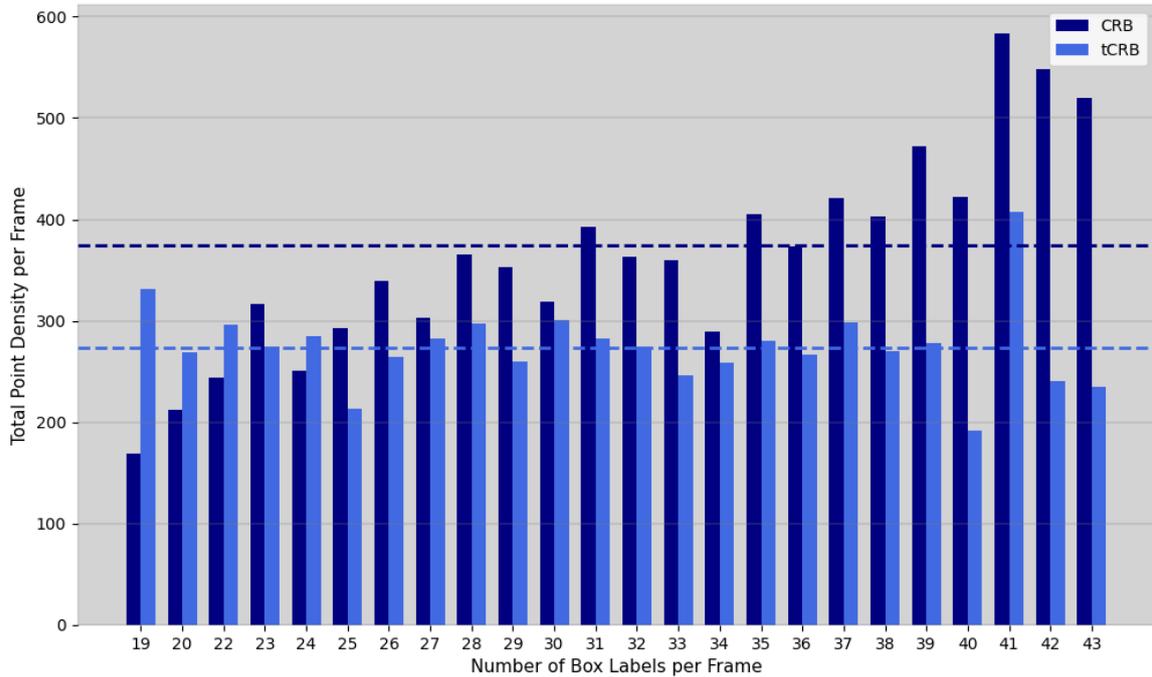


**Figure 5.8:** The figure shows each class’s point density distribution at stages 1 and 3, comparing the results of CRB and tCRB. Additionally, black dashed lines represent the mean point density for each class at the final active training cycle.

- In Figure 5.8, for major classes like cars, bicycles, and pedestrians, the point densities of CRB and tCRB are notably similar at both stages. The average point density of these classes additionally supports this.
- The overall analysis of Figure 5.8 shows no significant differences in the maximum or average point densities between CRB and tCRB, with only minor variations observed. This suggests that the temporal constraint introduced by tCRB does not strongly alter the representativeness of the selected frames in terms of point density distribution.
- In Figure 5.9, CRB generally exhibits higher point density per number of 3D bounding boxes, which implies that CRB favors frames with denser points. This might contribute to CRB’s higher performance by providing the model with richer information per frame, contributing to a better learning process.
- On the other hand, tCRB exhibits a narrower variance in point densities per number of 3D bounding boxes, suggesting a more uniform selection of frames in terms of point distribution. The temporal constraint in tCRB might lead to selecting sequences of frames with more regular point density distributions.

In conclusion, the proximity of point density distributions between CRB and tCRB for major classes reflects a shared focus on key object classes. However, the difference in point density variance and relation to bounding box count highlights the different approaches both

strategies employ to frame selection complexity and diversity, which may influence the effectiveness of the model training in the active training process.



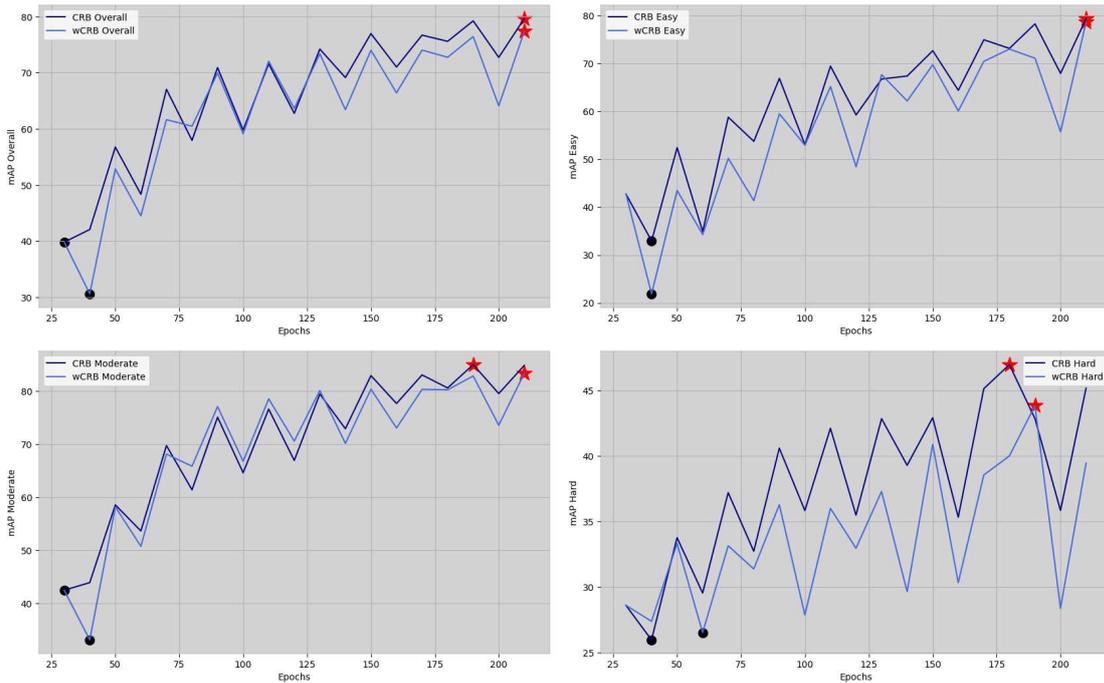
**Figure 5.9:** The figure presents the point density distribution and the number of predicted 3D bounding boxes per frame for CRB and tCRB at the final active training cycle.

In this ablation study, we compared the baseline CRB with its temporal variant, tCRB, across their three hierarchical stages. The analysis revealed that while CRB generally outperforms tCRB, the temporal aspect of tCRB aligns closely with CRB in major object classes. However, the temporal constraint in tCRB limits diversity in frame selection, leading to a narrower variance in point density distribution, affecting its overall performance. The temporal constraint also resulted in less aggregate entropy of the selected frames, hence less uncertainty. The gradient analysis shows that CRB and tCRB start their learning path similarly, but they start to diverge as the active training progresses. We attribute this divergence to the effect of the temporal constraint on the diversity and uncertainty of the selected frames.

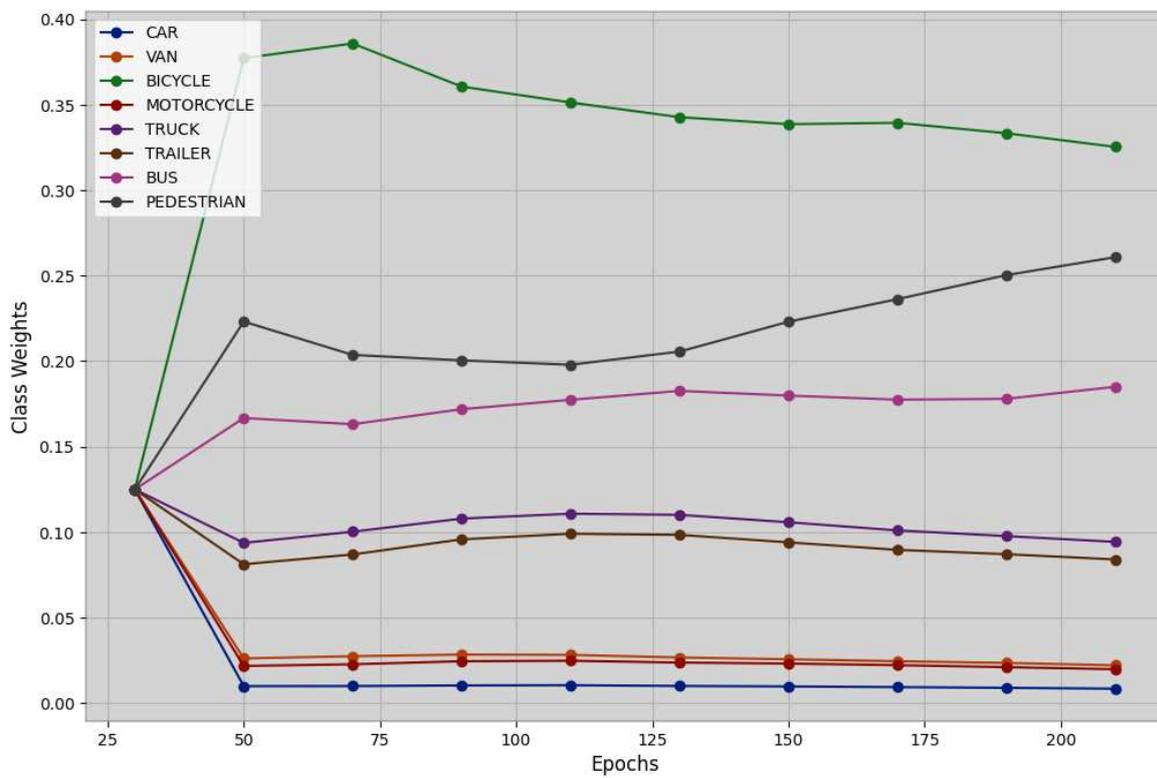
### 5.3.2 Active Class Weighting in CRB

As discussed in Section 4.3, we re-formulate the entropy computation in stage 1 of CRB to incorporate class weights, where the weights are determined dynamically by the class counts in the growing training set. This modification addresses the class imbalance issue, potentially steering the selection process toward a more balanced distribution of class labels. In this section, we evaluate the effectiveness of this modification. We call the modified variant of CRB, **wCRB**.

Looking at the mAP scores across the three difficulty levels in Figure 5.11, we observe only slight variations in performance between CRB and tCRB. Both strategies show similar performance curves, suggesting that the weighted entropy modification does not strongly impact the frame selection and hence the model performance.



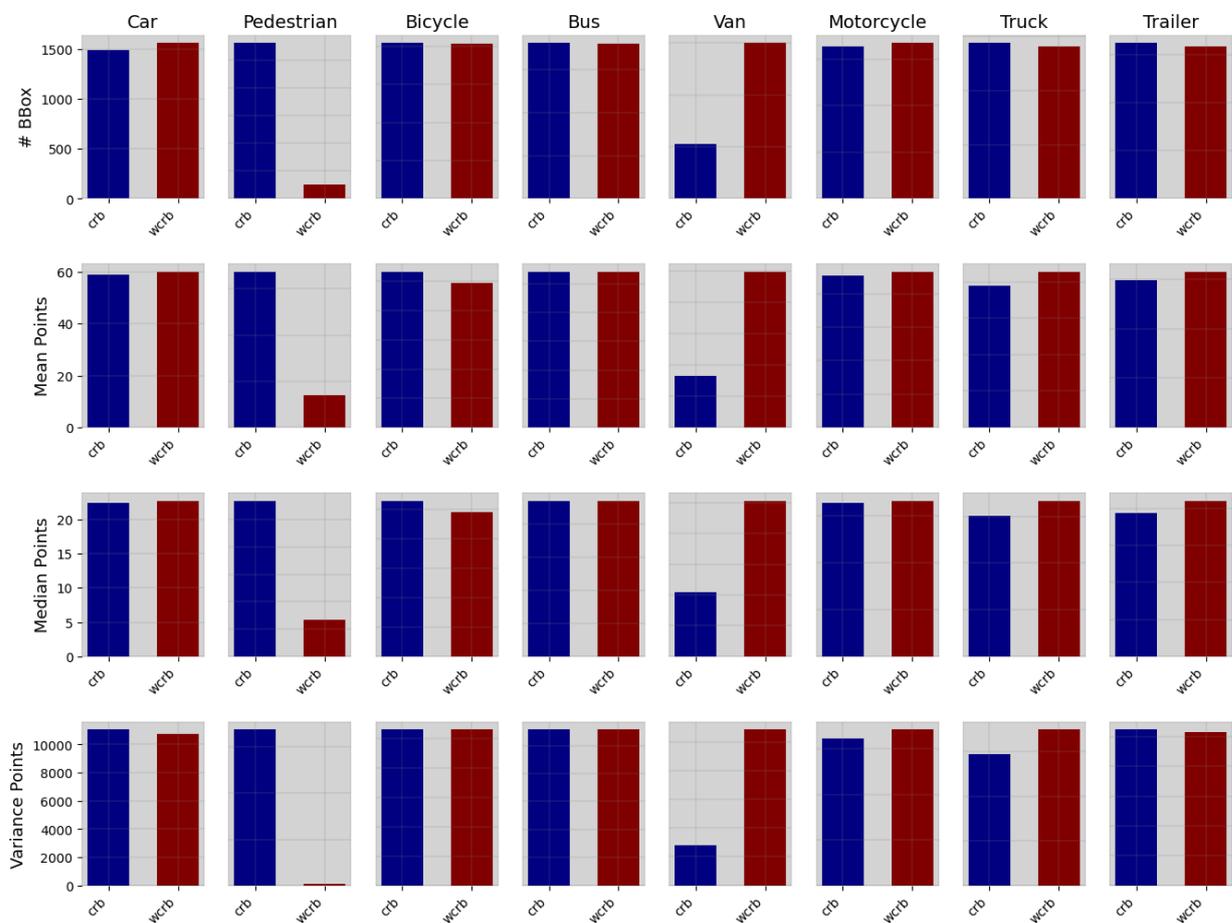
**Figure 5.10:** The figures display the mAP scores across the difficulty levels (Overall, Easy, Moderate, Hard) arranged from the top-left to the bottom right, respectively. The highest mAP score is marked with a red star, and the lowest mAP score is marked with a black circle.



**Figure 5.11:** The weights of the object classes computed at every active training cycle as the reciprocal of the class count in the current training set.

Figure 5.11 shows class weights typical of a long-tailed distribution often seen in autonomous vehicles datasets. Common classes like cars and vans are more frequent than rarer ones like bicycles and pedestrians. Despite class weight calculations being responsive to class frequencies, they seem to have minimal impact on balancing the training set through frame selection. For example, the selection strategy tends to choose frames with many cars, despite cars having the lowest class weight and being less prioritized during the first stage (i.e., during entropy computation). Furthermore, the class weight for the *bicycle* category remains high throughout, even though it's given priority in the entropy computation. This pattern is clear in Figure 5.12, where *w*CRB is observed to select fewer *pedestrians* and *bicycles* (classes with higher weights) than expected while unexpectedly choosing more *vans*, despite vans having a high class count and consequently a higher class weight.

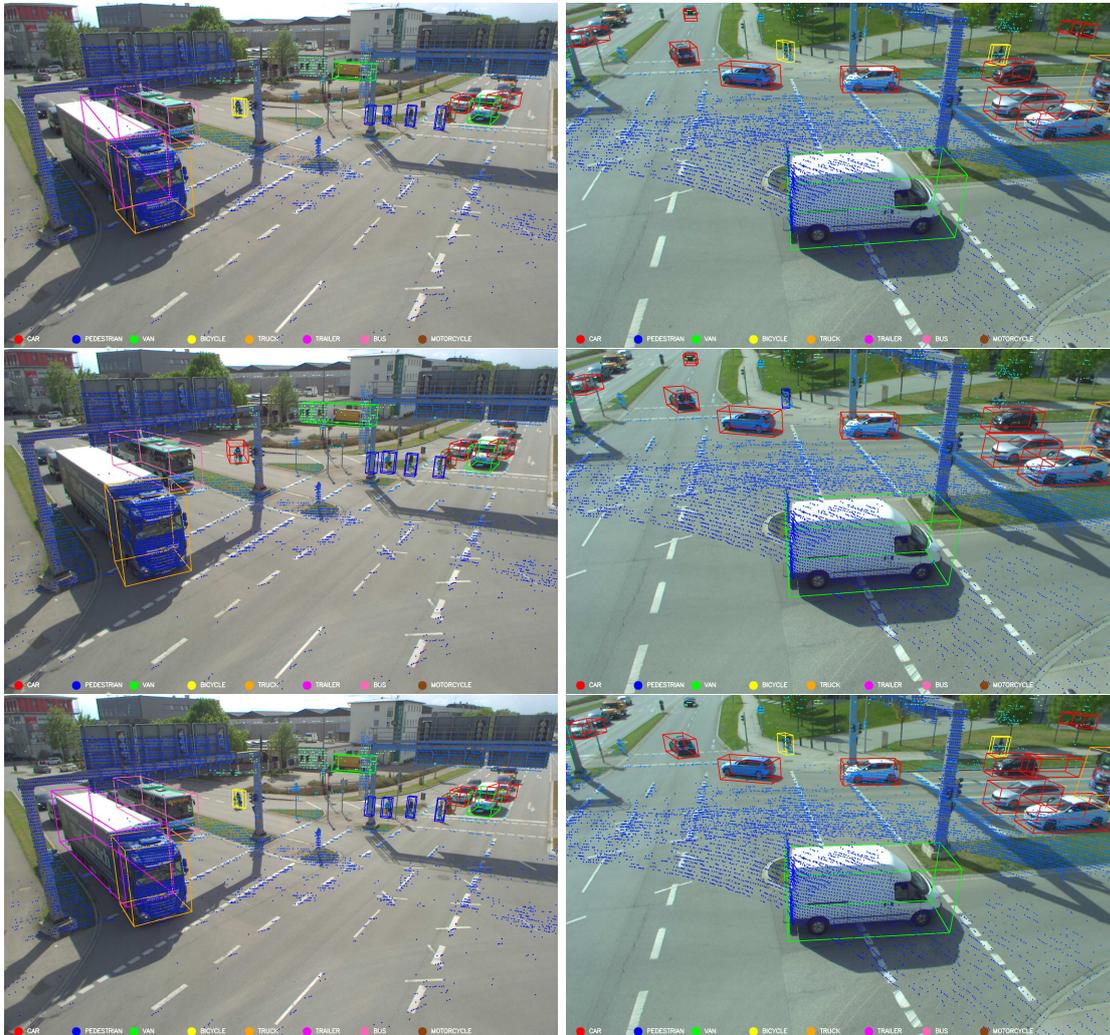
In conclusion, our results suggest that the weighted entropy approach is simplistic and ineffective in addressing the complex issue of class imbalance. This method failed to adequately address the challenges posed by the class imbalance issue.



**Figure 5.12:** The figure displays a statistical comparison of the object classes in the selected frames by CRB and *t*CRB. The statistics are averaged across all active training cycles.

## 5.4 Qualitative Results

In this section, we present a variety of qualitative outcomes, including captures for active training using both CRB and tCRB strategies, alongside results from the Oracle model. Additionally, we highlight an example from the *Inference* feature in proAnno++. These examples aim to visually demonstrate the discussions in earlier sections.



**Figure 5.13:** The figure shows the detection results of the PV-RCNN model across three scenarios: Firstly, the first row shows the detection outcomes when employing the CRB query strategy for active training. Secondly, the second row shows the results of the tCRB query strategy for active training. Lastly, the third row shows the detection results of the oracle model, which has been trained on the complete training dataset. It should be noted that the two columns in the figure correspond to two neighboring cameras, which capture different perspectives of the same traffic scene.

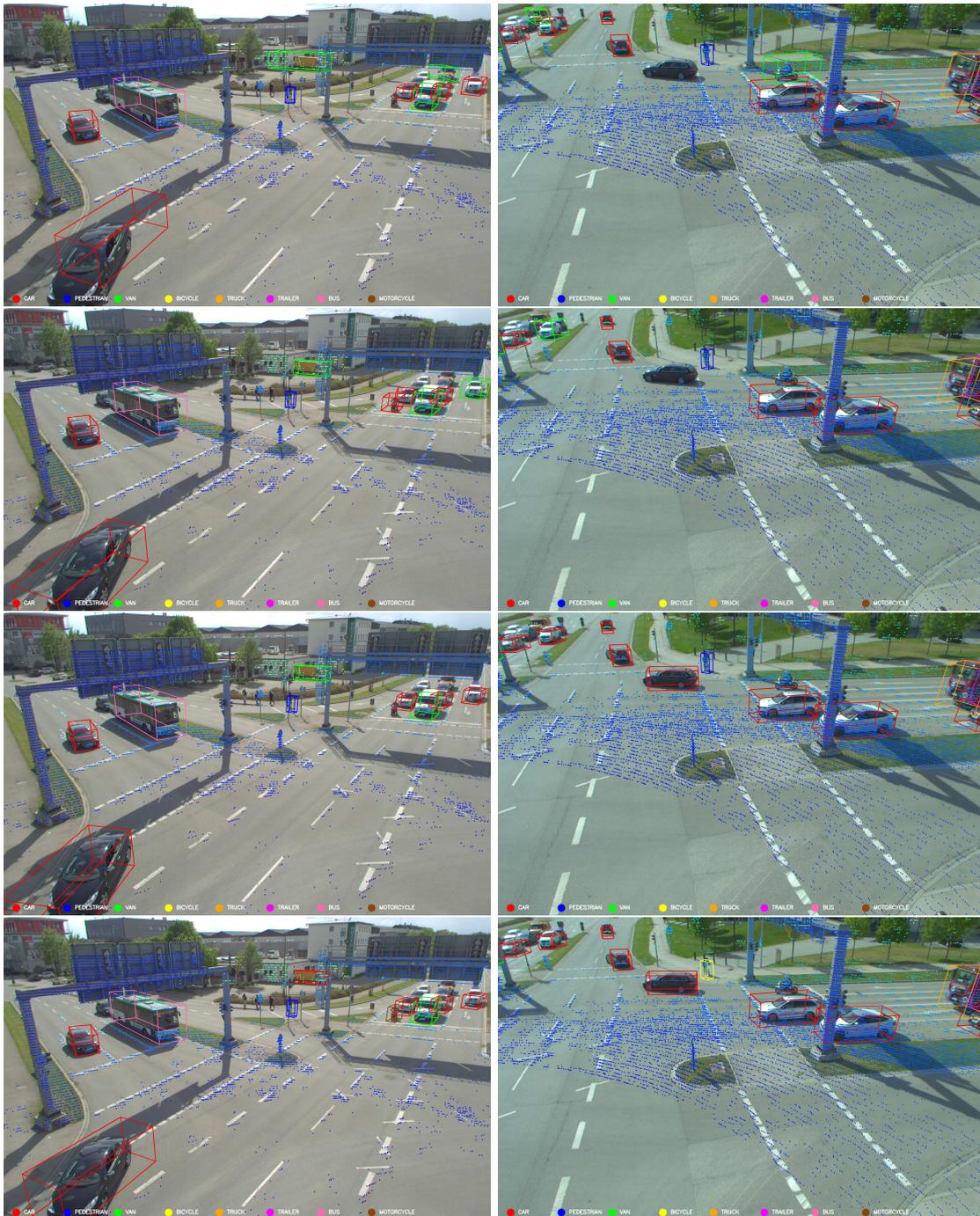
Figure 5.13 reflects our observations in the quantitative analysis Section 5.2. With the CRB strategy, the model effectively detects and localizes objects like bicycles, pedestrians, and motorcycles at moderate distances. Additionally, it accurately identifies challenging objects at greater distances, such as the parked car visible in the top right of the right camera and the parked van in the left camera. On the other hand, tCRB faces challenges in accurately classifying bicycles. tCRB misclassifies a bicycle as a pedestrian in the right camera and misclassifies a bicycle as a car in the left camera. Although the tCRB correctly classifies the far van object in the left camera, the inferred dimensions are incorrect. This may highlight



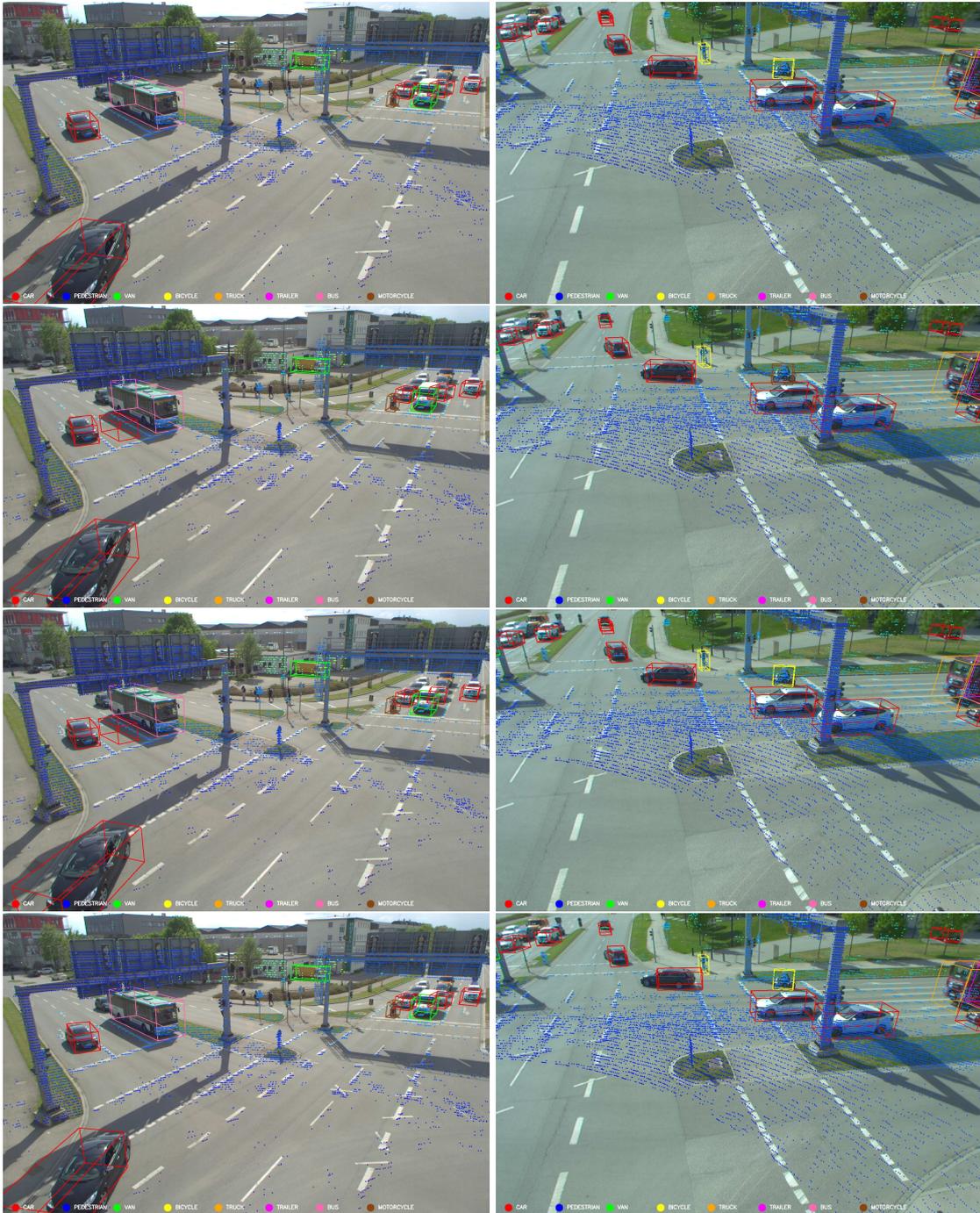
**Figure 5.14:** The figure shows the detection results of the PV-RCNN model in active training using CRB and tCRB query strategies.

the difference in the effectiveness of the learning process between CRB and tCRB, where CRB identifies the van boundaries without being misguided by the points of the neighboring building. Figure 5.14 further demonstrates that the tCRB strategy is less effective with distant objects. This challenge may arise from the way temporal consistency influences the diversity and complexity of frames selected in each active training cycle, hence reducing the model's learning efficiency for distant objects. Conversely, the CRB strategy shows better performance with distant objects, potentially due to its diverse selection approach, which leads to a more effective learning process for the model.

Figures 5.15 and 5.16 present a visual comparison of the detection performance at four successive stages towards the end of the active training process. These figures demonstrate the consistency of the CRB strategy in terms of learning efficiency, as shown by its stable detection across different stages, particularly noticeable in distant objects and less common classes like bicycles. In contrast, tCRB shows less consistency, as evident by the parked van example in the left camera, where the model trained actively with tCRB correctly detects it in an earlier cycle then detects it with wrong dimension in a later cycle. The consistency in detection throughout the active training cycles can be attributed to the diversity in the selected frames. Selecting a more diverse set of frames, especially in terms of object distances and point cloud distribution, can contribute to a more stable and robust learning process.

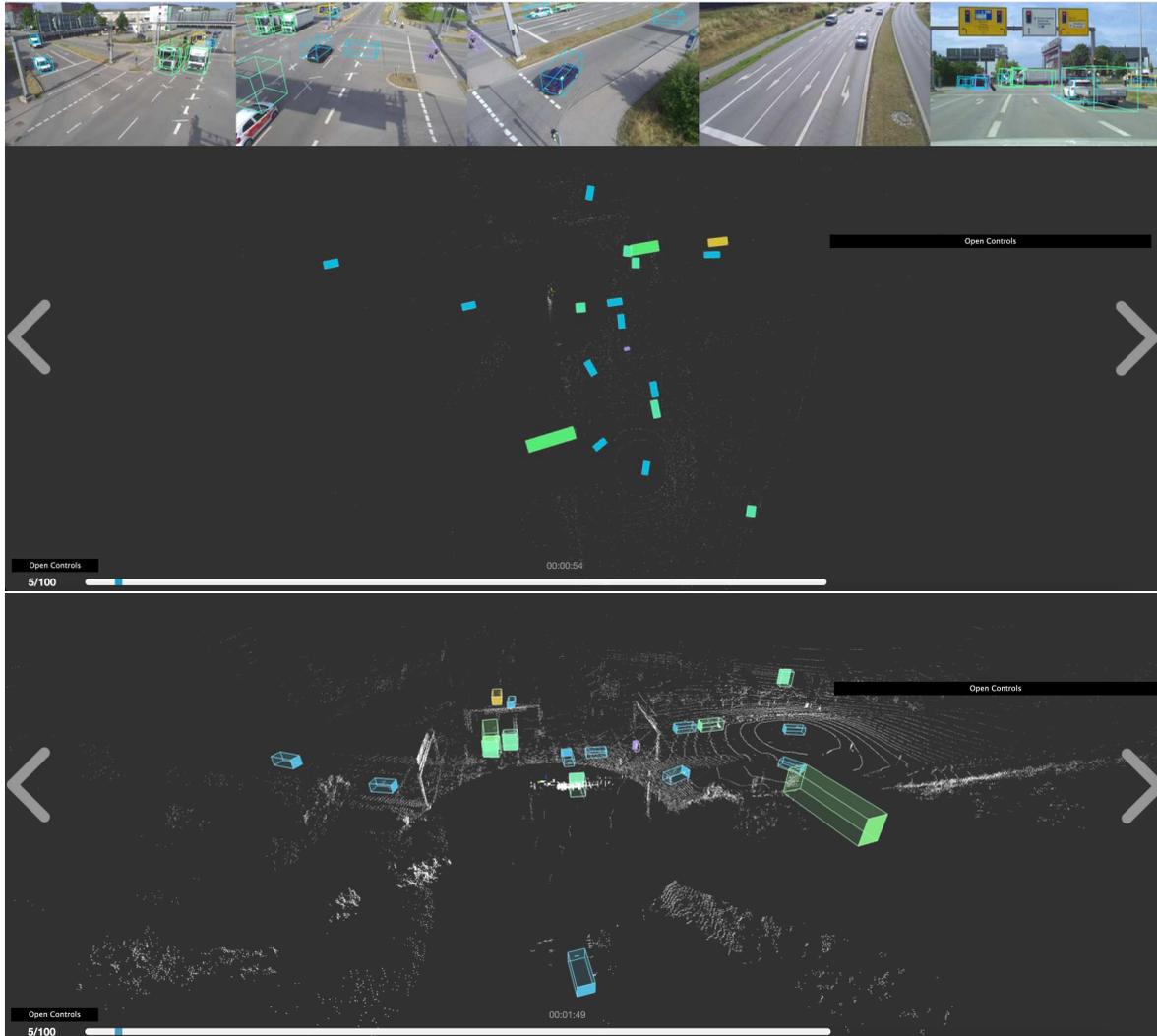


**Figure 5.15:** The figure shows the detection results of the PV-RCNN model on the same point cloud frame at different stages in the active training process, employing the tCRB strategy. The first row corresponds to the detections of the model's state at epoch 130 (the 6th selection cycle). The second row corresponds to epoch 150 (the 7th selection cycle). The third row corresponds to epoch 170 (the 8th selection cycle), and finally, the last row corresponds to epoch 190 (the 9th selection cycle).



**Figure 5.16:** The figure shows the detection results of the PV-RCNN model on the same point cloud frame at different stages in the active training process, employing the CRB strategy. The first row corresponds to the detections of the model's state at epoch 130 (the 6th selection cycle). The second row corresponds to epoch 150 (the 7th selection cycle). The third row corresponds to epoch 170 (the 8th selection cycle), and finally, the last row corresponds to epoch 190 (the 9th selection cycle).

Last but not least, Figure 5.17 demonstrates the 3D bounding box detections of the PV-RCNN model on a point cloud frame. Although the model is able to correctly and accurately detect several objects, there are notable concerns regarding its detection accuracy. The model overlooks multiple objects (*i.e.*, false negatives), generates several false positives, and has inaccuracies in object orientation. This highlights the necessity of proper pre-training of the PV-RCNN detector model or any similar 3D detector before its utilization in the annotation process. Without proper pre-training, the model could potentially complicate rather than ease the annotation process.



**Figure 5.17:** The figure displays the results of the PV-RCNN inference on a point cloud frame within the proAnno++ interface. This interface includes a view of five different cameras: four cameras mounted on a gantry bridge, and one is mounted on top of a moving vehicle. The visualization presents 3D bounding boxes, rendered both in the default perspective view and in an orthographic view.



## Chapter 6

### Conclusion

This thesis has thoroughly investigated how active learning can be effectively applied to LiDAR-based 3D object detection and 3D annotation tasks. The CRB query strategy demonstrated remarkable results, achieving performance comparable to that of an Oracle model using only 50% of the available point cloud data. Specifically, CRB achieves an overall mAP of approximately 80, closely approaching the Oracle’s overall mAP of approximately 83. This highlights the significant capability of active learning in identifying the most informative point cloud frames for 3D object detection and validates the canonical objective of active learning of addressing the data-hunger problem in training deep learning models. Additionally, the study found that continuous training methods, mainly incremental fine-tuning with a final training cycle on the entire dataset, are resource-efficient. In quantitative terms, our adopted continuous training method—*tunedAll*—requires approximately 72% of the training steps compared to the Oracle model to reach an equivalent overall mAP performance. These continuous training methods leverage the knowledge accumulated in prior active training cycles, thereby confirming our hypothesis that active learning methods can be both computationally efficient and highly effective for such downstream tasks. Moreover, the adjustments made to the CRB strategy for ensuring temporal consistency, namely tCRB, offer annotators the flexibility to choose sequential point cloud frames for labeling. However, our comprehensive ablation study reveals that tCRB does not necessarily select the most diverse and informative frames. As a result, tCRB does not yield as high a detection performance as the original CRB strategy. To sum up, our evaluation suggests the need of active learning strategies for careful fine-tuning to align with the specific requirements of the downstream task. Therefore, the effectiveness and efficiency of active learning methods compared to traditional methods remain open to debate and further exploration. Conversely, we have improved the 3D annotation process in proAnno by integrating AI-assisted features such as active learning. We successfully integrated the active learning pipeline—*Active3D*—into proAnno, enabling human annotators to focus on the most informative point cloud frames, potentially reducing annotation costs. We consider this approach promising and pave the way for more advanced human-in-the-loop machine learning that leverages both human expertise and the computational effectiveness of deep learning models. In proAnno, we have also improved several usability features, including the ability to zoom into camera images, show helper views when a 3D bounding box is selected, copy labels from one frame to another, and effortlessly switch between different datasets or sequences within the same dataset. These enhancements further streamline the annotation process, making it more intuitive and user-friendly for users.



# Chapter 7

## Future Work

The research presented in this thesis lays a foundation for several promising directions for future exploration. Building upon the insights and outcomes derived from our work, we outline the potential future work as follows:

- **Active Learning for Multi-modal 3D Object Detection:** Integrating active learning with fusion models is a promising approach, as it leverages the diverse information from various sensors such as LiDAR point clouds and RGB images. This integration enables a thorough evaluation of data sample informativeness from different perspectives.
- **Bayesian CRB:** The existing version of the CRB query strategy operates under the assumption that the test set (*i.e.*, unlabeled data pool) possesses both a uniform label distribution and uniform point density distribution. By incorporating a Bayesian approach, benefiting from the iterative process of active training and the evolving training set, it may become possible to accurately mirror the test set. This adaptation allows for a more detailed evaluation of the data informativeness based on the expanding training set.
- **proAnno++: Guided Active Selection:** The active query strategy can be tailored to suit the preferences of a human annotator. For instance, if an annotator prefers to label frames/images with a higher occurrence of specific classes, the strategy can be modified to evaluate the informativeness of these data samples concerning those particular classes. Consequently, it would prioritize returning the data samples most informative about the desired classes to the annotator. This feature allows for more targeted and efficient annotation, potentially improving the quality and focus of the dataset.
- **proAnno++: Local Inference:** Currently, proAnno++ conducts model inference across the entire selected point cloud; a future enhancement could include the ability for localized inference, focusing on the neighborhood around the cursor's position. This localized approach can lead to more efficient and faster processing, as it narrows down the area of analysis and potentially enhances the accuracy of predictions by focusing on specific regions of interest.
- **proAnno++: 2D object detection:** A useful feature involves enabling inference on images. In this scenario, the model would create 2D bounding boxes in images, which could be projected onto point clouds. Object dimensions could be estimated based on a predefined set of dimensions depending on the predicted class. However, this functionality may be impacted by occlusions, different daytime, and severe weather conditions.

- **proAnno++: Export 2D Bounding Boxes:** When a 3D bounding box is annotated in proAnno, it is projected onto camera images to form 2D bounding boxes. However, proAnno doesn't include these projected 2D bounding boxes in the output annotation file. This functionality could be added in future updates.
- **proAnno++: Usability Features:** Features designed to enhance usability can significantly simplify the annotation process, making it more intuitive and user-friendly, addressing its common difficulties and challenges. Potential features include highlighting the section of the point cloud that aligns with the field of view (FOV) of a camera image, which is especially useful when multiple cameras are involved. Another helpful addition could be displaying object IDs directly on the 3D bounding boxes within the point clouds. Furthermore, showing statistics of the currently labeled data can provide valuable insights during the annotation process.
- **proAnno++: Improving the Helper Views:** While currently, the helper views (Top, Side, and BEV views) are presented upon selection of 3D bounding box, a future enhancement would include enlarging the point size inside these views for better visibility. Additionally, crop the point cloud around the selected object offering a more focused analysis of the object being annotated.
- **proAnno++: 2D and 3D Masks:** After an object is labeled with a 3D bounding box, and its corresponding 2D bounding box is projected onto the image, a segmentation network can be used to identify the specific points (in the 3D box) and pixels (in the 2D box) belonging to the object or the background and therefore create 2D and 3D masks. This feature is beneficial as it provides a more precise and granular annotation level, capturing objects' shapes and contours more precisely. Additionally, it offers the necessary training data for 2D and 3D segmentation tasks.
- **proAnno++: Support Multiple Datasets:** Adapt proAnno architecture to support multiple datasets, including Waymo Open Dataset [45], nuScenes [4], and Argoverse 2.0 [4], converting annotations into a generic format adaptable across the datasets and exporting annotations to dataset-specific formats.
- **proAnno++: Large Language Models:** A promising direction for enhancing proAnno would be integrating language and vision foundational models into its 3D annotation framework. This integration could exploit the advanced capabilities of these models in understanding and interpreting complex visual and textual data. By leveraging language models, proAnno could introduce features such as natural language processing (NLP) to interpret and annotate 3D objects based on descriptive text inputs. This would allow users to annotate objects using simple language commands, streamlining the annotation process.
- **proAnno++: Denoising Point Clouds:** Incorporating a point cloud denoising feature into our 3D annotation tool, proAnno, could greatly enhance its utility. By preprocessing the point cloud data to reduce noise, the subsequent annotation process becomes more accurate and efficient. Users can focus on the essential features of the data without the distraction and potential errors introduced by noise. This improvement in data quality would be especially beneficial in applications requiring high precision, such as autonomous vehicle navigation, urban planning, or virtual reality environments.

## Bibliography

- [1] Arief, H. A., Arief, M., Zhang, G., Liu, Z., Bhat, M., Indahl, U. G., Tveite, H., and Zhao, D. “SAnE: smart annotation and evaluation tools for point cloud data”. In: *IEEE Access* 8 (2020), pp. 131848–131858.
- [2] *ASAM OpenLABEL Standard*. 12 Nov 2021. URL: <https://www.asam.net/standards/detail/openlabel/>.
- [3] Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. *Deep Batch Active Learning by Diverse, Uncertain Gradient Lower Bounds*. 2020. arXiv: 1906.03671 [cs.LG].
- [4] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. “nusenes: A multimodal dataset for autonomous driving”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11621–11631.
- [5] Chen, W., Edgley, A., Hota, R., Liu, J., Schwartz, E., Yizar, A., Peri, N., and Purtilo, J. *ReBounce: An Open-Source 3D Bounding Box Annotation Tool for Active Learning*. 2023. arXiv: 2303.06250 [cs.CV].
- [6] Chen, W., Edgley, A., Hota, R., Liu, J., Schwartz, E., Yizar, A., Peri, N., and Purtilo, J. “ReBounce: An Open-Source 3D Bounding Box Annotation Tool for Active Learning”. In: *AutomationXP @ CHI 2023* (2023).
- [7] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. “Multi-view 3D Object Detection Network for Autonomous Driving”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 6526–6534. DOI: 10.1109/CVPR.2017.691.
- [8] Creß, C., Zimmer, W., Strand, L., Fortkord, M., Dai, S., Lakshminarasimhan, V., and Knoll, A. “A9-Dataset: Multi-Sensor Infrastructure-Based Dataset for Mobility Research”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 965–970. DOI: 10.1109/IV51971.2022.9827401.
- [9] Dayoub, F., Sunderhauf, N., and Corke, P. I. “Episode-Based Active Learning With Bayesian Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. July 2017.
- [10] Doula, A., Güdelhöfer, T., Matviienko, A., Mühlhäuser, M., and Guinea, A. S. “Point-CloudLab: An Environment for 3D Point Cloud Annotation with Adapted Visual Aids and Levels of Immersion”. In: *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 11640–11646. DOI: 10.1109/ICRA48891.2023.10160225.
- [11] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD’96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.
- [12] Feng, D., Wei, X., Rosenbaum, L., Maki, A., and Dietmayer, K. *Deep Active Learning for Efficient Training of a LiDAR 3D Object Detector*. 2019. arXiv: 1901.10609 [cs.R0].

- [13] Foundation, L. A. *bibinitperiod D. Xtreme1 - The Next GEN Platform For Multisensory Training Data*. Software available from <https://github.com/xtreme1-io/xtreme1/>. 2023. URL: <https://xtreme1.io/>.
- [14] Gal, Y., Islam, R., and Ghahramani, Z. *Deep Bayesian Active Learning with Image Data*. 2017. arXiv: 1703.02910 [cs.LG].
- [15] Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., Shahzad, M., Yang, W., Bamler, R., and Zhu, X. X. *A Survey of Uncertainty in Deep Neural Networks*. 2022. arXiv: 2107.03342 [cs.LG].
- [16] Geiger, A., Lenz, P., and Urtasun, R. “Are we ready for autonomous driving? The KITTI vision benchmark suite”. In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.
- [17] He, C., Zeng, H., Huang, J., Hua, X.-S., and Zhang, L. “Structure Aware Single-Stage 3D Object Detection From Point Cloud”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11870–11879. DOI: 10.1109/CVPR42600.2020.01189.
- [18] Hekimoglu, A., Brucker, A., Kayali, A. K., Schmidt, M., and Marcos-Ramiro, A. *Active Learning for Object Detection with Non-Redundant Informative Sampling*. 2023. arXiv: 2307.08414 [cs.CV].
- [19] Hekimoglu, A., Friedrich, P., Zimmer, W., Schmidt, M., Marcos-Ramiro, A., and Knoll, A. C. *Multi-Task Consistency for Active Learning*. 2023. arXiv: 2306.12398 [cs.CV].
- [20] Hekimoglu, A., Schmidt, M., and Marcos-Ramiro, A. *Monocular 3D Object Detection with LiDAR Guided Semi Supervised Active Learning*. 2023. arXiv: 2307.08415 [cs.CV].
- [21] Hekimoglu, A., Schmidt, M., Marcos-Ramiro, A., and Rigoll, G. “Efficient Active Learning Strategies for Monocular 3D Object Detection”. In: *2022 IEEE Intelligent Vehicles Symposium (IV)*. 2022, pp. 295–302. DOI: 10.1109/IV51971.2022.9827454.
- [22] “JSON schema Draft 7 specification”. In: (12 Nov 2021). URL: <https://json-schema.org/draft-07/json-hyper-schema-release-notes>.
- [23] Kao, C.-C., Lee, T.-Y., Sen, P., and Liu, M.-Y. *Localization-Aware Active Learning for Object Detection*. 2018. arXiv: 1801.05124 [cs.CV].
- [24] Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. “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.
- [25] Li, B., Zhang, T., and Xia, T. *Vehicle Detection from 3D Lidar Using Fully Convolutional Network*. 2016. arXiv: 1608.07916 [cs.CV].
- [26] Li, E., Wang, S., Li, C., Li, D., Wu, X., and Hao, Q. “SUSTech POINTS: A Portable 3D Point Cloud Interactive Annotation Platform System”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 1108–1115. DOI: 10.1109/IV47402.2020.9304562.
- [27] Li, Z., Wang, F., and Wang, N. “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.
- [28] Loshchilov, I. and Hutter, F. *Decoupled Weight Decay Regularization*. 2019. arXiv: 1711.05101 [cs.LG].

- [29] Luo, Y., Chen, Z., Wang, Z., Yu, X., Huang, Z., and Baktashmotlagh, M. “Exploring Active 3D Object Detection from a Generalization Perspective”. In: *The Eleventh International Conference on Learning Representations*. 2023. URL: <https://openreview.net/forum?id=2RwXVje1rAh>.
- [30] Miao, Z., Chen, J., Pan, H., Zhang, R., Liu, K., Hao, P., Zhu, J., Wang, Y., and Zhan, X. “PVGNet: A Bottom-Up One-Stage 3D Object Detector with Integrated Multi-Level Features”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 3278–3287. DOI: 10.1109/CVPR46437.2021.00329.
- [31] Nefedov, G. *proAnno - An Automatic and Intelligent 3D Sensor Data Annotation Framework for Autonomous Driving*. Master’s thesis. Munich, Germany, June 2022.
- [32] Qi, C. R., Su, H., Mo, K., and Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].
- [33] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (J3016\_YYYYMMDD)*. 2014. URL: [https://www.sae.org/standards/content/j3016\\_202104/](https://www.sae.org/standards/content/j3016_202104/).
- [34] Sager, C., Zschech, P., and Kühl, N. *labelCloud: A Lightweight Domain-Independent Labeling Tool for 3D Object Detection in Point Clouds*. 2021. arXiv: 2103.04970 [cs.CV].
- [35] Sager, C., Zschech, P., and Kühl, N. *labelCloud: A Lightweight Domain-Independent Labeling Tool for 3D Object Detection in Point Clouds*. 2021. arXiv: 2103.04970 [cs.CV].
- [36] Schmidt, S., Rao, Q., Tatsch, J., and Knoll, A. “Advanced Active Learning Strategies for Object Detection”. In: *2020 IEEE Intelligent Vehicles Symposium (IV)*. 2020, pp. 871–876. DOI: 10.1109/IV47402.2020.9304565.
- [37] Sener, O. and Savarese, S. *Active Learning for Convolutional Neural Networks: A Core-Set Approach*. 2018. arXiv: 1708.00489 [stat.ML].
- [38] Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., and Li, H. “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.
- [39] Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., and Li, H. “PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection”. In: *International Journal of Computer Vision* 131.2 (2023), pp. 531–551.
- [40] Shi, S., Wang, X., and Li, H. “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.
- [41] Shi, W. and Rajkumar, R. “Point-gnn: Graph neural network for 3d object detection in a point cloud”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 1711–1719.
- [42] Sinha, S., Ebrahimi, S., and Darrell, T. *Variational Adversarial Active Learning*. 2019. arXiv: 1904.00370 [cs.LG].
- [43] Song, S., Lichtenberg, S. P., and Xiao, J. “SUN RGB-D: A RGB-D scene understanding benchmark suite”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 567–576. DOI: 10.1109/CVPR.2015.7298655.
- [44] Song, S. and Xiao, J. “Sliding Shapes for 3D Object Detection in Depth Images”. In: *Computer Vision – ECCV 2014*. Ed. by Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. Cham: Springer International Publishing, 2014, pp. 634–651. ISBN: 978-3-319-10599-4.

- [45] Sun, P., Kretschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Tomifeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D. “Scalability in Perception for Autonomous Driving: Waymo Open Dataset”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.
- [46] Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., and Han, S. “Searching efficient 3d architectures with sparse point-voxel convolution”. In: *European conference on computer vision*. Springer. 2020, pp. 685–702.
- [47] Team, O. D. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. <https://github.com/open-mmlab/OpenPCDet>. 2020.
- [48] Transportation, U. S. D. o. “Automated Vehicles for Safety”. In: (2021). URL: <https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.%20Retrieved%20on%2011.09.2023>.
- [49] Urbietta, I., Mujika, A., Piérولا, G., Irigoyen, E., Nieto, M., Loyo, E., and Aginako, N. “WebLabel: OpenLABEL-compliant multi-sensor labelling”. In: *Multimedia Tools and Applications* (2023). [Online]. Available: <https://doi.org/10.1007/s11042-023-16664-4>.
- [50] Wang, B., Wu, V., Wu, B., and Keutzer, K. “Latte: accelerating lidar point cloud annotation via sensor fusion, one-click annotation, and tracking”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2019, pp. 265–272.
- [51] Wang, D. and Shang, Y. “A new active labeling method for deep learning”. In: *2014 International Joint Conference on Neural Networks (IJCNN)*. 2014, pp. 112–119. DOI: 10.1109/IJCNN.2014.6889457.
- [52] Wang, L., Hu, X., Yuan, B., and Lu, J. “Active learning via query synthesis and nearest neighbour search”. In: *Neurocomputing* 147 (2015). Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012 (WSOM 2012), pp. 426–434. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2014.06.042>. URL: <https://www.sciencedirect.com/science/article/pii/S0925231214008145>.
- [53] Welch, G. and Bishop, G. *An Introduction to the Kalman Filter*. Tech. rep. USA, 1995.
- [54] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., Ramanan, D., Carr, P., and Hays, J. *Argoverse 2: Next Generation Datasets for Self-Driving Perception and Forecasting*. 2023. arXiv: 2301.00493 [cs.CV].
- [55] Yan, Y., Mao, Y., and Li, B. “Second: Sparsely embedded convolutional detection”. In: *Sensors* 18.10 (2018), p. 3337.
- [56] Yang, Z., Sun, Y., Liu, S., and Jia, J. “3dssd: Point-based 3d single stage object detector”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 11040–11048.
- [57] Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. “STD: Sparse-to-Dense 3D Object Detector for Point Cloud”. In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 1951–1960. URL: <https://api.semanticscholar.org/CorpusID:198229603>.
- [58] Yin, T., Zhou, X., and Krahenbuhl, P. “Center-based 3d object detection and tracking”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 11784–11793.

- [59] Yoo, D. and Kweon, I. S. *Learning Loss for Active Learning*. 2019. arXiv: 1905.03677 [cs.CV].
- [60] Yu, H., Luo, Y., Shu, M., Huo, Y., Yang, Z., Shi, Y., Guo, Z., Li, H., Hu, X., Yuan, J., and Nie, Z. *DAIR-V2X: A Large-Scale Dataset for Vehicle-Infrastructure Cooperative 3D Object Detection*. 2022. arXiv: 2204.05575 [cs.CV].
- [61] Zhou, Y., Cai, L., Cheng, X., Gan, Z., Xue, X., and Ding, W. *OpenAnnotate3D: Open-Vocabulary Auto-Labeling System for Multi-modal 3D Data*. 2023. arXiv: 2310.13398 [cs.CV].
- [62] Zhou, Y. and Tuzel, O. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4490–4499. DOI: 10.1109/CVPR.2018.00472.
- [63] Zhou, Y. and Tuzel, O. “Voxelnet: End-to-end learning for point cloud based 3d object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4490–4499.
- [64] Zimmer, W., Creß, C., Nguyen, H. T., and Knoll, A. C. *A9 Intersection Dataset: All You Need for Urban 3D Camera-LiDAR Roadside Perception*. 2023. arXiv: 2306.09266 [cs.CV].
- [65] Zimmer, W., Rangesh, A., and Trivedi, M. *3D BAT: A Semi-Automatic, Web-based 3D Annotation Toolbox for Full-Surround, Multi-Modal Data Streams*. 2019. arXiv: 1905.00525 [cs.CV].