3D Object Detection with Track-based Auto-Labelling using Very Sparsely Labelled Data

Mei Qi Tang, Vahdat Abdelzad, Chengjie Huang, Sean Sedwards, Krzysztof Czarnecki
{meiqi.tang,vahdat.abdelzad,c.huang,sean.sedwards,k2czarne}@uwaterloo.ca
University of Waterloo

***Abstract*— In the context of LiDAR-based 3D object detection, we consider the problem of generating high-quality pseudo-labels from very sparsely labelled data. We focus on track-based auto-labelling, which is a class of state-of-the-art pseudo-labelling methods that exploits the sequential nature of point cloud collection, but typically expects training data to be densely labelled. In this work, we analyze different ways to adapt a particular track-based auto-labelling approach to sparsely labelled sequential data from the Waymo Open Dataset, valuing balanced performance on stationary and dynamic vehicles. We thus propose methods that achieve high performance on both of these categories, with as few as one labelled frame per sequence.**

I. Introduction

This paper concerns 3D object detection based on Light Detection And Ranging (LiDAR) technology, which offers high-resolution 3D sensing capabilities and robustness to varied lighting conditions. Vehicle-mounted LiDAR sensors provide a dynamic perspective: as the vehicle moves, LiDAR data is collected as a sequence of frames at a fixed frequency, typically ≥ 10 Hz, such that every frame provides a slightly different viewpoint of the scene. We focus on methods that exploit the temporal continuity and smoothly varying perspectives provided by such sequential data.

To train a deep model that can accurately detect objects from LiDAR scans, it is necessary to obtain a large amount of training data with high-quality annotations for supervision. Such annotations are typically provided by human annotators and are therefore expensive to obtain. To reduce the human annotation effort, various strategies make use of unlabelled data, since simply collecting LiDAR sequences is relatively cheap. Such strategies include the following:

- *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.
- *Pseudo-labelling*, where a pre-trained model is used to detect and (pseudo-)label objects in raw sensor data; pseudo-labels have no inherent identities related to human labels.
- *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.

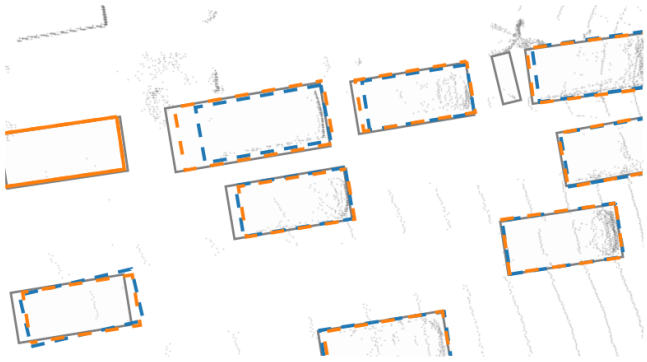


Fig. 1: Our track-based auto-labeller trained with only 1.5% labels (orange) has greater accuracy and more true positives than a standard high-performance model trained using 100% labels (blue), illustrated against ground-truth vehicles (grey). Solid borders indicate detection by a single model; dashed borders indicate detection by both. For a fair comparison, only predictions with scores > 0.5 are shown.

- *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.

In this work, we consider the problem of generating the best possible pseudo-labels with the fewest human-annotated labels. We analyze various techniques and ultimately propose methods that achieve high performance with as little as 0.5% labelled data. Figure 1 illustrates typical results of our proposed methods.

To reduce labelling, some datasets provide unlabelled sequences in addition to fully labelled sequences [13, 3], while others annotate all sequences at a lower frequency [1, 9, 10]. We refer to the latter as *sparse labelling*. Most state-of-the-art (SOTA) track-based auto-labelling methods assume full availability of annotations in either all sequences [5, 19, 20] or a subset of them [11, 7], then train a model on the fully labelled sequences. However, reducing the number of human-labelled sequences inevitably results in a loss of data diversity. Hence, [11, 7] also make use of unlabelled sequences to further train under the semi-supervised learning paradigm.

Some datasets use simple linear interpolation to create

intermediate labels from adjacent human-created annotations [1, 9]. The interpolated labels are then treated as ground truth. Interpolation is thus an obvious candidate to overcome the problem of sparsely labelled data, but even the use of sophisticated interpolation models is not enough at the very sparse labelling frequencies that are of interest to us. We nevertheless show how interpolated labels can be combined with other pseudo-labels in a highly effective way.

It has been observed that commonly used public datasets, such as the Waymo Open Dataset [13] and nuScenes [1], have a strong bias towards stationary objects [6]. While some training approaches make explicit use of this fact [6], others do so implicitly, achieving high overall performance by increasing the detection performance of stationary objects. In contrast to these, we seek a balanced performance on stationary and dynamic objects, to avoid amplifying any existing biases in the data.

In our experiments, we use a standard CenterPoint architecture for our base models, and choose CTRL [5] as the basis of our track-based auto-labelling pipeline (described in Sect. III-B). We perform training experiments on the Waymo Open Dataset [13], whose LiDAR data is collected in sequences that each contain approximately 200 frames. We consider labelling frequencies of $(1, 2, 3, 5, 9)$ frames per sequence (fps), which amount to approximately $(0.5, 1.0, 1.5, 2.5, 4.5)\%$ of the data. These frequencies correspond to levels of sparsity that 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 contributions are as follows:

- We adapt a track-based auto-labelling pipeline for use with sequential training data that is not fully labelled.
- We conduct experiments with extreme levels of label sparsity that 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 a combination of 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.

II. Related Work

A. Track-based auto-labelling

Earlier works in LiDAR-based 3D auto-labelling either use a pre-trained 2D model [21, 12], focus only on stationary objects [15], or are model-free and rely instead on tracking and pose estimation [4, 8]. However, despite requiring little to no supervision, they struggle to produce high-quality pseudo-labels that are comparable to human annotations. More recent works [11, 20, 5, 7, 19] adopt a two-stage object trajectory refinement approach: 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 [11], while others do not differentiate by object motion [20, 5, 7, 19].

Many of these methods [20, 19, 5] only train under full supervision, expecting all sequences to contain ground-truth annotations at every frame. While [11, 7] evaluate their approaches under the framework of semi-supervised learning, they still expect a subset of sequences to be fully annotated, in conjunction with a subset of fully unlabelled sequences. We have not found existing work in offline auto-labelling that specifically handles sparsely labelled data.

B. Datasets with sequential data

A large number of driving datasets have been developed following the rapid advancements in autonomous driving research. They vary by characteristics such as size, location, modalities, and annotation strategy. Focusing on the latter, densely labelled datasets like the Waymo Open Dataset [13], Argoverse 2 Sensor Dataset [17], and PandaSet [18] provide annotations for all LiDAR point clouds, resulting in a labelling frequency equal to the LiDAR's operating frequency. In contrast, nuScenes [1], H3D [9], and CADC [10] only label keyframes, at fixed intervals, leaving intermediate frames unlabelled. 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 propose linearly interpolating labels for unlabelled frames, suggesting that 2 Hz is robust to up-sampling, but there is a lack of evaluation results.

III. 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. 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 2 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 [5], which performs track-based supervision on tracks resulting from the tracker acting on detections from the Base detector.

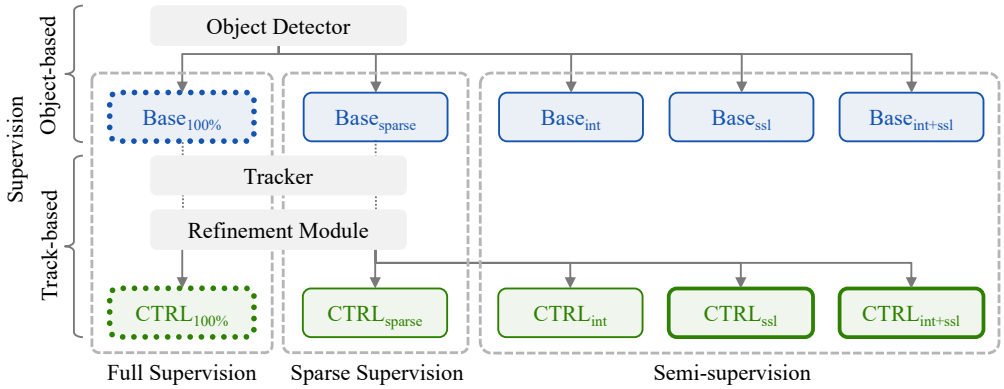


Fig. 2: Overview and relationship between the methods and the resulting models we train, classified by their architecture and supervision. Methods with a dotted border are our baselines; our proposed methods have bolded borders.

In Fig. 2, 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 $\text{CTRL}_{100\%}$ uses the fully supervised $\text{Base}_{100\%}$ as its base detector. All other CTRL models use $\text{Base}_{\text{sparse}}$ 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 [14] 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 [16], 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 $\text{CTRL}_{\text{sparse}}$ experiments. In this case, we reduced the number of epochs until overfitting was no longer observed.

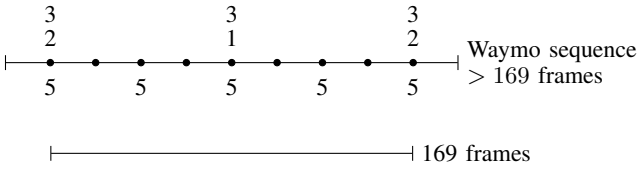


Fig. 3: Sparse GT: numbers identify labelled frames for given fps (frames per sequence); dots indicate 9 fps, with 20 frames between any two adjacent dots.

A. Dataset and sequence subsampling

We use the Waymo Open Dataset [13] due to its large scale and high data diversity. We use only the vehicle class to demonstrate our findings, since vehicle is the majority class and has sufficient depth and diversity to give meaningful results that we believe will generalize to other classes. We experiment with all 798 training and 202 validation sequences. We use all available GT labels in fully supervised training and subsample uniformly across all sequences to create sparse annotations for sparsely supervised training. We illustrate our sampling approach in Fig. 3. 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 frames per sequence (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. 2 in more detail.

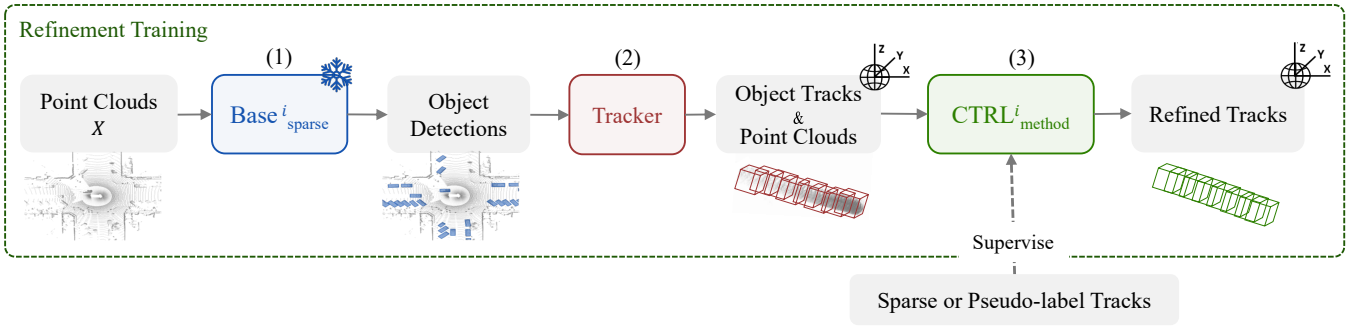


Fig. 4: Overview of the refinement training pipeline in track-based auto-labelling using sparsely labelled data. (1) A base detector ($\text{Base}^i_{\text{sparse}}$), pre-trained on sparse GT labels of sparsity i, infers initial detections. (2) A multi-object tracker builds tracks of bounding boxes and point clouds using those detections. (3) The refinement module ($\text{CTRL}^i_{\text{method}}$) extracts features from those input tracks and supervises them individually using either sparse or pseudo-label tracks, depending on the method.

B. Track-based auto-labelling pipeline

We choose SOTA auto-labelling method CTRL [5] as the basis of our two-stage refinement experiments. Referring to Fig. 4, CTRL consists of three main components: (1) an off-the-shelf object detector localizes initial object instances within each input LiDAR point cloud (PC); (2) a multi-object tracker tracks as many objects as possible across subsequent frames, while forming coarse object trajectories; (3) a refinement module learns to refine the trajectories using both track- and object-level features. Internally, the refinement module first uses the aggregated PCs of the input track to perform track-level feature extraction. Next, it uses the track proposals to crop those track-level features around each object to perform object-level feature extraction. Finally, after assigning a GT bounding box to each proposal, it supervises their refinement individually through classification and regression losses. We refer the reader to [5] for more architectural details on CTRL.

C. Notation

We refer to the first stage object detector models as Base^i_{method} and the second stage refinement models as CTRL^i_{method}, 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. 2).

We define the following sets to represent the various input PCs and GT labels used in training:

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

where $i \in \{1, 2, 3, 5, 9\}$ and $f \in \{\text{interpolate}, \text{pseudo-label}\}$. 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.

D. Full supervision baselines: $\text{Base}_{100\%}$ & $\text{CTRL}_{100\%}$

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 $\text{Base}_{100\%}$ and $\text{CTRL}_{100\%}$. Once $\text{Base}_{100\%}$ has completed training, we can use it for second-stage refinement training, as described in Sect. III-B. We run inference with $\text{Base}_{100\%}$ on X to produce output detections that are then parsed by the tracker to obtain the track inputs needed to train $\text{CTRL}_{100\%}$.

E. Sparse supervision models: $\text{Base}_{\text{sparse}}$ & $\text{CTRL}_{\text{sparse}}$

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, $\text{Base}^i_{\text{sparse}}$ 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 $\text{CTRL}_{\text{sparse}}$, 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.

F. 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.

1) 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 $\overline{Y_i}^{\text{int}}$, 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 $\overline{Y_i}^{\text{int}} \cup Y_i$.

2) Pseudo-label models Base_{ssl} & CTRL_{ssl}*:* An alternative method to obtain pseudo-labels for the unlabelled frames $\overline{X_i}$ is to run inference with a previously trained model, such as $\text{Base}^i_{\text{sparse}}$. The output detections are converted into pseudo-labels $\overline{Y_i}^{\text{ssl}}$ after discarding those with a score of less than 0.5, following [2].

Similar to the interpolation-based approach, we combine our pseudo-labels with the sparse GT labels to form the new set of pseudo-labels $\overline{Y_i}^{\text{ssl}} \cup Y_i$. Again, $\text{Base}^i_{\text{ssl}}$ can be trained the same way as $\text{Base}_{100\%}$, but supervised using $\overline{Y_i}^{\text{ssl}} \cup Y_i$ instead of Y. Training $\text{CTRL}^i_{\text{ssl}}$, 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 $\overline{Y_i}^{\text{ssl}} \cup Y_i$ to obtain pseudo-label tracks. $\text{CTRL}^i_{\text{ssl}}$ can then be trained the same way as $\text{CTRL}_{100\%}$, supervised with the newly generated pseudo-label tracks.

3) Hybrid models $\text{Base}_{\text{int+ssl}}$ & $\text{CTRL}_{\text{int+ssl}}$*:* 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.

a) Stationary track segments: We determine if a track segment is stationary using Waymo's $0.2\,\text{m/s}$ 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 $0.2\,\text{m/s}$ as stationary and the rest as dynamic. Next, from $\overline{Y_i}^{\text{int}}$, we keep only interpolated labels that fall within the time intervals of stationary segments (denoted *int-stat*) to create the new pseudo-label set $\overline{Y_i}^{\text{int-stat}}$.

b) Dynamic track segments: All segments not identified as stationary from the previous step are considered dynamic. We now need to gather pseudo-labels from $\overline{Y_i}^{\text{ssl}}$ 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 $\overline{Y_i}^{\text{ssl}}$ and discard those that overlap with the interpolated labels from stationary segments $\overline{Y_i}^{\text{int-stat}}$ using a fixed 3D Intersection over Union threshold of 0.1, resulting in the merged set of pseudo-labels $\overline{Y_i}^{\text{int-stat}} \cup \overline{Y_i}^{\text{ssl-dyn}}$.

Finally, we combine with sparse GT labels to form the pseudo-labels set $\overline{Y_i}^{\text{int-stat}} \cup \overline{Y_i}^{\text{ssl-dyn}} \cup Y_i$. $\text{Base}^i_{\text{int+ssl}}$ can

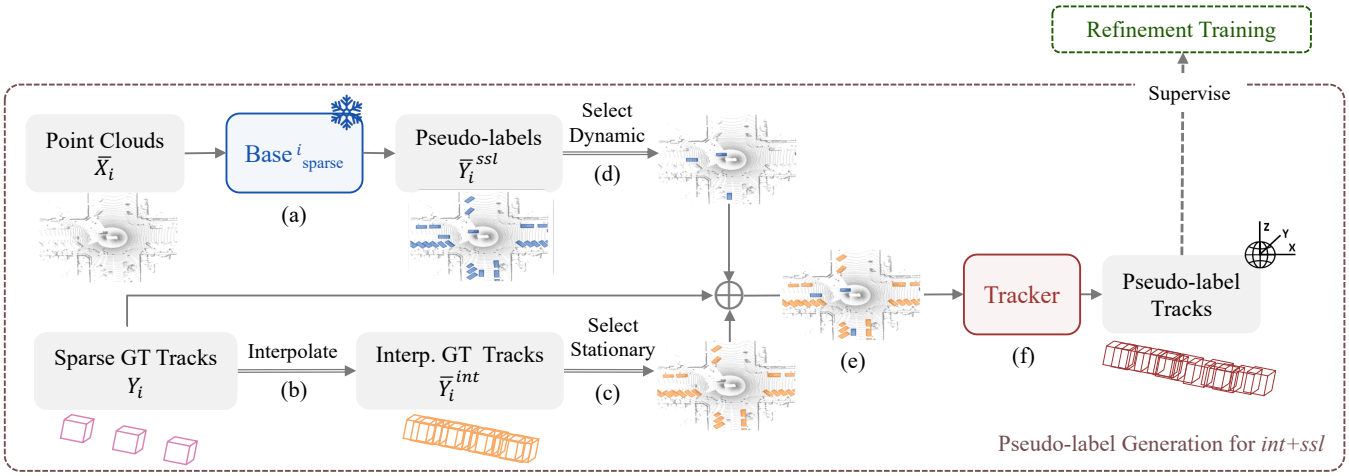


Fig. 5: Constructing pseudo-label tracks for hybrid semi-supervised model training. (a) A sparse base detector infers initial pseudo-labels. (b) All object tracks are interpolated from sparse GT labels. (c) Only interpolated labels from stationary track segments are selected. (d) Only pseudo-labels that 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.

be supervised with this set directly, but $\text{CTRL}^i_{\text{int+ssl}}$ needs pseudo-label tracks obtained from the tracker.

G. 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.

IV. Experiments

We present the main results of our training experiments in Fig. 6. Figure 6a gives the average precision of the vehicle class against the numbers of labelled frames per sequence, for different training methods. Figures 6b and 6c 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 blue line) and $\text{CTRL}_{100\%}$ (dotted red 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 learns track-based features while exploiting knowledge from both the past and future, the performance of $\text{CTRL}_{100\%}$ is significantly greater than that of $\text{Base}_{100\%}$. In the following sections, we break down the analysis of Base and CTRL experiments separately.

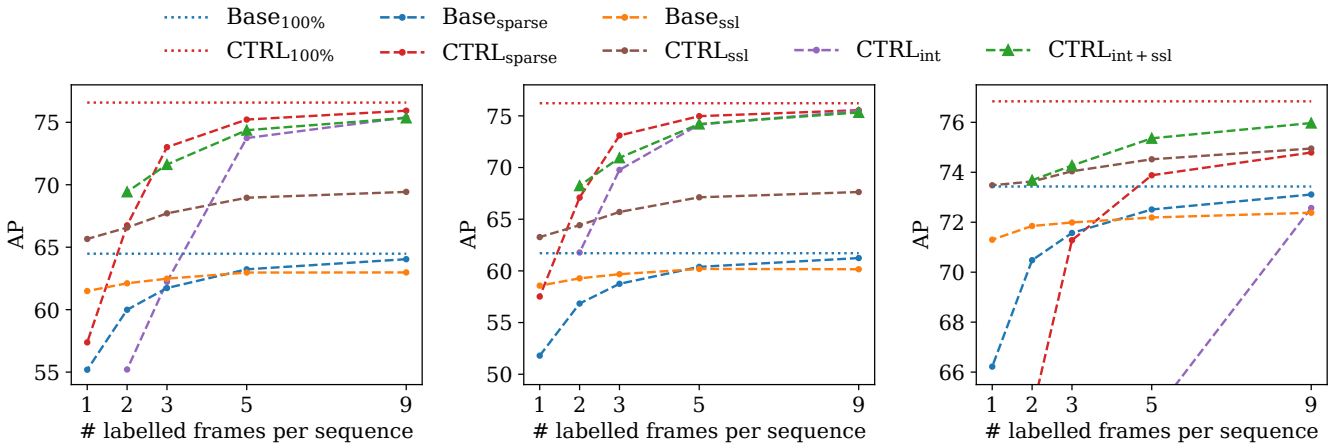
A. Base *performance*

Starting with the sparse supervision results obtained from $\text{Base}^i_{\text{sparse}}$ (dashed blue line), we observe that $\text{Base}^9_{\text{sparse}}$ achieves results surprisingly close to those of $\text{Base}_{100\%}$, with only 4.5% (9 fps) labelled data. $\text{Base}^1_{\text{sparse}}$ is less good than $\text{Base}^9_{\text{sparse}}$, 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, $\text{Base}^i_{\text{ssl}}$ (dashed orange line) shows that pseudo-labels improve performance relative to $\text{Base}^i_{\text{sparse}}$ 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 $\text{Base}^i_{\text{sparse}}$, there is no apparent bias regarding stationary and dynamic vehicles. Previous work [2] 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 in [2] 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.

B. CTRL *performance*

Supervising with adaptively interpolated pseudo-labels is entirely detrimental for low label frequencies. $\text{CTRL}^i_{\text{int}}$ (dashed purple 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. $\text{CTRL}^i_{\text{sparse}}$ (dashed red 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. 6a, $\text{CTRL}^i_{\text{sparse}}$ outperforms $\text{Base}^i_{\text{sparse}}$ for all sparsities $i \in \{1, 2, 3, 5, 9\}$, and is better than $\text{Base}_{100\%}$ and $\text{Base}^i_{\text{ssl}}$ for $i > 1$. However, since stationary vehicles are dominant,



(a) All vehicles (b) Stationary vehicles (c) Dynamic vehicles

Fig. 6: Level 2 vehicle 3D AP vs GT label frequency for different training methods. Base_{int} and $\text{Base}_{\text{int+ssl}}$ show similar performance trends to their corresponding CTRL experiments, however their curves are excluded to reduce clutter.

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. 6c, we see that $\text{CTRL}^i_{\text{sparse}}$ has much worse performance than both $\text{Base}^i_{\text{sparse}}$ and $\text{Base}^i_{\text{ssl}}$ 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 $\text{CTRL}_{\text{int+ssl}}$. $\text{CTRL}^i_{\text{ssl}}$ (dashed brown line) overcomes such imbalance by dramatically improving the average precision for dynamic vehicles when $i < 5$. Moreover, $\text{CTRL}^i_{\text{ssl}}$ is the best choice for $i = 1$ fps. Similar to the semi-supervision behaviour of $\text{Base}^i_{\text{ssl}}$, 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. $\text{CTRL}^i_{\text{int+ssl}}$ (dashed green line) displays the highest dynamic performance for all levels of sparsity that support interpolation, while its stationary performance is very close to that of $\text{CTRL}^i_{\text{sparse}}$. 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. $\text{CTRL}^i_{\text{sparse}}$.

V. Conclusions

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. 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 [5], 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 $\text{CTRL}_{\text{sparse}}$ 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 $\text{CTRL}_{\text{int+ssl}}$. Employing adaptive interpolation for stationary track segments and pseudo-labels for dynamic tracks, $\text{CTRL}_{\text{int+ssl}}$ 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 $\text{CTRL}_{\text{int+ssl}}$ with stationary objects is slightly below that of $\text{CTRL}_{\text{sparse}}$ for sparsities $i \geq 3$ fps. We hypothesize that this is due to the severe performance bias of $\text{CTRL}_{\text{sparse}}$ 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 $\text{CTRL}_{\text{int+ssl}}$ to be superior to $\text{CTRL}_{\text{sparse}}$. For annotation frequencies greater than or equal to 9 fps, one can consider using $\text{CTRL}_{\text{sparse}}$ 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 $\text{CTRL}_{\text{int+ssl}}$ or using the pseudo-labels produced by $\text{CTRL}_{\text{int+ssl}}$ 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.

ACKNOWLEDGMENT

This work was partially funded by the Natural Sciences and Engineering Research Council (NSERC) of Canada.

REFERENCES

[1] Holger Caesar, Varun Bankiti, et al. "nuScenes: A Multimodal Dataset for Autonomous Driving". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[2] Benjamin Caine, Rebecca Roelofs, Vijay Vasudevan, Jiquan Ngiam, Yuning Chai, Zhifeng Chen, and Jonathon Shlens. *Pseudo-labeling for Scalable 3D Object Detection*. 2021. arXiv: 2103.02093.

[3] Ming-Fang Chang, John Lambert, et al. "Argoverse: 3D Tracking and Forecasting with Rich Maps". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[4] 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". In: *IEEE Robotics and Automation Letters* 7.3 (2022), pp. 6107–6114.

[5] 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)*. 2023, pp. 19820–19829.

[6] 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.

[7] Tao Ma, Xuemeng Yang, et al. "DetZero: Rethinking Offboard 3D Object Detection with Long-term Sequential Point Clouds". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 6713–6724.

[8] Ziqi Pang, Zhichao Li, and Naiyan Wang. "Model-free Vehicle Tracking and State Estimation in Point Cloud Sequences". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 8075–8082.

[9] 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: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 9552–9557.

[10] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. "Canadian Adverse Driving Conditions dataset". In: *The International Journal of Robotics Research* 40.4-5 (2021), pp. 681–690.

[11] 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)*. 2021, pp. 6134–6144.

[12] 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*. 2020, pp. 4144–4152.

[13] Pei Sun, Henrik Kretzschmar, et al. "Scalability in Perception for Autonomous Driving: Waymo Open Dataset". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[14] OpenPCDet Development Team. *OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds*. https://github.com/open-mmlab/OpenPCDet. 2020.

[15] 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)*. IEEE. 2020, pp. 134–141.

[16] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. *Immortal Tracker: Tracklet Never Dies*. 2021. arXiv: 2111.13672.

[17] Benjamin Wilson, William Qi, et al. "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.

[18] Pengchuan Xiao, Zhenlei Shao, et al. "PandaSet: Advanced Sensor Suite Dataset for Autonomous Driving". In: *IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE. 2021, pp. 3095–3101.

[19] Anqi Joyce Yang, Sergio Casas, et al. "LabelFormer: Object Trajectory Refinement for Offboard Perception from LiDAR Point Clouds". In: *Proceedings of The 7th Conference on Robot Learning*. Vol. 229. Proceedings of Machine Learning Research. PMLR, 2023, pp. 3364–3383.

[20] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. *Auto4D: Learning to Label 4D Objects from Sequential Point Clouds*. 2021. arXiv: 2101.06586.

[21] 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)*. 2020, pp. 12224–12233.