Scalable Scene Flow From Point Clouds in the Real World

Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens

***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 dataset for scene flow estimation derived from corresponding tracked 3D objects, which is $\sim 1,000\times$ larger than previous real-world datasets in terms of the number of annotated frames. We demonstrate how previous works were bounded by 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 such as 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 FastFlow3D architecture which provides real time inference on the full point cloud. Additionally, we design human-interpretable metrics that better capture real world aspects by accounting for ego-motion and providing breakdowns per object type. We hope that this dataset may provide new opportunities for developing real world scene flow systems.**

***Index Terms*—Deep learning for visual perception, data sets for robot learning.**

I. Introduction

Motion is a prominent cue that enables humans to navigate complex environments [1]. 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 [2]. Accurate scene flow prediction enables an AV to identify potential obstacles, estimate the trajectories of objects [3], [4], and aid downstream tasks such as detection, segmentation and tracking [5], [6].

Recently, approaches that learn models to estimate scene flow from LiDAR have demonstrated the potential for LiDAR-based motion estimation, outperforming camera-based methods [7]–[9]. 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 [6]. 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 methods that employ auxiliary information to make strong inferences about the motion signal in order to bootstrap annotation labels [10], [11]. Such an approach suffers from the fact that motion annotations are extremely limited (e.g. 400 frames in [10], [11]) and often rely on pretraining a model based on synthetic data [12] exhibits distinct noise and sensor properties from real data. Furthermore, previous datasets cover a smaller area, e.g., the KITTI scene flow dataset covers 1/5th the area of our proposed dataest. This allows for different subsampling tradeoffs and inspired a class of models that are not able to tractably scale training and inference beyond ~10 K points [7]–[9], [13], [14], making the usage of such models impractical in real world AV scenes which often contain 100 K - 1000 K points.

In this work, we address these shortcomings of this field by introducing a large scale dataset for scene flow geared towards AVs. We derive per-point labels for motion estimation by bootstrapping from tracked objects densely annotated in a scene from a recently released large scale AV dataset [15]. The resulting scene flow dataset contains 198 K 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 [10], [11]. By working with a large scale dataset for scene flow, we identify several indications that the problem is quite distinct from current approaches:

- Learned models for scene flow are heavily bounded by the amount of data.
- Heuristics for operating on point clouds (e.g. downsampling) heavily degrade predictive performance. This observation motivates the development of a new class of models tractable on a full point cloud scene.
- Previous evaluation metrics ignore notable systematic biases across classes of objects (e.g. predicting pedestrian versus vehicle motion).

We discuss each of these points in turn as we investigate working with this new dataset. Recognizing the limitations of

Manuscript received September 8, 2021; accepted December 6, 2021. Date of publication December 31, 2021; date of current version January 14, 2022. This work was supported in part by Google Brain and in part by Waymo LLC. This letter was recommended for publication by Associate Editor R. Liu and Editor D. Popa upon evaluation of the reviewers' comments. *(Philipp Jund and Chris Sweeney contributed equally to this work.) (Corresponding author: Chris Sweeney.)*

Philipp Jund, Zhifeng Chen, and Jonathon Shlens are with the Google Brain, Mountain View, CA 94043 USA (e-mail: pjund@google.com; zhifengc@google.com; jon.from.california@gmail.com).

Chris Sweeney and Nichola Abdo are with the Waymo LLC, Mountain View, CA 94043 USA (e-mail: cjsweeney@waymo.com; abdon@informatik.uni-freiburg.de).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3139542, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3139542

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

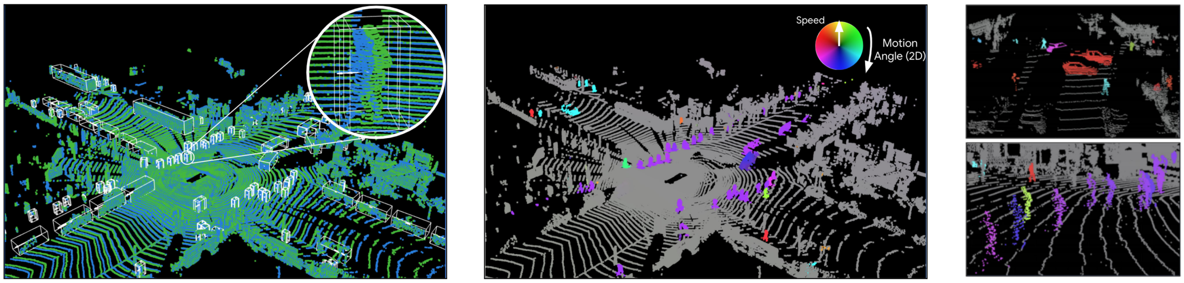

Fig. 1. LiDAR scene flow estimation for autonomous vehicles. Left: Overlay of two consecutive point clouds (green and blue, respectively) sampled at 10 Hz from the Waymo Open Dataset [15]. White boxes are tracked 3D bounding boxes for human annotated vehicles and pedestrians. Middle: Predicted scene flow for each point colored by direction, and brightened by speed based on overlaid frames.[1] Right: Two qualitative examples of bootstrapped annotations.

previous works, we develop a new baseline model architecture, *FastFlow3D*, that is tractable on the complete point cloud with the ability to run in real time (i.e. < 100 ms) on an AV. Fig. 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 based on the ability to predict the motion of unlabeled objects. We suspect the degree to which the fields of semi-supervised learning address 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 new forms of learned scene flow models.

II. Related Work

A. Benchmarks for Scene Flow Estimation

Early datasets focused on the related problems of inferring depth from a single image [16] or stereo pairs of images [17], [18]. Previous datasets for estimating optical flow were small and largely based on synthetic imagery [19]–[22]. Subsequent datasets focused on 2D motion estimation in movies or sequences of images [23]. The KITTI Scene Flow dataset represented a huge step forward, providing non-synthetic imagery paired with accurate ground truth estimates. However, it contained only 200 scenes for training and involved preprocessing steps that alter real-world characteristics [10]. FlyingThings3D offered a modern large-scale synthetic dataset comprising $\sim$20 K frames of high resolution data from which scene flow may be bootstrapped [12]. Internal datasets by [24], [25] are constructed similarly to ours, but are not publicly available and do not offer a detailed description. Recently, [26] created two scene flow datasets in a similar fashion, subsampling 2,691 and 1,513 training scenes from the Argoverse [27] and nuScenes [28] datasets, resepectively. However, even without subsampling, larger datasets in terms of scenes and number of points are needed to train more accurate scene flow models as shown in [26] and Fig. 4.

B. Datasets for Tracking in AVs

Recently, there have been several works introducing large-scale datasets for autonomous vehicle applications [11], [15], [27]–[29]. 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 opportunity to propose a methodology to construct point-wise flow annotations from such data (Section III-B).

We extend the Waymo Open Dataset (WOD) to construct a large-scale scene flow benchmark for dense point clouds [15]. 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 [28]) and contain $\sim 5\times$ the number of returns per LiDAR frame (Table I, [15]). In addition, the Waymo Open Dataset also provides $\sim 10\times$ more scenes and annotated LiDAR frames than Argoverse [27]. Recently, [29] released a large-scale dataset with 1,000+ hours of driving data. However, their tracked object annotations are not human-annotated but based on the results of the onboard perception system.

TABLE I
Inference Latency Varying Point Cloud Sizes

	32K	100K	255K	1000K
HPLFlowNet [9]	431.1	1194.5	OOM	OOM
FlowNet3D [7]	205.2	520.7	1116.4	3819.0
FastFlow3D (ours)	49.3	51.9	63.1	98.1

All numbers report latency in ms on a NVIDIA tesla P100 with batch size $= 1$. The timings for HPLFlowNet [9] differ from reported results as we include the required preprocessing on the raw point clouds. OOM indicates out of memory.

C. Models for Learning Scene Flow

There is a rich literature of building learned models for scene flow using end-to-end learned architectures [7], [8], [13], [14], [25], [30]–[32] as well as hybrid architectures [33]–[35]. We discuss these in Section V in conjunction with building a scalable baseline model that operates in real time. Recently, Lee *et al.* presented an approach for predicting *pillar-level* flow [25]. Whereas our model leverages a similar pillar-based architecture, we tackle the full scope of the scene flow problem and predict *point-level* flow while being tractable enough for real-time applications.

Moreover, many previous works train models on synthetic datasets like FlyingThings3D [12] and evaluate and/or fine-tune on KITTI Scene Flow [10], [11]. Typically, these models are limited in their ability to leverage synthetic data in training. This

[1]Please note that we predict 3D flow, but color the direction of flow with respect to the x-y plane for the visualization.

observation is in line with the robotics literature and highlights the challenges of generalization from simulation to the real world [36]–[39].

III. Constructing a Scene Flow Dataset

In this section, we present an approach for generating scene flow annotations bootstrapped from existing labeled datasets. We first formalize the scene flow problem definition. We then detail our method for computing per-point flow vectors by leveraging the motion of 3D object label boxes. We emphasize that many details abound in the assumptions behind such annotations, how to calculate various transformations in the track labels, as well as how to handle important edge cases.

A. Problem Definition

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 where v^d is the velocity in the d directions in m/s.

Following the scene flow literature, we predict flow given two consecutive point clouds of the scene, $\mathbf{P}_{-1}$ and $\mathbf{P}_0$. The scene flow encodes the motion between the previous and current time steps, $t_{\text{-}1}$ and t_0, respectively. We predict the scene flow at the current time step, $\mathbf{P}_0$ in order to make the predictions practical for real time operation.

B. From Tracked Boxes to Flow Annotations

Obtaining ground truth scene flow from standard real-world LiDAR data is a challenging task. One challenge is the lack of point-wise correspondences between subsequent LiDAR frames. Manual annotation is too expensive and humans must contend with ambiguity due to changes in viewpoint and partial occlusions. Therefore, we focus on a scalable automated approach bootstrapped from existing labeled, tracked objects in LiDAR data sequences.

The annotation procedure is straightforward. We assume that labeled objects are rigid and calculate point velocities using a *secant line approximation*. For each point $\mathbf{p}_0$ at time t_0, we compute the flow annotation as $\mathbf{f} = \frac{1}{\Delta_t}(\mathbf{p}_0 - \mathbf{p}_{-1})$, where $\Delta_t = t_0 - t_{\text{-}1}$, $\mathbf{p}_{-1} = \mathbf{T}_\Delta\, \mathbf{p}_0$ is the corresponding point at $t_{\text{-}1}$, and $\mathbf{T}_\Delta$ is a homogeneous transformation inferred from the track labels of the object to which the point belongs. If there is no label at $t_{\text{-}1}$, the flow is annotated as invalid. This captures how a moving object may have varying per-point flow magnitudes and directions. Though our rigidity assumption does not necessarily apply to non rigid objects (e.g. pedestrians), the high frame rate (10 Hz) minimizes non-rigid deformations between adjacent frames.

In order to calculate the transformation $\mathbf{T}_\Delta$, we compensate for the ego motion of the AV because this leads to superior predictive performance since a learned model does not need to additionally infer the AV's motion (most AVs are equipped with an IMU/GPS system to provide such information). Furthermore, compensating for ego motion improves the interpretability of the evaluation metrics (Section IV) since the predictions are now independent of the AV motion. We use this approach to compute the flow vectors for all points in $\mathbf{P}_0$ belonging to labeled objects. Points outside the labeled objects are assigned a flow of 0 m/s. This stationary assumption works well in practice but has a notable gap when considering unlabeled moving objects in the scene. See Section VI-C for an in depth analysis on how our model can generalize to unlabeled moving objects.

In this work, we apply this methodology on the Waymo Open Dataset [15]. The dataset offers a large scale with diverse LiDAR scenes where objects have been manually and accurately annotated with 3D boxes at 10 Hz. Finally, the accurate AV pose information permits compensating for ego motion. We note that the method for scene flow annotation is general and may be used to estimate 3D flow vectors of the label box poses available in other datasets [15], [28], [29], [40].

IV. 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 [7], [24]. In this work, we additionally propose modifications to improve the interpretability of the results.

Breakdown by object type: Objects within the AV scene (e.g. vehicles, pedestrians) have different speed distributions dictated by the object class (Section VI-A). This becomes especially apparent after accounting for ego motion. Reporting a single error ignores these systematic differences. In practice, we find it more meaningful to report all prediction performances delineated by the object label.

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" which captures a useful rudimentary signal. We employ this metric exclusively for the more difficult task of semi-supervised learning (Section VI-C) where learning is more challenging. In particular, we assign a binary label to each reflection as either *moving* or *stationary* based on a threshold, $|\mathbf{f}| \geq f_{\text{min}}$. Accordingly, we compute precision and recall metrics for these binary labels across an entire scene. Selecting a 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.

V. FastFlow3D: A Scalable Baseline Model

The average scene from the Waymo Open Dataset consists of 177 K points (Table II), even though most models [7]–[9], [13], [14] were designed to train with 8,192 points (16,384 points in [14]). This design choice favors algorithms that scale poorly to O(100 K) regimes. For instance, many methods require preprocessing techniques such as nearest neighbor lookup. Even with efficient implementations [46], [47], increasing fractions of

TABLE II
COMPARISON OF POPULAR DATASETS FOR SCENE FLOW ESTIMATION

	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

[11] is computed through a semi-supervised procedure [10]. [12] is computed from a depth map based on a geometric procedure [7]. #LiDAR frames counts annotated LiDAR frames for training and validation. ‡ indicates that only 400 frames of the KITTI dataset were annotated for scene flow (200 available for training). †indicates the average number of points with distance from the camera ? 35.

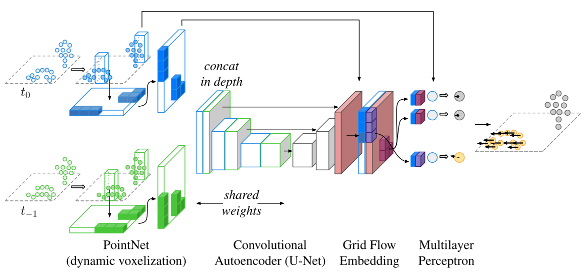

Fig. 2. Diagram of FastFlow3D model. FastFlow3D consists of 3 stages employing a PointNet encoder with dynamic voxelization [41], [42], a convolutional autoencoder [43], [44] with weights shared across two frames, and a shared MLP to regress an embedding on to a point-wise motion prediction. For details, see Section V and Appendix in [45].

inference time are dedicated to preprocessing instead of the core inference operation.

For this reason, we propose a new model that exhibits favorable scaling properties and may operate on O(100 K) in a real time system. We name this model *FastFlow3D* (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 (Fig. 2).

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 III-B). 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 [7]–[9].

The encoder computes embeddings at different spatial resolutions for both point clouds. The encoder is a variant of PointPillars [44] 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 [42], 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 [42], [44].

One can draw an analogy of our pillar-based point featurization to more computationally expensive sampling techniques used by previous works [7], [8]. 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 with an increased density of points.

The decoder is a 2D convolutional U-Net [43]. 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 [48]. 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 I). Note that we measure performance up to 1 M points in order to accommodate multi-frame perception models which operate on point clouds from multiple time frames concatenated together [49] [2]. As mentioned earlier, previously proposed baseline models rely on nearest neighbor search for pre-processing, and even with an efficient implementation [46], [47] result in poor scaling behavior (see Section VI-B for details.) making it prohibitively expensive to train and run these models on large, realistic datasets like the Waymo Open Dataset[3]. 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 1 M points is such that predictions may finish within the period of the scan as is required for real-time operation.

VI. 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 VI-A). In the process, we discuss dataset statistics and how this affects our selection of

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

[3]In Section VI-B we demonstrate that downsampling the point cloud severely degrades predictive performance, further motivating architectures that can natively operate on the entire point cloud in real time.

evaluation metrics. Next, in Section VI-B we present the FastFlow3D baseline architecture trained on the resulting dataset. With this model, we showcase 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 VI-C 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.

A. A Large-Scale Dataset for Scene Flow

The Waymo Open Dataset provides an accurate source of tracked 3D objects and an opportunity for deriving a large-scale scene flow dataset across a diverse and rich domain [15]. As previously discussed, scene flow ground truth does not exist in real-world point cloud datasets based on standard time-of-flight LiDAR because no correspondences exist 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 [15]. Following the methodology in Section III-B, we derived a supervised label (v^x, v^y, v^z) for each point in the scene across time. Fig. 1 (right) 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.

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 [15]. Hence, the training and validation splits contain 158,081 and 39,987 frames[4]. The total dataset comprises 24.3B and 6.1B LiDAR returns in each split, respectively. Table II indicates that the resulting dataset is orders of magnitude larger than the standard KITTI scene flow dataset [10], [11] and even surpasses the large-scale synthetic dataset FlyingThings3D [12] often used for pretraining.

Fig. 3 provides a summary of the scene flow constructed from the Waymo Open Dataset. Across 7,029,178 objects labeled across all frames,[5] we find that ∼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. The motion signature of each class of labeled object becomes even more distinct when examining the distribution of moving objects (Fig. 3, bottom). Note that the average speed

	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)

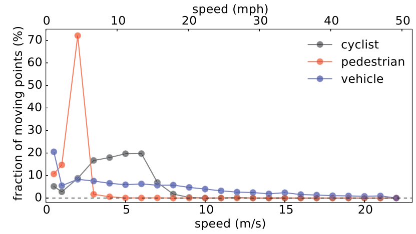

Fig. 3. Distribution of moving and stationary LiDAR points. Statistics computed from training set split. Top: Distribution of moving and stationary points across all frames (raw counts in parenthesis). We consider points with a flow magnitude below 0.1 m/s to be stationary. Bottom: Distribution of speeds for moving points.

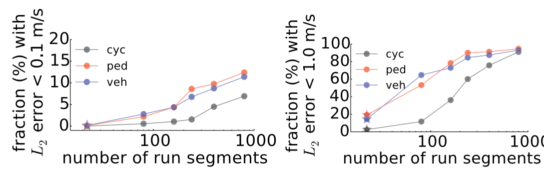

Fig. 4. Accuracy of scene flow estimation is bounded by the amount of data. Each point corresponds to the cross validated accuracy of a model trained on increasing amounts of data (see text). Y-axis reports the fraction of LiDAR returns contained within *moving* objects whose motion vector is estimated 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 in [10], [11].

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 significantly. This variability of motion across object types emphasizes our selection of evaluation metrics that consider the prediction of each class separately.

B. A Scalable Model Baseline for Scene Flow

We train the FastFlow3D architecture on the scene flow data. Briefly, the architecture consists of 3 stages employing established techniques: (1) a PointNet encoder with dynamic voxelization [41], [42], (2) a convolutional autoencoder with skip connections [43] in which the first half of the architecture [44] consists of shared weights across two frames, and (3) a shared MLP to regress an embedding on to a point-wise motion prediction.

The resulting model contains 5.23 M parameters, a vast majority of which reside in the convolution architecture (4.21 M). 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 (4,483). These latter sets of parameters are purposefully small, constraining computational cost as they are applied across all N points in a point cloud.

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

We first approach the question of what the appropriate dataset size is given the prediction task. Fig. 4 provides an ablation study in which we systematically subsample the number of run

[4]Please see the Appendix in [45] for more details on downloading and accessing this new dataset.

[5]A single instance of an object may be tracked across N frames. We count a single instance as N labeled objects.

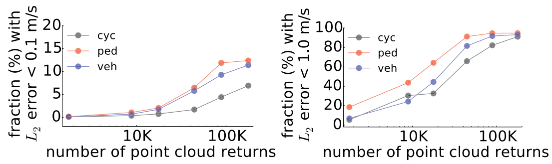

Fig. 5. Accuracy of scene flow estimation requires the full density of the point cloud scene. Each point corresponds to the cross validated accuracy of a model trained on an increasing density of point cloud points. Y-axis reports the fraction of LiDAR returns contained within *moving* vehicles, pedestrians and cyclists whose motion vector is correctly estimated within 0.1 m/s (top) and 1.0 m/s (bottom) L_2 error.

segments employed for training.[6] We observe that predictive performance improves significantly as the model is trained on increasing numbers of run segments. 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 (Fig. 3). Secondly, we observe that the cross validated accuracy is far from saturating behavior when approximating the amount of data available in the KITTI scene flow dataset [10], [11] (Fig. 4, stars). 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 [15]).

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 I) 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 (Fig. 5). 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 returns from the ground in order to limit the number of points to predict [7]–[9]. 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 removing points 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 an architecture which may be tractably trained on all point cloud returns instead of one that only trains on a sample of the returns.

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 III). Producing baseline comparisons for previous nearest neighbor based models is prohibitively expensive due to their poor scaling behavior.[7] We hope to motivate a new class of real time scene flow models that are capable of training on our dataset.

Table III indicates 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 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 dominant, we likewise observe the error to be smallest.

C. Generalizing to Unlabeled Moving Objects

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, the residuals of 92.8% and 99.8% of moving and stationary vehicle points have an error below 1.0 m/s. Next, we also investigate how the prediction accuracy for classes like pedestrians and cyclists can be cast as a discrete task distinguishing moving and stationary points.

Our supervised method for generating flow ground truth relies on every moving object having an accompanying tracked box. Without a tracked 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, etc.). Even though the labeled data does not capture such objects, we find qualitatively that a trained model does capture some motion in these objects (Fig. 6). 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 to predict motion in spite of this disadvantage. Additionally, we coarsely label points as *moving* if their annotated speed (flow vector magnitude) is $\geq$ 0.5 m/s ($f_{\min}$) and query the model to quantify the precision and recall for moving classification. This latter measurement of

[6]We subsample the number of run segments and not frames because subsequent frames within a single run segment are heavily correlated.

[7]We did try experiments involving cropping and downsampling the point clouds to make comparison to the baselines feasible. However, these modifications distorted the points too much to serve as practical input data.

TABLE III
PERFORMANCE OF BASELINE ON SCENE FLOW IN LARGE-SCALE DATASET

error metric	vehicle all	vehicle moving	vehicle stationary	pedestrian all	pedestrian moving	pedestrian stationary	cyclist all	cyclist moving	cyclist stationary	background
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
$\leq$ 0.1 m/s	70.0%	11.6%	90.2%	33.0%	14.0%	71.4%	13.4%	4.8%	78.0%	95.7%
$\leq$ 1.0 m/s	97.7%	92.8%	99.4%	96.7%	95.4%	99.4%	89.5%	88.2%	99.6%	96.7%

Mean pointwise L_2 error (top) and percentage of points with error below 0.1 m/s and 1.0 m/s (bottom). Most errors are $\leq$ 1.0 m/s. Additionally, we investigate the error for stationary and moving points where a point is coarsely considered moving if the flow vector magnitude is ≥ 0.5 m/s.

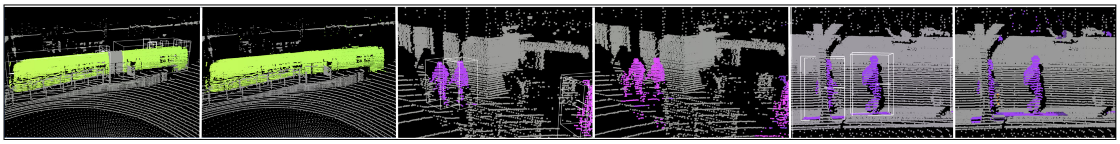

Fig. 6. Generalizing to unlabeled moving objects. Three examples each with the bootstrapped annotation (left) and model prediction (right) (Color code from Fig. 1). Left example: Despite missing flow annotation for the middle of a bus, our model can generalize well. Middle example: The model generalizes an unlabeled object (moving shopping cart). Right example: failures of generalization as motion is incorrectly predicted for the ground and parts of the tree.

TABLE IV
GENERALIZATION OF MOTION ESTIMATION

	method	L_2 error (m/s) all	moving	stationary	prec	recall
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

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.

detecting moving objects is particularly important for guiding planning in an AV [50]–[52].

Table IV reports 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 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, however the recall is quite poor as many moving points are not identified. 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 challenges in recall. Notably, we observe a large discrepancy in recall between 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. Removing the entire class of pedestrian labels removes much more of the ground truth labels for moving objects.

Although our model has some capacity to generalize to unlabeled moving object points, this capacity is limited. Ignoring labeled points does mitigate the error rate for cyclists and pedestrians but can result in other systematic errors. For instance, in earlier experiments, ignoring stationary labels 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 problem and new ideas are warranted for approaching this dataset. We suspect that many opportunities exist for applying semi-supervised learning techniques for generalizing to unlabeled objects and leave this to future work [53], [54].

VII. DISCUSSION

In this work, we presented a new large-scale scene flow dataset measured from LiDAR in AVs. Specifically, by leveraging the supervised tracking labels from the Waymo Open Dataset, we bootstrapped a motion vector annotation for each LiDAR return. The resulting dataset is $\sim 1000\times$ larger than previous real world scene flow datasets. We also propose 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 developed a scalable baseline model that achieves reasonable predictive performance and may be deployed for real time operation. Interestingly, our setup opens opportunities for self-supervised and semi-supervised methods. We hope that this dataset may provide a useful baseline for exploring such techniques and developing generic methods for scene flow estimation in AVs in the future.

ACKNOWLEDGMENT

The authors 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. We thank the Google Brain and Waymo Perception teams for their support.

REFERENCES

[1] D. A. Forsyth *et al.*, *Computer Vision: A Modern Approach*. Englewood Cliffs, NJ, USA: Prentice Hall, 2002.

[2] S. Thrun *et al.*, “Stanley: The robot that won the DARPA grand challenge,” *J. Field Robot.*, vol. 23, pp. 661–692, 2006.

[3] S. Casas *et al.*, “Intentnet: Learning to predict intention from raw sensor data,” in *Proc. Conf. Robot Learn.*, 2018, pp. 947–956.

[4] Y. Chai *et al.*, “Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction,” in *Proc. Conf. Robot Learn.*, 2019, pp. 86–99.

[5] W. Luo, B. Yang, and R. Urtasun, “Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3569–3577.

[6] R. Mahjourian, M. Wicke, and A. Angelova, “Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5667–5675.

[7] X. Liu, C. R. Qi, and L. J. Guibas, “FlowNet3D: Learning scene flow in 3D point clouds,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.

[8] W. Wu *et al.*, “PointPWC-Net: Cost volume on point clouds for (self-) supervised scene flow estimation,” *European Conf. Comput. Vision*, Springer, pp. 88–107, 2020, *arXiv:1911.12408*.

[9] X. Gu, Y. Wang, C. Wu, Y. J. Lee, and P. Wang, “HPLFlowNet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 3249–3258.

[10] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3061–3070.

[11] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[12] N. Mayer *et al.*, “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4040–4048.

[13] Z. Wang, S. Li, H. Howard-Jenkins, V. A. Prisacariu, and M. Chen, “FlowNet3D++: Geometric losses for deep scene flow estimation,” in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2020, pp. 91–98.

[14] X. Liu, M. Yan, and J. Bohg, “MeteorNet: Deep learning on dynamic 3D point cloud sequences,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 9246–9255.

[15] P. Sun *et al.*, “Scalability in perception for autonomous driving: Waymo open dataset,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2446–2454.

[16] A. Saxena *et al.*, “Learning depth from single monocular images,” in *Proc. Adv. Int. Conf. Neural Inf. Process. Syst.*, 2006, pp. 1161–1168.

[17] D. Scharstein *et al.*, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” *Int. J. Comput. Vis.*, vol. 47, pp. 7–42, 2002.

[18] D. Pfeiffer, S. Gehrig, and N. Schneider, “Exploiting the power of stereo confidences,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 297–304.

[19] S. Baker *et al.*, “A database and evaluation methodology for optical flow,” *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, 2011.

[20] D. Kondermann *et al.*, “On performance analysis of optical flow algorithms,” in *Outdoor and Large-Scale Real-World Scene Analysis*. Berlin, Germany: Springer, 2012, pp. 329–355.

[21] S. Morales *et al.*, “Ground truth evaluation of stereo algorithms for real world applications,” in *Proc. Asian Conf. Comput. Vis.*, 2010, pp. 152–162.

[22] L. Ladickỳ *et al.*, “Joint optimization for object class segmentation and dense stereo reconstruction,” *Int. J. Comput. Vis.*, vol. 100, pp. 122–133, 2012.

[23] D. J. Butler *et al.*, “A naturalistic open source movie for optical flow evaluation,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 611–625.

[24] S. Wang, S. Suo, W. -C. Ma, A. Pokrovsky, and R. Urtasun, “Deep parametric continuous convolutional neural networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2589–2597.

[25] K.-H. Lee *et al.*, “Pillarflow: End-to-end birds-eye-view flow estimation for autonomous driving,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 2007–2013.

[26] J. Pontes, J. Hays, and S. Lucey, “Scene flow from point clouds with or without learning,” *Int. Conf. 3D Vis.*, 2020, pp. 261–270.

[27] M.-F. Chang *et al.*, “Argoverse: 3D tracking and forecasting with rich maps,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8748–8757.

[28] H. Caesar *et al.*, “nuScenes: A multimodal dataset for autonomous driving,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11621–11631.

[29] J. Houston *et al.*, “One thousand and one hours: Self-driving motion prediction dataset,” 2020, [Online]. Available: https://level-5.global/level5/data/.

[30] A. Behl, D. Paschalidou, S. Donné, and A. Geiger, “PointFlowNet: Learning representations for rigid motion estimation from point clouds,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7962–7971.

[31] H. Fan *et al.*, “Pointrnn: Point recurrent neural network for moving point cloud processing,” 2019, *arXiv:1910.08287*.

[32] P. Wu, S. Chen, and D. N. Metaxas, “MotionNet: Joint perception and motion prediction for autonomous driving based on bird’s eye view maps,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020.

[33] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, “Rigid scene flow for 3D LiDAR scans,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1765–1770.

[34] A. Ushani, R. W. Wolcott, J. M. Walls, and R. M. Eustice, “A learning approach for real-time temporal scene flow estimation from LiDAR data,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 5666–5673.

[35] A. K. Ushani *et al.*, “Feature learning for scene flow estimation from LiDAR,” in *Proc. Conf. Robot Learn.*, 2018, pp. 283–292.

[36] K. Bousmalis *et al.*, “Using simulation and domain adaptation to improve efficiency of deep robotic grasping,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 4243–4250.

[37] A. Saxena *et al.*, “Robotic grasping of novel objects using vision,” *Int. J. Robot. Res.*, vol. 27, pp. 157–173, 2008.

[38] U. Viereck *et al.*, “Learning a visuomotor controller for real world robotic grasping using simulated depth images,” *Conf. Robot Learn.*, pp. 291–300, 2017.

[39] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, “High precision grasp pose detection in dense clutter,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 598–605.

[40] F. Yu *et al.*, “BDD100K: A diverse driving dataset for heterogeneous multitask learning,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2636–2645.

[41] C. R. Qi *et al.*, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.

[42] Y. Zhou *et al.*, “End-to-end multi-view fusion for 3D object detection in LiDAR point clouds,” in *Proc. Conf. Robot Learn.*, 2019, pp. 923–932.

[43] O. Ronneberger *et al.*, “U-Net: Convolutional networks for biomedical image segmentation,” in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2015, pp. 234–241.

[44] A. H. Lang *et al.*, “PointPillars: Fast encoders for object detection from point clouds,” *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 12697–12705, 2019.

[45] P. Jund *et al.*, “Scalable scene flow from point clouds in the real world,” 2021, *arXiv:2103.01306*.

[46] K. Zhou *et al.*, “Real-time KD-tree construction on graphics hardware,” *ACM Trans. Graph.*, vol. 27, no. 5, pp. 1–11, 2008.

[47] Y. Chen *et al.*, “Fast neighbor search by using revised *k*-d tree,” *Inf. Sci.*, vol. 472, pp. 145–162, 2019.

[48] A. Odena *et al.*, “Deconvolution and checkerboard artifacts,” *Distill*, vol. 1, no. 10, 2016, Art. no. e3.

[49] Z. Ding *et al.*, “1st place solution for waymo open dataset challenge-3D detection and domain adaptation,” 2020, *arXiv:2006.15505*.

[50] M. McNaughton, C. Urmson, J. M. Dolan, and J. Lee, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 4889–4895.

[51] K. Chu, M. Lee, and M. Sunwoo, “Local path planning for off-road autonomous driving with avoidance of static obstacles,” *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.

[52] D. Dolgov *et al.*, “Practical search techniques in path planning for autonomous driving,” in *Proc. AAAI Conf. Artif. Intell.*, vol. 1001, 2008, pp. 18–80.

[53] G. Papandreou, L. Chen, K. P. Murphy, and A. L. Yuille, “Weakly-and semi-supervised learning of a deep convolutional net for semantic image segmentation,” in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1742–1750.

[54] L.-C. Chen *et al.*, “Leveraging semi-supervised learning in video sequences for urban scene segmentation,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 695–714.