

Multi-source aggregation for automatic 3D point cloud annotation in road scenarios

Daniel Górniak, and Robert M. Nowak

Abstract—The creation of high-quality 3D annotations for lidar point clouds involves human experts, it is a time-consuming and costly process, limiting the development of large datasets for autonomous driving. This work reviews existing datasets, annotation tools, and automated and semi-automated methods for generating 3D annotations in road scenarios, with particular emphasis on approaches based on point clouds, multimodal data, and weakly supervised learning. Based on this review, an annotation-support method is proposed that combines 3D detections generated by multiple independent sources. Their outputs were aggregated using an IoU-based strategy, and the resulting annotations were evaluated using Average Precision (AP), Intersection over Union (IoU), and detection statistics. The experiments show that aggregating detections can improve annotation quality when only a small amount of training data is available. As the amount of training data increased, the benefit of aggregation decreased, reaching no improvement for the full training set. These results indicate that combining heterogeneous annotation sources can be an effective way of supporting the creation of 3D point cloud annotations, particularly in the early stages of dataset development.

Keywords—autonomous driving; 3D annotation; point clouds; lidar; weakly supervised learning; annotation aggregation

I. INTRODUCTION

THE development of autonomous driving systems relies heavily on the ability to perceive and interpret the surrounding environment. Among the sensing technologies used for this purpose, LiDAR provides a direct representation of the three-dimensional structure of a scene in the form of a point cloud. This information enables the localization of road users and other objects in three dimensions and is therefore an important input to modern perception systems.

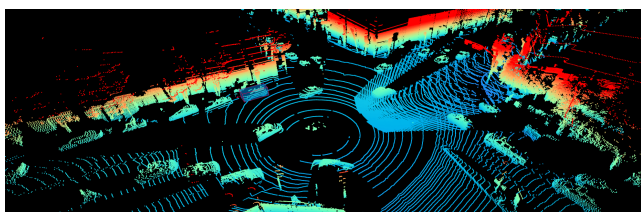


Fig. 1. Point cloud from the PandaSet dataset visualized in segments.ai.

D. Górniak is with Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland (e-mail: daniel.gorniak@pw.edu.pl).

R. Nowak is with Faculty of Electronics and Information Technology, Warsaw University of Technology, Poland (e-mail: robert.nowak@pw.edu.pl).

The performance of learning-based perception models depends strongly on the quality and diversity of the data used for training. In particular, 3D object detection requires annotations describing the position, dimensions, and orientation of objects within the point cloud. Creating such annotations is considerably more demanding than annotating conventional images. LiDAR point clouds are sparse and three-dimensional, and distant or partially occluded objects may be represented by only a small number of measurements. Consequently, manually defining accurate 3D bounding boxes requires substantial time and expertise. As autonomous-driving datasets continue to grow, manual annotation becomes an increasingly significant bottleneck in dataset development.

One approach to reducing this bottleneck is to use automatic annotation methods to generate preliminary 3D bounding boxes. Instead of creating every annotation from scratch, a human annotator can review and correct automatically generated predictions. This changes the annotation task from fully manual label creation to verification and refinement, potentially reducing the time and cost required to construct a dataset. The practical usefulness of such a workflow, however, depends directly on the quality of the automatically generated annotations.

Different automatic annotation methods exploit different properties of the available sensor data and therefore exhibit different error patterns. A LiDAR-based detector may provide accurate localization while failing on objects with few observed points, whereas a method exploiting image information may recover objects that are difficult to detect from LiDAR alone but introduce different localization errors. Consequently, predictions generated by different methods may contain complementary information.

This observation motivates the central research question of this work:

Can the quality of automatically generated 3D annotations be improved by aggregating predictions produced by complementary annotation methods?

Combining predictions from different sources is not, however, equivalent to simply taking their union. Predictions corresponding to the same physical object must first be associated, and the system must determine which detections should be merged and which should remain independent. In addition, the order in which sources are combined and the spatial-overlap threshold used for matching can influence the resulting

annotations. The aggregation strategy therefore becomes an important part of the annotation pipeline.

This work investigates this problem in the context of 3D vehicle annotation for autonomous-driving scenes. Three automatic annotation methods with different characteristics are considered: TED, a LiDAR-based 3D object detector; SphereFormer, a LiDAR-based method employing spherical attention; and FGR, a geometry-based approach combining camera detections with LiDAR measurements. Their predictions are first evaluated independently using different amounts of available training data. The predictions are then combined using a configurable aggregation algorithm based on the spatial overlap between 3D bounding boxes.

The aggregation procedure in our approach is a strategy-search problem. A constrained space of possible combinations of annotation sources and 3D IoU thresholds is defined, and candidate strategies are evaluated to identify configurations that provide the best detection performance. This formulation allows the effect of source ordering, source selection, and overlap thresholds to be studied systematically rather than relying on a manually chosen combination rule.

A second focus of the study is the relationship between training-data availability and the benefit obtained from aggregation. TED and SphereFormer are trained using different numbers of manually annotated scenes, while FGR does not require 3D training annotations. This setup represents a practical dataset-development scenario in which only a limited portion of a newly collected dataset has initially been annotated. The experiments investigate whether aggregation is particularly useful in this low-data regime, where individual learned annotation methods are expected to be less reliable.

Finally, the computational cost of the aggregation procedure is evaluated separately from the annotation-generation methods. In particular, the relationship between the number of 3D IoU calculations and the execution time of the aggregation algorithm is measured. This analysis determines whether the proposed approach remains computationally practical when applied to large-scale annotation datasets.

The main contributions of this work are:

- an experimental comparison of three complementary automatic 3D annotation methods under different amounts of available training data;
- a configurable algorithm for aggregating 3D annotations generated by multiple independent sources;
- a systematic search procedure for identifying effective aggregation strategies from a constrained space of source combinations and 3D IoU thresholds;
- an analysis of how the benefit of annotation aggregation changes with the amount of manually annotated training data; and
- an empirical evaluation of the computational cost of the aggregation procedure in relation to the number of 3D IoU calculations.

The remainder of this paper is organized as follows. Section II discusses existing approaches to autonomous-driving datasets, 3D annotation, automatic annotation, and weakly supervised data generation. Section III describes the dataset,

experimental setup, automatic annotation methods, and evaluation protocol used in this study. Section IV presents the proposed annotation aggregation algorithm, the aggregation-strategy search procedure, and the runtime evaluation methodology. Section V presents and discusses the experimental results, including the individual annotation methods, the aggregation experiments, and the computational analysis. Finally, Section VI summarizes the main findings and discusses directions for future work.

II. RELATED WORK

The annotation of three-dimensional data is an important component of the development of perception systems for autonomous driving. Existing research covers a broad spectrum of approaches, ranging from fully manual annotation tools to systems that automatically generate 3D annotations from LiDAR, camera images, or a combination of multiple sensors.

A. 3D Annotation of Autonomous Driving Data

An important characteristic of modern datasets supporting autonomous driving systems development is the use of multiple types of annotations. Objects can be represented using two-dimensional bounding boxes, three-dimensional bounding cuboids, or segmentation masks assigning semantic labels to individual pixels or LiDAR points. Datasets such as KITTI-360 [1] additionally provide correspondences between observations from different sensors, enabling annotations created in one representation to be transferred to another. The aiMotive dataset [2] represents another example of a large multimodal dataset in which automatic methods were used to generate annotations for training data, while human annotation was retained for validation data.

The increasing size of these datasets makes manual annotation increasingly difficult to scale. Consequently, annotation pipelines have increasingly incorporated algorithms that either assist human annotators or generate initial annotations automatically.

B. Semi-Automatic Annotation Tools

A number of annotation systems aim to reduce the effort required to create 3D annotations while retaining a human annotator in the process. One common approach is to provide interactive operations that automatically fit a cuboid to the points selected by the user. SUSTech Points [3], for example, provides interactive mechanisms for creating 3D bounding boxes from LiDAR point clouds. Other systems focus on reducing the number of interactions required to create an annotation. Nguyen et al. [4] proposed a single-click annotation method that projects the point cloud into a top-down representation and uses an object detector to identify objects around the location selected by the annotator. Another important feature of annotation systems is the exploitation of temporal information. Linear interpolation can be effective for objects with approximately constant motion, although it does not explicitly model changes in velocity or direction. Stumpf et al. [5], for example, investigated a hybrid interpolation approach for RGB-D video sequences. Temporal information can also be exploited through object tracking.

C. Automatic LiDAR-Based Annotation

A different group of methods attempts to generate 3D annotations directly from LiDAR point clouds with little or no human interaction. These methods are closely related to 3D object detection, since a detector produces the location and dimensions of objects in three-dimensional space.

Modern LiDAR-based detectors can be broadly divided according to how they represent the input point cloud. One family of approaches converts the irregular point cloud into a structured representation before applying convolutional or transformer-based networks. Voxel-based methods divide the scene into three-dimensional cells and extract features from occupied voxels. Vote3Deep [6] uses a sparse 3D representation to reduce the computational cost associated with processing empty regions of the scene. PointPillars [7] instead represents the point cloud using vertical columns, allowing efficient processing with architectures originally developed for two-dimensional data.

Another group of methods projects the point cloud onto a horizontal plane. PIXOR [8], for example, represents the scene in a bird's-eye-view representation and predicts objects directly from this representation. Complex-YOLO [9] similarly uses a projected representation containing information such as height, intensity, and point density.

Other detectors operate more directly on individual points. Point-based architectures avoid explicitly converting the complete point cloud into a regular voxel representation and can therefore preserve detailed geometric information. 3DSSD [10] is an example of a point-based single-stage detector, while PointRCNN [11] performs proposal generation directly from point clouds.

In addition to conventional object detection, semantic segmentation can be used as an intermediate representation for 3D annotation. SphereFormer [12], for example, uses a spherical transformer architecture for LiDAR-based 3D recognition. The spherical attention mechanism allows information to be aggregated over a larger spatial region while accounting for the sparse structure of LiDAR data. Such approaches can provide detailed information about the points belonging to individual objects, which can subsequently be used to construct 3D object annotations.

Temporal information provides another source of information for automatic LiDAR annotation. Qi et al. [13] proposed an approach that operates on sequences of point clouds rather than treating each frame independently. Linking observations across time allows the method to exploit both static and dynamic characteristics of objects and to generate annotations for sequences using a limited amount of manually annotated data.

These approaches demonstrate that modern LiDAR-based detectors can generate large numbers of 3D object proposals automatically. However, their performance generally depends on the availability of sufficiently representative training data, creating a close relationship between detector quality and the amount and diversity of manually annotated data.

D. Multimodal 3D Annotation

LiDAR provides accurate geometric information but is sparse, particularly for distant objects. Camera images, in contrast, contain dense visual information but do not directly provide accurate three-dimensional geometry. Consequently, several approaches combine information from both modalities when generating or refining 3D detections.

Multimodal 3D detection methods differ in how information from the two sensors is combined. Multi-view 3D Object Detection [14] combines image and point-cloud features in a common detection framework. Other approaches, such as [15], investigate different strategies for aggregating information from multiple views and sensors. EPNet [16] enhances point features using semantic information obtained from images, while Dual-view 3D Object Recognition [17] combines camera and LiDAR observations for object recognition and detection. Virtual Sparse Convolution [18] further uses image information to generate virtual points that supplement the sparse LiDAR representation.

A related family of methods uses two-dimensional detections as an explicit source of information for generating three-dimensional annotations. F-PointNet [19] first detects objects in an image and projects the resulting 2D bounding boxes into 3D space, producing frustums that constrain the points considered during 3D detection. Frustum ConvNet [20] follows a similar principle, using 2D detections to generate local 3D regions from which 3D object proposals are produced.

FGR [21] approaches the problem without requiring a dedicated 3D detector. The method uses 2D object detections together with geometric reasoning over the corresponding LiDAR points. Ground points are removed, points associated with a 2D detection are segmented, and the resulting point set is used to estimate a 3D bounding box. GAL [22] extends this general direction by using graph-based point grouping and adaptive learning to localize objects in three-dimensional space.

Transformer-based architectures have also been applied to multimodal automatic annotation. MAP-Gen [23] uses multimodal attention to generate additional points and construct 3D annotations from image and LiDAR information. Liu et al. [24] proposed a multimodal transformer for automatic 3D annotation and object detection, while Qian et al. [25] incorporated contextual information into a transformer-based annotation framework. These methods demonstrate the increasing use of attention mechanisms to model relationships between image and point-cloud features.

E. Weakly Supervised 3D Annotation

The methods described above can substantially reduce the amount of manual work required to produce 3D annotations. An important extension is to reduce the amount of manually annotated training data required to construct the automatic annotation system itself. This problem has been studied using weakly supervised learning, where a model is trained using incomplete or less expensive annotations and subsequently used to generate additional 3D annotations.

Several methods discussed in the previous subsection have been evaluated in this setting. FGR [21] and GAL [22] demonstrate that geometric and learning-based methods can generate useful 3D detections without requiring manually provided 3D bounding boxes for every training sample. This is particularly relevant when two-dimensional annotations are available at a substantially lower annotation cost than their three-dimensional counterparts.

Qi et al. [13] investigated the generation of 3D annotations from a limited subset of manually annotated sequences. Their method uses point-cloud sequences to propagate information across frames and generate annotations for previously unannotated data.

MAP-Gen [23], the multimodal transformer proposed by Liu et al., and the context-aware transformer of Qian et al. [25] also demonstrate that a relatively small subset of annotated sequences can be used to generate additional training annotations. The automatically generated annotations can then serve as training data for conventional 3D detectors.

These studies illustrate a common pattern in automatic annotation pipelines: a limited amount of manually annotated data is used to obtain an initial source of supervision, after which an automatic method generates annotations for a substantially larger dataset. The quality of the resulting annotations depends on both the initial supervision and the properties of the method used to generate them. Different approaches can consequently exhibit different error patterns, including differences in the number of missed objects, false detections, localization accuracy, and object coverage.

Overall, existing research demonstrates substantial progress toward reducing the cost of 3D annotation.

III. METHODS

This section describes the dataset, experimental design, automatic annotation methods, and evaluation metrics used in this study. The purpose of the experimental setup is to characterize the quality of individual automatic annotation sources under different amounts of available training data and to provide a common set of predictions for the annotation aggregation experiments described in Section IV.

A. Dataset

The experiments were conducted using the nuScenes dataset [26], a multimodal autonomous-driving dataset containing synchronized measurements from cameras, LiDAR, radar, and other vehicle sensors. The dataset is organized into scenes, where each scene represents a continuous driving sequence composed of temporally ordered samples.

The official nuScenes dataset is divided into training, validation, and test sets. The training set contains 700 scenes, while the validation and test sets contain 150 scenes each. Since ground-truth annotations for the test set are not publicly available, all quantitative experiments in this study were conducted on the validation set.

The experiments were restricted to the *car* category. This restriction reduces the complexity of the experimental analysis while preserving the main characteristics of the 3D annotation

problem. Predictions belonging to other object categories were therefore excluded from the evaluation and subsequent aggregation experiments.

The validation set was used exclusively for evaluation and was not used during model training. To investigate the influence of the amount of available training data, five training-set sizes were considered: 7, 70, 175, 350, and 700 scenes. These subsets correspond to approximately 1%, 10%, 25%, 50%, and 100% of the available training scenes, respectively.

The resulting experimental setting represents a partially annotated dataset scenario. A limited subset of scenes is assumed to have manually generated 3D annotations, which are then used to train automatic annotation models. The trained models can subsequently be applied to the remaining data to generate candidate annotations.

B. Experimental Design

Three automatic annotation methods were selected to provide complementary sources of 3D bounding-box predictions: TED [27], SphereFormer [12], and FGR [21]. TED and SphereFormer operate directly on LiDAR point clouds and require supervised training, whereas FGR combines image-based detections with LiDAR geometry and does not require manually annotated 3D bounding boxes for training.

The experimental procedure consisted of two stages. First, the three annotation methods were evaluated independently to determine how their performance depends on the amount of available training data. Second, the predictions generated by these methods were used as input to the annotation aggregation procedure described in Section IV.

TED and SphereFormer were trained independently for each of the five training-set sizes. For each configuration, only the corresponding subset of the nuScenes training set was used during training. The resulting models were then applied to all scenes in the validation set.

FGR was evaluated independently of the training-set size. Since FGR does not require manually annotated 3D bounding boxes for training, the same FGR configuration was used for all experimental settings.

The predictions generated by all three methods were converted into a common representation containing the estimated 3D bounding-box parameters and the associated confidence score. This representation was used both for the individual evaluation of the annotation methods and as input to the aggregation algorithm.

The original implementations released by the authors of the respective methods were used as the basis of the experiments. Where necessary, the implementations were adapted to support the nuScenes data format and the experimental protocol described above. The trainable models were trained using computational resources provided by the Warsaw University of Technology.

C. Automatic Annotation Methods

The three annotation methods were selected to represent different sources of information and different approaches to generating 3D annotations. The resulting differences in their

input data and processing mechanisms provide complementary prediction sources for the aggregation experiments.

1) *TED*: The implementation provided by the authors was used together with the OpenPCDet framework. The implementation was adapted to the nuScenes dataset and modified to support training only on *car* class and using the five training-set sizes defined in Section III-A.

2) *SphereFormer*: The implementation supplied by the authors was also based on OpenPCDet, allowing the training and inference pipelines of TED and SphereFormer to be implemented within the same framework. The implementation was adapted to the nuScenes dataset and trained using the same five training-set sizes as TED.

Several modifications were introduced to the original configuration. First, velocity prediction was disabled because velocity was not required for the annotation task. Second, the voxel dimensions and maximum number of voxels were modified to obtain a point-cloud representation suitable for the experimental setup.

During preliminary experiments, training the model using all available object classes resulted in better performance than restricting training to the *car* category. Therefore, all object classes were retained during training, while only predictions belonging to the *car* category were retained for evaluation and aggregation.

Due to GPU memory limitations, the batch size was reduced from four to three. Predictions with a confidence score below 0.1 were discarded before further processing.

3) *FGR*: The original FGR implementation was developed for the KITTI dataset and was therefore adapted to operate on nuScenes. An important difference between the two datasets is their camera configuration. KITTI provides a single forward-facing camera, whereas nuScenes contains multiple cameras covering different directions around the vehicle.

In the adapted implementation, the cameras associated with each nuScenes sample were processed independently. When an object was observed by multiple cameras, the resulting 3D bounding boxes were compared using their spatial overlap. Predictions representing the same object were subsequently merged to obtain a single 3D annotation.

Several modifications were also introduced to address geometric failure cases observed during preliminary experiments. When the estimated object height was considered unreliable, a predefined height value was used. In addition, cases in which incorrect vertices were selected while extending the estimated bounding box to the boundaries of the corresponding frustum were detected and corrected. These modifications prevented relevant LiDAR points from being excluded and reduced erroneous displacement of the resulting bounding boxes.

The 2D detections provided to FGR were generated using YOLOv8 trained on the COCO dataset. The confidence score produced by the 2D detector was retained as the confidence score associated with the corresponding FGR prediction.

D. Evaluation Metrics

The quality of the automatically generated annotations was evaluated using three-dimensional Intersection over Union

(IoU) and the official nuScenes Average Precision (AP) metric. The numbers of true positive (TP), false positive (FP), and false negative (FN) detections were additionally recorded to characterize the detection behavior of individual annotation methods.

1) *Three-Dimensional Intersection over Union*: The three-dimensional IoU measures the spatial overlap between a predicted bounding cuboid and the corresponding ground-truth cuboid. It is defined as

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (1)$$

where A denotes the predicted cuboid and B denotes the ground-truth cuboid.

IoU was calculated for matched prediction-ground-truth pairs. The resulting average IoU provides a measure of the geometric accuracy of the generated annotations. TP, FP, and FN counts were additionally recorded to distinguish between differences in localization accuracy and differences in object coverage.

2) *nuScenes Average Precision*: The second evaluation measure was the official nuScenes Average Precision (AP) metric. Unlike conventional IoU-based detection metrics, nuScenes AP determines whether a prediction is a true positive using the two-dimensional distance between the predicted and ground-truth object centers on the ground plane.

Four center-distance thresholds are considered: 0.5, 1, 2, and 4 meters. For each threshold, predictions are matched to ground-truth objects according to their center distance and confidence score. The resulting precision-recall curves are used to calculate AP, and the values obtained for the four thresholds are averaged to obtain the final AP value.

The two evaluation measures capture different properties of the generated annotations. IoU directly evaluates the geometric overlap of predicted and reference cuboids, whereas nuScenes AP evaluates object detection and localization according to the official nuScenes detection protocol. Therefore, both measures were retained throughout the experiments.

E. Individual Annotation Evaluation

The first experimental analysis focused on the individual annotation methods. TED and SphereFormer were evaluated independently for each of the five training-set sizes. Their predictions were generated for the complete nuScenes validation set and evaluated using the metrics described in Section III-D.

FGR was evaluated on the same validation set. Since the method does not require manually annotated 3D training data, its performance was independent of the training-set sizes used for TED and SphereFormer.

These experiments provide the baseline performance of each annotation source before any combination of predictions is performed. The resulting predictions are subsequently used in Section IV to investigate whether combining multiple annotation sources can improve the quality of the generated annotations.

IV. ANNOTATION AGGREGATION

The individual annotation methods described in Section III produce predictions with different characteristics and error patterns. This section describes the proposed procedure for combining these predictions into a single set of 3D annotations.

The aggregation procedure has two objectives. First, it provides a common mechanism for combining predictions originating from an arbitrary number of annotation sources. Second, it enables different aggregation configurations to be evaluated systematically rather than relying on a manually selected combination rule.

A. Aggregation Algorithm

The aggregation algorithm operates independently on each frame and receives as input the predictions generated by the available annotation methods. Each prediction consists of a 3D bounding box and its associated confidence score and source identifier.

An aggregation strategy is represented as an ordered sequence of rules. Each rule specifies a subset of annotation sources and an IoU threshold. The rule determines which predictions should be considered together and the minimum spatial overlap required for them to be treated as observations of the same object.

The order of rules is significant. The algorithm processes the rules sequentially, and predictions that have already been assigned to an aggregated annotation are removed from subsequent processing. This prevents the same prediction from contributing to multiple output annotations.

When predictions from multiple sources are associated with the same object, they are combined into a single annotation according to the aggregation operation defined by the algorithm. Predictions that cannot be associated with another source remain available as individual annotations and are processed according to the subsequent rules in the strategy.

B. Prediction Matching

A rule containing n annotation sources does not perform an exhaustive comparison between all possible pairs of sources. Instead, the first source specified by the rule acts as the reference source and is compared with each of the remaining sources.

Consequently, a rule containing n sources requires $n - 1$ groups of pairwise comparisons, rather than $\frac{n(n-1)}{2}$ comparisons required by an exhaustive pairwise procedure.

A further ambiguity occurs when a prediction from one source overlaps with multiple predictions from another source. Such duplicate associations are resolved by selecting the combination with the highest average IoU. When multiple candidate associations involve the same predictions, cases with a larger number of associated detections are processed first. Once a prediction has been assigned to an aggregated annotation, it cannot participate in another aggregation within the same processing stage.

C. Aggregation Strategy Representation

The aggregation algorithm allows the order of rules, the participating annotation sources, and the IoU threshold of each rule to be varied. Consequently, even with only three annotation sources, a large number of possible aggregation strategies can be constructed.

To make systematic evaluation feasible, the strategy space was constrained to the three IoU thresholds $\{0.2, 0.5, 0.7\}$.

The structure of candidate strategies was further constrained by ordering rules according to the number of participating annotation sources. Rules combining all three sources were considered first, followed by rules combining pairs of sources, and finally rules containing a single source. Individual rules could be omitted from a strategy.

Under these constraints, 7,000 candidate aggregation strategies were generated for each training-set configuration.

The constrained search space provides a compromise between flexibility and computational feasibility. It allows the experiment to evaluate different combinations and processing orders of the three annotation sources without requiring an exhaustive search over an unrestricted space of possible strategies and continuous IoU thresholds.

D. Strategy Search Procedure

The objective of the strategy-search experiment was to identify an aggregation configuration that maximized annotation quality for each training-set size.

Evaluating all candidate strategies on the complete validation set would require a substantial number of repeated IoU calculations. Therefore, a two-stage evaluation procedure was adopted.

In the first stage, all 7,000 candidate strategies were evaluated on the first 30 validation scenes. The resulting AP values were used to rank the candidate strategies. The 100 strategies with the highest AP values were retained for further evaluation.

In the second stage, these 100 candidate strategies were evaluated on the complete validation set. The strategy obtaining the highest AP on the complete validation set was selected as the best-performing aggregation strategy for the corresponding training-set size.

The search was repeated independently for each training-set configuration. Consequently, the selected strategy was allowed to depend on the amount of training data available to the underlying LiDAR-based annotation methods.

The strategy-search procedure is intended to evaluate whether the optimal way of combining annotation sources changes as the quality of the individual sources changes with increasing amounts of training data.

E. Runtime Evaluation

In addition to annotation quality, the computational cost of the aggregation procedure was evaluated. The runtime experiment was designed to characterize the relationship between the number of spatial comparisons performed by the algorithm and its execution time.

The experiment used predictions generated by TED and SphereFormer models trained on all 700 training scenes together with predictions generated by FGR. Different aggregation configurations and different numbers of processed scenes were used to obtain a range of numbers of IoU calculations.

The primary quantity recorded during the experiment was the number of three-dimensional IoU calculations performed by the aggregation algorithm. The total execution time of the aggregation procedure was measured independently of data loading and result serialization.

The measured execution time was then analyzed as a function of the number of IoU calculations. This experiment was used to characterize the computational behavior of the proposed aggregation algorithm and to determine whether its cost remains practical when processing larger numbers of automatically generated annotations.

V. RESULTS

This section presents the experimental results obtained for the three automatic annotation methods and the proposed aggregation algorithm. The analysis is divided into five parts. First, the individual annotation methods are compared using the proposed geometric measures and the official nuScenes Average Precision (AP) metric. Second, the effect of the amount of training data on the two trainable methods is examined. Third, the results of combining predictions from the three annotation sources are presented. The fourth part analyzes the aggregation strategies selected by the strategy-search procedure. Finally, the computational cost of the aggregation algorithm is evaluated.

A. Individual Annotation Methods

Table I summarizes the results obtained by TED, SphereFormer, and FGR. For TED and SphereFormer, results are reported for all five training-set sizes. FGR does not require 3D training annotations and therefore produces the same results independently of the training-set size.

TABLE I
RESULTS OF THE INDIVIDUAL AUTOMATIC ANNOTATION METHODS FOR DIFFERENT NUMBERS OF TRAINING SCENES. TP, FP, AND FN DENOTE THE NUMBERS OF TRUE-POSITIVE, FALSE-POSITIVE, AND FALSE-NEGATIVE DETECTIONS, RESPECTIVELY. AVERAGE IOU IS CALCULATED OVER MATCHED PREDICTIONS. AP DENOTES THE OFFICIAL NUSCENES AVERAGE PRECISION.

Method	Metric	Training scenes				
		7	70	175	350	700
TED	TP	25428	36928	40367	42507	42481
	FP	58653	42872	44778	45623	43873
	FN	54576	43076	39637	37497	37523
	Avg. IoU	0.506	0.585	0.611	0.624	0.634
	AP	0.213	0.468	0.539	0.585	0.595
SphereFormer	TP	38343	47485	51691	52286	53295
	FP	79997	95277	124077	104108	103934
	FN	41661	32519	28313	27718	26709
	Avg. IoU	0.449	0.506	0.537	0.566	0.574
	AP	0.432	0.602	0.678	0.717	0.737
FGR	TP	28870				
	FP	15459				
	FN	51134				
	Avg. IoU	0.438				
	AP	0.275				

The results show clear differences in the behavior of the three annotation methods. SphereFormer produces the largest number of true-positive detections for every training-set size. For the model trained on all 700 scenes, SphereFormer detects 53,295 objects, compared with 42,481 detected by TED. This corresponds to approximately 25% more true-positive detections. At the same time, SphereFormer also produces substantially more false positives: 103,934 compared with 43,873 for TED. Thus, SphereFormer provides greater object coverage at the cost of a substantially larger number of incorrect detections.

The difference between the methods is also reflected in the average IoU. TED achieves a higher average IoU than SphereFormer for every training-set size. With 700 training scenes, the average IoU of TED reaches 0.634, whereas SphereFormer obtains 0.574. This indicates that the detections produced by TED tend to have better geometric overlap with the corresponding ground-truth objects when a prediction is correctly matched.

The behavior of FGR is different from both LiDAR-based methods. It produces 28,870 true-positive detections and only 15,459 false positives, resulting in substantially fewer incorrect detections than either TED or SphereFormer. In particular, FGR produces approximately 0.53 false-positive detections per true positive. For comparison, this ratio is approximately 1.03 for TED trained on 700 scenes and 1.95 for SphereFormer trained on the same number of scenes. FGR therefore provides a substantially more conservative set of annotations.

This characteristic is particularly relevant in an annotation-assisted workflow. A false positive still has to be identified and removed by a human annotator, whereas an object that is not detected at all has to be located and annotated manually. Consequently, a method with a relatively small number of false positives can provide useful initial annotations even when its overall detection coverage is lower than that of a stronger detector.

The AP results provide a different view of the three methods. SphereFormer achieves the highest AP for every training-set size, increasing from 0.432 with 7 training scenes to 0.737 with 700 scenes. TED improves from 0.213 to 0.595, while FGR obtains an AP of 0.275. The consistently higher AP of SphereFormer indicates that its larger number of detections provides a better precision-recall trade-off according to the nuScenes evaluation protocol, despite its lower average 3D IoU.

The difference between the IoU and AP results is important for the interpretation of the annotation methods. IoU directly evaluates the geometric overlap of matched bounding boxes, whereas nuScenes AP uses the distance between the predicted and ground-truth object centers on the ground plane. Therefore, a prediction can contribute positively to AP even when its complete 3D bounding box has relatively limited overlap with the ground-truth cuboid. The two measures consequently provide complementary information about the usefulness of automatically generated annotations.

B. Effect of Training-Set Size

The results demonstrate a strong dependence between the amount of training data and the quality of the two trainable annotation methods. Both TED and SphereFormer improve substantially when increasing the number of training scenes from 7 to 70. The improvement becomes progressively smaller as more training data are added.

For TED, the average IoU increases from 0.506 with 7 training scenes to 0.585 with 70 scenes, and subsequently reaches 0.634 with the complete training set. Similarly, AP increases from 0.213 to 0.468 between 7 and 70 scenes, and finally reaches 0.595 with 700 scenes. The number of true-positive detections increases from 25,428 to 36,928 in the first interval and reaches 42,481 with the complete training set.

SphereFormer follows the same general trend. Its average IoU increases from 0.449 with 7 scenes to 0.506 with 70 scenes and reaches 0.574 with 700 scenes. AP increases from 0.432 to 0.602 between 7 and 70 scenes and eventually reaches 0.737. The number of true-positive detections increases from 38,343 to 53,295 over the full range of training-set sizes.

The largest improvements therefore occur when moving from 7 to 70 training scenes. In contrast, the changes between 350 and 700 scenes are considerably smaller. This suggests that the first portion of the training data provides the models with a large amount of previously unseen variation, while additional scenes increasingly contain examples of patterns already represented in the training data.

The effect is particularly visible in AP. For TED, increasing the training set from 7 to 70 scenes increases AP by 0.255, whereas the increase from 350 to 700 scenes is only 0.010. For SphereFormer, the corresponding improvements are 0.170 and 0.020, respectively. Thus, although additional training data continue to improve both methods, the marginal benefit of additional scenes decreases substantially.

This observation is relevant to the intended annotation scenario. A small initial manually annotated subset can already provide a substantial improvement over a model trained with extremely limited supervision. However, increasing the amount of annotated data remains beneficial, particularly when the objective is to obtain high-quality automatic annotations rather than simply generate a large number of candidate detections.

C. Aggregation Results

The second part of the evaluation investigates whether the predictions generated by TED, SphereFormer, and FGR can be combined to improve the resulting annotations. For each training-set size, the aggregation strategy with the highest AP on the complete validation set was selected using the strategy-search procedure described in Section IV.

Table II presents the resulting TP, FP, FN, average IoU, and AP values.

The aggregation results show that combining the three annotation sources provides the largest benefit when the underlying trainable detectors have access to little training data. With 7 training scenes, aggregation increases AP from 0.432 for

TABLE II
RESULTS OBTAINED USING THE BEST AGGREGATION STRATEGY FOR EACH TRAINING-SET SIZE.

Metric	7	70	175	350	700
TP	38594	47497	51693	52286	53293
FP	80469	95265	124075	104108	103936
FN	41410	32507	28311	27718	26711
Avg. IoU	0.486	0.536	0.536	0.571	0.570
AP	0.454	0.611	0.680	0.718	0.737

SphereFormer to 0.454, corresponding to an absolute improvement of 0.022, or approximately 5.1% relative improvement. The number of true-positive detections also increases from 38,343 to 38,594.

The benefit becomes smaller as the amount of training data increases. At 70 scenes, AP increases from 0.602 to 0.611, while at 175 scenes it increases from 0.678 to 0.680. With 350 scenes, the aggregation result is 0.718 compared with 0.717 for SphereFormer. Finally, with 700 training scenes, both SphereFormer and the aggregation strategy achieve an AP of 0.737. Thus, no AP improvement is obtained from aggregation in the full-data setting.

This trend indicates that aggregation is most useful when the individual annotation methods contain complementary errors. When the strongest individual method is trained using only a small amount of data, additional sources can provide detections that are absent from its output. As the training set becomes larger, SphereFormer itself becomes increasingly effective and the additional information provided by TED and FGR becomes less valuable under the AP objective.

The geometric measures support this interpretation. For 700 training scenes, the aggregation result is almost identical to SphereFormer: the number of true positives is 53,293 compared with 53,295 for SphereFormer, while the number of false positives is 103,936 compared with 103,934. The average IoU is also similar, with values of 0.570 and 0.574, respectively. Therefore, the selected aggregation strategy effectively reproduces the behavior of the strongest individual source in the full-data setting.

At lower training-set sizes, aggregation can increase the number of detected objects, but this does not necessarily result in an improvement in geometric accuracy. For example, with 7 training scenes the aggregation method increases the number of true positives substantially relative to TED, but its average IoU is lower than that of TED. This again illustrates the distinction between increasing detection coverage and improving the geometric accuracy of individual bounding boxes.

D. Strategy-Search Results

The aggregation algorithm provides a configurable sequence of merging rules. The constrained search described in Section IV was used to identify the strategy that maximized AP for each training-set size. The selected strategies are shown below:

A clear trend can be observed in the complexity of the selected strategies. The strategy selected for 7 training scenes contains five rules, whereas the strategies selected for 175, 350, and 700 scenes contain only two rules. Thus, as the

TABLE III
BEST AGGREGATION STRATEGIES FOR DIFFERENT TRAINING-SET SIZES.
THE NUMBER IN PARENTHESES DENOTES THE IOU THRESHOLD.

Scenes	Aggregation rule
7	TED + FGR + SphereFormer (0.2)
	TED + SphereFormer (0.7)
	SphereFormer + FGR (0.2)
	TED + FGR (0.2)
	SphereFormer
70	TED + FGR + SphereFormer (0.5)
	TED + SphereFormer (0.5)
	SphereFormer + FGR (0.2)
	SphereFormer
175	SphereFormer + FGR (0.5)
	SphereFormer
350	TED + FGR + SphereFormer (0.5)
	SphereFormer
700	SphereFormer + FGR (0.5)
	SphereFormer

quality of the underlying trainable detectors improves, the search procedure tends to select simpler aggregation strategies.

An important common characteristic is that every selected strategy ends with the rule containing only SphereFormer. This rule ensures that all SphereFormer predictions that were not incorporated into an earlier merge remain in the final annotation set. This behavior is consistent with SphereFormer’s superior AP for every training-set size. Since the search objective is AP, retaining the complete set of SphereFormer predictions provides a strong baseline on top of which additional predictions from TED and FGR can be incorporated.

The selected strategies also show that the aggregation algorithm does not simply favor a fixed IoU threshold. For the 7-scene and 70-scene configurations, multiple thresholds and combinations of annotation sources are used. With more training data, however, the strategy becomes progressively simpler and relies increasingly on SphereFormer predictions.

This result provides evidence that the proposed strategy-search procedure can adapt the aggregation process to the quality of its input annotation sources. Rather than assuming that all methods should contribute equally, the search identifies combinations that are appropriate for the current quality of the underlying detectors.

E. Runtime Results

The computational cost of the aggregation algorithm was evaluated separately from the cost of generating the individual annotations. The runtime experiment focused on the calculation of 3D IoU values between predictions originating from different annotation sources, which represents the main computational operation performed by the aggregation procedure.

The experiment used predictions generated by TED and SphereFormer models trained on all 700 training scenes together with predictions generated by FGR using YOLOv8 detections. Different aggregation strategies and numbers of scenes were used to obtain different numbers of IoU calculations. Only the execution time of the aggregation algorithm was measured; loading the prediction files and saving the resulting annotations were excluded.

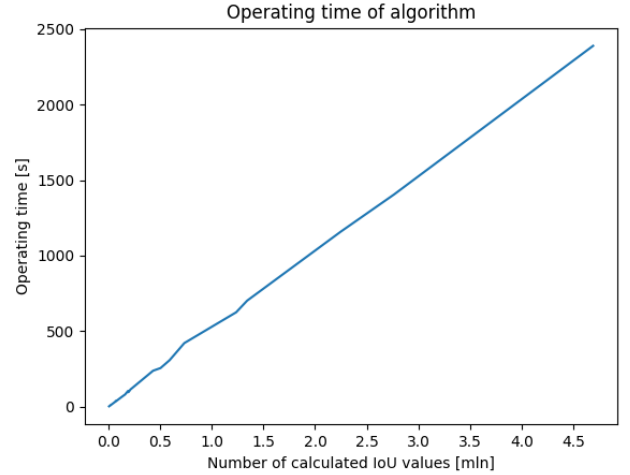


Fig. 2. Runtime of the aggregation algorithm as a function of the number of calculated 3D IoU values.

Figure 2 shows an approximately linear relationship between the number of IoU calculations and the total execution time. This behavior is consistent with the structure of the proposed aggregation algorithm, where the dominant operation consists of independent pairwise comparisons between detections selected by the aggregation rules.

The measured runtime remains practical for the scale considered in this study. For approximately 4.5 million IoU calculations, the execution time does not exceed 2,500 s, corresponding to approximately 42 minutes. Since the aggregation operates on already generated predictions and does not include the substantially more expensive model training or inference stages, this computational cost is acceptable for an offline annotation pipeline.

The runtime experiment also supports the use of the constrained aggregation strategy described in Section IV. By limiting the comparisons performed by each rule to the reference source and the remaining sources, the number of IoU calculations can be reduced compared with exhaustive pairwise comparison of all sources. The approximately linear scaling observed experimentally indicates that the computational cost can be estimated from the number of comparisons generated by a given strategy.

Overall, the runtime results indicate that the proposed aggregation procedure does not introduce a prohibitive computational overhead. Its execution time is substantially smaller than the time required to manually create the corresponding 3D annotations, making the approach suitable for large-scale offline annotation workflows.

VI. CONCLUSION

This work investigated whether automatically generated 3D annotations can be improved by combining predictions produced by complementary annotation methods. The experimental results demonstrate a strong relationship between the amount of training data and the quality of automatically generated annotations. Three proposed methods provide complementary characteristics that are exploited through aggregation.

The aggregation procedure was selected automatically, and it can improve detection performance, particularly when only a small amount of training data is available. With 7 training scenes, the AP increased from 0.432 for SphereFormer to 0.454 after aggregation, corresponding to an improvement of approximately 5.1%. The improvement for 70 scenes was to 0.611 from 0.602 and became marginal for larger training sets. With all 700 training scenes, aggregation did not improve the AP over SphereFormer, with both approaches achieving an AP of 0.737.

The selected aggregation strategies also provide an important insight into the behavior of the proposed approach. As the amount of training data increased, the selected strategies generally became simpler, indicating that the additional information provided by the other annotation sources became less valuable once the strongest individual detector had access to sufficient training data. This suggests that annotation aggregation is most useful in the low-data regime, where individual models have not yet learned a sufficiently broad representation of the target objects.

Several limitations provide directions for future work. First, the current aggregation procedure relies primarily on spatial overlap and does not explicitly incorporate the confidence scores produced by the individual annotation methods. Incorporating prediction confidence could allow more reliable detections to have a greater influence on the resulting annotation. Second, the current method does not explicitly identify all false positive detections. Additional criteria, such as the number and distribution of LiDAR points within a predicted bounding box or temporal consistency obtained through object tracking, could be investigated to improve false positive filtering. Third, reducing the computational cost of IoU calculations would make it possible to explore a larger space of aggregation strategies and potentially identify better combinations of annotation sources.

Overall, the results indicate that combining multiple automatic annotation sources can provide a useful complement to individual 3D detectors, particularly when only a small manually annotated training set is available.

REFERENCES

- [1] Y. Liao, J. Xie, and A. Geiger, “Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3292–3310, 2022.
- [2] T. Matuszka, I. Barton, Á. Butykai, P. Hajas, D. Kiss, D. Kovács, S. Kunsági-Máté, P. Lengyel, G. Németh, L. Pető *et al.*, “aimotive dataset: A multimodal dataset for robust autonomous driving with long-range perception,” *arXiv preprint arXiv:2211.09445*, 2022.
- [3] E. Li, S. Wang, C. Li, D. Li, X. Wu, and Q. Hao, “Sustech points: A portable 3d point cloud interactive annotation platform system,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1108–1115.
- [4] T. Nguyen, B.-S. H. Duc, T. Nguyen, and D. Phung, “Single-click 3d object annotation on lidar point clouds,” in *Proc. NeurIPS DCAI Workshop*, 2021.
- [5] D. Stumpf, S. Krauß, G. Reis, O. Wasenmüller, and D. Stricker, “Salt: A semi-automatic labeling tool for rgb-d video sequences,” *arXiv preprint arXiv:2102.10820*, 2021.
- [6] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1355–1361.
- [7] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” in *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, 2019, pp. 12 689–12 697.
- [8] B. Yang, W. Luo, and R. Urtasun, “Pixor: Real-time 3d object detection from point clouds,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 7652–7660.
- [9] M. Simon, S. Milz, K. Amende, and H.-M. Gross, “Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds,” in *European Conference on Computer Vision*. Springer, 2018, pp. 197–209.
- [10] Z. Yang, Y. Sun, S. Liu, and J. Jia, “3dssd: Point-based 3d single stage object detector,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020, pp. 11 037–11 045.
- [11] S. Shi, X. Wang, and H. Li, “Pointcnn: 3d object proposal generation and detection from point cloud,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 770–779.
- [12] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia, “Spherical transformer for lidar-based 3d recognition,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023, pp. 17 545–17 555.
- [13] C. R. Qi, Y. Zhou, M. Najibi, P. Sun, K. Vo, B. Deng, and D. Anguelov, “Offboard 3d object detection from point cloud sequences,” in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2021, pp. 6130–6140.
- [14] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “January 21-26). multi-view 3d object detection network for autonomous driving. proceedings of the ieee conference on computer vision and pattern recognition, honolulu, hi, usa,” 2017.
- [15] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, “Joint 3d proposal generation and object detection from view aggregation,” in *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018, pp. 1–8.
- [16] T. Huang, Z. Liu, X. Chen, and X. Bai, “Epnet: Enhancing point features with image semantics for 3d object detection,” in *European conference on computer vision*. Springer, 2020, pp. 35–52.
- [17] J. Li, R. Li, J. Li, J. Wang, Q. Wu, and X. Liu, “Dual-view 3d object recognition and detection via lidar point cloud and camera image,” *Robotics and Autonomous Systems*, vol. 150, p. 103999, 2022.
- [18] H. Wu, C. Wen, S. Shi, X. Li, and C. Wang, “Virtual sparse convolution for multimodal 3d object detection,” in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023, pp. 21 653–21 662.
- [19] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum pointnets for 3d object detection from rgb-d data,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, 2018, pp. 918–927.
- [20] Z. Wang and K. Jia, “Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1742–1749.
- [21] Y. Wei, S. Su, J. Lu, and J. Zhou, “Fgr: Frustum-aware geometric reasoning for weakly supervised 3d vehicle detection,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4348–4354.
- [22] D. Yin, H. Yu, N. Liu, F. Yao, Q. He, J. Li, Y. Yang, S. Yan, and X. Sun, “Gal: Graph-induced adaptive learning for weakly supervised 3d object detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 9, pp. 9684–9697, 2023.
- [23] C. Liu *et al.*, “Map-gen: An automated 3d-box annotation flow with multimodal attention point generator,” in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 1148–1155.
- [24] —, “Multimodal transformer for automatic 3d annotation and object detection,” in *European conference on computer vision*. Springer, 2022, pp. 657–673.
- [25] X. Qian, C. Liu, X. Qi, S.-C. Tan, E. Lam, and N. Wong, “Context-aware transformer for 3d point cloud automatic annotation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 2082–2090.
- [26] H. Caesar *et al.*, “nuscenes: A multimodal dataset for autonomous driving,” in *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. IEEE, 2020, pp. 11 618–11 628.
- [27] H. Wu, C. Wen, W. Li, X. Li, R. Yang, and C. Wang, “Transformation-equivariant 3d object detection for autonomous driving,” 2022. [Online]. Available: <https://arxiv.org/abs/2211.11962>