Scalable Scene Flow from Point Clouds in the Real World

Philipp Jund *,[1], Chris Sweeney *,[2], Nichola Abdo [2], Zhifeng Chen [1], Jonathon Shlens [1]
[1] Google Brain, [2] Waymo
{abdon,cjsweeney}@waymo.com

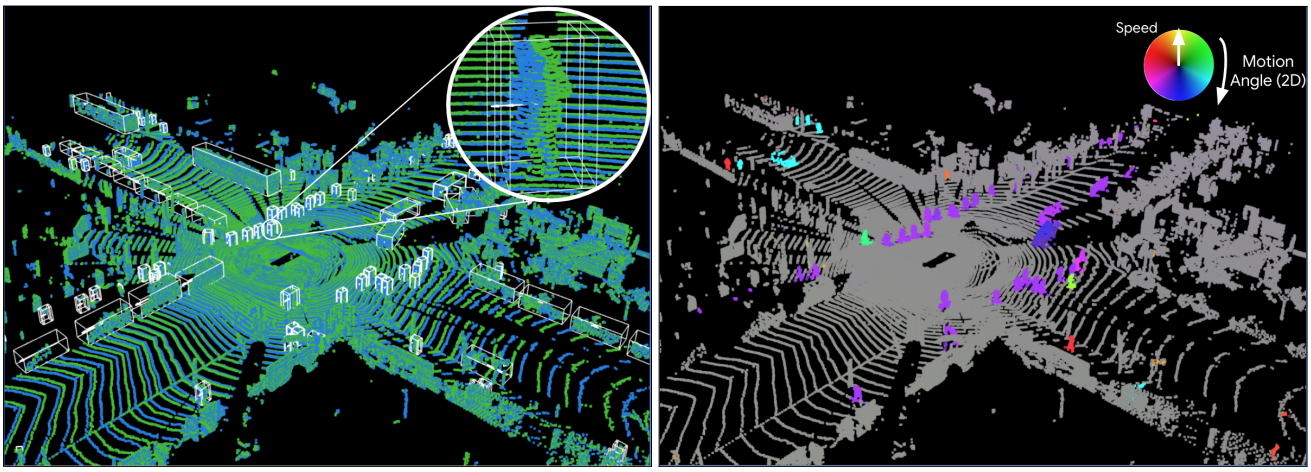

Figure 1: **Estimating scene flow in real world LiDAR point clouds from an autonomous vehicle.** Left: Overlay of two consecutive frames of point clouds (green and blue, respectively) sampled at 10 Hz from the Waymo Open Dataset [48]. White boxes indicate tracked 3D bounding boxes for human annotated vehicles and pedestrians. Right: Predicted scene flow for each point colored by direction, and brightened by the magnitude of motion based on overlaid frames†. Note that pedestrians and vehicles have distinctly different colors based on the direction of movement.

Abstract

Autonomous vehicles operate in highly dynamic environments necessitating an accurate assessment of which aspects of a scene are moving and where they are moving to. A popular approach to 3D motion estimation – termed scene flow – is to employ 3D point cloud data from consecutive LiDAR scans, although such approaches have been limited by the small size of real-world, annotated LiDAR data. In this work, we introduce a new large scale benchmark for scene flow based on the Waymo Open Dataset. The dataset is ∼1,000× larger than previous real-world datasets in terms of the number of annotated frames and is derived from the corresponding tracked 3D objects. We demonstrate how previous works were bounded based on the amount of real LiDAR data available, suggesting that larger datasets are required to achieve state-of-the-art predictive performance. Furthermore, we show how previous heuristics for operating on point clouds such as artificial down-sampling heavily degrade performance, motivating a new class of models that are tractable on the full point cloud. To address this issue, we introduce the model architecture FastFlow3D that provides real time inference on the full point cloud. Finally, we demonstrate that this problem is amenable to techniques from semi-supervised learning by highlighting open problems for generalizing methods for predicting motion on unlabeled objects. We hope that this dataset may provide new opportunities for developing real world scene flow systems and motivate a new class of machine learning problems.

1. Introduction

Motion is a prominent cue that enables humans to navigate complex environments [17]. Likewise, understanding and predicting the 3D motion field of a scene – termed the *scene flow* – provides an important signal to enable autonomous vehicles (AVs) to understand and navigate highly dynamic environments [49]. Accurate scene flow prediction enables an AV to identify potential obstacles, estimate the trajectories of objects [6, 7], and aid downstream tasks such as detection, segmentation and tracking [30, 31].

Recently, approaches that learn models to estimate scene

* Denotes equal contributions.
† Please note that we predict 3D flow, but color the direction of flow with respect to the x-y plane for the visualization.

flow from LiDAR have demonstrated the potential for LiDAR-based motion estimation, outperforming camera-based methods [28, 56, 21]. Such models take two consecutive point clouds as input and estimate the scene flow as a set of 3D vectors, which transform the points from the first point cloud to best match the second point cloud. One of the most prominent benefits of this approach is that it avoids the additional burden of estimating the depth of sensor readings as is required in camera-based approaches [31]. Unfortunately, for LiDAR based data, ground truth motion vectors are ill-defined and not tenable because no correspondence exists between LiDAR returns from subsequent time points. Instead, one must rely on semi-supervised learning methods that employ auxiliary information to make strong inferences about the motion signal in order to bootstrap annotation labels [35, 19]. Such an approach suffers from the fact that motion annotations are extremely limited (e.g. 400 frames in [35, 19]) and often rely on pretraining a model based on synthetic data [32] which exhibit distinct noise and sensor properties from real data [1]. Furthermore, even if one trains with such limited data, the resulting models are not able to tractably scale beyond $\sim$10K points [28, 56, 21, 54, 29] making the usage of such models impractical in a real world AV scenes which often contains 100K - 1000K points.

In this work, we address these shortcomings of this field by deriving a new large-scale benchmark for scene flow from the Waymo Open Dataset [48]. We derive per-point labels for motion estimation by bootstrapping from tracked objects densely annotated in each scene. The resulting scene flow dataset contains 230K frames of motion estimation annotations. This amounts to roughly $\sim$1,000$\times$ larger training set than the largest, commonly used real world dataset (200 frames) for scene flow [35, 19]. By working with a large scale dataset for scene flow, we identify several indications that the problem is quite distinct from previous pursuits in this area.

- Learned models for scene flow are heavily bounded by the amount of data. Even operating with the complete dataset, we find indications that even more data may be necessary to achieve a saturating regime.
- Heuristics for operating on point clouds – such as artificially downsampling – heavily degrades predictive performance. This observation necessitates the development of a new class of models which are tractable on a full point cloud scene, and may operate in real time on an AV.
- Previous evaluation metrics averaged and ignored notable systematic biases across classes of objects that have strong practical implications (e.g. predicting pedestrian versus vehicle speed).

We discuss each of these points in turn as we investigate working with this new dataset. We employ our investigation of the dataset statistics to motivate new evaluation criteria. Furthermore, recognizing the limitations of previous works, we develop a new baseline model architecture, named *FastFlow3D*, that is tractable on the complete point cloud with the ability to run in real time (i.e. $<$ 100 ms) on an AV. Figure 1 shows scene flow predictions from FastFlow3D, trained on our scene flow dataset. Finally, we identify and characterize an under-appreciated problem in semi-supervised learning research literature based on the ability to predict the motion of unlabeled objects. We suspect that the degree to which the fields of semi-supervised learning attack this problem may have strong implications for the real-world application of scene flow in AVs. We hope that the resulting dataset presented in this paper may open the opportunity for qualitatively different forms of learned models for scene flow.

2. Related Work

2.1. Benchmarks for scene flow estimation

Early datasets focused on the related problems of learning depth from a single image [43] and computing depth from stereo pairs of images [45, 40]. Previous datasets for estimating optical flow based on image sequences were small and largely based on synthetic imagery [1, 25, 36, 26]. Subsequent datasets focused on 2D motion estimation in movies or sequences of images [4]. The KITTI Scene Flow dataset represented a huge step forward providing the first dataset with non-synthetic imagery and accurate ground truth estimates for LiDAR point clouds [35]. Unfortunately, this dataset provided only 200 scenes for training and involved preprocessing steps that alter real-world characteristics. The FlyingThings3D dataset comprised a more modern synthetic dataset that provided a large-scale dataset comprising $\sim$20K frames of high resolution data from which scene flow may be bootstrapped [32] (see Appendix B in [28]). The internal dataset by [53] is constructed similarly to ours, but is not publicly available and does not offer a detailed description.

2.2. Datasets for tracking in AV's

Recently, there have been several works introducing large-scale datasets for autonomous vehicle applications, offering trade-offs in terms of annotated object categories, point cloud density, annotation frequency, and area covered [19, 8, 5, 23, 48]. While these datasets do not directly provide scene flow labels, they provide vehicle localization data, as well as raw LiDAR data and bounding box annotations for perceived tracklets. These recent datasets offer an

[1] Although techniques addressing domain adaptation may mitigate such challenges [3, 44, 52, 22], such approaches achieve sub-optimal performance especially when compared to data from the target domain of interest.

opportunity to propose and leverage a methodology to construct point-wise flow annotations from such datasets (Section 3.2).

We extend the Waymo Open Dataset to construct a large-scale scene flow benchmark for dense point clouds [48]. We select the Waymo Open Dataset because the bounding box annotations are at a higher acquisition frame (10 Hz) than competing datasets (e.g. 2 Hz in [5]) and contain $\sim 5\times$ the number of returns per LiDAR frame (Table 1, [48]). In addition, the Waymo Open Dataset also provides $\sim 10\times$ more scenes and annotated LiDAR frames than Argoverse [8]. Recently, [23] released a large-scale dataset comprising over 1,000 hours of driving data along with rich semantic map information. Although this dataset exceeds the size of the Waymo Open Dataset in the number of labeled scenes, we found it not suitable for our methodology for bootstrapping scene flow annotations, because the tracked objects provided are based on the results of the onboard perception system and not human-annotated bounding boxes of tracked objects.

2.3. Models for learning scene flow

There has been a rich literature of building learned models for scene flow using an assortment of end-to-end learning architectures [2, 15, 56, 28, 29, 54, 55] as well as hybrid architectures [12, 51, 50]. We discuss these baseline models in more depth in Section 5 in conjunction with a discussions about building a scalable baseline model that operates in real time.

Many of these approaches train a model initially on a synthetic dataset like FlyingThings3D [32] and evaluate and/or fine-tune on KITTI Scene Flow [19, 35]. Typically, these models are limited in their ability to leverage synthetic data in training. This observation is in line with what has been reported in the robotics literature and highlights the challenges of generalization from the simulated to the real world [3, 44, 52, 22].

3. Constructing a Scene Flow Dataset

In this section, we present our approach for generating scene flow annotations bootstrapped from existing labeled datasets. We first formalize the scene flow problem definition and relevant notation. We then detail our method for computing per-point flow vectors by leveraging the motion of 3D object label boxes in the scene. Finally, we mention the practical considerations and caveats of this approach. We reserve Appendix D to provide details about the specifics for accessing this dataset.

3.1. Problem definition and notation

We consider the problem of estimating 3D scene flow in settings where the scene at time t_i is represented as a point cloud $\mathbf{P}_i$ as measured by a LiDAR sensor mounted on the AV. Specifically, we define scene flow as the collection of 3D motion vectors $\mathbf{f} := (v^x, v^y, v^z)^\top$ for each point in the scene. Here, v^x, v^y, and v^z are the velocity components in the x, y, and z directions in m/s respectively, represented in the reference frame of the AV.

Following the scene flow literature, we aim to predict flow given two consecutive point clouds of the scene, $\mathbf{P}_{-1}$ and $\mathbf{P}_0$. As such, the scene flow encodes the motion between the previous and current time steps, $t_{\text{-}1}$ and t_0, respectively. One challenge inherent to real-world point clouds is the lack of correspondence between the observed points in $\mathbf{P}_{-1}$ and $\mathbf{P}_0$. We choose to make flow predictions for the points at the current time step, $\mathbf{P}_0$. As opposed to doing so for $\mathbf{P}_{-1}$, we believe that explicitly assigning flow predictions to the points in the most recent frame is advantageous to an AV that needs to reason about and react to the environment in real time. Additionally, the motion between $\mathbf{P}_{-1}$ and $\mathbf{P}_0$ is a reasonable approximation for the flow at t_0 when considering a high LiDAR acquisition frame rate and assuming a constant velocity between two consecutive frames.

3.2. From tracked boxes to flow annotations

A major goal of this work is to provide a large-scale dataset for estimating scene flow from real-world point clouds. Providing an AV with the capability to infer scene flow is important for reasoning about the future position of all objects in the scene and safely navigating the environment [49, 18]. However, obtaining ground truth scene flow from standard real-world time-of-flight LiDAR data is a challenging task. One challenge is the lack of point-wise correspondences between subsequent LiDAR frames. Additionally, changes in viewpoint and partial occlusions add a source of ambiguity for any point-level manual annotation. Therefore, we focus on a scalable automated approach bootstrapped from existing labeled, tracked objects in LiDAR data sequences. In these annotations, objects are represented by 3D label bounding boxes and unique IDs.

The idea of our scene flow annotation procedure is straightforward. By assuming that labeled objects are rigid, we can leverage the 3D label boxes to circumvent the point-wise correspondence problem between $\mathbf{P}_0$ and $\mathbf{P}_{-1}$ and estimate the position of the points belonging to an object in $\mathbf{P}_0$ as they *would have been observed* at $t_{\text{-}1}$. We can then compute the flow vector for each point at t_0 using its displacement over the duration of $\Delta_t = t_0 - t_{\text{-}1}$. Let point clouds $\mathbf{P}_{-1}$ and $\mathbf{P}_0$ be represented in the reference frame of the AV at their corresponding time steps. We identify the set of objects $\mathcal{O}_0$ at t_0 based on the annotated 3D boxes of the corresponding scene. We express the pose of an object o in the AV frame as a homogeneous transformation matrix $\mathbf{T}$ consisting of 3D translation and rotational components, which we construct from the pose of its corresponding la-

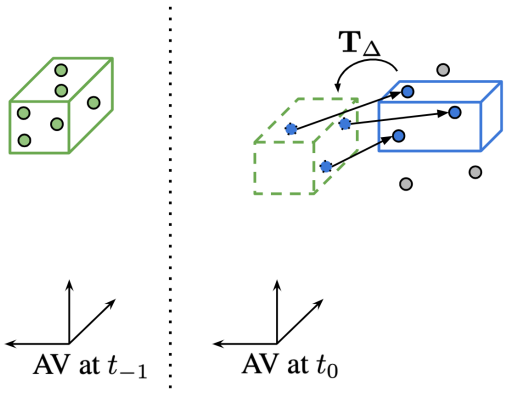

Figure 2: **Overview of our method for computing scene flow annotations from labeled objects in point cloud data.** The solid green (left) and blue (right) boxes depict the label boxes of the same object observed at time $t_{\text{-}1}$ and t_0, respectively. An object may change location because either the object moved or the AV moved. To remove the confound of the latter, we first compensate for the ego motion of the AV and compute the pose of the object at $t_{\text{-}1}$ in the reference frame of the AV at t_0 (dashed green box). Next, we compute a transform $\mathbf{T}_\Delta$ expressing the motion of the object between $t_{\text{-}1}$ and t_0, and use this to transform all points belonging to the object at t_0 to their corresponding positions at $t_{\text{-}1}$ (dotted blue points). Next, we compute per-point flow vectors based on each point's displacement between $t_{\text{-}1}$ and t_0, allowing us to model varying intra-object flow vectors. Finally, all points outside labeled boxes are annotated as stationary as depicted by the gray points.

bel box. Furthermore, we assume knowledge of the correspondences between objects at $t_{\text{-}1}$ and t_0, which is typically provided in annotated LiDAR datasets of tracked objects. Figure 2 provides a visual outline of the method we now describe in detail.

Although object poses are often expressed in the reference frame of the AV at each time step, we found that compensating for ego motion leads to superior performance with respect to a model's ability to predict scene flow. Additionally, compensating for ego motion allows us to reason about the motion of an object with respect to the ground, improving the interpretablity of evaluation metrics independent of the motion of the AV (Section 4). Therefore, for each object $o \in \mathcal{O}_0$, we first use its pose $\mathbf{T}_{-1}$ relative to the AV at $t_{\text{-}1}$ and compensate for ego motion to compute its pose $\mathbf{T}^*_{-1}$ at $t_{\text{-}1}$ but with respect to the AV frame at t_0. This is straightforward given knowledge of the poses of the AV at the corresponding time steps in the dataset, e.g. from a localization system. Accordingly, we compute the rigid body transform $\mathbf{T}_\Delta$ used to transform points belonging to object o at time t_0 to their corresponding position at $t_{\text{-}1}$, i.e. $\mathbf{T}_\Delta \coloneqq \mathbf{T}^*_{-1} \cdot \mathbf{T}_0^{-1}$.

Given the label boxes, we are able to identify the set of observed points at t_0 belonging to object o (i.e. contained in its label box). For each point $\mathbf{p}_0 = (p_0^x, p_0^y, p_0^z, 1)^\top$ expressed in homogeneous coordinates, we compute the corresponding point at $t_{\text{-}1}$ as $\mathbf{p}_{-1} = (p_{-1}^x, p_{-1}^y, p_{-1}^z, 1)^\top = \mathbf{T}_\Delta \cdot \mathbf{p}_0$. We compute the flow vector annotation $\mathbf{f}$ for each such point $\mathbf{p}_{t_0}$ as

$$\mathbf{f} = \frac{1}{\Delta_t} \left(p_0^x - p_{-1}^x,\ p_0^y - p_{-1}^y,\ p_0^z - p_{-1}^z\right)^\top$$

where we assume that a first order linear approximation between successive frames provides a reasonable approximation of the underlying speed. Instead of coarsely assigning all points to an object to the same motion vector, this approach allows us to compute different per-point flow values for each object point, e.g. capturing the fact that points on a turning vehicle have different flow directions and magnitudes. We use this approach to compute the flow vectors for all points in $\mathbf{P}_0$ belonging to labeled objects $\mathcal{O}_0$.

Many recently released datasets provide 3D bounding boxes and tracklets for a variety of object types, allowing our method to be applied to any such dataset [5, 23, 48, 57]. In this work, we apply this methodology on the Waymo Open Dataset [48] as discussed in Section 6.1. In addition to the scale of this dataset, it offers rich LiDAR scenes where objects have been manually and accurately annotated with 3D boxes at 10Hz. Combined with accurate AV pose information, this allows us to accurately compensate for ego motion when computing per-point flow vectors. Finally, our scene flow annotation approach is general in its ability to estimate 3D flow vectors based on $SE(3)$ label box poses. However, note that the Waymo Open Dataset assumes that label boxes can only rotate around the z-axis, which is sufficient to capture most relevant moving objects that change 3D position and heading orientation over time.

3.3. Practical considerations

Section 3.2 describes the general algorithm for computing per-point flow for objects labeled in two consecutive frames. Here we discuss assumptions and practical issues for generating flow annotations for all points in the scene.

Rigid body assumption. Our approach for scene flow annotation assumes the 3D label boxes correspond to rigid bodies, allowing us to compute the point-wise correspondences between two frames. Although this is a common assumption in the literature (especially for labeled vehicles [35]), this does not necessarily apply to non-rigid objects such as pedestrians. However, we found this to be a reasonable approximation in our work on the Waymo Open Dataset for two reasons. First, we derive our annotations from frames measured at high frequency (i.e. 10 Hz) such that object deformations

are minimal between adjacent frames. Second, the number of observed points on objects like pedestrians is typically small making any deviations from a rigid assumption to be of statistically minimal consequence.

Objects with no matching previous frame labels. In some cases, an object $o \in \mathcal{O}_{t_0}$ with a label box at t_0 will not have a corresponding label at $t_{\text{-}1}$, e.g. the object is first observable at t_0. Without information about the motion of the object between $t_{\text{-}1}$ and t_0, we choose to annotate its points as having invalid flow. While we can still use them to encode the scene and extract features during model training, this annotation allows us to exclude them from model weight updates and scene flow evaluation metrics.

Background points. Since typically most of the world is stationary (e.g. buildings, ground, vegetation), it is important to reflect this in the dataset. Having compensated for ego motion, we assign zero motion for all unlabeled points in the scene, and additionally annotate them as belonging to the "background" class (Appendix D). Although this holds for the vast majority of unlabeled points, there will always exist rare moving objects in the scene that were not manually annotated with label boxes (e.g. animals). In the absence of label boxes, points of such objects will receive a stationary annotation by default. Nonetheless, we recognize the importance of enabling a model to predict motion on unlabeled objects, as it is crucial for an AV to safely react to rare, moving objects. In Section 6.3, we highlight this challenge and discuss opportunities for employing this dataset as a benchmark for semi-supervised and self-supervised learning.

Coordinate frame of reference. As opposed to most other works [19, 35], we account for ego motion in our scene flow annotations. Not only does this better reflect the fact that most of the world is stationary, but it also improves the interpretability of flow annotations, predictions, and evaluation metrics. In addition to compensating for ego motion when computing flow annotations at t_0, we also transform $\mathbf{P}_{-1}$, the scene at $t_{\text{-}1}$, to the reference frame of the AV at t_0 when learning and inferring scene flow. We argue that this is more realistic for AV applications in which ego motion is available from IMU/GPS sensors [49]. Furthermore, having a consistent coordinate frame for both input frames lessens the burden on a model to correspond moving objects between frames [16] as explored in Appendix B.

4. Evaluation Metrics for Scene Flow

Two common metrics used for 3D scene flow are mean L_2 error of pointwise flow and the percentage of predictions with L_2 error below a given threshold [28, 53]. In this work, we additionally propose modifications to improve the interpretability of the results.

Breakdown by object type. Objects within the AV scene have different speed distributions dictated by the object class. This becomes especially apparent after accounting for ego motion. For instance, pedestrians walk far more slowly than vehicles drive (Section 6.1). Hence, reporting a single error ignores these systematic differences and associated systematic errors. In practice, we find it far more meaningful to report all prediction performances broken down by the known label of an object.

Binary classification formulation. One important practical application of predicting scene flow is enabling an AV to distinguish between *moving* and *stationary* parts of the scene. In that spirit, we formulate a second set of metrics that represent a "lower bar" than the L_2 error metric which captures an extremely useful rudimentary signal. We employ this metric exclusively for the more difficult task of semi-supervised learning where learning is even more challenging (Section 6.3). In particular, we assign a binary label to each point cloud as either *moving* or *stationary* based on a threshold decision $|\mathbf{f}| \geq f_{\text{min}}$. Accordingly, we compute standard precision and recall metrics for these binary labels across an entire scene. Selecting an appropriate threshold, f_{min}, is not straightforward as there is an ambiguous range between very slow and stationary objects. For simplicity, we select a conservative threshold of $f_{\text{min}} = 0.5$ m/s (1.1 mph) to assure that things labeled as moving are actually moving.

5. FastFlow3D: A Scalable Baseline Model

The average scene from the Waymo Open Dataset consists of 177K points (Table 2), even though most models [28, 56, 21, 54, 29] were designed to train with 8,192 points (16,384 points in [29]). These design choices result in favoring algorithms that scale poorly to O(100K) regimes. For instance, many methods require simple preprocessing techniques that have poor scaling properties, such as nearest neighbor lookup. Even with the application of more efficient implementations [58, 10], increasing fractions of inference time are associated with preprocessing instead of the core inference operation [2].

For this reason, we propose a new baseline model that exhibits favorable scaling properties and may operate on O(100K) in a real time system. We name this model *Fast-*

[2] A simple solution to this problem is to degrade the LiDAR sensor data by artificially donwsampling the point cloud and only perform inference on a subset of points. In Section 6.2 we demonstrate that such a strategy severely degrades predictive performance further motivating the development of architectures that can natively operate on the entire point cloud in real time.

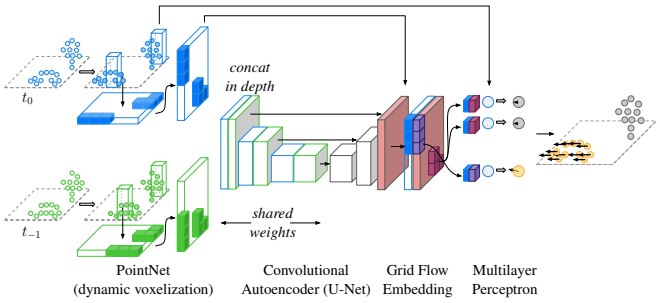

Figure 3: **Diagram of the FastFlow3D scene flow estimation model.** FastFlow3D consists of 3 stages employing established techniques: (1) a PointNet encoder [41] with a dynamic voxelization to place each feature in a spatial grid layout [59], (2) a convolutional autoencoder [42] in which the first half of the architecture [27] consists of shared weights across two frames (3) a shared MLP to regress an embedding on to a point-wise motion prediction. Note that the left-most points in the diagram are shifted horizontally with respect to one another corresponding to an inferred horizontal motion in the yellow dots shown on the right. For additional details, see Section 5 and Table 7.

Flow3D (FF3D). In particular we exploit the fact that LiDAR point clouds are dense, relatively flat along the z dimension, but cover a large area along the x and y dimensions. The proposed model is composed of three parts: a scene encoder, a decoder fusing contextual information from both frames, and a subsequent decoder to obtain point-wise flow (Figure 3). See Appendix A and Table 7 for more thorough details about the model architecture.

FastFlow3D operates on two successive point clouds where the first cloud has been transformed into the coordinate frame of the second. The target annotations are correspondingly provided in the coordinate frame of the second frame. The result of these transformation is to remove apparent motion due to the movement of the AV (Section 3.3). We train the resulting model with the average L_2 loss between the final prediction for each LiDAR returns and the corresponding ground truth flow annotation [56, 28, 21].

The encoder computes embeddings at different spatial resolutions for both point clouds. The encoder is a variant of PointPillars [27] and offers a great trade-off in terms of latency and accuracy by aggregating points within fixed vertical columns (i.e "pillars") followed by a 2D convolutional network to decrease the spatial resolution. Each pillar center is parameterized through its center coordinate (c_x, c_y, c_z). We compute the offset from the pillar center to the points in the pillar $(\Delta_x, \Delta_y, \Delta_z)$, and append the pillar center and laser features (l_0, l_1), resulting in an 8D encoding $(c_x, c_y, c_z, \Delta_x, \Delta_y, \Delta_z, l_0, l_1)$. Additionally, we employ dynamic voxelization [59], computing a linear transformation and aggregating *all* points within a pillar instead of sub-sampling points. Furthermore, we find that summing the featurized points in the pillar outperforms the max-pooling operation used in previous works [27, 59].

	32K	100K	255K	1000K
HPLFlowNet [21]	431.1	1194.5	OOM	OOM
FlowNet3D [28]	205.2	520.7	1116.4	3819.0
FastFlow3D (ours)	49.3	51.9	63.1	98.1

Table 1: **Inference latency for varying point cloud sizes.** All numbers report latency in ms on a NVIDIA Tesla P100 GPU with a batch size of 1. Note that the reported timings for HPLFlowNet [21] differ from reported results as we include the required preprocessing steps on the raw point clouds. OOM indicates out of memory. Appendix C provides details of this measurement.

One can draw an analogy of our pillar-based point featurization to more computationally expensive sampling techniques used by previous works [28, 56]. Instead of choosing representative sampled points based on expensive *farthest point sampling* and computing features relative to these points, we use a fixed grid to sample the points and compute features relative to each pillar in the grid. The pillar based representation allows our net to cover a larger area [3] with an increased density of points.

The decoder is a 2D convolutional U-Net [42]. First, we concatenate the embeddings of both encoders at each spatial resolution. Subsequently, we use a 2D convolution to obtain contextual information at the different resolutions. These context embeddings are used as the skip connections for the U-Net, which progressively merges context from consecutive resolutions. To decrease latency, we introduce bottleneck convolutions and replace deconvolution operations (i.e. transposed convolutions) with bilinear upsampling [38]. The resulting feature map of the U-Net decoder represents a grid-structured flow embedding. To obtain point-wise flow, we introduce the unpillar operation, which for each point retrieves the corresponding flow embedding grid cell, concatenates the point feature, and uses a multi layer perceptron to compute the flow vector.

As proof of concept, we showcase how the resulting architecture achieves favorable scaling behavior up to and beyond the number of laser returns in the Waymo Open Dataset (Table 1). Note that we measure performance up to 1M points in order to accommodate multi-frame perception models which operate on point clouds from multiple time frames concatenated together [13] [4].

[3] Note that due to the pillar grid representation, points outside our grid are marked as invalid points, and receive no predictions. See Appendix A for more model details.

[4] Many unpublished efforts employ multiple frames as detailed at https://waymo.com/open/challenges

	KITTI	FlyingThings3D	Ours
Data	LiDAR	Synth.	LiDAR
Label	Semi-Sup.	Truth	Super.
Scenes	22	–	1150
# LiDAR Frames	200 ‡	28K	198K
Avg Points/Frame	208K	220K †	177K

Table 2: **Comparison of popular datasets for scene flow estimation**. KITTI Scene Flow data is computed through a semi-supervised procedure [19, 35]. FlyingThings3D [32] is computed from a depth map based on a geometric procedure ([28], Appendix B). # LiDAR Frames indicates the number of annotated LiDAR frames available for training and validation (i.e. not the test split). ‡ indicates that only 400 of the entire 1.5 hour KITTI dataset was annotated for scene flow, although only 200 are available for training. † indicates the average number of points in a frame whose distance from the camera is ≤ 35.

As mentioned earlier, previously proposed baseline models rely on nearest neighbor search for pre-processing, and even with an efficient implementation [10, 58] result in poor scaling behavior (see Appendix C for details). In contrast, our baseline model exhibits nearly linear growth with a small constant. Furthermore, the typical period of a LiDAR scan is 10 Hz (i.e. 100 ms) and the latency of operating on 1M points is such that predictions may finish within the period of the scan as is required for real-time operation.

6. Results

We first present results describing the generated scene flow dataset and discuss how it compares to established baselines for scene flow in the literature (Section 6.1). In the process, we discuss dataset statistics and how this affects our selection of evaluation metrics. Next, in Section 6.2 we present the FastFlow3D baseline architecture trained on the resulting dataset. We showcase with this model the necessity of training with the full density of point cloud returns as well as the complete dataset. These results highlight deficiencies in previous approaches which employed too few data or employed sub-sampled points for real-time inference. Finally, in Section 6.3 we discuss an extension to this work in which we examine the generalization power of the model and highlight an open challenge in the application of self-supervised and semi-supervised learning techniques.

6.1. A large-scale benchmark for scene flow

The Waymo Open Dataset provides a rich and accurate source of tracked 3D objects and an exciting opportunity for deriving a large-scale scene flow dataset across a diverse and rich domain [48]. As previously discussed, scene flow ground truth does not exist in real-world point cloud datasets based on standard time-of-flight LiDAR sensors because there exist no correspondences between points from subsequent frames.

To generate a reasonable set of scene flow labels, we leveraged the human annotated tracked 3D objects from the Waymo Open Dataset [48]. Following the methodology in Section 3.2, we derived a supervised label (v^x, v^y, v^z) for each point in the scene across time. Figure 4 highlights some qualitative examples of the resulting annotation of scene flow using this methodology. In the selected frames, we highlight the diversity of the scene and difficulty of the resulting bootstrapped annotations. Namely, we observe the challenges of working with real LiDAR data including the noise inherent in the sensor reading, the prevalence of occlusions and variation in object speed. All of these qualities result in a challenging predictive task.

We release a new version of the Waymo Open Dataset with per-point flow annotations (see Appendix D for details). The dataset comprises 800 and 200 scenes, termed *run segments*, for training and validation, respectively. Each run segment is 20 seconds recorded at 10 Hz [48]. Hence, the training and validation splits contain 158,081 and 39,987 frames, respectively. The total dataset comprises 24.3B and 6.1B LiDAR returns in each split, respectively. Table 2 indicates that the resulting dataset is orders of magnitude larger than the standard KITTI scene flow dataset [19, 35] and even surpasses the large-scale 3D synthetic dataset FlyingThings3D [32], which is often used for pretraining.

Figure 5 provides a statistical summary of the scene flow constructed from the Waymo Open Dataset. Across 7,029,178 objects labeled across all frames [5], we find that $\sim$64.8% of the points within pedestrians, cyclists and vehicles are stationary. This summary statistic belies a large amount of systematic variability across object class. For instance, the majority of points within vehicles (68.0%) are parked and stationary, whereas the majority of points within pedestrians (73.7%) and cyclists (84.7%) are actively moving. This motion signature of each class of labeled object becomes even more distinct when examining the distribution of moving objects (Figure 5, bottom). Note that the average speed of moving points corresponding to pedestrians (1.3 m/s or 2.9 mph), cyclists (3.8 m/s or 8.5 mph) and vehicles (5.6 m/s or 12.5 mph) vary quite significantly. This variability of motion across object types emphasizes our selection of evaluation metrics that consider the prediction of each class separately.

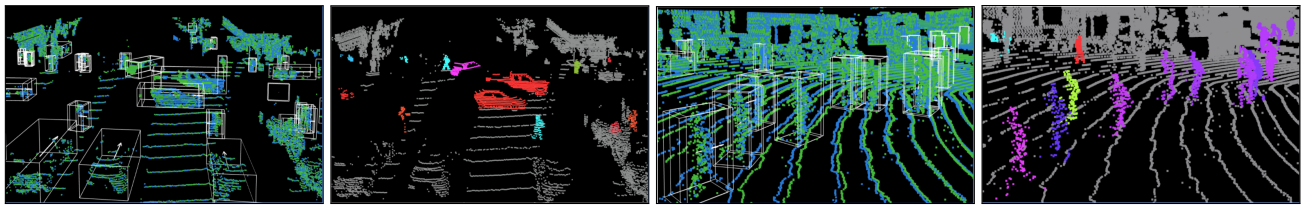

Figure 4: **Qualitative examples of bootstrapped annotations.** Two examples shown in pairs of images. Left: Overlay of two consecutive frames of point clouds (blue and green, respectively). Right: Bootstrapped annotations for each point colored by direction, and brightened by magnitude of motion based on overlaid frames. See Figure 1 for color legend. White boxes indicate original 3D bounding box annotations.

	moving		stationary	
vehicles	32.0%	(843.5M)	68.0%	(1,790.0M)
pedestrians	73.7%	(146.9M)	26.3%	(52.4M)
cyclists	84.7%	(7.0M)	15.2%	(1.6M)

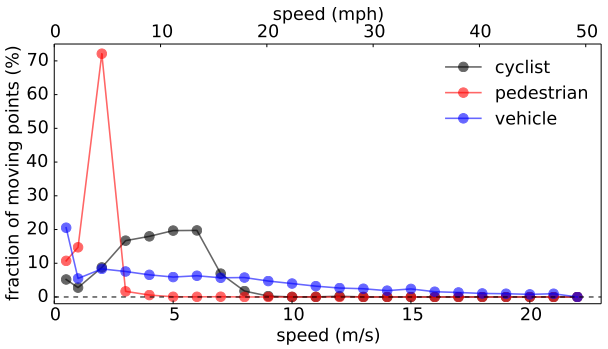

Figure 5: **Distribution of moving and stationary LiDAR points**. Statistics computed from training set split. Top: Distribution of moving and stationary points across all frames (raw counts in parenthesis). Here, we consider points with a flow magnitude below 0.1 m/s to be stationary. Bottom: Distribution of speeds for moving points.

6.2. A scalable model baseline for scene flow

We train the FastFlow3D architecture on the scene flow data generated. Briefly, the architecture consists of 3 stages employing established techniques: (1) a PointNet encoder with a dynamic voxelization [59, 41], (2) a convolutional autoencoder with skip connections [42] in which the first half of the architecture [27] consists of shared weights across two frames, and (3) a shared MLP to regress an embedding on to a point-wise motion prediction. For additional details about the training methods as well as a detailed description of the architecture, see Section 5 and Appendix A.

[5]A single instance of an object may be tracked across N frames, however we ignore the track ID annotation and instead count this single instance as N labeled objects.

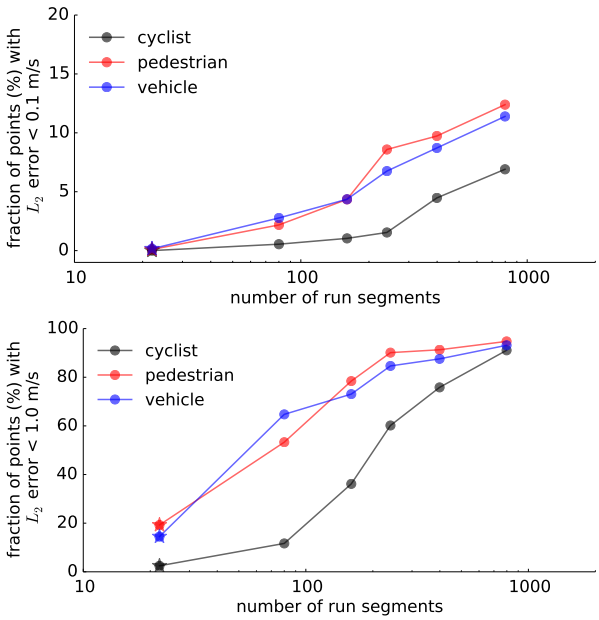

Figure 6: **Accuracy of scene flow estimation is bounded by the amount of data.** Each point corresponds to the cross validated accuracy of a scene flow model trained on increasing amounts of data. The amount of data is measured by number of independent *run segments* corresponding to roughly 200 slices of time. The y-axis reports the fraction of LiDAR returns contained within *moving* vehicles, pedestrians and cyclists whose motion vector is correctly estimated within 0.1 m/s (top) or 1.0 m/s (bottom) L_2 error. Higher numbers are better. The star indicates a model trained on the number of run segments the KITTI scene flow dataset samples from [19, 35]. Note that even the model employing all of the Waymo Open Dataset training run segments is not in a saturating regime.

The resulting model contains in total 5,233,571 parameters, a vast majority of which reside in the standard convolution architecture (4,212,736). A small number of parameters (544) are dedicated to featurizing each point cloud

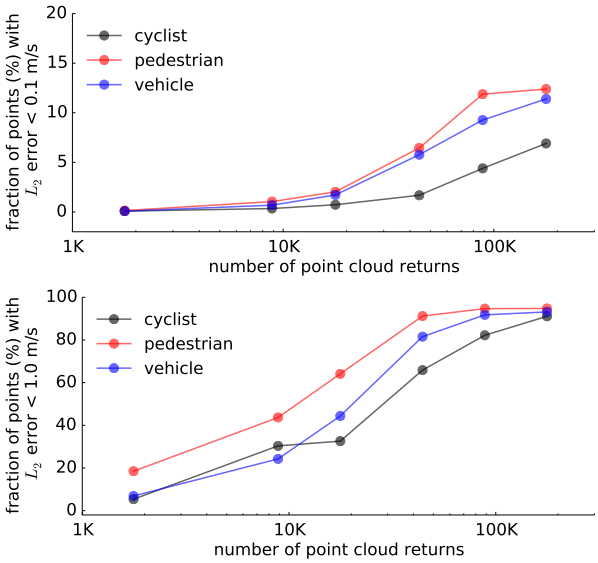

Figure 7: **Accuracy of scene flow estimation requires the full density of the point cloud scene.** Each point corresponds to the cross validated accuracy of a scene flow model trained on an increasing density of point cloud points. The y-axis reports the fraction of LiDAR returns contained within *moving* vehicles, pedestrians and cyclists whose motion vector is correctly estimated within 0.1 m/s (top) and 1.0 m/s (bottom) L_2 error.

point [41] as well as performing the final regression on to the motion flow (4,483). These latter sets of parameters are purposefully small in order to effectively constrain computational cost because they are applied across all N points in a LiDAR point cloud.

We evaluate the resulting model on the cross-validated split using the aforementioned metrics across an array of experimental studies to further justify the motivation for this dataset as well as demonstrate the difficulty of the prediction task.

We first approach the fundamental question of what the appropriate dataset size is given the prediction task. Figure 6 provides an ablation study in which we systematically subsample the number of run segments employed for training the model [6]. We observe that predictive performance improves significantly as the model is trained on increasing numbers of run segments. Interestingly, we find that cyclists trace out a curve quite distinct from pedestrians and vehicles, possibly indicative of the small number of cyclists in a scene (Figure 5). Secondly, we observe that the cross validated accuracy is far from saturating behavior when approximating the amount of data available in the standard KITTI scene flow dataset [19, 35] (Figure 6, stars). Interestingly, we observe that even with the complete dataset, our metrics do not appear to exhibit asymptotic behavior indicating that models trained on the Waymo Open Dataset may still be data bound. This result parallels detection performance reported in the original results (Table 10 in [48]).

We next investigate how scene flow prediction is affected by the density of the point cloud scene. This question is important because many baseline models purposefully operate on a smaller number of points (Table 1) and by necessity must heavily sub-sample the number of points in order to perform inference in real time. In stationary objects, we observe minimal detriment in performance (data not shown). This result is not surprising given that the vast majority of LiDAR returns arise from stationary, background objects (e.g. buildings, roads). However, we do observe that training on sparse versions of the original point cloud severely degrades predictive performance of *moving* objects (Figure 7). Notably, moving pedestrians and vehicle performance appear to be saturating indicating that if additional LiDAR returns were available, they would have minimal additional benefit in terms of predictive performance.

In addition to decreasing point density, previous works also filter out the numerous retuns from the ground in order to limit the number of points to predict [56, 28, 21]. Such a technique has a side benefit of bridging the domain gap between FlyingThings3D and KITTI Scene Flow, which differ in the inclusion of such points. We performed an ablation experiment to parallel this heuristic by training and evaluating with our annotations but with ground points removed using with a crude threshold of 0.2 m above ground. When removing ground points, we found that the mean L_2 error increased by 159% and 31% for points in moving and stationary objects, respectively. We take these results to indicate that the inclusion of ground points provide a useful signal for predicting scene flow. Taken together, these results provide post-hoc justification for building a baseline architecture which may be tractably trained on all point cloud returns instead of a model that only trains on a sample of the data points.

Finally, we report our results on the complete dataset and identify systematic differences across object class and whether or not an object is moving (Table 3). Notably, we find that *moving* vehicle points have a mean L_2 error of 0.54 m/s, corresponding to 10% of the average speed of moving vehicles (5.6 m/s). Likewise, the mean L_2 error of moving pedestrian and cyclist points are 0.32 m/s and 0.57 m/s, corresponding to 25% and 15% of the mean speed of each object class, respectively. Hence, the ability to predict vehicle speed is better than pedestrians and cyclists. We suspect that these imbalances are largely due to imbalances

[6] We note that we subsample the number of run segments and not the number of frames for this study because subsequent frames within a single run segment may be heavily correlated. Hence, the cross validated accuracy across sub-sampling frames may not be reflective of the real world performance of a model.

error metric	vehicle			pedestrian			cyclist			background
	all	moving	stationary	all	moving	stationary	all	moving	stationary	
mean (m/s)	0.18	0.54	0.05	0.25	0.32	0.10	0.51	0.57	0.10	0.07
mean (mph)	0.40	1.21	0.11	0.55	0.72	0.22	1.14	1.28	0.22	0.16
≤ 0.1 m/s	70.0%	11.6%	90.2%	33.0%	14.0%	71.4%	13.4%	4.8%	78.0%	95.7%
≤ 1.0 m/s	97.7%	92.8%	99.4%	96.7%	95.4%	99.4%	89.5%	88.2%	99.6%	96.7%

Table 3: **Performance of baseline on scene flow in large-scale dataset**. Reporting the mean pointwise L_2 error (top rows), as well as the percentage of points with error below 0.1 m/s and 1.0 m/s (bottom rows). Most errors are ≤ 1.0 m/s. All numbers are broken down by whether the points belong to vehicles, pedestrians, cyclists, or background. Additionally, we investigate the error for "stationary" and "moving" points where we coarsely consider a point to be moving if its annotated speed (flow vector magnitude) is ≥ 0.5 m/s.

	method	L_2 error (m/s)			prec	recall
		all	moving	stationary		
cyc	supervised	0.51	0.57	0.10	1.00	0.95
	stationary	1.13	1.24	0.06	1.00	0.67
	ignored	0.83	0.93	0.06	1.00	0.78
ped	supervised	0.25	0.32	0.10	1.00	0.91
	stationary	0.90	1.30	0.10	0.97	0.02
	ignored	0.88	1.25	0.10	0.99	0.07

Table 4: **Generalization of motion estimation.** Approximating generalization for moving objects by artificially excluding a class from training by either treating all its points as having zero flow (stationary) or as having no target label (ignored). We report the mean pointwise L_2 error and the precision and recall for moving point classification.

in the number of training examples for each label and the average speed of these objects. For instance, the vast majority of points are marked as *background* and hence have a target of zero motion. Because the background points are so dominant, we likewise observe the error to be smallest.

The mean L_2 error is averaged over many points, making it unclear if this statistic may be dominated by outlier events. To address this issue, we show the percentage of points in the Waymo Open Dataset evaluation set with L_2 errors below 0.1 m/s and 1.0 m/s. We observe that the vast majority of the errors are below 1.0 m/s (2.2 mph) in magnitude, indicating a rather regular distribution to the residuals. For example, this applies to 93.5% of moving vehicle points. This percentage increases to 99.8% for stationary vehicle points, which is aligned with the distribution of moving vs stationary vehicle point examples (Figure 5). In the next section, we also investigate how the prediction accuracy for classes like pedestrians and cyclists can be seen from the perspective of a discrete task distinguishing moving and stationary points.

6.3. Generalizing to unlabeled moving objects

Our supervised method for generating flow ground truth relies on every moving object having an accompanying 3D labeled box. Without a labeled box, we effectively assume the points on an object are stationary. Though this assumption holds for the vast majority of points, there are still a wide range of moving objects that our algorithm assumes to be stationary. For deployment on a safety critical system, it is important to capture motion for these objects (e.g. stroller, opening car doors, shopping carts, etc.). Even though the labeled data does not capture such objects, we find through qualitative inspection that a trained model does capture some motion in these objects (Figure 8). We next ask the degree to which a model trained on such data predicts the motion of unlabeled moving objects.

To answer this question, we construct several experiments by artificially removing labeled objects from the scene and measuring the ability of the model (in terms of the point-wise mean L_2 error) to predict motion in spite of this disadvantage. Additionally, we coarsely label points as *moving* if their annotated speed (flow vector magnitude) is ≥ 0.5 m/s (f_{min}) and query the model to quantify the precision and recall for moving classification. This latter measurement of *detecting* moving objects is particularly important for guiding planning in an AV [34, 11, 14].

Table 4 reports these results for selectively ablating the labels for pedestrian and cyclist. We ablate the labels in two methods: (1) *Stationary* treats points of ablated objects as background with no motion, (2) *Ignored* treats points of ablated objects as having no target label. We observe that *fixing* all points as background stationary results in a model with near perfect precision. However, the recall suffers enormously, particularly for pedestrians. Our results imply that unlabeled points predicted to be *moving* are almost perfectly correct (i.e. minimal false positives), however the recall is quite poor as many moving points are not identified (i.e. large number of false negatives). Furthermore, we find that treating the unlabeled points as *ignored* improves the performance slightly, indicating that even moderate information known about potential moving objects may alleviate the challenges in recall.

Notably, we observe a large discrepancy in recall be-

tween the ablation experiments for cyclists and pedestrians. We posit that this discrepancy is likely due to the much larger amount of pedestrian labels in the Waymo Open Dataset. Therefore, removing the entire class of pedestrian labels removes much more of the moving ground truth labels for moving objects.

Although a simple baseline model has some capacity to generalize to unlabeled moving object points, this capacity is clearly limited. Ignoring labeled points does mitigate the error rate for cyclists and pedestrians, however such an approach can result in other systematic errors. For instance, in earlier experiments, ignoring the stationary label for background points (i.e. no motion) results in a large increase in mean L_2 error in background points from 0.03 m/s to 0.40 m/s. Hence, such heuristics are only partial solutions to this learning problem and new ideas are warranted for approaching this dataset. We suspect that there are many opportunities for applying semi-supervised learning techniques for generalizing to unlabeled objects and leave this opportunity to future work [39, 46, 33, 9].

7. Discussion

In this work we extended the Waymo Open Dataset to provide a new benchmark for large-scale, scene flow estimation for LiDAR in autonomous vehicles. Specifically, by leveraging the supervised tracking labels, we bootstrapped a motion vector annotation for every LiDAR return. The resulting dataset is $\sim 1000\times$ larger than previous real world scene flow datasets. We also propose and discuss a series of metrics for evaluating the resulting scene flow with breakdowns based on criteria that are relevant for deploying in the real world.

Finally, we demonstrated a scalable baseline model trained on this dataset that achieves reasonable predictive performance and may be deployed for real time operation. Interestingly, training a model in such a fashion opens opportunities for self-supervised and semi-supervised training methods [39, 46, 33, 9]. We hope that this dataset may provide a useful baseline for exploring such techniques and developing generic methods for scene flow estimation in AV's in the future.

Acknowledgements

We thank Vijay Vasudevan, Benjamin Caine, Jiquan Ngiam, Brandon Yang, Pei Sun, Yuning Chai, Charles Qi, Dragomir Anguelov, Congcong Li, Jiyang Gao, James Guo, and Yin Zhou for their comments and suggestions. Additionally, we thank the larger Google Brain and Waymo Perception teams for their support.

References

[1] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011. 2

[2] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7962–7971, 2019. 3

[3] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 4243–4250. IEEE, 2018. 2, 3

[4] Daniel J Butler, Jonas Wulff, Garrett B Stanley, and Michael J Black. A naturalistic open source movie for optical flow evaluation. In *European conference on computer vision*, pages 611–625. Springer, 2012. 2

[5] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 2, 3, 4

[6] Sergio Casas, Wenjie Luo, and Raquel Urtasun. Intentnet: Learning to predict intention from raw sensor data. In *Conference on Robot Learning*, pages 947–956, 2018. 1

[7] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 1

[8] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 2, 3

[9] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D Collins, Ekin D Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Leveraging semi-supervised learning in video sequences for urban scene segmentation. In *European Conference on Computer Vision (ECCV)*, 2020. 11

[10] Yewang Chen, Lida Zhou, Yi Tang, Jai Puneet Singh, Nizar Bouguila, Cheng Wang, Huazhen Wang, and Jixiang Du. Fast neighbor search by using revised kd tree. *Information Sciences*, 472:145–162, 2019. 5, 7

[11] Keonyup Chu, Minchae Lee, and Myoungho Sunwoo. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1599–1616, 2012. 10

[12] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *2016*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 1765–1770. IEEE, 2016. 3

[13] Zhuangzhuang Ding, Yihan Hu, Runzhou Ge, Li Huang, Sijia Chen, Yu Wang, and Jie Liao. 1st place solution for waymo open dataset challenge–3d detection and domain adaptation. *arXiv preprint arXiv:2006.15505*, 2020. 6

[14] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Practical search techniques in path planning for autonomous driving. *Ann Arbor*, 1001(48105):18–80, 2008. 10

[15] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019. 3

[16] Artem Filatov, Andrey Rykov, and Viacheslav Murashkin. Any motion detector: Learning class-agnostic scene dynamics from a sequence of lidar point clouds. *arXiv preprint arXiv:2004.11647*, 2020. 5, 14

[17] David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002. 1

[18] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 3

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 2, 3, 5, 7, 8, 9

[20] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 17

[21] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019. 2, 5, 6, 9, 14

[22] Marcus Gualtieri, Andreas Ten Pas, Kate Saenko, and Robert Platt. High precision grasp pose detection in dense clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 598–605. IEEE, 2016. 2, 3

[23] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. One thousand and one hours: Self-driving motion prediction dataset. *arXiv preprint arXiv:2006.14480*, 2020. 2, 3, 4

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 14, 17

[25] Daniel Kondermann, Steffen Abraham, Gabriel Brostow, Wolfgang Förstner, Stefan Gehrig, Atsushi Imiya, Bernd Jähne, Felix Klose, Marcus Magnor, Helmut Mayer, et al. On performance analysis of optical flow algorithms. In *Outdoor and Large-Scale Real-World Scene Analysis*, pages 329–355. Springer, 2012. 2

[26] Lubor Ladickỳ, Paul Sturgess, Chris Russell, Sunando Sengupta, Yalin Bastanlar, William Clocksin, and Philip HS Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012. 2

[27] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*, 2018. 6, 8, 14

[28] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 2, 3, 5, 6, 7, 9, 14

[29] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9246–9255, 2019. 2, 3, 5

[30] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018. 1

[31] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018. 1, 2

[32] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2, 3, 7

[33] Geoffrey J McLachlan. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975. 11

[34] Matthew McNaughton, Chris Urmson, John M Dolan, and Jin-Woo Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 IEEE International Conference on Robotics and Automation*, pages 4889–4895. IEEE, 2011. 10

[35] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 2, 3, 4, 5, 7, 8, 9

[36] Sandino Morales and Reinhard Klette. Ground truth evaluation of stereo algorithms for real world applications. In *Asian Conference on Computer Vision*, pages 152–162. Springer, 2010. 2

[37] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019. 14

[38] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016. 6, 14

[39] George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *ICCV*, 2015. 11

[40] David Pfeiffer, Stefan Gehrig, and Nicolai Schneider. Exploiting the power of stereo confidences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 297–304, 2013. 2

[41] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 6, 8, 9, 14

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 6, 8

[43] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng. Learning depth from single monocular images. In *Advances in neural information processing systems*, pages 1161–1168, 2006. 2

[44] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008. 2, 3

[45] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002. 2

[46] H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 1965. 11

[47] Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019. 14

[48] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 1, 2, 3, 4, 7, 9

[49] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9):661–692, 2006. 1, 3, 5

[50] Arash K Ushani and Ryan M Eustice. Feature learning for scene flow estimation from lidar. In *Conference on Robot Learning*, pages 283–292, 2018. 3

[51] Arash K Ushani, Ryan W Wolcott, Jeffrey M Walls, and Ryan M Eustice. A learning approach for real-time temporal scene flow estimation from lidar data. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5666–5673. IEEE, 2017. 3

[52] Ulrich Viereck, Andreas ten Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. *arXiv preprint arXiv:1706.04652*, 2017. 2, 3

[53] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2, 5

[54] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020. 2, 3, 5, 14

[55] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11385–11395, 2020. 3

[56] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:1911.12408*, 2019. 2, 3, 5, 6, 9

[57] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020. 4

[58] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. *ACM Transactions on Graphics (TOG)*, 27(5):1–11, 2008. 5, 7

[59] 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 *Conference on Robot Learning (CoRL)*, 2019. 6, 8

Appendix

A. Model Architecture and Training Details

Figure 3 provides an overview of FastFlow3D. We discuss each section of the architecture in turn and provide additional parameters in Table 7.

The model architecture contains in total 5,233,571 parameters. The vast majority of the parameters (4,212,736) reside in the standard convolution architecture [27]. An additional large set of parameters (1,015,808) reside in later layers that perform upsampling with a skip connection [38]. Finally, a small number of parameters (544) are dedicated to featurizing each point cloud point [41] as well as performing the final regression on to the motion flow (4483). Note that both of these latter sets of parameters are purposefully small because they are applied to all N points in the LiDAR point cloud.

FastFlow3D uses a top-down U-Net to process the pillarized features. Consequently, the model can only predict flow for points inside the pillar grid. Points outside the x-y dimensions of the grid or outside the z dimension bounds for the pillars are marked as invalid and receive no predictions. To extend the scope of the grid, one can either make the pillar size larger or increase the size of the pillar grid. In our work, we use a 170×170 m grid (centered at the AV) represented by 512×512 pillars ($\sim 0.33 \times 0.33$ m pillars). For the z dimension, we consider the valid pillar range to be from -3 m to $+3$ m.

The model was trained for 19 epochs on the Waymo Open Dataset training set using the Adam optimizer [24]. The model was written in Lingvo [47] and forked from the open-source repository version of PointPillars 3D object detection [27, 37] [7]. The training set contains a label imbalance, vastly over-representing background stationary points. In early experiments, we explored a hyperparameter to artificially downweight background points and found that weighing down the L_2 loss by a factor of 0.1 provided good performance.

B. Compensating for Ego Motion

In Section 3.2, we argue that compensating for ego motion in the scene flow annotations improves the interpretability of flow predictions and highlights important patterns and biases in the dataset, e.g. slow vs fast objects. When training our proposed FastFlow3D model, we also compensate for ego motion by transforming both LiDAR frames to the reference frame of the AV at t_0, the time step at which we predict flow. This is convenient in practice given that ego motion information is easily available from the localization module of an AV. We hypothesize that this lessens the burden on the model, because the model does not have to implicitly learn to compensate for the motion of the AV.

We validate this hypothesis in a preliminary experiment where we compare the performance of the model reported in Section 6 to a model trained on the same dataset but without compensating for ego motion in the input point clouds. Consequently, this model has to implicitly learn how to compensate for ego motion. Table 6 shows the mean L_2 error for two such models. We observe that mean L_2 error increases substantially when ego motion is not compensated for across all object types and across moving and stationary objects. This is also consistent with previous works [16]. We also ran a similar experiment where the model consumes non ego motion compensated point clouds, but instead subtracts ego motion from the predicted flow during training and evaluation. We found slightly better performance for moving objects for this setup, but the performance is still far short of the performance achieved when compensating for ego motion directly in the input. Further research is needed to effectively learn a model that can implicitly account for ego motion.

[7] https://github.com/tensorflow/lingvo/

ID	label	description
-1	no flow	No flow information
0	unlabeled	Not contained in a bounding box.
1	vehicle	Contained within a vehicle label box.
2	pedestrian	Contained within a pedestrian label box.
3	sign	Contained within a sign label box.
4	cyclist	Contained within a cyclist label box.

Table 5: **Taxonomy of scene flow annotations**. Note that *unlabeled* points are considered *background* in this paper.

C. Measurements of Latency

In this section we provide additional details for how the latency numbers for Table 1 were calculated. All calculations were performed on a standard NVIDIA Tesla P100 GPU with a batch size of 1. The latency is averaged over 90 forward passes, excluding 10 warm up runs. Latency for the baseline models, HPLFlowNet [21] and FlowNet3D [28, 54] included any preprocessing necessary to perform inference. For HLPFlowNet and FlowNet3D, we used the implementations provided by the authors and did not alter hyperparameters. Note that this is in favor of these models, as they were tuned for point clouds covering a much smaller area compared to the Waymo Open Dataset.

D. Dataset Format for Annotations

In order to access the data, please go to http://www.waymo.com/open and click on Access Waymo Open Dataset, which requires

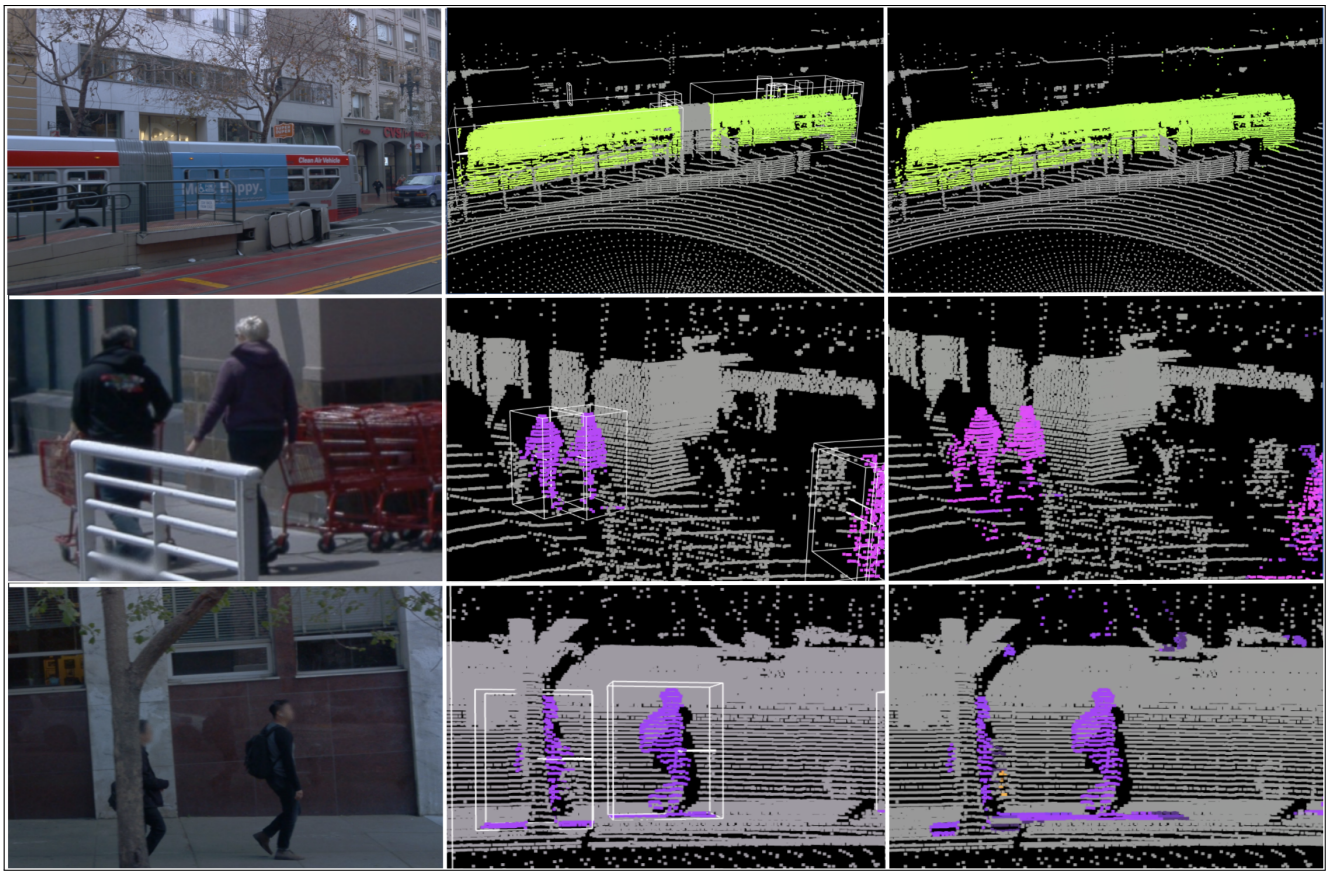

Figure 8: **Qualitative examples of generalizing to unlabeled moving objects.** Each row corresponds to a qualitative example. Left panel is the camera image; middle panel is the bootstrapped annotation; right panel is the model prediction. Color code follows Figure 1. Top row shows an example where part of a moving bus is annotated with zero flow since it is missing a 3D bounding box label. Nonetheless, the model correctly predicts the corresponding points as moving. Middle row shows an example of the model generalizing an unlabeled object (i.e. moving shopping cart). Bottom row shows failures of generalization as motion is incorrectly predicted for the ground and parts of the tree.

a user to sign in with Google and accept the Waymo Open Dataset license terms. After logged in, please visit `https://pantheon.corp.google.com/storage/browser/waymo_open_dataset_scene_flow` to download the labels.

We extend the Waymo Open Dataset to include the scene flow labels for the training and validation datasets splits. For each LiDAR, we add a new range image through the field `range_image_flow_compressed` in the message `dataset.proto:RangeImage`. The range image is a 3D tensor of shape $[H, W, 4]$ where H and W are the height and width of the LiDAR scan. For the LiDAR returns at point (i, j), we provide annotations in the range image where $[i, j, 0{:}3]$ corresponds to the estimated velocity components for the return along x, y and z axes, respectively. Finally, the value stored in the range image at $[i, j, 3]$ contains an integer class label following Table 5.

ego motion compensated for	vehicle			pedestrian			cyclist			background
	all	moving	stationary	all	moving	stationary	all	moving	stationary	
yes	0.18	0.54	0.05	0.25	0.32	0.10	0.51	0.57	0.10	0.07
no	0.36	1.16	0.08	0.49	0.63	0.17	1.21	1.34	0.14	0.07

Table 6: **Accounting for ego motion in the input significantly improves performance**. All reported values correspond to the mean L_2 error in m/s. The first column refers to whether or not we transform the point cloud at $t_{\text{-}1}$ to the reference frame of the AV at t_0. For both models, we evaluate the error in predicting scene flow annotations as described in Section 3.2, i.e. with ego motion compensated for.

Meta-Arch	Name	Input(s)	Operation	Kernel	Stride	BN?	Output Size	Depth	# Param
Architecture									
	A	N pts	MLP	–	–	Yes	N pts	64	544
	B	A	Snap-To-Grid	–	–	–	512×512	64	0
	C	B	Convolution	3×3	2	Yes	256×256	64	37120
	D	C	Convolution	3×3	1	Yes	256×256	64	37120
	E	D	Convolution	3×3	1	Yes	256×256	64	37120
	F	E	Convolution	3×3	1	Yes	256×256	64	37120
	G	F	Convolution	3×3	2	Yes	128×128	128	74240
	H	G	Convolution	3×3	1	Yes	128×128	128	147968
	I	H	Convolution	3×3	1	Yes	128×128	128	147968
	J	I	Convolution	3×3	1	Yes	128×128	128	147968
	K	J	Convolution	3×3	1	Yes	128×128	128	147968
	L	K	Convolution	3×3	1	Yes	128×128	128	147968
	M	L	Convolution	3×3	2	Yes	64×64	256	295936
	N	M	Convolution	3×3	1	Yes	64×64	256	590848
	O	N	Convolution	3×3	1	Yes	64×64	256	590848
	P	O	Convolution	3×3	1	Yes	64×64	256	590848
	Q	P	Convolution	3×3	1	Yes	64×64	256	590848
	R	Q	Convolution	3×3	1	Yes	64×64	256	590848
	S	R, L, 128	Upsample-Skip	–	–	No	128×128	128	540672
	T	S, F, 128	Upsample-Skip	–	–	No	256×256	128	311296
	U	T, B, 64	Upsample-Skip	–	–	No	512×512	64	126976
	V	U	Convolution	3×3	1	No	512×512	64	36864
	W	V	Ungrid	–	–	–	N pts	64	0
	X	W, B	Concatenate	–	–	–	N pts	128	0
	Y	X	Linear	–	–	No	N pts	32	4384
	Z	Y	Linear	–	–	No	N pts	3	99
Upsample-Skip $(\alpha, \beta, \mathsf{d})$									
	U1	α	Convolution	1×1	1	No	$h_\alpha \times w_\alpha$	d	
	U2	U1	Bilinear Interp.	–	$\frac{1}{2}$	No	$h_\beta \times w_\beta$	d	
	U3	β	Convolution	1×1	1	No	$h_\beta \times w_\beta$	d	
	U4	U2, U3	Concatenate	–	–	No	$h_\beta \times w_\beta$	d	
	U5	U4	Convolution	3×3	1	No	$h_\beta \times w_\beta$	d	
	U6	U5	Convolution	3×3	1	No	$h_\beta \times w_\beta$	d	
Padding mode		SAME							
Optimizer		Adam [24] ($\alpha = 1\text{e}{-6}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)							
Batch size		64							
Weight initialization		Xavier-Glorot [20]							
Weight decay		None							

Table 7: **FastFlow3D architecture and training details.** The network receives as input N 3D points with additional point cloud features and outputs N 3D points corresponding to the predicted 3D motion vector of each point. This output corresponds to the output of layer Z. All layers employ a ReLU nonlinearity except for layers $S - Z$ which employ no nonlinearity. The shape of tensor x is denoted $h_x \times w_x$. The Upsample-Skip layer receives two tensors α and β and a scalar depth d as input and outputs tensor $U6$.