

CADC++: Extending CADc with a Paired Weather Domain Adaptation Dataset for 3D Object Detection in Autonomous Driving

by

Mei Qi Tang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

© Mei Qi Tang 2024

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The thesis is based on collaborative work by the author, Chengjie Huang, Dr. Vahdat Abdelzad, Dr. Michal Antkiewicz, Prof. Sean Sedwards, and Prof. Krzysztof Czarnecki. Each collaborator’s contributions are detailed below.

Chengjie Huang, Dr. Vahdat Abdelzad, and Prof. Sean Sedwards provided guidance and insights on 3D object detection, domain adaptation, semi-supervised learning, and auto-labelling. Chengjie Huang helped me familiarize with the training environment, tools, and frameworks. He also provided useful data manipulation and visualization scripts. Dr. Michal Antkiewicz sets up and maintains servers for model training and storing the datasets. He selected the third-party labelling company, Segments.ai and wrote scripts necessary to convert and upload our dataset on their labelling platform. Additionally, Dr. Michal Antkiewicz helped me debug the ROS and sensor calibration issues we encountered during dataset development. Prof. Krzysztof Czarnecki supervised the overall projects, making sure the budget and timelines aligned. Prof. Sean Sedwards provided more fine-grained guidance for each project, ensuring the methodology and experiments were adequate, logical, and scientific. He also mainly developed the sequence matching algorithms for selecting CADC++ sequences and wrote a great part of Sect. 4.2. Finally, Prof. Sean Sedwards and Prof. Krzysztof Czarnecki provided feedback and edits on the writing and the figures.

Abstract

Lidar sensors enable precise 3D object detection for autonomous driving under clear weather but face significant challenges in snowy conditions due to signal attenuation and backscattering. While prior studies have explored the effects of snowfall on lidar returns, its impact on 3D object detection performance remains underexplored. Conducting such an evaluation objectively requires a dataset with abundant labelled data from both weather conditions and ideally captured in the same driving environment. Current driving datasets with lidar data either do not provide enough labelled data in both snowy and clear weather conditions, or rely on simulation methods to generate data for the weather domain with insufficient data. Simulations, nevertheless, often lack realism, introducing an additional domain shift that impedes accurate evaluations. This thesis presents our work in creating CADC++, a paired weather domain adaptation dataset that extends the existing snowy dataset, CADC, with clear weather data. Our CADC++ clear weather data have been recorded on the same roads and around the same days as CADC. We pair each CADC sequence with a clear weather one as closely as possible, both spatially and temporally. Our curated CADC++ achieves similar object distributions as CADC, enabling minimal domain shift in environmental factors beyond the presence of snow. Additionally, we propose track-based auto-labelling methods to overcome a limited labelling budget. Our approach, evaluated on the Waymo Open Dataset, achieves a balanced performance across stationary and dynamic objects and still surpasses a standard 3D object detector when using as low as 0.5% of human-annotated ground-truth labels.

Acknowledgements

I would like to thank Prof. Krzysztof Czarnecki, Prof. Sean Sedwards, Dr. Vahdat Abdelzad, Dr. Michal Antkiewicz, and Chengjie Huang for the thorough discussions, guidance, and constructive feedback during our weekly meetings. It has allowed me to learn quickly and also shaped me into a better researcher. I especially thank Prof. Sean Sedwards for closely mentoring and guiding me into doing high-quality research, including teaching me the methodology and tools necessary for diagnosing issues and planning experiments. Moreover, I would like to thank Chengjie Huang for the huge help in showing me the tools, frameworks, and relevant knowledge needed to get started in this line of research.

Finally, I am forever grateful to my parents and Zak for being tremendously supportive, understanding, and accommodating during this journey.

Dedication

To my loved ones, whose patience, understanding, and sacrifices made this journey possible.

Table of Contents

Author's Declaration	ii
Statement of Contributions	iii
Abstract	iv
Acknowledgements	v
Dedication	vi
List of Figures	x
List of Abbreviations	xiii
1 Introduction	1
2 Background	5
2.1 Lidar Sensors	5
2.2 Lidar Sensors in Adverse Weather	6
2.3 Lidar-based 3D Object Detection	8
2.4 The CADC Dataset	9
2.5 Clear Weather Data from the 1000 km Drives	11
2.6 Track-based auto-labelling pipeline	14
2.7 CTRL	15

3	Related Work	17
3.1	Autonomous Driving Datasets	17
3.2	Snowy Weather Driving Datasets	18
3.3	Simulated Weather Domain Adaptation Datasets	18
3.3.1	Generating Clear Weather Datasets	19
3.3.2	Generating Snowy Weather Datasets	19
3.4	Auto-labelling	20
3.5	Sparsely labelled datasets	20
4	The CADC++ Dataset	22
4.1	Problem Statement	22
4.2	Weather-paired Dataset Matching Method	22
4.2.1	Spatial Matching	23
4.2.2	Temporal Matching	26
4.3	Dataset Labelling Strategy	28
4.4	Pre-labelling for Segments.ai	28
4.4.1	Averaging static tracks	31
4.5	Dataset statistics	33
5	Track-based Auto-labelling for Sparsely Labelled Data	39
5.1	Problem statement	39
5.2	Methods	40
5.2.1	Dataset and sequence subsampling	41
5.2.2	Notation	42
5.2.3	Full supervision baselines: $Base_{100\%}$ & $CTRL_{100\%}$	43
5.2.4	Sparse supervision models: $Base_{sparse}$ & $CTRL_{sparse}$	43
5.2.5	Semi-supervision	44
5.2.6	Evaluation	47

5.3	Experiments	47
5.3.1	Base performance	48
5.3.2	CTRL performance	49
5.4	Discussion	50
6	Conclusion and Future Work	52
	References	53

List of Figures

2.1	Overview of ToF-based Light Detection And Ranging (LiDAR) sensors [65]	6
2.2	Decreasing visibility from normal conditions (dotted lines) for different rain-fall intensities, target wetness, and LiDAR sensors [65].	7
2.3	Effects of snowfall on LiDAR point clouds [84, 28]. The voids result from strong spurious snow returns at close ranges blocking target returns at further ranges.	7
2.4	Overview of the 3D object detection pipeline in autonomous driving applications [38].	8
2.5	A LiDAR point cloud and a front camera image from an example frame in Canadian Adverse Driving Conditions (CADC) [46]. The red ground-truth 3D bounding boxes are projected for visualization purposes only. The cloud of white particles around the ego vehicle (yellow prism) are LiDAR returns from falling snow.	9
2.6	Sensor setup of the Autonomoose vehicle platform for CADC’s data collection.	10
2.7	Example of a clear weather sequence recorded on the same road as a snowy sequence from CADC.	11
2.8	Geographical coverage overview of the clear weather data compared to CADC. Clear weather data recorded in further cities like Cambridge and Guelph were not considered in this work.	13
2.9	Overview of a track-based auto-labelling pipeline. The snowflake icon refers to a trained model with frozen weights, used in inference mode. The globe with axes symbolizes that those data have been transformed into the global world frame.	14
2.10	Architectural overview of CTRL’s refinement module.	15

4.1	Demonstrating the main steps of our spatial and temporal matching methods using visual examples.	23
4.2	Some examples of snowy CADC sequences matched with clear weather sequences in CADC++. Each sequence is assigned a number as its unique ID. Below the sequence name, we indicate the number of frames and sampling frequency that were selected as part of the match. d_{max} is the distance between the furthest two points from a CADC sequence and its clear weather match.	27
4.3	Precision-Recall (PR) curves for the pre-labelling models evaluate on CADC's validation set. Each PR curve demonstrates the performance of a single-class model that was pre-trained on the Waymo Open Dataset and then fine-tuned on CADC. The score thresholds s maximize the F1 scores, the harmonic mean between precision and recall.	29
4.4	Comparison of our generated pre-labels with the ground-truth labels corrected by human annotators.	30
4.5	Common errors and inaccuracies in our pre-labels.	30
4.6	Coverage overview of matched sequences from CADC++ and CADC. Overlapping sequences were successfully matched spatially and non-overlapping ones were manually selected according to the similarity of scenes and object distributions.	33
4.7	Number of object instances per labelled category in CADC and CADC++.	34
4.8	Average speed distributions of object tracks in CADC and CADC++.	36
4.9	Distributions of point count per annotation and object count per frame for the Car class in CADC and CADC++.	37
4.10	Distributions of point count per annotation and object count per frame for lesser represented classes in CADC and CADC++.	38
5.1	Overview and relationships between the methods and the resulting models we train, classified by their architecture and supervision. Methods with a dotted border are our baselines; our proposed methods have bolded borders. We use Base _{sparse} as the first stage detector in all our non-baseline CTRL experiments to obtain comparable results. In practice, for best results, the highest-performing Base model should be used to produce the best possible initial detections.	40

5.2	Sparse ground-truth (GT): numbers identify labelled frames for given frames per <i>sequence</i> (fps); dots indicate 9fps, with 20 frames between any two adjacent dots.	42
5.3	Constructing pseudo-label tracks for hybrid semi-supervised model training. (a) A sparse base detector infers initial pseudo-labels. (b) All object tracks are interpolated from sparse GT labels. (c) Only interpolated labels from stationary track segments are selected. (d) Only pseudo-labels that do not overlap with those from (c) are selected. (e) All selected interpolated labels and pseudo-labels are combined with sparse GT labels before being fed into the tracker in (f) to generate pseudo-label tracks.	46
5.4	Level 2 vehicle 3D AP vs GT label frequency for different training methods. Base _{int} and Base _{int+ssl} show similar performance trends to their corresponding CTRL experiments, however their curves are excluded to reduce clutter. Base results are in lighter equivalent colours than CTRL results.	48
5.5	Qualitative detection performance of our track-based auto-labeller trained on sparsely labelled data compared to a standard detector trained on fully labelled data. Our method effectively refines detections along an object track, reducing the overall jitteriness that standard detectors might introduce when not considering the object's full trajectory.	50

List of Abbreviations

AVs autonomous vehicles [6](#)

BEV Bird’s Eye View [9](#)

CADC Canadian Adverse Driving Conditions [x](#), [xi](#), [1](#), [3](#), [4](#), [8–13](#), [18](#), [20](#), [22](#), [24](#), [26–30](#), [33–38](#), [52](#)

EM electromagnetic [6](#)

FOV field-of-view [6](#)

fps frames per *sequence* [xii](#), [3](#), [42](#), [43](#), [48–51](#)

GT ground-truth [xii](#), [16](#), [31](#), [41–48](#), [50](#)

LiDAR Light Detection And Ranging [x](#), [1](#), [2](#), [4](#), [6–10](#), [12](#), [14](#), [19](#), [20](#), [28](#), [39](#), [44](#), [45](#)

MOT multi-object tracker [14](#), [15](#)

NMS non-maximum suppression [41](#)

PCs point clouds [42](#), [43](#)

PR Precision-Recall [xi](#), [29](#)

SOTA state-of-the-art [3](#), [14](#), [15](#), [29](#)

ToF time-of-flight [5](#)

Chapter 1

Introduction

Under clear weather conditions, [LiDAR](#) sensors can generate accurate 3D maps of the environment by measuring distances using laser pulses, making it ideal for 3D object detection in autonomous driving. However, adverse weather conditions might affect the laser pulses that these devices emit. More specifically, hydrometeor particles such as snowflakes have been shown to affect [LiDAR](#) scans due to signal attenuation and backscattering, resulting in potential additions of extra spurious points, a reduced number of signal returns, and points with weaker intensity [\[53, 28, 33\]](#). While these studies have explored the effects of snowfall on [LiDAR](#) sensors and their resulting point clouds, research on its impact on 3D object detectors remains limited.

To ensure objective analysis of the impact of snowy conditions, 3D object detectors need to be evaluated on weather-paired datasets, where snow is the only varying factor, with otherwise identical scenes. However, creating such datasets is impractical without collecting data in controlled environments, which is unrealistic and unrepresentative of real-world driving conditions. Current weather domain adaptation datasets are unpaired and often either contain only data from the adverse domain [\[46, 5\]](#) or do not have enough labelled data in one of the domains to train an object detector [\[5, 16, 4\]](#). To fill this gap, we build CADC++, a snowy weather-paired dataset that extends [CADC](#) [\[46\]](#), by matching each [CADC](#) sequence from the snowy weather domain to a new clear weather sequence to the best of our abilities. Even though CADC++ data was recorded on the same roads as [CADC](#) on neighbouring days, the scenes remain not identical due to the uncontrollable nature of the public environment. However, we devise matching algorithms to select sequences that minimize spatial and temporal discrepancies. The matched CADC++ sequences display object type and speed distributions similar to those in [CADC](#), enabling domain adaptation experiments with minimal domain shift from environmental factors beyond snow.

After selecting the CADC++ sequences, they required annotation. However, we could not afford human-labelled ground-truths for all new CADC++ frames due to budget limitations. To mitigate this, we opt for sparsely labelling the training set, then we explore methods that automatically generate high-quality labels for the unlabelled frames. A fully labelled validation set is still necessary to validate our auto-labelling model.

The following methods use existing labelled frames or other annotated datasets to automatically generate labels for the unlabelled frames, thereby reducing the need for human annotation.

- *Pseudo-labelling*, where a pre-trained model is used to detect and (pseudo-)label objects in raw sensor data; unlike human-annotated labels, pseudo-labels do not inherently assign unique track IDs to objects.
- *Semi-supervised learning*, where pseudo-labels from a base model are used in conjunction with human annotations to train a new model that potentially produces better pseudo-labels.

In most autonomous driving data collection platforms, [LiDAR](#) sensors are mounted on the vehicle and capture sequences of frames at a fixed frequency, typically ≥ 10 Hz, as the vehicle drives. This setup produces a dynamic perspective, with each frame offering a slightly different viewpoint of the scene. In addition to the above methods, the following approaches can be employed to further enhance pseudo-labelling quality by exploiting the temporal continuity and the smoothly varying perspectives of sequential [LiDAR](#) data.

- *Interpolation*, where temporally intermediate annotations are automatically inferred from the available human annotations within a sequence; interpolated objects inherit the human-labelled identities and dimensions of the objects from which they were interpolated.
- *Track-based auto-labelling* is a form of offline pseudo-labelling that takes advantage of the track-based nature of sequential data; having access to both past and future trajectories of objects achieves levels of performance that are unachievable with online approaches that only have access to the past.

To best auto-label CADC++, we adapt track-based auto-labelling for sparse labelling and propose methods that make use of interpolation and semi-supervised learning to achieve competitive quality as ground-truth labels.

For this work, we experiment with the Waymo Open Dataset [61] instead of CADC, given its larger scale and more diverse data to yield more objective results. We analyze various techniques, comparing between object-based and track-based supervision. Moreover, we explore models trained with full supervision, sparse supervision, and semi-supervision. We employ a standard CenterPoint [80] architecture for our base models, and choose state-of-the-art (SOTA) method CTRL [21] as the basis of our track-based auto-labelling pipeline. Since the Waymo Open Dataset is fully labelled, we can study the effects of sparse labelling by subsampling the sequences to obtain our desired sparsity levels. We trim Waymo sequences to 169 frames and consider labelling frequencies of (1, 2, 3, 5, 9) fps, corresponding to approximately (0.5, 1.0, 1.5, 2.5, 4.5)% of the data. These levels of sparsity are sufficient to illustrate interesting phenomena that we believe will also occur with other datasets. Choosing fewer than one frame per sequence is equivalent to choosing a subset of the sequences, which would create a somewhat orthogonal and out-of-scope problem. The performance we achieve using 9 fps with simple training approaches leaves little headroom for further improvements, making higher frequencies uninteresting in the present context.

Our main contributions can be summarized as follows:

- We develop CADC++, a paired weather domain adaptation dataset that matches clear weather sequences to the snowy sequences from CADC.
- We devise sequence matching and selection algorithms for minimizing spatial discrepancies between two trajectories while satisfying a minimum sequence length.
- Our CADC++ shares a similar dataset size and object distributions as CADC, enabling domain adaptation experiments with minimal domain shift other than that caused by snowy conditions.
- We implement a sparse labelling strategy and adapt a track-based auto-labelling pipeline to use sparsely labelled sequential data during training.
- We conduct experiments with extreme levels of label sparsity, using as little as 0.5% labelled data, which we believe have not been explored before.
- We investigate sparse labels and three pseudo-label generation methods for training an auto-labelling model, settling on an approach that combines interpolation and semi-supervised learning.
- We thus propose track-based auto-labelling methods that work well with very sparsely labelled data and that outperform a standard detector trained on fully labelled data. Our methods also exhibit balanced detection on stationary and dynamic vehicles.

We organize this thesis according to the following structure:

- Chapter 2 provides the general background about [LiDAR](#) sensors, adverse weather, 3D object detection, the [CADC](#) dataset, our clear weather data, and track-based auto-labelling.
- Chapter 3 introduces the related work in various types of autonomous driving datasets and auto-labelling.
- Chapter 4 details the development process of our weather-paired dataset CADC++, including our matching algorithms, sparse labelling strategy, dataset pre-labelling methods, and dataset statistics.
- Chapter 5 presents our work in developing track-based auto-labelling methods for sparsely labelled data to overcome budget limitations.
- Lastly, in Chapter 6, we conclude and outline our plan for future work.

Chapter 2

Background

This chapter provides the necessary background for understanding our contributions in both the dataset development of CADC++ and track-based auto-labelling for sparsely labelled data.

2.1 Lidar Sensors

Lidar sensors measure distances by emitting laser light pulses and analyzing the time it takes for them to return after reflecting off objects, also called the [time-of-flight \(ToF\)](#) principle [65]. Illustrated in Fig. 2.1, light pulses are emitted from laser diodes, then reflect off an object, and finally are received by a photosensor receiver. When the light pulses are emitted, the internal clock's timing circuit gets triggered and starts measuring the elapsed time. The clock only gets disabled when the reflected light pulse is received.

The distance R to a reflection point can be calculated with:

$$R = \frac{1}{2n}c\Delta t$$

where n is the index of refraction of the propagation medium (≈ 1 for air), c is the speed of light in a vacuum, and Δt is the round-trip ToF. Then, the received power $P(R)$ of a pulse reflected at distance R can be obtained with:

$$P(R) = P_0 \rho \frac{A_0}{\pi R^2} \mu_0 e^{-2\gamma R}$$

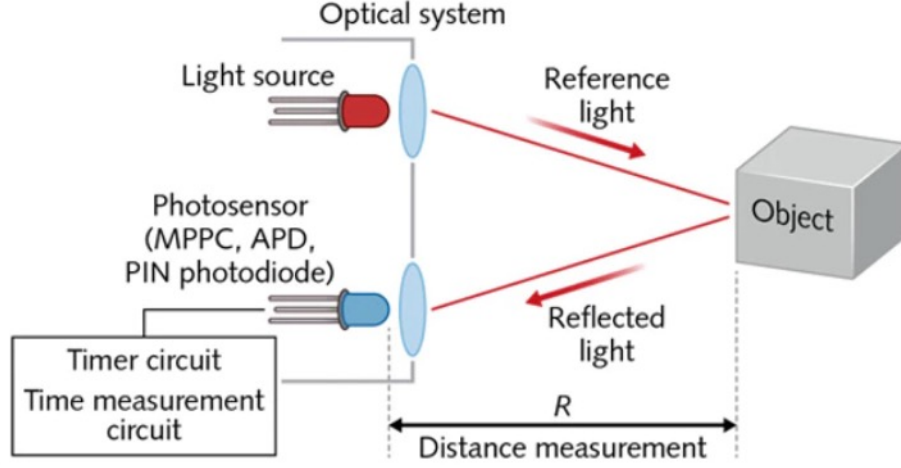


Figure 2.1: Overview of ToF-based LiDAR sensors [65]

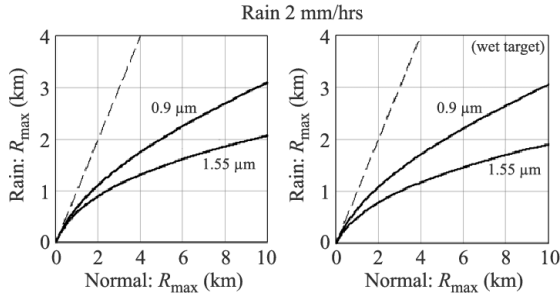
where P_0 is the peak output power of transmitted laser pulse, ρ is the target's reflectivity, A_0 is the receiver's aperture area, μ_0 is the receiving optics spectral transmission, and γ is the atmospheric extinction coefficient [65].

Common lidars in autonomous vehicles (AVs) operate at either around 905 nm or 1550 nm wavelengths [71] and have either a mechanically spinning mechanism or use flash technology to flood the scene within the field-of-view (FOV) of the detector with light [65]. The LiDAR sensor involved in this work is a Velodyne VLP-32C mechanical LiDAR that has 32 laser channels, rotates at 600 rpm, and operates at 903 nm [46].

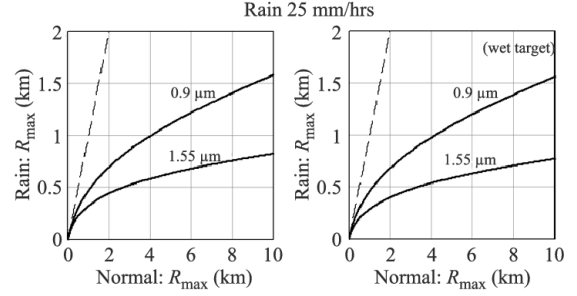
2.2 Lidar Sensors in Adverse Weather

Adverse weather conditions such as rain, fog, and snow introduce small hydrometeor particles that can interfere with electromagnetic (EM) signals, like LiDAR pulses, as they propagate through the medium. When the signal's wavelength is similar to or smaller than the droplet diameter, Mie scattering can occur, leading to signal attenuation through absorption of EM energy, as well as additional returns caused by backscattering [65].

Figure 2.2 shows that during 2 mm/h rainfall, the maximum detection range (R_{max}), also called LiDAR visibility, decreases from 2 km to approximately 1.2 km for the 905 nm LiDAR and 0.9 km for the 1550 m LiDAR. At 25 mm/h, there is a much greater degradation to approximately 0.7 km and 0.4 km [65].

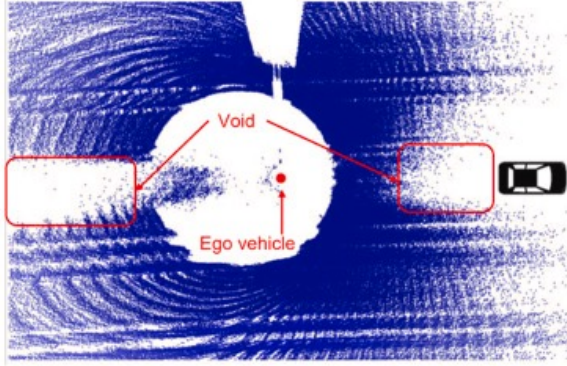


(a) Rainfall rate of 2 mm/h

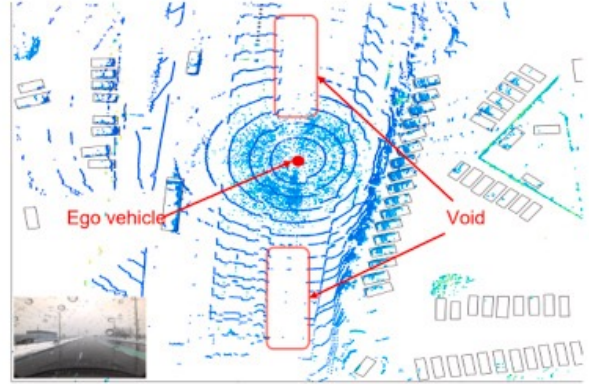


(b) Rainfall rate of 25 mm/h

Figure 2.2: Decreasing visibility from normal conditions (dotted lines) for different rainfall intensities, target wetness, and LiDAR sensors [65].



(a) Ouster OS1-64 point cloud in snow swirl. A few points of powder snow around the ego vehicle; missing view on both forward and back due to the turbulent snow caused by the leading vehicle and ego vehicle itself.



(b) Snow swirl effect without a leading vehicle. Black rectangles correspond to surrounding vehicles. Similar voids can be observed both in front of and behind the ego vehicle. The falling snow is sensed as similar to dense fog clouds around the ego vehicle. Point cloud scene from CADC Dataset. Points color depth indicates intensities: dark blue corresponds to points with higher intensity (cars); light blue corresponds to points with medium intensities (snowfall); cyan corresponds to points with lower intensities (buildings).

Figure 2.3: Effects of snowfall on LiDAR point clouds [84, 28]. The voids result from strong spurious snow returns at close ranges blocking target returns at further ranges.

Snowflakes, unlike rain droplets, can form larger, irregular solid clumps, increasing the likelihood of becoming obstacles that cause false detections or block the line of sight for valid detections [84]. Figure 2.3 illustrates the spurious points generated by powdered snow around the ego vehicle, as well as the void areas that result from these obstructions. In addition to attenuation and backscattering, Kutilla et al. [33] also note that mechanical sensors might freeze under extremely low temperatures and salted roads might cause dirt to cover the sensor box.

2.3 Lidar-based 3D Object Detection

The task of 3D object detection aims to predict a 3D bounding box around each detected object and classify it. The final output typically includes the predicted cuboid’s centre position ($centre_x, centre_y, centre_z$), dimensions (w, l, h), yaw orientation (θ), confidence score (s), and object class (c). This can be done using only camera images [10, 8, 40, 25, 29], only **LiDAR** point clouds [34, 51, 74, 80, 57, 14, 22, 13], or both [85, 48, 30, 11]. Figure 2.4 shows the different methods in solving 3D object detection. This work focuses on 3D object detection using only **LiDAR** point clouds. However, our CADC++ dataset is multimodal like CADC, allowing others to use camera images alongside **LiDAR** point clouds if desired.

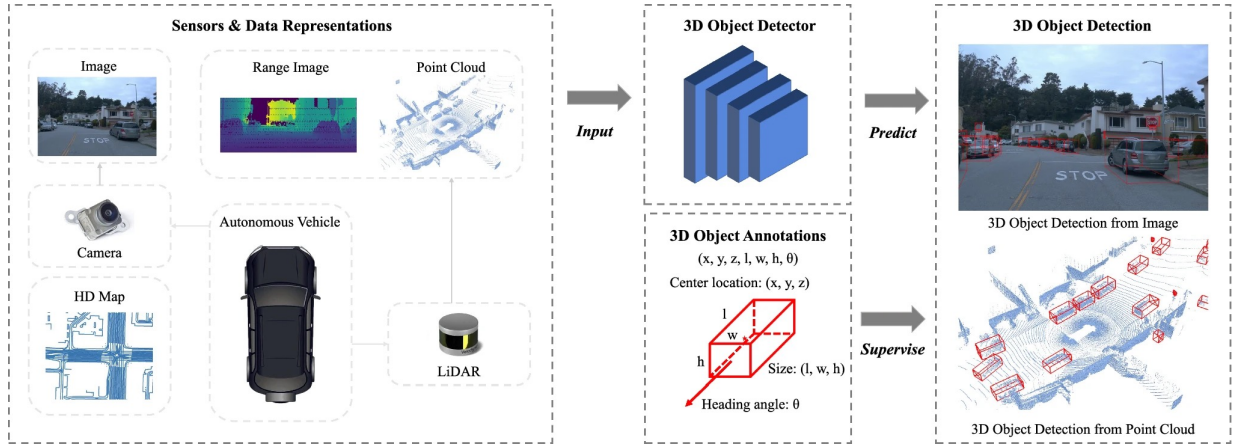


Figure 2.4: Overview of the 3D object detection pipeline in autonomous driving applications [38].

Lidar-based 3D object detectors are typically point-based [49, 51, 47, 59, 78], voxel-based [74, 77, 34, 18, 22], point-voxel based [57, 58], or range image-based [39, 20, 3, 62].

Despite the inherent sparsity and irregularity of [LiDAR](#) point cloud data, which contrasts with the grid-structured nature of image data, many 3D detectors still inspire from 2D object detection frameworks and require a dense detection head. For example, VoxelNet [86] performs dense convolution on 3D voxel features, PIXOR [77] and PointPillars [34] employ 2D dense convolution on [Bird’s Eye View \(BEV\)](#) features, and others [74, 80, 57, 18] adopt a sparse-to-dense scheme where a sparse convolution backbone is followed by a dense 2D detection head. These designs result in inefficient and complicated detection pipelines. Recently, fully sparse detectors [22, 14] avoid such issues by predicting directly from sparse 3D voxel features.

Additionally, many current detectors leverage multiple frames of [LiDAR](#) point clouds and demonstrate superior performance as compared to a single frame: some architectures [87, 80, 58, 79] perform simple concatenation, MPPNet [13] adopts hierarchical multi-frame feature encoding and interactions, and FSD++ [22] reduces the temporal redundancy in multi-frame concatenation. However, these multi-frame detectors are often intended for onboard usage due to computational constraints and perform best with a few frames from a short temporal interval as the aggregated box size and trajectory might not be consistent or smooth over time. On the other hand, given an offboard context, methods then utilize the whole sequence of frames to perform aggregation for quasi-static objects [27] or object trajectory refinement [76, 50, 21, 75] (more on this in Sect. 3.4).

2.4 The CADC Dataset

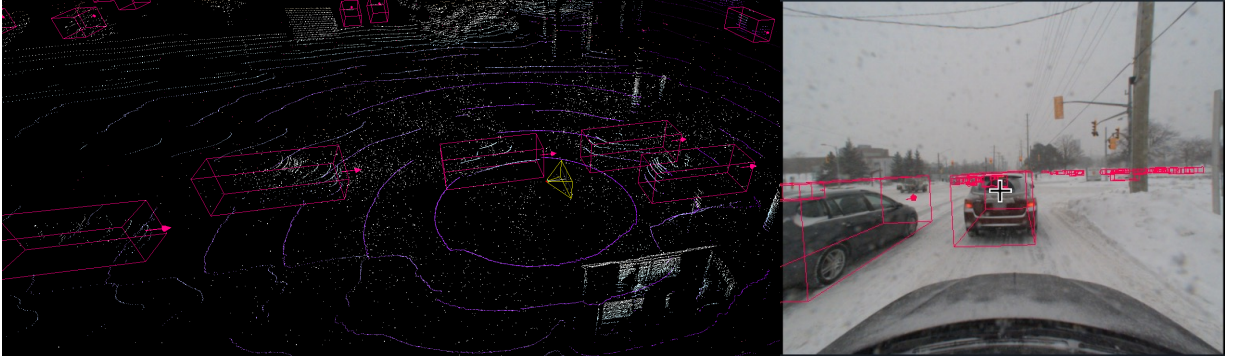


Figure 2.5: A [LiDAR](#) point cloud and a front camera image from an example frame in [CADC](#) [46]. The red ground-truth 3D bounding boxes are projected for visualization purposes only. The cloud of white particles around the ego vehicle (yellow prism) are [LiDAR](#) returns from falling snow.

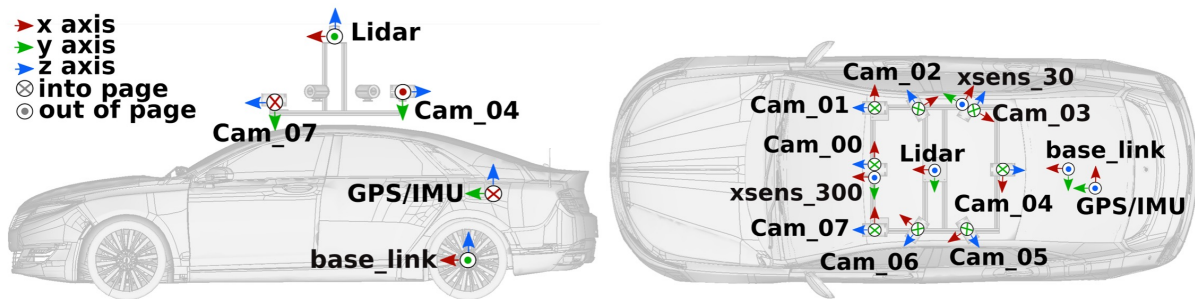


Figure 2.6: Sensor setup of the Autonomoose vehicle platform for [CADC](#)'s data collection.

The [CADC](#) dataset [46] was the first autonomous driving dataset that targeted snowy adverse weather conditions. Consisting of 75 sequences recorded during 3 days in the winters of 2018 - 2019, it covers approximately 20 km around the region of Waterloo, Ontario. All sequences have either active falling snow or accumulated snow on the roads, or both. Autonomoose, the vehicle platform used to collect [CADC](#) data is shown in Fig. 2.6. It consists of a 10 Hz 32-beam centre [LiDAR](#) (Velodyne VLP-32C), 8 cameras (Ximea MQ013CG-E2) providing 360° field of view coverage, and a GNSS/INS system (Novatel OEM638). While the [LiDAR](#) sensor operates at 10 Hz, only every 3rd frame has human-annotated 3D bounding boxes, yielding a total of 7000 labelled and 32887 unlabelled point clouds. Its taxonomy consists of 10 classes commonly seen on the road: Car, Truck, Bus, Bicycle, Horse and Buggy, Pedestrian, Pedestrian with Object, Animal, Garbage Container on Wheels, and Traffic Guidance Object.

2.5 Clear Weather Data from the 1000 km Drives

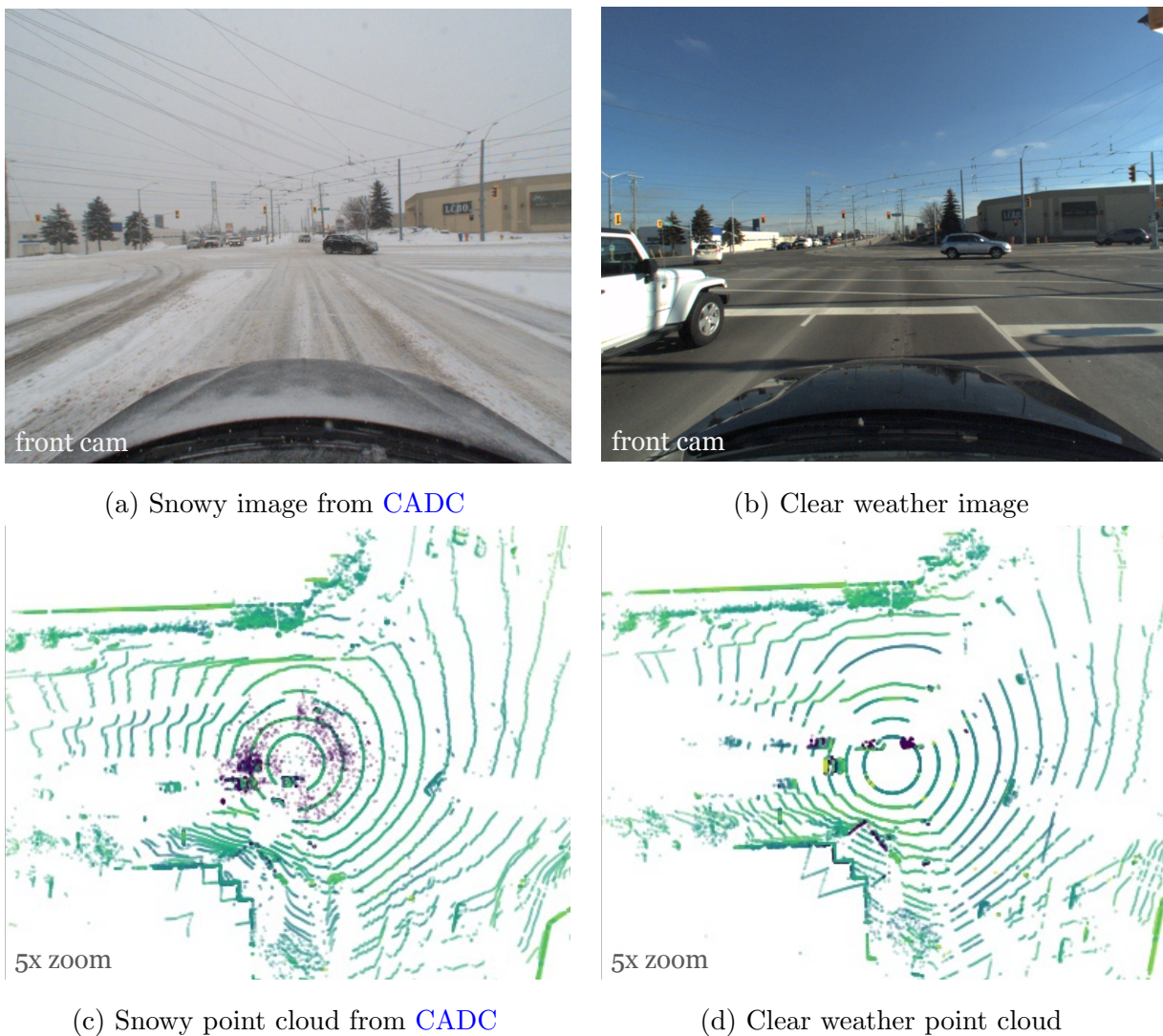


Figure 2.7: Example of a clear weather sequence recorded on the same road as a snowy sequence from CADC.

During CADC's development, a bigger data collection effort recorded about 1000 km of driving data, using Autonomoose, not only around Waterloo but also Kitchener, Cambridge, and Guelph. Considering only recordings close to Waterloo, the data includes 33

drives collected over 17 days between 2017 and 2018. They encompass a range of weather conditions, including not only snowy weather but also clear and cloudy conditions. Unlike other clear weather datasets that typically feature summer scenes, our clear weather data were recorded in the winter, with some collected only one day apart from the [CADC](#) data. This helps reduce the potential domain gap introduced by the roads and driving scene, resulting in a more accurate weather-paired dataset. Some of the earlier recordings that either lacked the Novatel RTK or used a different [LiDAR](#) sensor model did not get considered for this work.

As we aim to pair [CADC](#) with clear weather data collected from the same roads, we are left with only 7 clear weather drives, from 4 days, that occurred under either sunny or cloudy conditions around the city of Waterloo. Each of these drives is further divided into multiple ROS bag recordings of approximately 500 frames, resulting in a total of more than 400 available sequences. [Section 4.2](#) details our method for further pairing each sequence on a finer level to achieve a more accurate match.

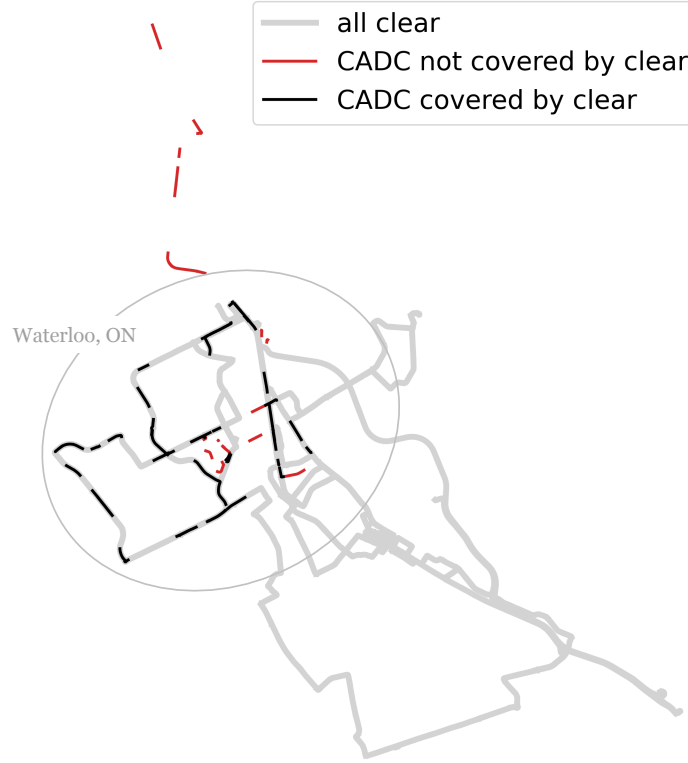


Figure 2.8: Geographical coverage overview of the clear weather data compared to CADC. Clear weather data recorded in further cities like Cambridge and Guelph were not considered in this work.

Figure 2.8 illustrates the geographic coverage that the clear weather data provides relative to CADC. In red are CADC sequences not covered by the clear weather data, in black are those that have spatial coverage with clear data, and in grey are clear data that do not spatially overlap with CADC. Since these clear weather recordings were collected in the past and Autonomoose now has a different sensor configuration, we are unfortunately unable to record new drives on those roads. However, we manually inspect sequences from similar driving scenes and try to match them by types of roads or types of road agents.

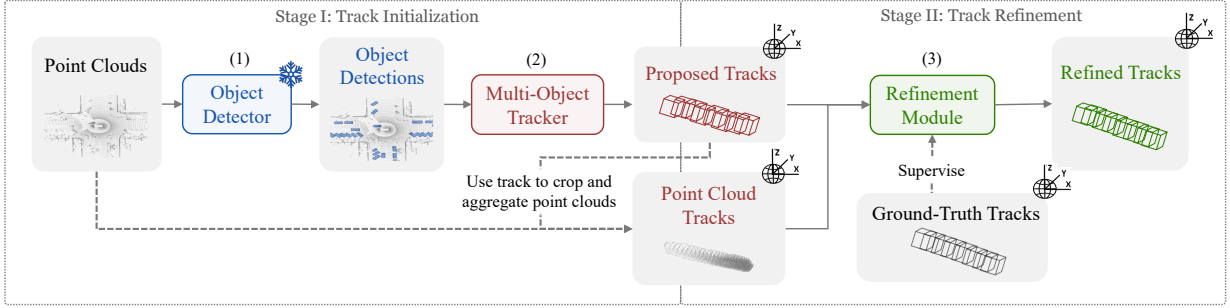


Figure 2.9: Overview of a track-based auto-labelling pipeline. The snowflake icon refers to a trained model with frozen weights, used in inference mode. The globe with axes symbolizes that those data have been transformed into the global world frame.

2.6 Track-based auto-labelling pipeline

Auto-labelling methods are typically designed for offline use cases, where there are no runtime constraints, and the entire dataset, including past and future frames, is accessible. This enables models to utilize temporal data and consider entire object trajectories, adopting a track-centric design instead of the frame-centric approach in online detection.

As illustrated in Fig. 2.9, a typical track-based auto-labelling pipeline consists of three main components: (1) an object detector localizes initial object instances within each input LiDAR point cloud; (2) a multi-object tracker (MOT) tracks as many objects as possible across subsequent frames, while forming coarse object trajectories; (3) a refinement module extracts both track- and object-level features from those tracks and learns to refine the trajectories by individually supervising them. Once the MOT creates tracks from the detected objects, the respective point clouds from these track frames can be cropped and combined to form aggregated point cloud tracks. This allows the refinement module to learn from both 3D boxes and point clouds. Lastly, ground-truth tracks are used to supervise the proposed tracks. This pipeline is also often referred to as a two-stage paradigm: the first stage initializes object trajectories, while the second stage refines each trajectory individually.

Different track-based auto-labelling methods primarily differ in their design of the refinement module, particularly in the feature extraction and supervision steps. It is preferable to pick a high-performing SOTA object detector and MOT since high-quality detections result in better object tracks, making it easier for the refinement module to learn effectively.

2.7 CTRL

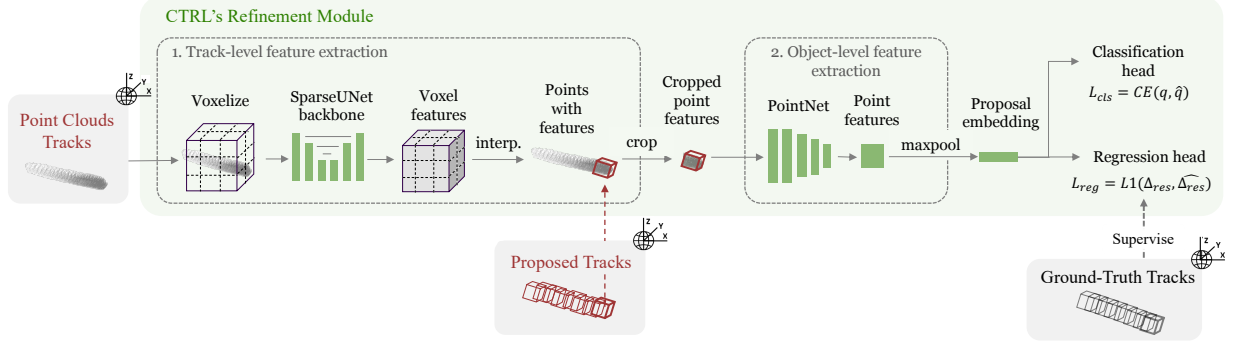


Figure 2.10: Architectural overview of CTRL’s refinement module.

We choose SOTA auto-labelling method CTRL [21] as the basis of our track-based refinement experiments. CTRL follows the pipeline described in Sect. 2.6, requiring the refinement module to be trained in a fully supervised manner. The authors of CTRL chose to use FSD [19] with multi-frame aggregation as the base detector. They aggregate 9 frames in total, sampling every other frame from both the past and future relative to the current frame. For the MOT, ImmortalTracker [67] was selected, despite its original design for online use, which only supports forward propagation of object tracks. To utilize past temporal data as well, CTRL extended it with backward track propagation using a backward motion model, enabling bidirectional tracking.

As illustrated in Fig. 2.10, CTRL’s refinement module first uses the aggregated point cloud tracks to extract some track-level features. It voxelizes an input space that can cover most tracks in the Waymo Open Dataset, then uses a sparse UNet with aggressive downsampling to extract the voxel features. These features get mapped back to the points via interpolation. Next, to perform object-level feature extraction, it uses the boxes in the proposed tracks to crop out points with features that belong to individual objects. A PointNet is subsequently used to extract object-level features.

Before a ground-truth track can be used to supervise the proposed tracks, they need to be matched to each other. Since tracks created from object detections are not perfect, a track-based IoU metric is used to assign a ground-truth track to a proposed track. The Track IoU (TIOU) metric is defined in eq. (2.1).

$$\text{TIOU}(T_a^{S_a}, T_b^{S_b}) = \frac{\sum_{i=1}^{|S_a \cap S_b|} \text{IoU}(B_a^i, B_b^i)}{|S_a \cup S_b|} \quad (2.1)$$

where given two tracks T_a and T_b with timestamp indices S_a and S_b , boxes B_a^i and B_b^i are the i -th overlapping box in each track. This metric computes the average IoU of each overlapping box pair, normalized by the length of the union of two tracks.

TIoU is computed for each potential GT and proposed track pair, with an assignment being valid only if the TIoU exceeds a predefined threshold. As a result, proposed tracks may be associated with either multiple or no GT tracks. During the refinement module training, each matched pair undergoes a second round of assignment, performed frame by frame. For each frame, if there are multiple matches, the GT box with the highest TIoU is matched to the proposed box. This approach allows the track-informed but object-level features to be supervised by individual GT boxes, enabling the model to learn to refine proposed tracks on a frame-by-frame basis.

For classification, a quality-aware loss that computes the cross-entropy between predictions and the soft classification targets is used. Soft targets are formulated as in eq. (2.2), using the IoU between each proposed box and its corresponding GT. The predictions $\hat{q}$ are outputted by an MLP after maxpooling the point features from the PointNet into a proposal embedding.

$$q = \min(1, \max(0, 2\text{IoU} - 0.5)) \quad (2.2)$$

For regression, L1 loss is used to compute between the ground-truth residual Δ_{res} and the predicted residual $\widehat{\Delta_{res}}$, also obtained from the proposal embedding.

Finally, the total loss is defined in eq. (2.3).

$$L = L_{cls} + \alpha L_{reg} \quad (2.3)$$

Chapter 3

Related Work

This chapter outlines the relevant work in various categories of autonomous driving datasets and auto-labelling.

3.1 Autonomous Driving Datasets

Autonomous vehicle data collection platforms are typically equipped with a variety of sensors, including cameras, radars, lidars, inertial measurement units, and high-precision positioning systems [82]. These data collection vehicles then drive on public roads to record synchronized data from all these sensors. As a result, datasets in autonomous driving are commonly multi-modal and contain sequential segments. Each segment is also called a sequence, containing frames spaced equally in time. These datasets then provide human-labelled ground-truth annotations in various sensor modalities and tasks [38].

The KITTI dataset [23] is a pioneering dataset to benchmark 3D object detection methods. However, with only around 7.5k labelled frames available for training, more recent datasets [61, 6, 37] have focused on increasing scale and data diversity. They provide more than 10 times more labelled frames and feature more diverse driving environments. Unlike KITTI’s daytime driving recordings from a single city, the Waymo Open Dataset [61] includes nighttime scenes and nuScenes [6] spans two cities across different continents. Despite their large scale and data diversity for urban driving scenarios, these datasets offer limited labelled data for adverse weather conditions, primarily focusing on rain rather than snowfall.

3.2 Snowy Weather Driving Datasets

Since most commonly used datasets primarily cover clear or rainy weather conditions, a significant research gap exists in snowy weather perception [84]. To bridge this gap, several dataset works [41, 5, 32, 16, 4, 46] focused on collecting data in snowy conditions.

The GROUNDED [41] and Boreas [5] datasets contain snowy data but lack 3D bounding box annotations to support 3D object detection tasks. The WADS [32] dataset offers pointwise semantic labels that could potentially be converted into 3D bounding boxes, but these are available only for snowy scenes. Similarly, the CADC [46] dataset provides 3D bounding box annotations but exclusively for snowy conditions. On the other hand, the Ithaca365 [16] and DENSE [4] datasets label both clear and snowy weather data with 3D bounding boxes. However, the number of labelled frames in each weather domain is insufficient to train modern deep learning-based object detectors. Ithaca365 only has approximately 4.7k labelled frames in clear weather and 1.8k in snow. DENSE offers slightly more, with about 6.2k clear weather frames and 4.7k snowy frames. However, that is still roughly half the size of the KITTI dataset, especially when split into training and validation sets. Additionally, DENSE’s snowy scenes primarily feature light snowfall, in contrast to the heavier snowfall captured in the CADC and WADS datasets.

Without sufficient labelled data in both clear and snowy weather domains, the impact of snowfall on 3D object detection performance cannot be empirically analysed. Moreover, these datasets do not attempt to pair sequences from the two domains, failing to ensure, as much as possible, that snow is the only varying factor.

3.3 Simulated Weather Domain Adaptation Datasets

Due to the difficulty of controlling weather and dynamic objects in real-world driving scenes, a practical approach to creating weather-paired datasets is with simulation. Real point clouds from a source domain can be transformed into a simulated target domain while preserving the overall scene and its agents. This way, only the desired weather conditions are introduced. Object annotations from the source domain also remain valid in the simulated data, eliminating any additional labelling efforts that would otherwise be needed.

Once data from both domains is obtained, two separate models can be trained on each domain to compare the potential difference in 3D object detection performance caused by snow. There is, however, an additional domain gap between the simulated and real data

that cannot be distinguished from the weather domain gap. This motivates our efforts to collect real data for both weather domains, then try our best at pairing each sequence from source to target domain.

3.3.1 Generating Clear Weather Datasets

Simulated clear weather data can be generated by transforming real snowy data. Due to the effects of backscattering, many studies [9, 31, 1, 43, 69, 56, 35, 73, 68] treat spurious snowflake returns as noise and developed denoising methods to recover clear weather data from snowy recordings. Early denoising approaches mainly relied on statistical filtering, using either the distance values of LiDAR points [9, 31, 1], the intensity values [43], or a combination of both [69] to remove snowy outliers. With the release of the WADS dataset [32], which provides pointwise semantic labels for training and validation, studies have advanced supervised learning-based methods to segment snowy points out [26, 45], as well as unsupervised techniques [81, 2]. More recently, researchers have furthermore used both spatial and temporal features in adjacent point clouds [56, 35, 73, 68]. While identifying and removing noisy snow points from a point cloud can effectively reduce backscattering effects, attenuation effects remain unaddressed. Additionally, inaccurate denoising algorithms will mistakenly remove valuable points. To revert the effects of signal attenuation, a paired LiDAR dataset with both snowy and clear weather data is needed but is not available in the current literature. We aim to bridge this gap in this work.

3.3.2 Generating Snowy Weather Datasets

Simulated snowy point clouds can be generated from clear-weather data using physics-based models. Hahner et al. [24] developed a simulation model that accounts for both backscattering and attenuation effects. Each LiDAR beam is raytraced from the sensor and returns are recorded as the beam interacts with soft targets like snowflakes or hard targets like road objects. Parameters such as snowflake density, size distribution, and snowfall rate are adjustable based on values from established meteorological studies [24]. SnowyKITTI [56] dataset has been generated using this model. However, despite efforts to capture the effects of airborne snowflakes, fallen snow accumulating on roads and objects was not modelled in the method proposed by Hahner et al. [24]. This lack of realism nonetheless introduces a domain gap between the simulated and real data. To overcome the lack of realism, a paired weather adaptation dataset is once again crucial but not available in the current literature.

3.4 Auto-labelling

Earlier works in LiDAR-based 3D auto-labelling either use a pre-trained 2D model [83, 52], focus only on stationary objects [66], or are model-free and rely instead on tracking and pose estimation [12, 42]. However, despite requiring little to no supervision, they struggle to produce high-quality pseudo-labels that are comparable to human annotations. More recent works [50, 76, 21, 36, 75] adopt the two-stage object trajectory refinement approach previously shown in Sect. 2.6: the initial stage model generates coarse object predictions using an object detector and a multi-object tracker, then the refinement stage model, usually trained in a supervised manner, improves these predictions by considering their entire trajectories. Some methods handle stationary and dynamic objects separately [50], while others do not differentiate by object motion [76, 21, 36, 75].

Many of these methods [76, 75, 21] only train under full supervision, expecting all training sequences to contain ground-truth annotations at every frame. Only labelling a subset of those sequences is a strategy to reduce labelling costs, but this inevitably results in a loss of data diversity. Hence, some works [50, 36] also make use of the remaining unlabelled sequences to further train under the framework of semi-supervised learning. Nevertheless, all current methods expect either all sequences or a subset of them to be fully annotated. We have not found existing work in offline auto-labelling that specifically handles sparsely labelled data, that is, where all sequences are annotated at a lower frequency to reduce the labelling costs while still maximizing data diversity.

3.5 Sparsely labelled datasets

In addition to differences in size, location, and sensor suite, some autonomous driving datasets also vary in their annotation strategies. Densely labelled datasets like the Waymo Open Dataset [61], Argoverse 2 Sensor Dataset [70], and PandaSet [72] provide annotations for all LiDAR frames, resulting in a labelling frequency equal to the LiDAR’s operating frequency. In contrast, nuScenes [6], H3D [44], and CADC [46] only label keyframes, at fixed intervals, leaving intermediate frames unlabelled. We refer to this annotation strategy as *sparse labelling*. Specifically, nuScenes annotates at 2 Hz, given a 20 Hz LiDAR, whereas H3D and CADC annotate at 2 Hz and 3.3 Hz, respectively, given a 10 Hz LiDAR.

nuScenes and H3D use simple linear interpolation to create intermediate labels from adjacent human-created annotations [6, 44], suggesting that 2 Hz is robust to up-sampling, but there is a lack of evaluation results. Their interpolated labels are then treated as

ground truth. While interpolation is an obvious candidate to overcome the problem of sparsely labelled data, we are interested in very sparse labelling frequencies for which even the use of sophisticated interpolation models might not be enough. We therefore explore how interpolated labels can be combined with other pseudo-labels in a highly effective way.

Chapter 4

The CADC++ Dataset

4.1 Problem Statement

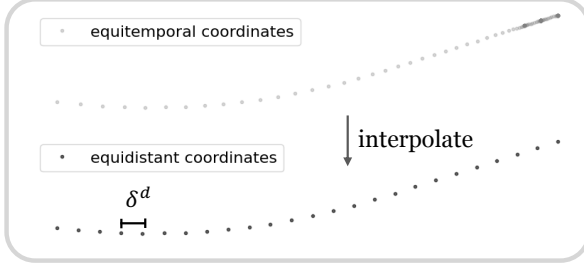
Given the set of 75 snowy CADC sequences and the set of 400 clear weather CADC++ sequences to choose from, how can we maximally match each CADC sequence with a similar-length sequence from CADC++? Ideally, we aim for each snowy frame to have a matching clear frame that is observed from the same position and orientation, as well as recorded under the same driving speed and traffic density. However, since the CADC++ clear drives were not purposely collected to satisfy these requirements, we define matching objectives and devise a method that matches sequences with non-zero distance. Additionally, since CADC++ sequences can be up to five times longer, we determine the precise frame endpoints and frame rates required to extract a 100-frame matching segment.

The resulting CADC++ should be comparable in size to the original CADC, with similar numbers of sequences. Moreover, each CADC sequence should be paired with a unique CADC++ sequence to ensure that the training and validation sets maintain similar sizes and distributions.

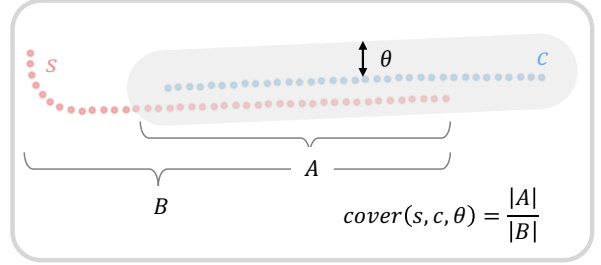
4.2 Weather-paired Dataset Matching Method

We select 100-frame CADC++ matches based on the following two-step process. First, we find the best spatially matched clear weather CADC++ sequence for each CADC sequence using their GPS coordinates. Then, we perform temporal matching to select a

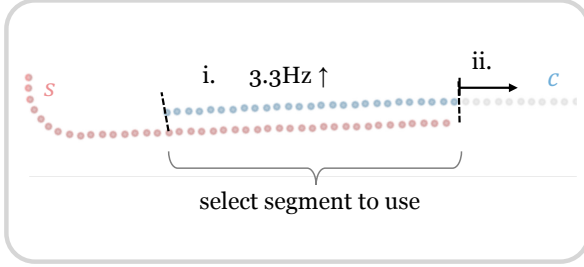
specific 100-frame segment with the desired frame endpoints and frame rate. These two steps are detailed in Sects. 4.2.1 and 4.2.2 below.



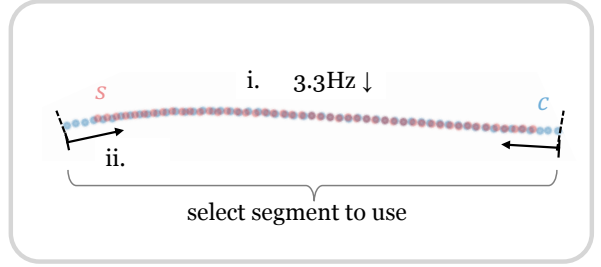
(a) Build spatial models of all sequences.



(b) Compute coverage ratio to all possible sequence pairs.



(c) Choose sampling frequency and endpoints when the clear sequence is too short (< 100 frames). We can i) increase sampling frequency up to 10 Hz, and if necessary, ii) extend endpoints beyond the coverage area.



(d) Choose sampling frequency and endpoints when the clear sequence is too long (> 150 frames). We can i) decrease sampling frequency down to 2 Hz, and if necessary, ii) shorten endpoints even if it reduces the coverage area.

Figure 4.1: Demonstrating the main steps of our spatial and temporal matching methods using visual examples.

4.2.1 Spatial Matching

We define the following objectives to match two sequences in space based on their driven paths, where $\succ$ denotes “better than” and $\gg$ denotes “much better than”:

1. On the same road $\gg$ on an adjacent road. Adjacent roads might display very different driving environments.

2. In the same direction $\succ$ in the opposite direction. Different driving directions only view the environment from slightly different viewpoints.
3. In the same lane $\succ$ in an adjacent lane. An adjacent lane might lead to different driving behaviors, but ultimately causes marginal viewpoint differences.
4. Number of frames $\geq$ number of snowy frames. As some [CADC](#) sequences will be unmatched (blue roads in Fig. 2.8), we allow for CADC++ sequences to have slightly more frames if needed to partially compensate for the loss of frames.
5. Equal number $\succ$ greater number of other road users than in snowy sequence. We favor the same traffic density, but when not possible, we prefer a more interesting scene, which usually has more traffic agents.

To solve for these objectives, we first need to model each sequence in space to enable spatial trajectory comparisons between two sequences. GPS coordinates are recorded equitemporally but are not uniformly distributed in space. Therefore, we employ linear interpolation and equidistant sampling to obtain equidistant points that model discretely a sequence’s driven path in space.

Let w_i^t denote the coordinates of frame i in a collected GPS data sequence g , where t emphasizes that consecutive frames are separated by constant time δ^t . Then, let w_j^d denote the coordinates of point j in its corresponding spatial model, where d emphasizes that consecutive points are separated by constant distance δ^d . We obtain spatial model points w_j^d from GPS points w_i^t as follows:

$$\begin{aligned} w_0^d &= w_0^t \\ w_j^d &= \alpha(w_i^t - w_{i-1}^t) + w_{i-1}^t \quad i, j > 0, \end{aligned} \tag{4.1}$$

with i, j constrained by

$$0 < j\delta^d \quad i = 1 \tag{4.2}$$

$$\sum_{k=1}^{i-1} \|w_k^t - w_{k-1}^t\| < j\delta^d \quad i > 1 \tag{4.3}$$

$$\sum_{k=1}^i \|w_k^t - w_{k-1}^t\| \geq j\delta^d, \tag{4.4}$$

and

$$\alpha = \frac{j\delta^d - \sum_{k=1}^{i-1} \|w_k^t - w_{k-1}^t\|}{\|w_i^t - w_{i-1}^t\|}. \quad (4.5)$$

Intuitively, we create a spatial model by resampling along the recorded trajectory of GPS points to create coordinates that are spatially equidistant from each other by a distance δ^d . Equation (4.1) specifies that the starting points should be the same, but subsequent points are placed proportionally, using a scaling factor α , to follow the original trajectory's direction while respecting the consistent spacing constraints defined by eqs. (4.2) to (4.4).

Once a discrete spatial model of each snowy and clear sequence is obtained, as shown in Fig. 4.1a, we can then calculate a longitudinal overlap ratio of every clear and snowy spatial sequence pair by counting the fraction of points in the snowy sequence that are within a specified distance of a point in the clear sequence, for a range of such distances. These distances correspond to the possible lateral distance between paths on the same road.

Let $\mathcal{S}$ and $\mathcal{C}$ denote the sets of snowy and clear spatial sequences, respectively, and let $\Theta = \{1, 2, 4, 8, 16\}$ m be the set of lateral distances we wish to consider. We thus define a coverage function $\text{cover} : \mathcal{S} \times \mathcal{C} \times \Theta \rightarrow [0, 1]$ that returns the overlap ratio, i.e., the fraction of points in a snowy sequence $s \in \mathcal{S}$ that are within $\theta \in \Theta$ m of a point in clear sequence $c \in \mathcal{C}$. The function cover is implemented as:

$$\text{cover}(s, c, \theta) = \frac{|\{w^s \in s \mid \exists w^c \in c \bullet \|w^s - w^c\| \leq \theta\}|}{|s|} \quad (4.6)$$

By breaking down a drive into a discrete spatial sequence of points, we favor maximizing coverage along the longitudinal direction of the driven paths rather than focusing on minimizing lateral displacement in the trajectories.

After calculating the coverage ratio for each snowy-clear sequence pair, as shown in Fig. 4.1b, we can identify those with the highest coverage. Then, we use the filtering criteria defined below to further refine our selection.

The set of clear sequences that have greatest coverage of snowy sequence s , given lateral distance threshold θ , is denoted $c_{s,\theta}^*$ and given by

$$c_{s,\theta}^* = \operatorname{argmax}_{c \in \mathcal{C}} \text{cover}(s, c, \theta). \quad (4.7)$$

Let $C_s^* = \bigcup \{c \in c_{s,\theta}^* \mid \text{cover}(s, c, \theta) > 0, \theta \in \Theta\}$ be the set of clear sequences with greatest nonzero coverage of snowy sequence s , for all θ , and let $\text{cons} : \mathcal{S} \times \mathcal{C} \rightarrow 2^\Theta$ be a consistency

function that gives the subset of Θ for which $c \in \mathcal{C}$ is the clear sequence that has greatest nonzero coverage of snowy sequence $s \in \mathcal{S}$. Function cons is implemented as:

$$\text{cons}(s, c) = \{\theta \in \Theta \mid c \in c_{s,\theta}^*, \text{cover}(s, c, \theta) > 0\} \quad (4.8)$$

C_s^* is refined into the following subsets, in decreasing order of desirability. A clear sequence is then chosen from the first nonempty subset:

1. $\{c \in C_s^* \mid \text{cover}(s, c, \theta) = 1, \theta = \min(\Theta)\}$
2. $\{c \in C_s^* \mid \exists \theta \in \Theta \bullet \text{cover}(s, c, \theta) \geq 0.95\}$
3. $\{c \in C_s^* \mid |C_s^*| = 1\}$
4. $\{c \in C_s^* \mid |\text{cons}(s, c)| > 1\}$
5. C_s^*

Informally, set 1 contains any clear sequences that are in the same lane and that completely cover the snowy sequence; set 2 contains any clear sequences that are in the same road and cover most of the snowy sequence. If there is just one clear sequence that has non-zero coverage of the snowy sequence, considering all lateral distances, it is in set 3. Set 4 contains clear sequences that have the greatest coverage of the snowy sequence for more than one lateral distance.

When the first non-empty subset is not a singleton, a clear sequence is chosen by human inspection, based on its coverage of the clear sequence and the number and diversity of the other road users it contains. In the few cases that a snowy sequence has no matching sequence by the above process, a clear sequence is chosen that is as qualitatively similar to the snowy sequence as possible.

4.2.2 Temporal Matching

Having identified the full-length CADC++ sequences using spatial models, we must choose specific data frames to obtain reduced sequences that also match CADC in length, as shown in Figs. 4.1c and 4.1d. CADC data sequences mostly contain 100 labelled frames sampled at 3.3 Hz, with a handful of sequences containing fewer frames. Since not all CADC sequences have a matching CADC++ sequence, we partially compensate for the reduction in the total number of frames by ensuring that all clear sequences have at least 100 frames. Given the

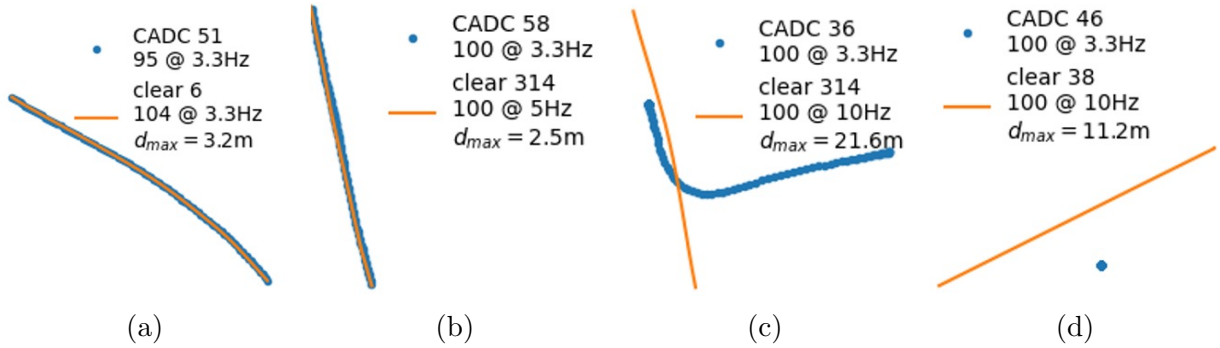


Figure 4.2: Some examples of snowy CADC sequences matched with clear weather sequences in CADC++. Each sequence is assigned a number as its unique ID. Below the sequence name, we indicate the number of frames and sampling frequency that were selected as part of the match. d_{max} is the distance between the furthest two points from a CADC sequence and its clear weather match.

inherently different driving speeds and accelerations that exist in clear and snowy weather, it is not feasible to simultaneously match the number of frames, the ends of the sequences, and the frame rates of the sequences. We therefore vary the frame sampling rate to achieve a minimum frame count of 100, as demonstrated by Fig. 4.1c, while also minimizing for distance disparities between snowy and clear frames. Our procedure is described below.

For each clear temporal sequence c matched to a snowy temporal sequence s , we first find the set of frames in c that are closest in euclidean distance to frames in s . From this set, we identify those with the earliest and latest timestamps, which demark a clear subsequence c' . The ends of c' usually correspond to the ends of s , but not always, due to the vehicle remaining stationary in some sequences, and the existence of measurement errors. We require a minimum of 100 frames from c , sampled at regular intervals, so we find the maximum sampling interval $\in \{1, 2, 3, 4, 5\}$ that samples c' into ≥ 100 frames. In rare cases where this produces more than 150 sampled frames, like the example shown in Fig. 4.1d, we increment the sampling interval and use exactly 100 frames. The sampled frames form the final clear sequence, which may have been sampled at 10, 5, 3.3, 2.5, or 2 Hz, depending on the sampling interval.

Figure 4.2a shows an example where a segment of a clear sequence was able to match exactly a CADC sequence while sharing the same sampling frequency and satisfying the 100-frame requirement. The pair in Fig. 4.2b also matches in length, but its sampling rate had to be increased to 5 Hz to reach 100 frames. Example in Fig. 4.2c, however, only has a match for part of the CADC trajectory. Additionally, the clear trajectory had to

extend beyond the closest matching frames to achieve 100 frames, increasing the maximum distance discrepancy. However, that was necessary since the highest possible sampling frequency of 10 Hz (our [LiDAR](#)’s operating frequency) has already been reached, therefore the sequence length cannot be further shortened by increasing the sampling frequency. This scenario is also applicable when [CADC](#) sequences are static, as illustrated in Fig. 4.2d.

Overall, [CADC](#) sequences only have one unique CADC++ match but a full CADC++ sequence might be split into multiple subsequences to match different [CADC](#) sequences. We also note that our CADC++ subsequences might overlap with each other spatially, but they must generally sample different frames by using different starting frames or sampling rates.

4.3 Dataset Labelling Strategy

After selecting the best matching sequences, CADC++ needed to be labelled with 3D bounding boxes for 3D object detection. However, we did not have enough budget to label all [LiDAR](#) frames. To resolve this limitation, we devised a sparse labelling strategy and auto-labelling methods for sparsely labelled data, reducing the original labelling effort to only 30%. The validation set, comprising 20% of the dataset, needs to be fully labelled regardless. A fully labelled validation set is crucial to evaluate the performance of the auto-labelling model that will be used to automatically generate annotations for the unlabelled frames. However, the remaining training set can be labelled sparsely, that is frames spaced at fixed intervals get labelled while the intermediate frames remain unlabelled. Our budget allowed every 10th frame of the training sequences to be labelled, accruing another 10% of the original annotation costs.

4.4 Pre-labelling for Segments.ai

Having human annotators label [LiDAR](#) point clouds with 3D bounding boxes is an expensive and time-consuming task. Our original quote from Deepen AI was over 30,000 US dollars for 7000 frames, with an estimated labelling time of up to three months. Looking for a more affordable alternative, we found Segments.ai offering lower labelling costs with a user-friendly interface that supports uploading pre-labels to help reduce the labelling efforts. Since their pricing is based on total labelling time, generating high-quality pre-labels is crucial to maximizing the number of frames we can afford to annotate. We outline below our approach to creating the best pseudo-labels to facilitate labelling.

To produce the best pseudo labels, we choose to pre-train on 20 % of the bigger and more diverse Waymo Open Dataset [61], then fine-tune on CADC [46]. Training on only 20 % has been shown to achieve good enough performance based on our prior experience. We train a separate detector for each class since we have previously achieved better results with a single detection head than with multiple ones. We choose to use the VoxelNext [14] architecture with 5-frame aggregation, enabling the model to learn from four preceding point clouds combined with the current frame. The car, truck, bus, and pedestrian models have been fine-tuned for 40, 20, 10, and 40 epochs, respectively, after pre-training for 24 epochs on the Waymo Open Dataset.

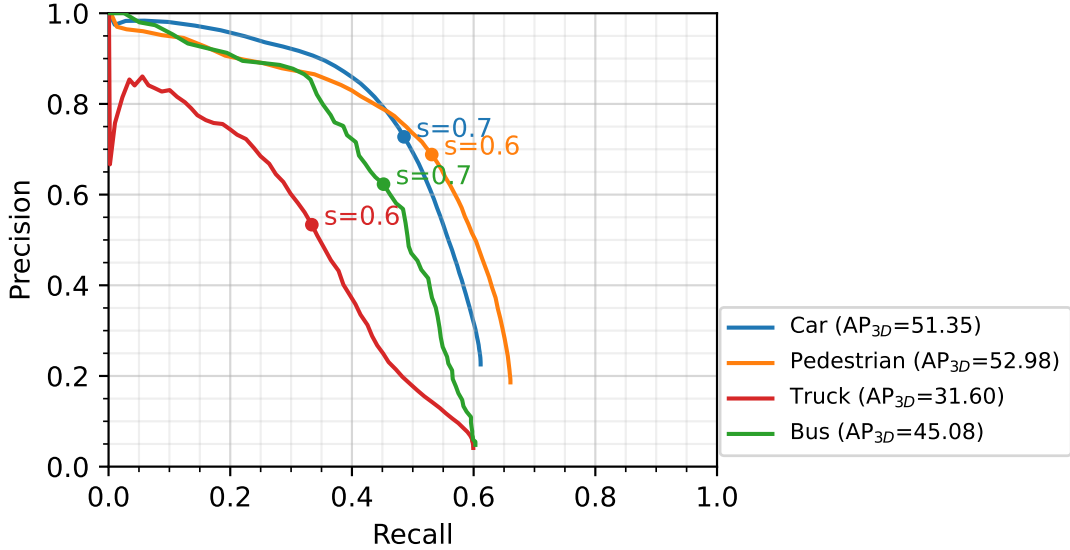
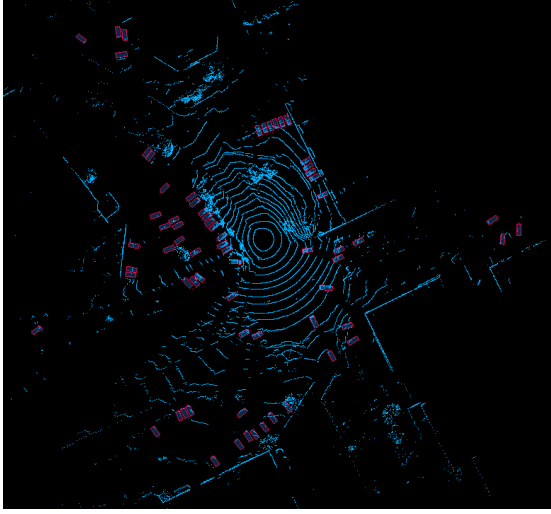


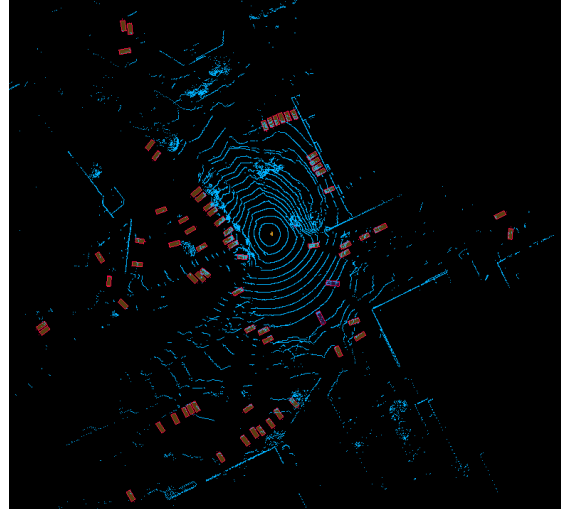
Figure 4.3: PR curves for the pre-labelling models evaluate on CADC’s validation set. Each PR curve demonstrates the performance of a single-class model that was pre-trained on the Waymo Open Dataset and then fine-tuned on CADC. The score thresholds s maximize the F1 scores, the harmonic mean between precision and recall.

Once the detectors have been trained, we run inference on the CADC++ point clouds and then run ImmortalTracker [67], a SOTA multi-object tracker, in the association mode, to assign track IDs to our detections. Score thresholds of 0.7, 0.6, 0.7, and 0.6, as depicted by the markers in Fig. 4.3, have initially been chosen for the car, truck, bus, and pedestrian classes, respectively, to maximize the F1 scores. We then visually assessed their performance and tweaked these score thresholds to either reduce the number of false negatives while allowing more false positives or to reduce the number of false positives. The final

score thresholds used are respectively 0.45, 0.7, 0.6, and 0.45. We believe that human labellers can efficiently remove extra detections but need more time identifying missed objects. Figures 4.4 and 4.5 show some examples of our pseudo labels. Given the similarity between CADC and CADC++ data, our models produced high-quality pre-labels overall, as can be seen in Fig. 4.4a. Although decent as initial pre-labels, they still needed human annotators to refine. Shown in Fig. 4.5, common errors include missed or false detections, as well as an inaccurate bounding box position, yaw, or size.

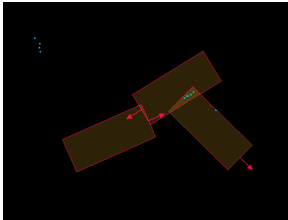


(a) Our generated pre-labels

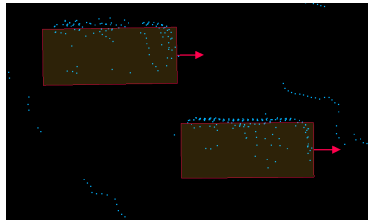


(b) Ground-truth labels

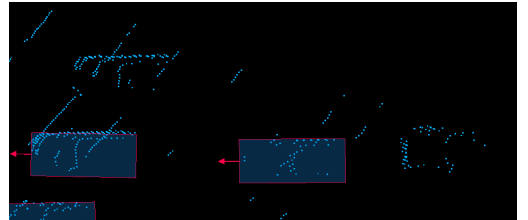
Figure 4.4: Comparison of our generated pre-labels with the ground-truth labels corrected by human annotators.



(a) False positives



(b) Inaccurate positions



(c) False negatives

Figure 4.5: Common errors and inaccuracies in our pre-labels.

4.4.1 Averaging static tracks

Algorithm 1 Averaging Static Tracks

```

function AVERAGE_STATIC_TRACKS( $\mathbf{T}, \tau_{static}, \tau_{dynamic}$ )
    Initialize empty arrays to store tracks  $\mathbf{T}'$ 
    for  $\mathbf{t}$  in  $\mathbf{T}$  do                                     ▷ iterate over all the tracks
         $\mathbf{p} \leftarrow \mathbf{t}.position$                              ▷ get all bounding box positions in track
         $\hat{p} \leftarrow mean(\mathbf{p})$                                ▷ get the mean position
         $\tau_{outlier} \leftarrow COMPUTE\_OUTLIER\_THRESHOLD(\mathbf{p}, \hat{p})$ 
         $isDynamic \leftarrow IS\_TRACK\_DYNAMIC(\mathbf{p}, \tau_{static}, \tau_{dynamic})$ 
        if  $isDynamic$  then
             $\mathbf{T}'.append(\mathbf{t})$                                 ▷ keep track as is
        else
             $\mathbf{p}' \leftarrow FILTER\_OUTLIERS(\mathbf{p}, \tau_{outlier})$ 
             $\mathbf{t}_0.position \leftarrow mean(\mathbf{p}')$              ▷ update track position with the average filtered
            position on the first frame
             $\mathbf{t} \leftarrow \mathbf{t}_0$                                ▷ only keep the first frame
             $\mathbf{T}'.append(\mathbf{t})$ 
        end if
    end for
    return  $\mathbf{T}'$ 
end function

```

As an extra request to speed up labelling, the Segments.ai team has asked us to average the detections of static vehicles and only upload them on the first frame in which they appear. Since static vehicles do not move, doing this avoids them having to tweak potentially different bounding boxes for each frame of a static track.

The first step in averaging static detections is to detect a static track. While static tracks can be easily detected with GT labels, tracks made of pseudo labels are more noisy. False positives, identity switches, and drift detections in a track can make it seem dynamic. As described in Alg. 1, we use statistical filtering on the bounding box positions over the entire track to determine if a track is static. Once a track is deemed static, its box positions are filtered to remove outliers and then averaged. The track is updated with the averaged position on its first frame, and the other instances from subsequent frames are discarded. If a track is deemed dynamic, we return the track without modification. $\tau_{static}, \tau_{dynamic}$ are thresholds we can specify and tune. We choose $\tau_{static} = 2$ to represent that a track segment

is considered static if it moves less than 2m between adjacent frames. $\tau_{dynamic} = 0.2$ is chosen to represent that a track is deemed dynamic if 20% of its segments are dynamic. $\tau_{outlier}$ is computed using the mean and standard deviation of the track positions.

This method has been validated and manually tuned by visualizing the detected static tracks on the Segments.ai tool.

Algorithm 2 Auxiliary functions

```

function COMPUTE_OUTLIER_THRESHOLD( $\mathbf{p}, \hat{p}$ )
   $\mathbf{d} \leftarrow \|\mathbf{p} - \hat{p}\|_2$ 
   $\tau_{outlier} \leftarrow mean(\mathbf{d}) + std(\mathbf{d})$ 
  return  $\tau_{outlier}$ 
end function

function IS_TRACK_DYNAMIC( $\mathbf{p}, \tau_{static}, \tau_{dynamic}$ )
   $n_{dynamic} \leftarrow COUNT\_DYNAMIC\_SEGMENTS(\mathbf{p}, \hat{p}, \tau_{static})$ 
   $n \leftarrow \mathbf{p}.length$ 
  return  $n_{dynamic}/n > \tau_{dynamic}$ 
end function

function COUNT_DYNAMIC_SEGMENTS( $\mathbf{p}, \hat{p}, \tau_{static}$ )
   $n_{dynamic} \leftarrow 0$ 
  for  $p$  in  $\mathbf{p}$  do
    if  $\|p - \hat{p}\|_2 > \tau_{static}$  then
       $n_{dynamic}++ = 1$ 
    end if
  end for
  return  $n_{dynamic}$ 
end function

function FILTER_OUTLIERS( $\mathbf{p}, \hat{p}, \tau_{outlier}$ )
  Initialize empty array  $\mathbf{p}'$  to store filtered positions
  for  $p$  in  $\mathbf{p}$  do
    if  $\|p - \hat{p}\|_2 \leq \tau_{outlier}$  then
       $\mathbf{p}'.append(p)$ 
    end if
  end for
  return  $\mathbf{p}'$ 
end function

```

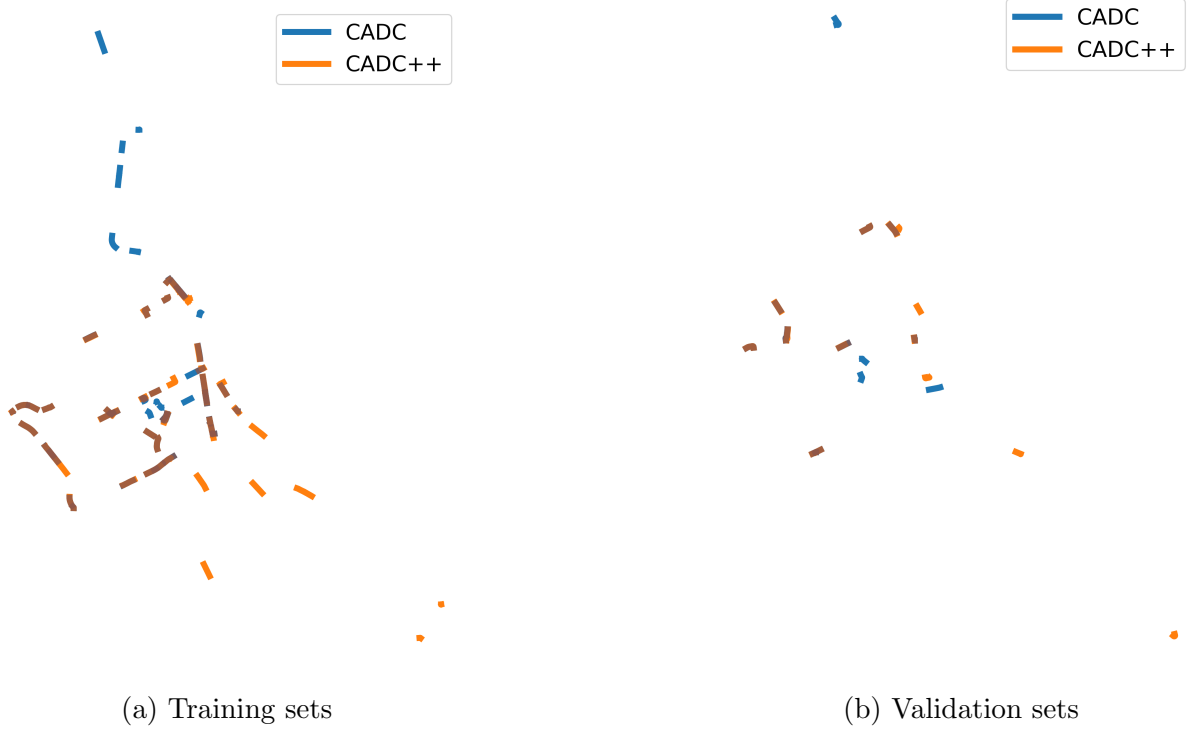


Figure 4.6: Coverage overview of matched sequences from CADC++ and CADC. Overlapping sequences were successfully matched spatially and non-overlapping ones were manually selected according to the similarity of scenes and object distributions.

4.5 Dataset statistics

All sequences in CADC have been matched one-to-one with a clear weather CADC++ sequence of at least 100 frames. Figure 4.6 shows the spatial coverage between CADC and the finalized CADC++, with separate visualizations for the training and validation sets. There is only one CADC sequence where we made the exception to match with two half-length sequences from CADC++, due to better alignment and data diversity. There is also one sequence in CADC where the annotations are corrupted, so we have not attempted to match it. As a result, CADC++ has 60 training sequences and 15 validation sequences, where two of the validation sequences are half-length.

Out of the 74 total CADC sequences, we were unable to spatially match 21 of them due to the lack of coverage illustrated in Fig. 2.8. Hence, for 15 of those, we were successful in manually finding a match from an alternate but similar scene (e.g., matching a parking lot

with a different parking lot). We also preferred a candidate sequence with more objects. For the remaining 6, we selected sequences matching the type of road agents while the scene differed slightly (e.g., a pedestrian-heavy intersection on the university campus was matched to a pedestrian-heavy road intersection).

In total, if we sample [CADC](#) labels sparsely, in the same way that we have labelled CADC++, we obtain distributions shown in Figs. 4.7 to 4.10.

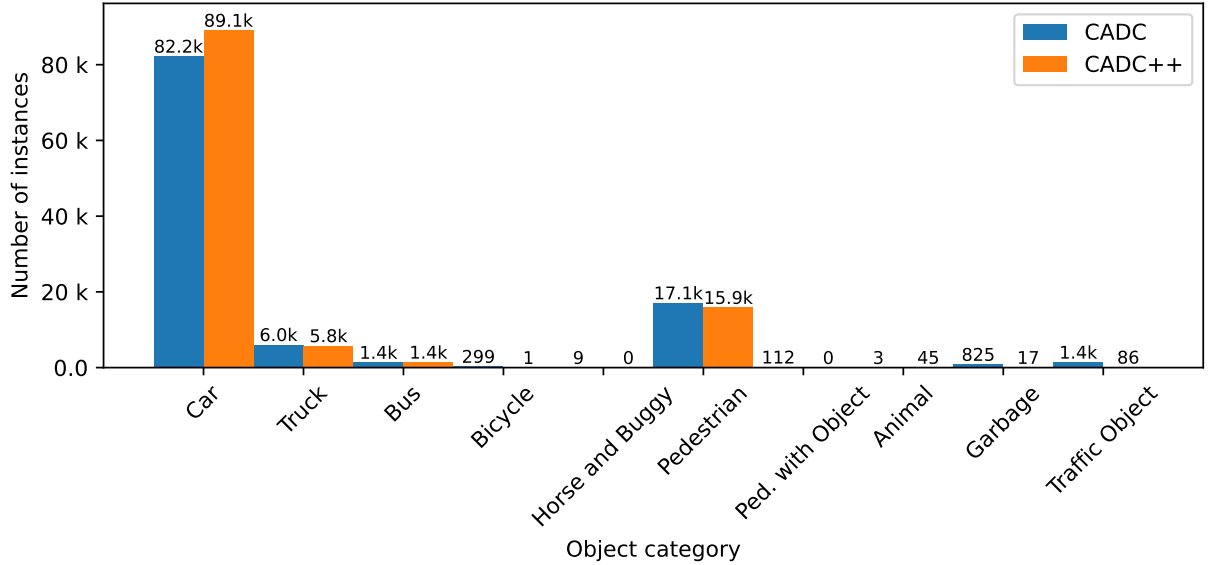


Figure 4.7: Number of object instances per labelled category in [CADC](#) and CADC++.

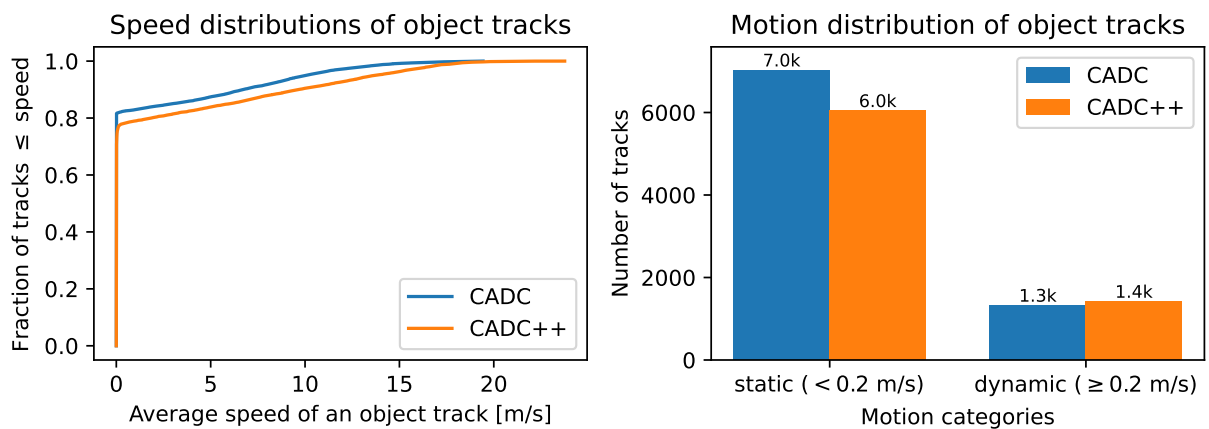
Figure 4.7 shows that for the four major classes (Car, Truck, Bus, Pedestrian), the object instance count in CADC++ is only slightly lower than in [CADC](#). The selection of [CADC](#) sequences prioritized scenes with a high number of road agents to include more interesting scenarios. In CADC++, however, our primary goal was to achieve spatial alignment with [CADC](#), making matching scene object density a secondary objective. In addition, the 6 [CADC](#) sequences where we could not find a close match were recorded on campus, which has a higher density of cars and pedestrians than city streets due to parking lots and students heading to class.

We gather all bounding boxes along an object’s trajectory within a sequence and plot their average speed in Fig. 4.8. We then use a threshold of 0.2 m/s to distinguish between stationary and dynamic vehicles, as defined in Waymo Open Dataset’s evaluation [61]. We lower the threshold to 0.1 m/s for pedestrians to accommodate their lower speed. We indeed

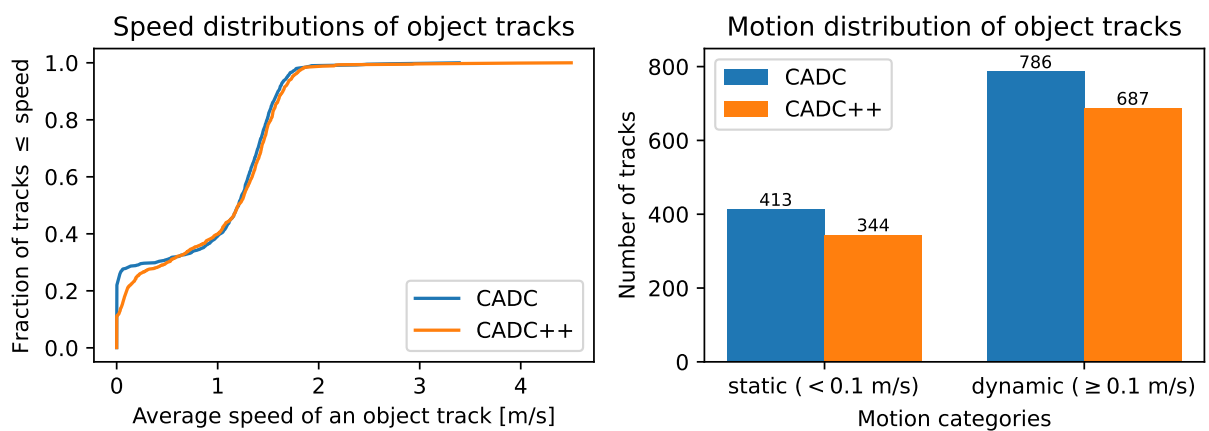
observe in Fig. 4.8a that the missing cars in CADC++ all exhibit stationary trajectories, likely originating from those unmatched parking lots. On the other hand, static and dynamic pedestrians are equally lacking in CADC++, according to Fig. 4.8b.

Despite CADC++ slightly lacking in its total number of objects, its distributions of point count per bounding box and annotation count per frame remain relatively close to those of CADC, as illustrated in Fig. 4.9. This alignment of object distributions is crucial for CADC++ to serve as a weather-based domain adaptation dataset, enabling an objective evaluation of snow’s impact on 3D object detection with minimal bias from other confounding factors.

However, the number of object instances per frame deviates more for lesser represented object classes, as the scenes are inherently different despite our best efforts at matching the driven paths. This can be observed in Fig. 4.10.

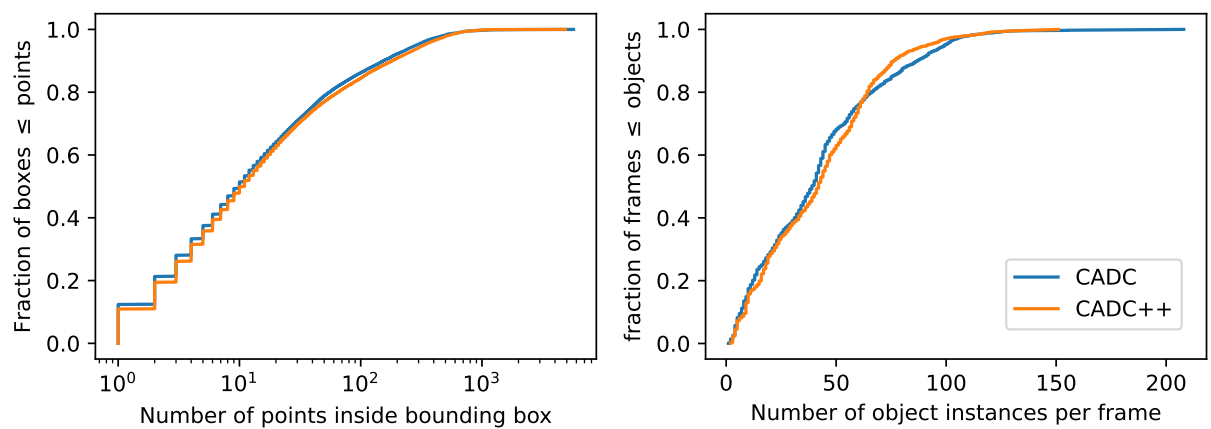


(a) Car



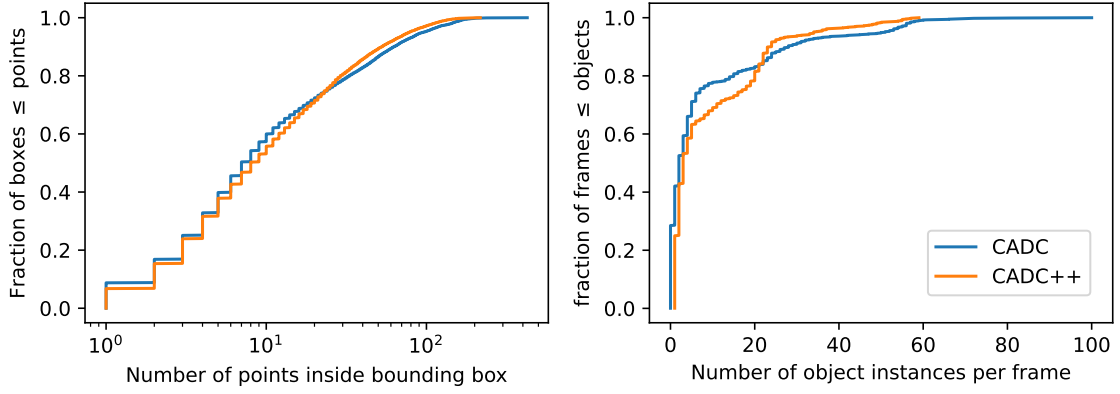
(b) Pedestrian

Figure 4.8: Average speed distributions of object tracks in CADC and CADC++.

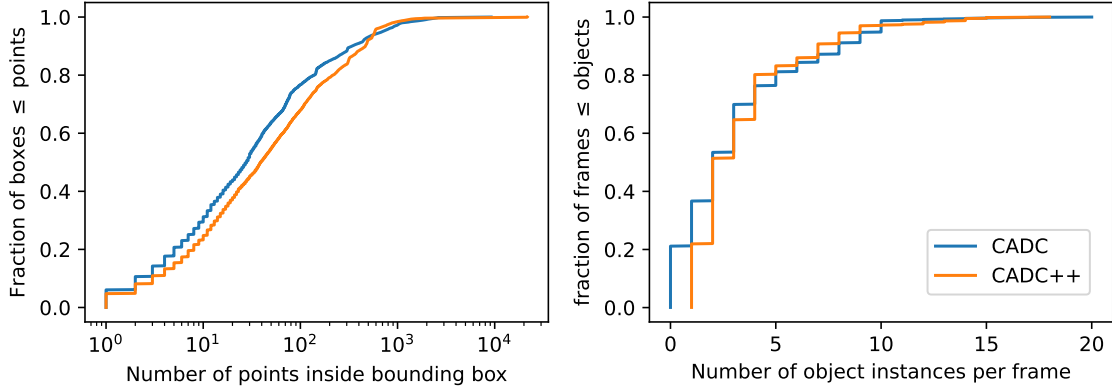


(a) Car

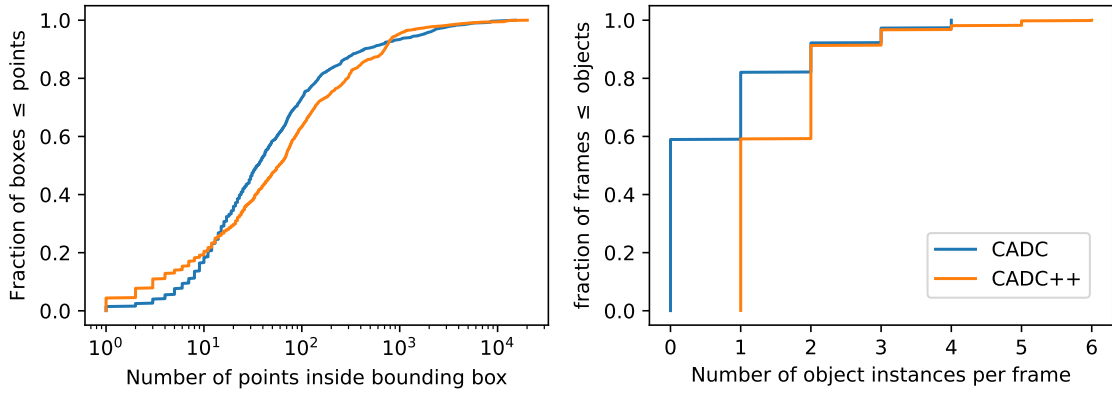
Figure 4.9: Distributions of point count per annotation and object count per frame for the Car class in CADC and CADC++.



(a) Pedestrian



(b) Truck



(c) Bus

Figure 4.10: Distributions of point count per annotation and object count per frame for lesser represented classes in CADC and CADC++.

Chapter 5

Track-based Auto-labelling for Sparsely Labelled Data

As CADC++ training sequences were labelled sparsely due to budget constraints, this chapter details our work in developing methods that can automatically label the unannotated intermediate frames with quality comparable to human annotations. While this work has been evaluated on the Waymo Open Dataset [61], we believe the methods can be applied to auto-label CADC++ and are likely to yield similar results.

5.1 Problem statement

We consider the problem of generating the best possible pseudo-labels while using the fewest number of human-annotated labels. In our context, the labels are available only for a fixed number of frames, sparsely sampled at constant intervals from a [LiDAR](#) sequence. In addition to label efficiency, we aim for a method that yields balanced performance on stationary and dynamic objects to avoid amplifying the existing bias towards stationary objects in common datasets [61, 6, 46]. High overall performance should not be achieved by solely increasing the detection performance of stationary objects. Lastly, we are not constrained by runtime requirements as we target an offline use case.

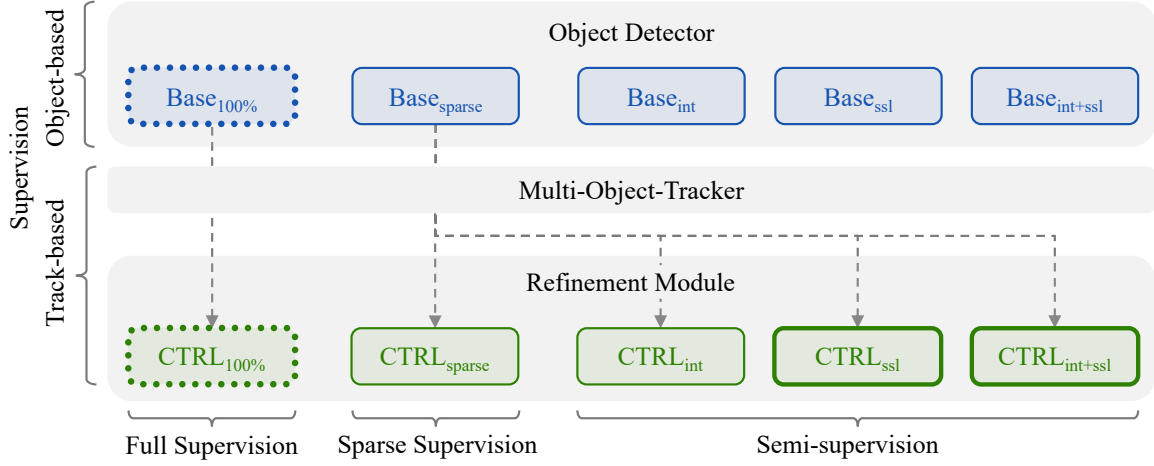


Figure 5.1: Overview and relationships between the methods and the resulting models we train, classified by their architecture and supervision. Methods with a dotted border are our baselines; our proposed methods have bolded borders. We use $Base_{sparse}$ as the first stage detector in all our non-baseline CTRL experiments to obtain comparable results. In practice, for best results, the highest-performing Base model should be used to produce the best possible initial detections.

5.2 Methods

We compare track-based and object-based pseudo-labelling methods to assess the effectiveness of track-based auto-labelling. Our experiments have one or two training stages, where each stage corresponds to training a model from randomized weights. When present, a second stage uses the output of the first stage.

Figure 5.1 gives an overview of the methods and the resulting models we evaluate in this work. The braces indicate two types of supervision based on their architecture and training pipeline, distinguishing between object- and track-based methods. In the first type, each Base model (in blue) is an object detector with object-based supervision, meaning that individual object instances are supervised independently. In the second type, each model (in green) is a refinement module, based on the architecture from CTRL [21], which performs track-based supervision on tracks resulting from the tracker acting on detections from the Base detector.

In Fig. 5.1, the dashed grey boxes classify the methods and models into three additional types of supervision depending on the quantity and type of labels used. Full supervision

means that all labels for all frames in all sequences are human-annotated and available during training, hence denoted by *100%*. Sparse supervision, however, denoted by *sparse*, has access to only sparse labels during training. These sparse labels are human-annotated *GT* labels that come from only a small number of evenly distributed frames within a sequence. Finally, semi-supervision involves training on a mixture of human-annotated *GT* labels and pseudo-labels. The latter can be created from either interpolation (denoted by *int*), a pre-trained model (denoted by *ssl*), or a combination of both (denoted by *int+ssl*).

The fully supervised refinement model CTRL_{100%} uses the fully supervised Base_{100%} as its base detector. All other CTRL models use Base_{sparse} as their base detector to ensure comparable results. Sparse and semi-supervision models are trained at different label sparsity levels, as explained shortly.

Our choice of model architecture and training parameters, which are shared across all baseline and subsequent experiments to ensure comparability, are as follows. For all Base experiments, we opt for the single-frame CenterPoint architecture for our off-the-shelf object detector, due to its popularity and familiarity. We use OpenPCDet’s [63] default configurations for single-frame CenterPoint, such as a *non-maximum suppression* (NMS) threshold of 0.7 and a one-cycle schedule with a maximum learning rate of 0.003 using the Adam optimizer. Data augmentation, including random world flip, rotation, scaling, and translation, is applied during training. For all CTRL experiments, we adopt CTRL’s version of ImmortalTracker [67], with a *NMS* threshold of 0.25 and a score threshold of 0.5. The refinement module is trained with default configurations, including noise augmentation for bounding box centres, sizes, and heading angles, as well as global track augmentation, such as random flip, rotation, scale, and translation.

Each of our fully supervised models is trained for 6 epochs, which is sufficient for the dataset size. Since models trained on sparser sets of labels receive less supervision, we compensate with more training time to ensure comparable results. Hence, for sparsely supervised models, we normalize the training time by increasing it proportionally to the fraction of labels available for supervision. We observed overfitting in only the CTRL_{sparse} experiments. In this case, we reduced the number of epochs until overfitting was no longer observed.

5.2.1 Dataset and sequence subsampling

The Waymo Open Dataset [61] has 1000 fully labelled sequences of approximately 200 frames each, split into training and validation sets of 798 and 202 sequences, respectively. Its fully annotated sequences enable experimentation with varying levels of label sparsity

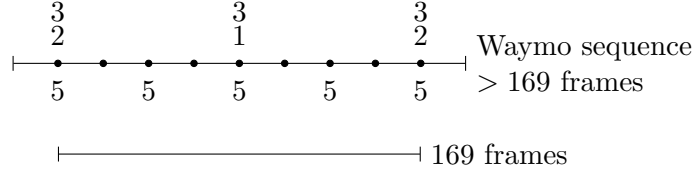


Figure 5.2: Sparse GT: numbers identify labelled frames for given fps; dots indicate 9 fps, with 20 frames between any two adjacent dots.

through subsampling. Additionally, its large scale and diverse driving scenes produce more objective results. We use only the vehicle class to demonstrate our findings, as it is the majority class and offers sufficient depth and diversity to give meaningful results that we believe will generalize to other classes.

We use all available GT labels in fully supervised training and subsample uniformly across all its training sequences to create sparse annotations for sparsely supervised training. We illustrate our sampling approach in Fig. 5.2. The original sequences have lengths in the range 171–200 frames, but we align the sequences by their midpoints and trim them symmetrically to a common length of 169 frames to ensure consistent subsampling intervals. This length allows us to have uniformly sampled ground-truth of 1, 2, 3, 5, and 9 labelled fps, while also making the sparser samples a subset of the denser ones.

The following subsections first give the necessary background on CTRL, our baseline track-based auto-labelling method, then describe each of the ten methods from Fig. 5.1 in more detail.

5.2.2 Notation

We refer to the first stage object detector models as Base_{method}^i and the second stage refinement models as CTRL_{method}^i , where $i \in \{1, 2, 3, 5, 9\}$ is the chosen sparsity level and $method \in \{100\%, \text{sparse}, \text{int}, \text{ssl}, \text{int+ssl}\}$ denotes the method name.

When describing the model type, rather than the specific model, we omit the sparsity level and use the simplified notation Base_{method} and CTRL_{method} (Fig. 5.1).

We define the following sets to represent the various input point clouds (PCs) and GT

labels used in training:

$$\begin{aligned}
X &= \{x \in \text{PCs from all frames}\} \\
X_i &= \{x \in \text{PCs from frames sampled at sparsity } i \text{ fps}\} \\
\overline{X}_i &= (X \setminus X_i) = \{x \in \text{PCs from frames not sampled}\} \\
Y &= \{y \in \text{GT labels from all frames}\} \\
Y_i &= \{y \in \text{GT labels from frames sampled at sparsity } i \text{ fps}\} \\
\overline{Y}_i &= f(Y_i, \overline{X}_i)
\end{aligned}$$

where $i \in \{1, 2, 3, 5, 9\}$ and $f \in \{\text{interpolate, pseudo-label}\}$. In general, $\overline{Y}_i$ can be seen as a set of pseudo-labels generated directly (by interpolation) or indirectly (by training a model) from the available GT data. The specific sets used by each of our methods are described in their respective sections.

5.2.3 Full supervision baselines: *Base*_{100%} & *CTRL*_{100%}

Fully supervised models are used as baselines to show the upper bound of what is achievable when the highest level of human annotation is available. All subsequent experiments demonstrate a comparatively reduced performance due to the effects of label sparsity. We thus use the set of all available point clouds X and their corresponding annotations Y to train *Base*_{100%} and *CTRL*_{100%}. Once *Base*_{100%} has completed training, we can use it for second-stage refinement training, as described in Sect. 2.6. We run inference with *Base*_{100%} on X to produce output detections that are then parsed by the tracker to obtain the track inputs needed to train *CTRL*_{100%}.

5.2.4 Sparse supervision models: *Base*_{sparse} & *CTRL*_{sparse}

Sparse supervision is a simple way to handle sparsely labelled data, i.e., a model is supervised solely using the sparse labels that are available. Since the base detector supervises individual object instances independently, regardless of the sequential nature of their trajectories, *Base*_{sparse} ^{i} can be trained using Y_i and the respective sparsely sampled point clouds X_i without any architectural modification. Prior to training, we simply subsample point clouds to create X_i , the same way we subsample labels Y_i . Then, we generate the sparse GT database for augmentation and train as usual.

The refinement module in CTRL, however, does not natively support sparse supervision. Input point clouds are collected at the LiDAR’s operational frequency of 10 Hz before going through initial detection by the Base model. The resulting object detections are fed into the tracker and converted into tracks without changing frequency. Finally, the tracks are matched one-to-one with track labels to undergo refinement training with the expectation that the labels are also of the same frequency. Since this is not the case, we devise CTRL_{sparse}, our adaptation of native CTRL for sparsely supervised refinement training, as described next.

Since CTRL’s refinement module performs track-level feature extraction, we choose to keep the input point clouds X at the full frequency of 10 Hz, maximizing the feature extraction potential by using all available point cloud data in a sequence. Then, sparsity is introduced at the supervision step by only using sparse labels instead of full track labels. Since some detections will now no longer have a corresponding label to supervise with, they can further be omitted from the object-level feature extraction step. In other words, we detect, track, and extract track-level features using X , but then extract object-level features and supervise with only sparse GT labels Y_i . Such adaptation of the refinement module supports sparse supervision while making use of all available sensor data.

5.2.5 Semi-supervision

Semi-supervision is a standard technique to make maximum use of limited GT labels. Hence, we develop a semi-supervised learning-based strategy for track-based auto-labelling. Pseudo-labels, used to train a semi-supervised model, can be obtained from unlabelled data in different ways. Described below, we experiment with the methods of interpolation, pseudo-labelling using a base detector, and a hybrid approach that involves both.

Interpolation models: $Base_{int}$ & $CTRL_{int}$

Since sparse annotations are uniformly sampled in time, a simple method of obtaining pseudo-labels for the unlabelled frames $\overline{X}_i$ is to interpolate from the sparse labels Y_i . To obtain interpolated pseudo-labels $\overline{Y}_i^{int}$, we first construct individual object tracks using the sparse labels Y_i of sparsity i . Then, for each sparse object track, we interpolate the object bounding box’s centre [*centre x*, *centre y*, *centre z*] and heading angle θ , while keeping the object’s unique identifier and dimensions [*length*, *width*, and *height*] the same throughout its entire track.

It is worth noting that while sequences are subsampled with a fixed number of frames, objects may only appear in some and not all of those sparsely sampled frames. This results in object tracks of various lengths, including empty tracks if the sparsity is low enough. Since different degrees of interpolation require different minimum numbers of data points, we adopt an *adaptive* approach: we perform cubic spline interpolation if a track has at least 4 frames, quadratic spline if 3, linear if 2, and none if only 1. Adaptive interpolation is applied to all vehicle tracks, regardless of their speed, acceleration, or trajectory curvature.

To obtain higher-quality interpolations with smoother trajectories, we perform two operations prior to interpolating. First, since object motions should be independent of the ego vehicle’s movement, we convert all labels to global world coordinates, as they are initially defined in local **LiDAR** coordinates. Another numerical detail we need to address is that **GT** heading angles are originally provided between $[-\pi, +\pi]$ rad, making it difficult to interpolate values that wrap around when exceeding this range. Hence, we use the standard phase unwrapping function from NumPy to minimize discontinuities in heading angles before interpolating.

Finally, we combine the interpolated pseudo-labels with sparse **GT** labels to supervise our interpolation-based semi-supervision experiments. After discarding any empty tracks, Base_{int} and CTRL_{int} can be trained the same way as our full supervision experiments, but with the newly devised set of pseudo-labels $\bar{Y}_i^{\text{int}} \cup Y_i$.

Pseudo-label models: Base_{ssl} & CTRL_{ssl}

An alternative method to obtain pseudo-labels for the unlabelled frames $\bar{X}_i$ is to run inference with a previously trained model, such as $\text{Base}_{\text{sparse}}^i$. The output detections are converted into pseudo-labels $\bar{Y}_i^{\text{ssl}}$ after discarding those with a score of less than 0.5, following [7].

Similar to the interpolation-based approach, we combine our pseudo-labels with the sparse **GT** labels to form the new set of pseudo-labels $\bar{Y}_i^{\text{ssl}} \cup Y_i$. Again, $\text{Base}_{\text{ssl}}^i$ can be trained the same way as $\text{Base}_{100\%}$, but supervised using $\bar{Y}_i^{\text{ssl}} \cup Y_i$ instead of Y . Training $\text{CTRL}_{\text{ssl}}^i$, however, requires pseudo-labels to become pseudo-label tracks first. Unlike sparse **GT** or interpolated labels, they do not have unique identifiers assigned. Therefore, we run the tracker on $\bar{Y}_i^{\text{ssl}} \cup Y_i$ to obtain pseudo-label tracks. $\text{CTRL}_{\text{ssl}}^i$ can then be trained the same way as $\text{CTRL}_{100\%}$, supervised with the newly generated pseudo-label tracks.

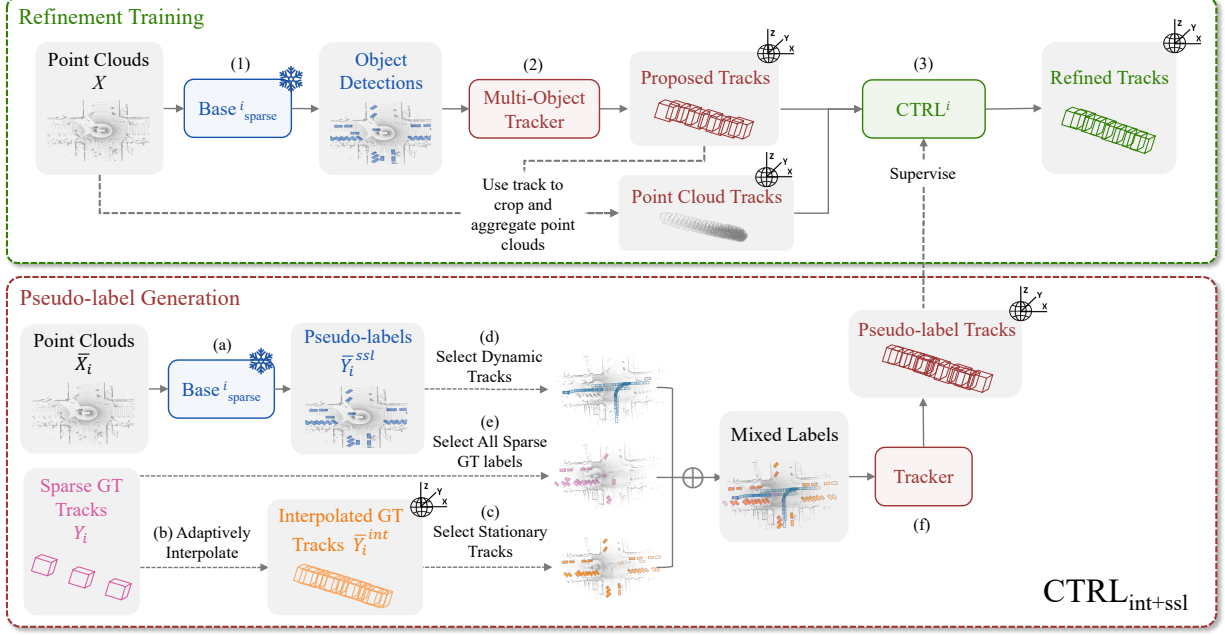


Figure 5.3: Constructing pseudo-label tracks for hybrid semi-supervised model training. (a) A sparse base detector infers initial pseudo-labels. (b) All object tracks are interpolated from sparse GT labels. (c) Only interpolated labels from stationary track segments are selected. (d) Only pseudo-labels that do not overlap with those from (c) are selected. (e) All selected interpolated labels and pseudo-labels are combined with sparse GT labels before being fed into the tracker in (f) to generate pseudo-label tracks.

Hybrid models: $Base_{int+ssl}$ & $CTRL_{int+ssl}$

We hypothesize that stationary objects can be interpolated with minimal error, as they do not vary significantly in speed or trajectory shape. In contrast, dynamic objects may exhibit more complex motion, resulting in poorer interpolations, especially as the sparsity increases. Therefore, we explore using interpolated labels for stationary objects, but resort to pseudo-labels for dynamic objects. To do so, we first need to classify object track segments as either stationary or dynamic, then we can combine the two types of labels. We define a *track segment* as a portion of an object trajectory that is bounded by two consecutive sparse GT labels. The next sections describe these steps in detail. An overview is depicted in Fig. 5.3.

Stationary track segments. We determine if a track segment is stationary using Waymo’s 0.2m/s threshold for stationary objects. First, the average speed of a track

segment is estimated using the displacement over time of its sparse GT endpoint bounding boxes. Then, we save all segments of estimated speed less than 0.2 m/s as stationary and the rest as dynamic. Next, from $\bar{Y}_i^{\text{int}}$, we keep only interpolated labels that fall within the time intervals of stationary segments (denoted *int-stat*) to create the new pseudo-label set $\bar{Y}_i^{\text{int-stat}}$.

Dynamic track segments. All segments not identified as stationary from the previous step are considered dynamic. We now need to gather pseudo-labels from $\bar{Y}_i^{\text{ssl}}$ that are part of dynamic track segments (denoted *ssl-dyn*) and merge them with the interpolated labels of stationary segments. However, due to the absence of object identities in pseudo-labels, we cannot work with individual tracks. Instead, we gather all pseudo-labels $\bar{Y}_i^{\text{ssl}}$ and discard those that overlap with the interpolated labels from stationary segments $\bar{Y}_i^{\text{int-stat}}$ using a fixed 3D Intersection over Union threshold of 0.1, resulting in the merged set of pseudo-labels $\bar{Y}_i^{\text{int-stat}} \cup \bar{Y}_i^{\text{ssl-dyn}}$.

Finally, we combine with sparse GT labels to form the pseudo-labels set $\bar{Y}_i^{\text{int-stat}} \cup \bar{Y}_i^{\text{ssl-dyn}} \cup Y_i$. $\text{Base}_{\text{int+ssl}}^i$ can be supervised with this set directly, but $\text{CTRL}_{\text{int+ssl}}^i$ needs pseudo-label tracks obtained from the tracker.

5.2.6 Evaluation

We evaluate using Waymo’s full validation set for all our experiments. We also use Waymo’s official evaluation metrics but combine all non-stationary velocity categories into one single dynamic category. The original threshold between stationary and dynamic objects (0.2 m/s) is retained. All results report Level 2 3D average precision (AP) for vehicle class.

5.3 Experiments

We present the main results of our training experiments in Fig. 5.4. Figure 5.4a gives the average precision of the vehicle class against the numbers of labelled frames per sequence, for different training methods. Figures 5.4b and 5.4c break down the results in terms of stationary and dynamic objects, respectively, as defined by the standard Waymo criteria.

Our full supervision baselines, $\text{Base}_{100\%}$ (dotted light grey line) and $\text{CTRL}_{100\%}$ (dotted grey line), indicate the maximum performance achievable using fully labelled data, given our models and training conditions. Since CTRL is an offline auto-labelling technique that

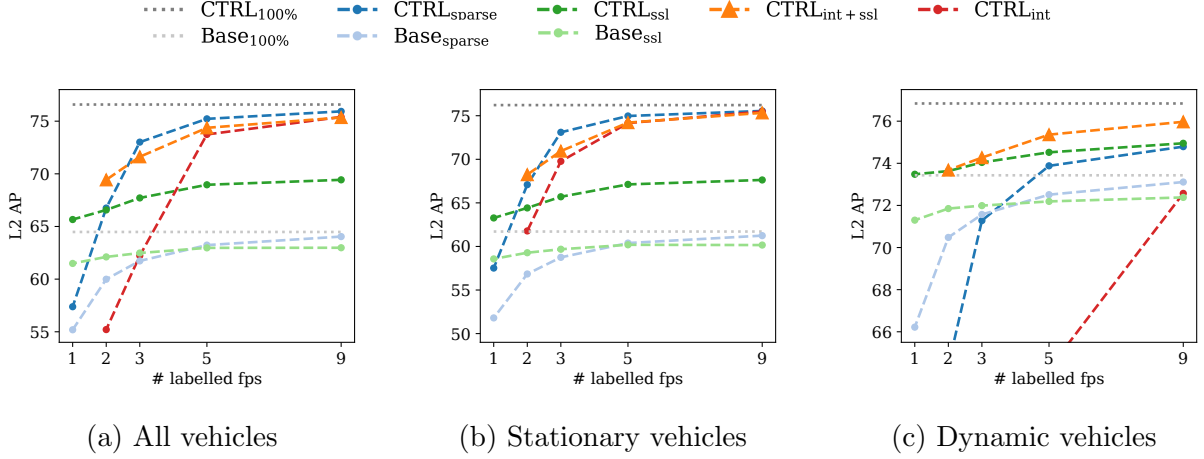


Figure 5.4: Level 2 vehicle 3D AP vs GT label frequency for different training methods. Base_{int} and Base_{int+ssl} show similar performance trends to their corresponding CTRL experiments, however their curves are excluded to reduce clutter. Base results are in lighter equivalent colours than CTRL results.

learns track-based features while exploiting knowledge from both the past and future, the performance of CTRL_{100%} is significantly greater than that of Base_{100%}. In the following sections, we break down the analysis of Base and CTRL experiments separately.

5.3.1 Base performance

Starting with the sparse supervision results obtained from Base_{sparse}^{*i*} (dashed light blue line), we observe that Base_{sparse}⁹ achieves results surprisingly close to those of Base_{100%}, with only 4.5% (9 fps) labelled data. Base_{sparse}¹ is less good than Base_{sparse}⁹, but nevertheless surprisingly performant, given that it uses only 0.5% (1 fps) labelled data. Additionally, the performance trends with stationary and dynamic vehicles are similar, suggesting no obvious bias towards either type.

As for the semi-supervision results, Base_{ssl}^{*i*} (dashed light green line) shows that pseudo-labels improve performance relative to Base_{sparse}^{*i*} when labelled data is very sparse ($i < 5$ fps), but deteriorate as labelled data become less sparse ($i \geq 5$ fps). We suppose this is due to the pseudo-labels contributing valuable diversity, while also containing low-quality labels and false positives that confuse the model. Similar to the trends in Base_{sparse}^{*i*}, there is no apparent bias regarding stationary and dynamic vehicles. Previous work [7] showed

that training with 100% of the labelled data combined with a large amount of pseudo-labelled data produces a significant increase in performance compared to training on the labelled data alone, implying that more pseudo-labels are generally better. However, the authors did not normalize their training times for the amount of data. Our results show that some of the performance gains reported by Caine et al. [7] are likely due to the additional training time. Moreover, we clearly show that training with pseudo-labels can actually deteriorate performance when the training time is normalized and the labelled data already have sufficient diversity.

5.3.2 CTRL performance

Supervising with adaptively interpolated pseudo-labels is entirely detrimental for low label frequencies. $\text{CTRL}_{\text{int}}^i$ (dashed red line) is worse than without interpolation for stationary objects and worse than all methods for dynamic objects. However, it shows promise for label frequencies above 9 fps. $\text{CTRL}_{\text{sparse}}^i$ (dashed blue line), on the other hand, demonstrates good overall performance compared to the $\text{CTRL}_{100\%}$ baseline. This is similar to the Base sparse supervision trends. Looking at the all-vehicle performance in Fig. 5.4a, $\text{CTRL}_{\text{sparse}}^i$ outperforms $\text{Base}_{\text{sparse}}^i$ for all sparsities $i \in \{1, 2, 3, 5, 9\}$, and is better than $\text{Base}_{100\%}$ and $\text{Base}_{\text{ssl}}^i$ for $i > 1$. However, since stationary vehicles are dominant, being approximately 80% of the training and validation data, the all-vehicle performance follows the performance of stationary vehicles and hides the much poorer performance of dynamic vehicles. In Fig. 5.4c, we see that $\text{CTRL}_{\text{sparse}}^i$ has much worse performance than both $\text{Base}_{\text{sparse}}^i$ and $\text{Base}_{\text{ssl}}^i$ on dynamic vehicles when $i < 5$.

The severe performance imbalance between stationary and dynamic vehicles of $\text{CTRL}_{\text{sparse}}$ and CTRL_{int} motivates our creation of CTRL_{ssl} and $\text{CTRL}_{\text{int+ssl}}$. $\text{CTRL}_{\text{ssl}}^i$ (dashed green line) overcomes such imbalance by dramatically improving the average precision for dynamic vehicles when $i < 5$. Moreover, $\text{CTRL}_{\text{ssl}}^i$ is the best choice for $i = 1$ fps. Similar to the semi-supervision behaviour of $\text{Base}_{\text{ssl}}^i$, training using pseudo-labels is beneficial for lower levels of sparsity, but eventually becomes detrimental, presumably because it introduces inaccuracies and false positives. Hence, we devise our hybrid approach $\text{CTRL}_{\text{int+ssl}}$ to exploit the high degree of interpolation accuracy of stationary objects, as well as the high dynamic performance of semi-supervised refinement training. $\text{CTRL}_{\text{int+ssl}}^i$ (dashed orange line) displays the highest dynamic performance for all levels of sparsity that support interpolation, while its stationary performance is very close to that of $\text{CTRL}_{\text{sparse}}^i$. Despite its improved dynamic detection, using pseudo-labels inevitably introduces confusion for stationary detection, as false positives are considered ground truth during supervision. This leads to a slight stationary performance degradation w.r.t. $\text{CTRL}_{\text{sparse}}^i$.

5.4 Discussion

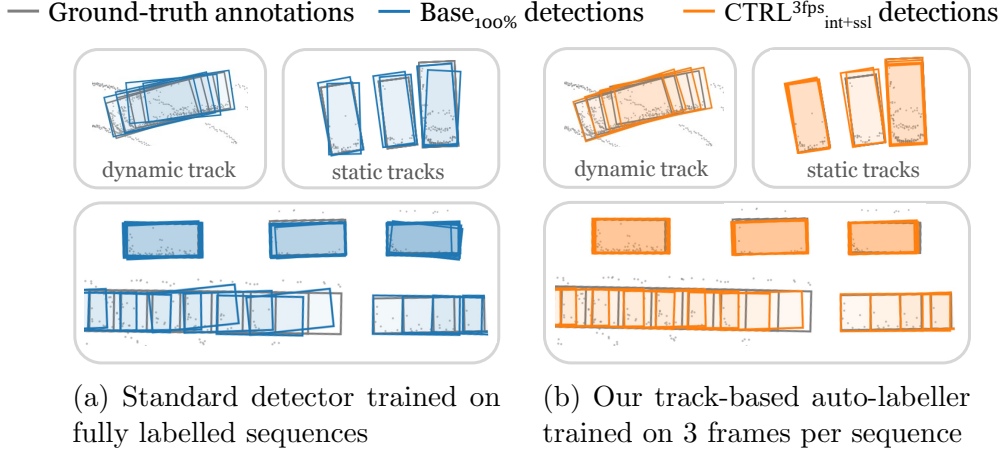


Figure 5.5: Qualitative detection performance of our track-based auto-labeller trained on sparsely labelled data compared to a standard detector trained on fully labelled data. Our method effectively refines detections along an object track, reducing the overall jitteriness that standard detectors might introduce when not considering the object’s full trajectory.

We find that annotating LiDAR sequences at a much lower frequency than their collection rate is a very effective strategy to save on human labelling costs. Figure 5.5 shows some detection examples of our sparse track-based auto-labeller compared to a standard detector trained on fully labelled data. Even our simplest method, training a Base model with standard augmentation, is effective and does not bias towards stationary vehicles. Semi-supervised learning, using pseudo-labels from a Base model, is even more effective for the most sparsely labelled data, but becomes detrimental as labelling becomes less sparse. Using an offline track-based auto-labelling approach, such as CTRL [21], can improve the performance of a Base model beyond what is possible when it is conventionally trained with fully labelled sequences.

After modifying native CTRL to support training on sparse annotations, our $\text{CTRL}_{\text{sparse}}$ adaptation produces the best detection results for stationary objects and sparsity $i > 2$ fps. However, such performance comes at the detriment of its detection capabilities for dynamic objects, being even worse than the Base model for sparsity $i \leq 3$ fps. In contrast, CTRL_{ssl} and $\text{CTRL}_{\text{int+ssl}}$ achieve balanced detection performance between stationary and dynamic objects. Using a combination of pseudo-labels and sparse human-generated GT labels to supervise the refinement module, CTRL_{ssl} not only achieves balanced stationary-dynamic

performance, but also surpasses CTRL_{sparse} in dynamic object detection. Moreover, it outperforms a Base model trained with fully labelled data, at all levels of sparsity. At 1 [fps](#), it is the superior approach in all evaluation categories. For the other levels of very sparse labels, we propose CTRL_{int+ssl}. Employing adaptive interpolation for stationary track segments and pseudo-labels for dynamic tracks, CTRL_{int+ssl} achieves the best dynamic detection performance and significantly improves on CTRL_{ssl}'s stationary object detection performance at all levels of sparsity. We note that the performance of CTRL_{int+ssl} with stationary objects is slightly below that of CTRL_{sparse} for sparsities $i \geq 3$ [fps](#). We hypothesize that this is due to the severe performance bias of CTRL_{sparse} in favour of stationary objects. We suppose that the model learns to avoid conflicts between detecting both stationary and dynamic vehicles by not detecting the dynamic ones. However, since we require a balanced detection performance, we deem CTRL_{int+ssl} to be superior to CTRL_{sparse}. For annotation frequencies greater than or equal to 9 [fps](#), one can consider using CTRL_{sparse} or simply adaptive interpolation.

We speculate that it might be possible to further increase performance by iterating some of our methods, such as using the Base_{ssl} model as the base model for CTRL_{int+ssl} or using the pseudo-labels produced by CTRL_{int+ssl} to train a new Base model. However, given a finite amount of ground truth, there are likely to be diminishing returns and possibly unforeseen interactions that reduce performance.

Chapter 6

Conclusion and Future Work

We construct CADC++, a paired weather domain adaptation dataset that extends the existing CADC dataset [46] with clear weather data. Each snowy sequence from CADC is matched with minimal spatial discrepancy along a driven trajectory to a unique sequence in clear weather. The resulting CADC++ shares similar size and object distributions as CADC, with minimized domain shifts from confounding factors other than falling and accumulated snow. Despite having to sparsely label CADC++’s training set at 10%, we can employ our track-based auto-labelling method to generate automatic annotations for the intermediate unlabelled frames. Evaluated on the Waymo Open Dataset [61], the auto-labels from our method demonstrated comparable quality to that of ground-truth labels, especially for the labelling sparsity level we used in CADC++.

With CADC++ ready for use alongside CADC, future work can tackle our initial research goal of evaluating the effect of snow on 3D object detection performance. We can train a 3D object detector in clear weather data and evaluate its potential performance degradation in snowy scenes. A snowy model evaluated on clear weather data might also provide interesting insight. Additionally, we can compare their performance with models trained on either simulated snowy or de-snowed data. At last, a weather-paired like CADC++ enables domain adaptation research in many other directions, such as improving current learning-based de-snowing or weather-based style transfer methods.

References

- [1] Haris Balta, Jasmin Velagic, Walter Bosschaerts, Geert De Cubber, and Bruno Siciliano. Fast statistical outlier removal based method for large 3d point clouds of outdoor environments. *IFAC-PapersOnLine*, 51(22):348–353, 2018. 12th IFAC Symposium on Robot Control SYROCO 2018.
- [2] Johan Bergius and Jesper Holmblad. Lidar point cloud de-noising for adverse weather. Master’s thesis, , School of Information Technology, 2022.
- [3] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3D object detection, 2020.
- [4] Mario Bijelic, Tobias Gruber, Fahim Mannan, Florian Kraus, Werner Ritter, Klaus Dietmayer, and Felix Heide. Seeing through fog without seeing fog: Deep multimodal sensor fusion in unseen adverse weather. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11682–11692, 2020.
- [5] Keenan Burnett, David J Yoon, Yuchen Wu, Andrew Z Li, Haowei Zhang, Shichen Lu, Jingxing Qian, Wei-Kang Tseng, Andrew Lambert, Keith YK Leung, Angela P Schoellig, and Timothy D Barfoot. Boreas: A multi-season autonomous driving dataset. *The International Journal of Robotics Research*, 42(1-2):33–42, 2023.
- [6] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [7] Benjamin Caine, Rebecca Roelofs, Vijay Vasudevan, Jiquan Ngiam, Yuning Chai, Zhifeng Chen, and Jonathon Shlens. Pseudo-labeling for scalable 3D object detection, 2021.

- [8] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Celine Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [9] Nicholas Charron, Stephen Phillips, and Steven L. Waslander. De-noising of lidar point clouds corrupted by snowfall. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 254–261, 2018.
- [10] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [11] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [12] Xieyuanli Chen, Benedikt Mersch, Lucas Nunes, Rodrigo Marcuzzi, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Automatic labeling to generate training data for online lidar-based moving object segmentation. *IEEE Robotics and Automation Letters*, 7(3):6107–6114, 2022.
- [13] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. MPPNet: Multi-frame feature intertwining with proxy points for 3D temporal object detection. In *European Conference on Computer Vision (ECCV)*, pages 680–697. Springer, 2022.
- [14] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. VoxelNeXt: Fully sparse VoxelNet for 3D object detection and tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21674–21683, 2023.
- [15] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: Towards high performance voxel-based 3D object detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(2):1201–1209, May 2021.
- [16] Carlos A. Diaz-Ruiz, Youya Xia, Yurong You, Jose Nino, Junan Chen, Josephine Monica, Xiangyu Chen, Katie Luo, Yan Wang, Marc Emond, Wei-Lun Chao, Bharath Hariharan, Kilian Q. Weinberger, and Mark Campbell. Ithaca365: Dataset and driving

- perception under repeated and challenging weather conditions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21383–21392, June 2022.
- [17] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. CenterNet: Keypoint triplets for object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6569–6578, 2019.
 - [18] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3D object detector with sparse transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8458–8468, 2022.
 - [19] Lue Fan, Feng Wang, Naiyan Wang, and ZHAO-XIANG ZHANG. Fully sparse 3D object detection. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 351–363. Curran Associates, Inc., 2022.
 - [20] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. RangeDet: In defense of range view for lidar-based 3D object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2918–2927, 2021.
 - [21] Lue Fan, Yuxue Yang, Yiming Mao, Feng Wang, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Once detected, never lost: Surpassing human performance in offline LiDAR based 3D object detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19820–19829, 2023.
 - [22] Lue Fan, Yuxue Yang, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Super sparse 3D object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12490–12505, 2023.
 - [23] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
 - [24] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. LiDAR snowfall simulation for robust 3D object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16364–16374, June 2022.

- [25] Tong He and Stefano Soatto. Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8409–8416, Jul. 2019.
- [26] Robin Heinzler, Florian Piewak, Philipp Schindler, and Wilhelm Stork. CNN-Based lidar point cloud de-noising in adverse weather. *IEEE Robotics and Automation Letters*, 5(2):2514–2521, 2020.
- [27] Chengjie Huang, Vahdat Abdelzad, Sean Sedwards, and Krzysztof Czarnecki. SOAP: Cross-sensor domain adaptation for 3D object detection using Stationary Object Aggregation Pseudo-labelling. In *IEEE/CVF Winter Conf. on Applications of Computer Vision (WACV)*, 2024.
- [28] Maria Jokela, Matti Kutila, and Pasi Pyykönen. Testing and validation of automotive point-cloud sensors in adverse weather conditions. *Applied Sciences*, 9(11), 2019.
- [29] Seong-heum Kim and Youngbae Hwang. A survey on deep learning based methods and datasets for monocular 3d object detection. *Electronics*, 10(4), 2021.
- [30] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L. Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8, 2018.
- [31] Akhil Kurup and Jeremy Bos. DSOR: A scalable statistical filter for removing falling snow from lidar point clouds in severe winter weather. *arXiv preprint arXiv:2109.07078*, 2021.
- [32] Akhil Kurup and Jeremy Bos. Winter adverse driving dataset (WADS): year three. In Michael C. Dudzik, Stephen M. Jameson, and Theresa J. Axenson, editors, *Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea and Space Vehicles and Infrastructure 2022*, volume 12115, page 121150H. International Society for Optics and Photonics, SPIE, 2022.
- [33] Matti Kutila, Pasi Pyykönen, Maria Jokela, Tobias Gruber, Mario Bijelic, and Werner Ritter. Benchmarking automotive lidar performance in arctic conditions. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, 2020.
- [34] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In

- IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019.
- [35] Boyang Li, Jieling Li, Gang Chen, Hejun Wu, and Kai Huang. De-snowing lidar point clouds with intensity and spatial-temporal features. In *International Conference on Robotics and Automation (ICRA)*, pages 2359–2365, 2022.
 - [36] Tao Ma, Xuemeng Yang, Hongbin Zhou, Xin Li, Botian Shi, Junjie Liu, Yuchen Yang, Zhizheng Liu, Liang He, Yu Qiao, Yikang Li, and Hongsheng Li. DetZero: Rethinking offboard 3D object detection with long-term sequential point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6713–6724, 2023.
 - [37] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Chunjing Xu, et al. One million scenes for autonomous driving: ONCE dataset. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.
 - [38] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision*, 131(8):1909–1963, 2023.
 - [39] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. LaserNet: An efficient probabilistic 3D object detector for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12677–12686, 2019.
 - [40] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
 - [41] Teddy Ort, Igor Gilitschenski, and Daniela Rus. GROUNDED: The localizing ground penetrating radar evaluation dataset. In *Robotics: Science and Systems*, volume 2, 2021.
 - [42] Ziqi Pang, Zhichao Li, and Naiyan Wang. Model-free vehicle tracking and state estimation in point cloud sequences. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8075–8082. IEEE, 2021.
 - [43] Ji-Il Park, Jihyuk Park, and Kyung-Soo Kim. Fast and accurate desnowing algorithm for lidar point clouds. *IEEE Access*, 8:160202–160212, 2020.

- [44] Abhishek Patil, Srikanth Malla, Haiming Gang, and Yi-Ting Chen. The H3D dataset for full-surround 3D multi-object detection and tracking in crowded urban scenes. In *International Conference on Robotics and Automation (ICRA)*, pages 9552–9557, 2019.
- [45] Florian Piewak, Peter Pinggera, Manuel Schafer, David Peter, Beate Schwarz, Nick Schneider, Markus Enzweiler, David Pfeiffer, and Marius Zollner. Boosting lidar-based semantic labeling by cross-modal training data generation. In *European Conference on Computer Vision (ECCV)*, September 2018.
- [46] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.
- [47] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3D object detection in point clouds. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9277–9286, 2019.
- [48] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [49] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.
- [50] Charles R. Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3D object detection from point cloud sequences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6134–6144, 2021.
- [51] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.
- [52] Zengyi Qin, Jinglu Wang, and Yan Lu. Weakly supervised 3D object detection from point clouds. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 4144–4152, 2020.
- [53] R. H. Rasshofer, M. Spies, and H. Spies. Influences of weather phenomena on automotive laser radar systems. *Advances in Radio Science*, 9:49–60, 2011.

- [54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
- [55] R. S. Sekhon and R. C. Srivastava. Snow size spectra and radar reflectivity. *Journal of Atmospheric Sciences*, 27(2):299 – 307, 1970.
- [56] Alvari Seppänen, Risto Ojala, and Kari Tammi. 4denoisenet: Adverse weather denoising from adjacent point clouds. *IEEE Robotics and Automation Letters*, 8(1):456–463, 2023.
- [57] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. PV-RCNN: Point-voxel feature set abstraction for 3D object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10529–10538, 2020.
- [58] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3D object detection. *International Journal of Computer Vision*, 131(2):531–551, 2023.
- [59] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.
- [60] Chenghao Sun, Pengpeng Sun, Jianqiu Wang, Ying Guo, and Xiangmo Zhao. Understanding lidar performance for autonomous vehicles under snowfall conditions. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–11, 2024.
- [61] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo Open Dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [62] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. RSN: Range sparse net for efficient, accurate lidar 3D object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5725–5734, 2021.

- [63] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3D object detection from point clouds. <https://github.com/open-mmlab/OpenPCDet>, 2020.
- [64] Sven Teufel, Georg Volk, Alexander Von Bernuth, and Oliver Bringmann. Simulating realistic rain, snow, and fog variations for comprehensive performance characterization of lidar perception. In *2022 IEEE 95th Vehicular Technology Conference: (VTC2022-Spring)*, pages 1–7, 2022.
- [65] Jorge Vargas, Suleiman Alsweiss, Onur Toker, Rahul Razdan, and Joshua Santos. An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors*, 21(16), 2021.
- [66] Sean Walsh, Jason Ku, Alex D Pon, and Steven L Waslander. Leveraging temporal data for automatic labelling of static vehicles. In *17th Conference on Computer and Robot Vision (CRV)*, pages 134–141. IEEE, 2020.
- [67] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. Immortal tracker: Tracklet never dies. *CoRR*, abs/2111.13672, 2021.
- [68] Weimin Wang, Ting Yang, Yu Du, and Yu Liu. Snow removal for lidar point clouds with spatio-temporal conditional random fields. *IEEE Robotics and Automation Letters*, 8(10):6739–6746, 2023.
- [69] Weiqi Wang, Xiong You, Lingyu Chen, Jiangpeng Tian, Fen Tang, and Lantian Zhang. A scalable and accurate de-snowing algorithm for LiDAR point clouds in winter. *Remote Sensing*, 14(6), 2022.
- [70] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *35th Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.
- [71] J. Wojtanowski, M. Zygmunt, M. Kaszczuk, Z. Mierczyk, and M. Muzal. Comparison of 905 nm and 1550 nm semiconductor laser rangefinders’ performance deterioration due to adverse environmental conditions. *Opto-Electronics Review*, 22(3):183–190, 2014.
- [72] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. PandaSet: Advanced sensor suite

- dataset for autonomous driving. In *IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021.
- [73] Xinyuan Yan, Junxing Yang, Xinyu Zhu, Yu Liang, and He Huang. Denoising framework based on multiframe continuous point clouds for autonomous driving lidar in snowy weather. *IEEE Sensors Journal*, 24(7):10515–10527, 2024.
 - [74] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10), 2018.
 - [75] Anqi Joyce Yang, Sergio Casas, Nikita Dvornik, Sean Segal, Yuwen Xiong, Jordan Sir Kwang Hu, Carter Fang, and Raquel Urtasun. LabelFormer: Object trajectory refinement for offboard perception from LiDAR point clouds. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pages 3364–3383. PMLR, 2023.
 - [76] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. Auto4D: Learning to label 4D objects from sequential point clouds, 2021.
 - [77] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7652–7660, 2018.
 - [78] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-based 3D single stage object detector. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11040–11048, 2020.
 - [79] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3D-MAN: 3D multi-frame attention network for object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1863–1872, 2021.
 - [80] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3D object detection and tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11784–11793, 2021.
 - [81] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. Lisownet: Real-time snow removal for lidar point clouds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6820–6826, 2022.
 - [82] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

- [83] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D objects with differentiable rendering of SDF shape priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12224–12233, 2020.
- [84] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177, 2023.
- [85] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 923–932. PMLR, 30 Oct–01 Nov 2020.
- [86] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.
- [87] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. CenterFormer: Center-based transformer for 3D object detection. In *European Conference on Computer Vision (ECCV)*, pages 496–513. Springer, 2022.