ZeroFlow: Fast Zero Label Scene Flow via Distillation

Kyle Vedder[1]* **Neehar Peri**[2] **Nathaniel Chodosh**[2] **Ishan Khatri**[3] **Eric Eaton**[1]
Dinesh Jayaraman[1] **Yang Liu**[4] **Deva Ramanan**[2] **James Hays**[5]
[1]University of Pennsylvania [2]Carnegie Mellon University [3]Motional
[4]Lawrence Livermore National Laboratory [5]Georgia Tech

Abstract

Scene flow estimation is the task of describing the 3D motion field between temporally successive point clouds. State-of-the-art methods use strong priors and test-time optimization techniques, but require on the order of tens of seconds for large-scale point clouds, making them unusable as computer vision primitives for real-time applications such as open world object detection. Feed forward methods are considerably faster, running on the order of tens to hundreds of milliseconds for large-scale point clouds, but require expensive human supervision. To address both limitations, we propose *Scene Flow via Distillation*, a simple distillation framework that uses a label-free optimization method to produce pseudo-labels to supervise a feed forward model. Our instantiation of this framework, *ZeroFlow*, produces scene flow estimates in real-time on large-scale point clouds at quality competitive with state-of-the-art methods while using zero human labels. Notably, at test-time ZeroFlow is over 1000× faster than label-free state-of-the-art optimization-based methods on large-scale point clouds and over 1000× cheaper to train on unlabeled data compared to the cost of human annotation of that data. To facilitate research reuse, we release our code, trained model weights, and high quality pseudo-labels for the Argoverse 2 and Waymo Open datasets.

1 Introduction

Scene flow estimation is an important primitive for open-world object detection and tracking [31, 47, 3, 16, 10]. For example, Najibi et al. [31] generate supervisory boxes for an open-world lidar detector via offline object extraction using high quality scene flow estimates from Neural Scene Flow Prior (NSFP) [25]. Although NSFP does not require human supervision, it takes tens of seconds to run on a single large-scale point cloud pair — if NSFP were both high quality and fast enough to run in real-time, its estimations could be directly used as a runtime primitive in the downstream detector, instead of being relegated to an offline pipeline. This runtime feature formulation is similar to Zhai et al. [47]'s use of scene flow from FlowNet3D [26] as an input primitive for their multi-object tracking pipeline; although FlowNet3D is fast enough for online processing of the modestly-sized point clouds, its supervised feed forward formulation requires significant in-domain human annotations.

Broadly, these exemplar methods are representative of the strengths and weakness of their class of approach: state-of-the-art scene flow methods train feed forward models using *extremely expensive* human annotations[1], or use test-time optimization techniques that operate without human annotations, but are too slow for online scene flow estimation[2].

*Corresponding email: kvedder@seas.upenn.edu

[1]At ∼$0.10 per cuboid per frame, the Argoverse 2 [45] train split cost ∼$750,000 to annotate. In comparison, ZeroFlow's pseudo-labels cost $394 at current cloud compute prices. See Supplemental C for details.

[2]Neural Scene Flow Prior [25] takes more than 26 seconds per point cloud pair on the Argoverse 2 [45] train split. See Supplemental C for details.

Preprint. Under review.

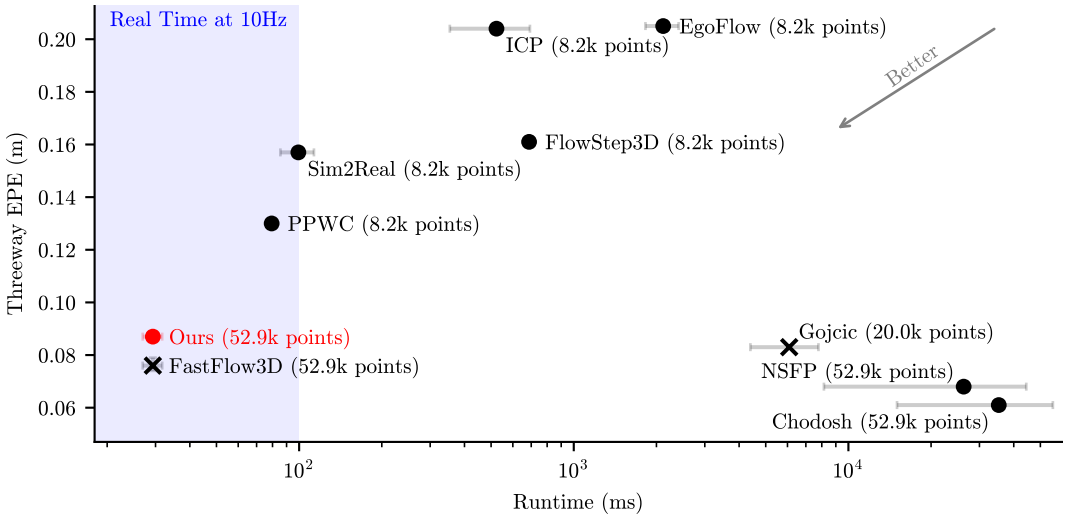


Figure 1: We plot the performance and run-time of recent scene flow methods on the Argoverse 2 [45] Sensor dataset, along with the size of the point cloud prescribed in the method's evaluation protocol. ZeroFlow produces competitive quality scene flow estimates on full-sized point clouds compared to optimization-based approaches (e.g. NSFP [25] and Chodosh et al. [6]), but is over 1000× faster. Methods that use *any* human labels are plotted with ✕, and zero-label methods are plotted with ●.

We propose *Scene Flow via Distillation* (SFvD), a framework that, while conceptually simple, generates a new class of scene flow estimation methods that combine the strengths of optimization-based and feed forward methods to achieve fast run-time and high accuracy without human supervision. We instantiate this pipeline into the method *Zero-Label Real-Time Scene Flow* (ZeroFlow), using components that scale to large-scale point clouds while providing high quality scene flow estimates. As a result, on both the Argoverse 2 (Figure 1) and Waymo Open [37] datasets, ZeroFlow produces competitive quality scene flow estimates in real-time on full point clouds using *zero* human labels.

Our primary contributions include:

- We introduce a simple yet effective distillation framework, *Scene Flow via Distillation* (SFvD), which uses a label-free optimization method to produce pseudo-labels to supervise a feed forward model, allowing us to approximate the performance of slow optimization-based approaches at the speed of feed forward methods.
- Using SFvD, we present *Zero-Label Real-Time Scene Flow* (ZeroFlow), a method that produces competitive quality scene flow in real-time on large-scale clouds, running over 1000× faster than state-of-the-art optimization methods on real point clouds and is over 1000× cheaper to train compared to the cost of human annotations.
- We release high quality flow pseudo-labels (representing 4.8 GPU months of compute) for the popular Argoverse 2 [45] and Waymo Open [37] autonomous vehicle datasets, alongside our code and trained model weights, to facilitate research reuse.

2 Background and Related Work

Preliminaries. Given point clouds P_t at time t and P_{t+1} at time $t+1$, scene flow estimators predict $\hat{F}_{t,t+1}$, which contains one 3D vector for each point in P_t to describe how it moves from t to $t+1$ (Figure 2). Traditionally, the performance of a scene flow method has been measured using the Endpoint Error (EPE) between the predicted flow $\hat{F}_{t,t+1}$ and ground truth flow $F^*_{t,t+1}$ (Equation 1):

$$\text{EPE}\left(P_t\right) = \frac{1}{\|P_t\|} \sum_{p \in P_t} \left\| \hat{F}_{t,t+1}(p) - F^*_{t,t+1}(p) \right\|_2 . \tag{1}$$

Related Work. Vedula et al. [41] first studied the problem of scene flow from stereo RGB to estimate dense non-rigid scene flow from optical flow. More recently, scene flow is commonly studied in

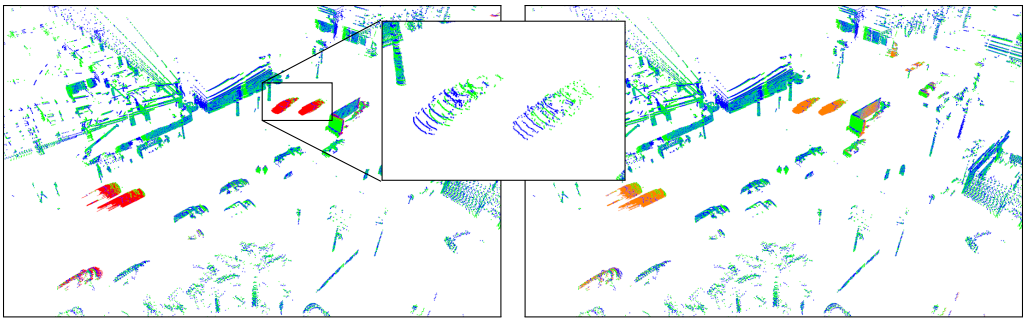


Figure 2: Scene flow estimation of two consecutive point clouds sampled 100 ms apart (green and blue, respectively) on Argoverse 2 [45]. **Left:** We show the ground truth scene flow annotations in red. These annotations are derived from the motion of amodal bounding boxes. **Right:** We show ZeroFlow's scene flow estimates estimates in orange, which closely match with the ground truth.

the context of lidar [1, 5, 13, 15, 14, 29, 33, 34, 43, 44]. Motivated by applications in autonomous vehicles, Dewan et al. [7] defined the problem of scene flow estimation between two point clouds.

Recent approaches for scene flow estimation can be split into three categories, including test-time optimization [35, 9, 25, 6], supervised deep learning [26, 4, 38, 20, 46, 36, 23, 18, 12, 2], and self-supervised deep learning methods [30, 3, 11, 8, 24]. Supervised deep learning methods are efficient at test-time; however, they require costly human annotations at train-time. Both test-time optimization and self-supervised methods seek to address this problem through the use of label-free proxy losses such as Chamfer distance [35], cycle-consistency [30], and various rigidity priors [7, 35, 24, 6, 3, 11]. Self-supervised methods achieve faster inference but do not match the quality of optimization-based methods [6]. Typically, self-supervised approaches also require more sophisticated network architectures compared to supervised methods [3, 11, 20]. In practice, this can make them slower and more difficult to train. In contrast to existing work, we take advantage of the quality of optimization-based methods as well as the efficiency and architectural simplicity of supervised networks. Our approach, ZeroFlow, uses label-free optimization methods [25] to produce pseudo-labels to supervise a feed forward model [18].

3 Method

We propose *Scene Flow via Distillation* (SFvD), a conceptually simple distillation framework which describes a new class of scene flow estimation methods that are fast, high quality, and require zero human supervision (Figure 3). Concretely, we use a label-free optimization-based method to generate scene flow pseudo-labels, which are used to supervise a fast feed forward model. While conceptually simple, efficiently instantiating SFvD requires careful construction; most online optimization methods and feed forward models are unable to efficiently scale to large-scale point clouds (Section 3.1).

Based on our scalability analysis, we propose *Zero-Label Real-Time Scene Flow* (ZeroFlow), an SFvD-based scene flow method that produces competitive quality scene flow in real-time for large-scale point clouds without any human labels (Algorithm 1). ZeroFlow uses Neural Scene Flow prior (NSFP) [25] to generate high quality, label-free pseudo-labels on full-size point clouds (Section 3.2) and the feed forward architecture of FastFlow3D [18] for efficient flow inference (Section 3.3).

3.1 Scaling *Scene Flow via Distillation* to Large Point Clouds

Popular AV datasets including Argoverse 2 [45] (collected with dual Velodyne VLP-32 sensors) and Waymo Open [37] (collected with a proprietary lidar sensor and subsampled) have large-scale point clouds with an average of 52,000 and 79,000 points per frame, respectively, after ground plane removal (Supplemental A and Figure 5). For practical applications, sensors such as the Velodyne VLP-128 in dual return mode produce up to 480,000 points per sweep [42] and proprietary sensors at full resolution can produce well over 1 million points per sweep. Therefore, scene flow methods must scale to large-scale point clouds for real-world applications.

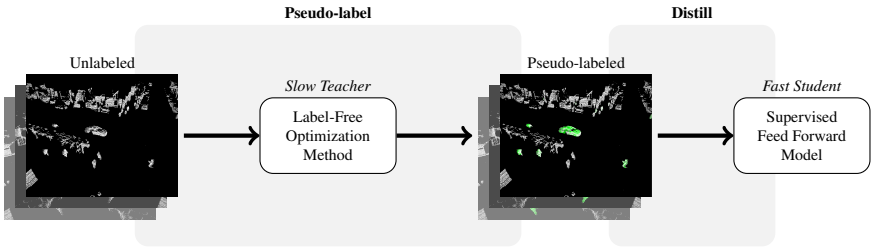


Figure 3: The *Scene Flow via Distillation* (SFvD) framework, which describes a new class of scene flow methods that produce high quality, human label-free flow at the speed of feed forward networks.

Unfortunately, most methods focus strictly on scene flow *quality* for toy point clouds constructed by randomly subsampling full point clouds down to 8,192 points [17, 38, 46, 20, 26, 25]. Motivated by real-world applications, we target scene flow estimation for the full point cloud. At this scale, architectural efficiency is of paramount importance. FastFlow3D [18], which uses a PointPillar-style [21] encoder, can process 1 million points in under 100 ms on an NVIDIA Tesla P1000 GPU (making it real-time for a 10Hz lidar), while methods like FlowNet3D [26], which use 3D point-wise convolutions, take almost 4 seconds to process the same point cloud.

We design our approach to effectively scale to large point clouds. For SFvD's pseudo-labeling step, method run-time is less of a concern — pseudo-labeling each point cloud pair is embarrassingly parallel, and high-quality methods like Neural Scene Flow Prior (NSFP) [25] require only a modest amount of GPU memory (under 3GB) when estimating scene flow on point clouds with 70K points. This enables fast pseudo-labeling using commodity GPUs; as described in Supplemental C, pseudo-labeling the Argoverse 2 train split with NSFP is over 1000$\times$ cheaper than human annotation. The runtime of SFvD's student feed forward model *is* critical, as it determines the method's test-time speed, motivating models like FastFlow3D [18] that can process large-scale point clouds in real-time.

3.2 Neural Scene Flow Prior is a Slow Teacher

Neural Scene Flow Prior (NSFP) [25] is an optimization-based approach to scene flow estimation. Notably, it does not use ground truth labels to generate high quality flows, instead relying upon strong priors from its learnable function class (determined by the architecture of its coordinate network) and proxy measures of scene flow (e.g. Equation 2). Point residuals are fit per point cloud pair P_t, P_{t+1} at runtime by randomly initializing two MLPs; one to describe the forward flow $\hat{F}^+$ from P_t to P_{t+1}, and one to describe the reverse flow $\hat{F}^-$ from $P_t + \hat{F}_{t,t+1}$ to P_t in order to impose a cyclic constraint. The forward flow $\hat{F}^+$ and backward flow $\hat{F}^-$ are then optimized jointly to minimize

$$\text{TruncatedChamfer}(P_t + \hat{F}^+, P_{t+1}) + \text{TruncatedChamfer}(P_t + \hat{F}^+ + \hat{F}^-, P_t) \ , \tag{2}$$

where TruncatedChamfer is the standard Chamfer distance with per-point distances above 2 meters set to zero to reduce the influence of outliers.

NSFP is able to produce high-quality scene flow estimations due to its choice of coordinate network architecture and cyclical constraint. The coordinate network's learnable function class is expressive enough to fit the low frequency signal of residuals for moving objects while restrictive enough to avoid fitting the high frequency noise from TruncatedChamfer, and the cyclic constraint acts as a smoothness regularizer for the forward flow. NSFP provides high quality estimates on full size point clouds (Figure 1), making it a good candidate for the pseudo-label step of SFvD.

3.3 FastFlow3D is a Fast Student

FastFlow3D [18] is an efficient feed forward network that learns using human supervisory labels $F^*_{t,t+1}$ and per-point foreground / background class labels. FastFlow3D minimizes a variation

of the End-Point Error (Equation 1) that reduces the importance of background points, thus minimizing

$$\frac{1}{\|P_t\|} \sum_{p \in P_t} \sigma(p) \left\| \hat{F}_{t,t+1}(p) - F^*_{t,t+1}(p) \right\|_2 \quad (3) \qquad \text{where} \qquad \sigma(p) = \begin{cases} 1 & \text{if } p \in \text{Foreground} \\ 0.1 & \text{if } p \in \text{Background} \end{cases}. \quad (4)$$

FastFlow3D uses a PointPillars-style encoder [21], traditionally used for efficient lidar object detection [39, 40], with a U-Net style backbone. The encoder of the U-Net processes P_t and P_{t+1} separately, and the decoder jointly processes both embeddings. A small MLP is used to extract scene flow for each point in P_t using the point's coordinates and its associated output pillar feature.

As discussed in Section 3.1, FastFlow3D's design choices make it highly scalable for large point clouds. While most feed forward networks are evaluated on a standard toy evaluation protocol with subsampled point clouds, FastFlow3D is able to scale up to full resolution point clouds while maintaining real-time performance and emitting competitive quality scene flow estimates, making it a good candidate for the distillation step of SFvD.

In order to train FastFlow3D using pseudo-labels, we need a replacement $\sigma(\cdot)$ scaling function (Equation 4) because our pseudo-labels do not provide foreground / background semantics. We propose a soft weighting based on pseudo-label flow magnitude: for the point p in the pseudo-label flow $F^*_{t,t+1}(p)$, where $s(p)$ represents its speed in meters per second, we linearly interpolate the weight of p between $0.1\times$ at 0.4 m/s and full weight at 1.0 m/s, i.e.

$$\sigma(p) = \begin{cases} 0.1 & \text{if } s(p) < 0.4 \text{ m/s} \\ 1.0 & \text{if } s(p) > 1.0 \text{ m/s} \\ 1.8s - 0.8 & \text{o.w.} \end{cases} \quad (5)$$

These thresholds are selected to down-weight approximately 80% of points by $0.1\times$, with the other 20% of points split between the soft and full weight region[3]. In Section 4.2, we show that our weighting scheme provides non-trivial improvements over uniform weighting (i.e. $\sigma(\cdot) = 1$).

Additionally, we depart from FastFlow3D's problem setup in two minor ways. Unlike FastFlow3D, we delete ground points using dataset provided maps, and use the standard scene flow problem setup of predicting residuals (Section 2) instead of predicting future flow vectors in meters per second given two prior frames. Our approach is given in Algorithm 1, with setup details specified in Section 4.1.

Algorithm 1 ZeroFlow

$D \leftarrow$ collection of unlabeled point cloud pairs ▷ Training Data
for $P_t, P_{t+1} \in D$ **do** ▷ Embarrasingly Parallel `For`
  $F^*_{t,t+1} \leftarrow \text{TeacherNSFP}(P_t, P_{t+1})$ ▷ SFvD *Pseudo-label* Step
for epoch $\in$ epochs **do**
  for $P_t, P_{t+1}, F^*_{t,t+1} \in D$ **do** ▷ SFvD's *Distill* Step
    $l \leftarrow \text{Equation 3}(\text{StudentFastFlow3D}_\theta(P_t, P_{t+1}), F^*_{t,t+1})$
    $\theta \leftarrow \theta$ updated w.r.t. l

4 Experiments

ZeroFlow is designed to provide competitive quality scene flow in real-time. In order to validate this and understand the impact of every component in the Scene Flow via Distillation framework used to create ZeroFlow, we perform extensive experiments on the Argoverse 2 [45] and Waymo Open [37] datasets. We compare to first party implementations of NSFP [25] and Chodosh et al. [6], implement FastFlow3D [18] ourselves (no first party implementation is available), and use the implementations from Chodosh et al. [6] for all other baselines.

While many scene flow methods are evaluated using average Endpoint Error (Equation 1), downstream applications often rely on good quality scene flow estimates for foreground points in particular;

[3]For Argoverse 2, exactly 78.1% of points are downweighted, 11.8% lie in the soft-weight region, and 10.1% lie in the full weight region; for Waymo Open 80.0% of points are downweighted, 7.9% lie in the soft-weight region, and 12.1% lie in the full-weight region respectively.

however, roughly 80% of typical point clouds are background. To prevent background points from dominating our evaluations, we use the metric proposed in Chodosh et al. [6], *Threeway EPE*:

$$\text{Threeway EPE}(P_t) = \text{Avg} \begin{cases} \text{EPE}(p \in P_t : p \in \text{Background}) & \text{(Static BG)} \\ \text{EPE}(p \in P_t : p \in \text{Foreground} \wedge F^*_{t,t+1}(p) \leq 0.5\text{m/s}) & \text{(Static FG)} \\ \text{EPE}(p \in P_t : p \in \text{Foreground} \wedge F^*_{t,t+1}(p) > 0.5\text{m/s}) & \text{(Dynamic FG)} \end{cases}. \tag{6}$$

4.1 How does ZeroFlow compare to state-of-the-art scene flow methods on real point clouds?

In order to characterize how ZeroFlow compares to other scene flow estimation methods, we evaluate it against many baselines on the Argoverse 2 Sensor dataset [45] and confirm these characteristics hold on the Waymo Open [37] dataset version 1.4. Both datasets are composed of real world point clouds collected from lidar on an autonomous vehicle, along with ego pose and map information (enabling efficient ego motion compensation and ground point removal), and amodal bounding boxes on moving objects (enabling human flow supervision).

Argoverse 2. The dataset contains 700 training and 150 validation sequences. Each sequence contains 15 seconds of 10Hz point clouds collected using two Velodyne VLP-32s mounted on the roof of a car. As part of the training protocol for ZeroFlow, FastFlow3D, and NSFP w/ Motion Compensation, we perform ego compensation, ground point removal, and restrict all points to be within a 102.4m $\times$ 102.4m area centered around the ego vehicle, resulting in point clouds with an average of 52,871 points (Figure 5a). The point cloud P_{t+1} is centered at the origin of the ego vehicle's coordinate system and P_t is projected into P_{t+1}'s coordinate frame. For ZeroFlow and FastFlow3D, the PointPillars encoder uses $0.2\text{m}\times0.2\text{m}$ pillars, with all architectural configurations matching [18]. For NSFP w/ Motion Compensation, we use the same architecture and early stopping parameters as the original method [25]. For FastFlow3D and the FastFlow3D student architecture of ZeroFlow, we train to convergence (50 epochs) with an Adam [19] learning rate of 2×10^{-6} and batch size 64. For all other methods in Table 1, we use the implementations provided by Chodosh et al. [6], which follow ground removal and ego compensation protocols from their respective papers.

Waymo Open. The dataset contains 798 training and 202 validation sequences. Each sequence contains 20 seconds of 10Hz point clouds collected using a custom lidar mounted on the roof of a car. We use the same preprocessing and training configurations used on Argoverse 2; after ego motion compensation and ground point removal, the average point cloud has 79,327 points (Figure 5b).

Table 1: Quantitative results on the Argoverse 2 Sensor validation split using the evaluation protocol from Chodosh et al. [6]. The methods used in this paper, shown in the first block of the table, are trained and evaluated on point clouds within a 102.4m $\times$ 102.4m area centered around the ego vehicle. However, following the protocol of Chodosh et al. [6], all methods report error on points in the 70m $\times$ 70m area centered around the ego vehicle. Runtimes are collected on an NVIDIA V100 with a batch size of 1 [22]. ZeroFlow and FastFlow3D have identical feed forward architectures and thus share the same runtime. Methods with an * have performance averaged over 3 training runs (see Supplemental B for details). Underlined methods require human supervision.

	Runtime (ms)		Point Cloud Subsample Size	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow* (Ours)	29.33±	2.38	Full Point Cloud	0.087	0.217	0.023	0.023	0.394	0.164
FastFlow3D* [18]	26285.0±	18139.3	Full Point Cloud	0.076	0.186	0.021	0.021	0.474	0.200
NSFP w/ Motion Comp [25]	35281.4±	20247.7	Full Point Cloud	0.068	0.131	0.036	0.036	0.699	0.472
Chodosh et al. [6]			Full Point Cloud	0.061	0.129	0.028	0.028	0.710	0.477
Odometry	—		Full Point Cloud	0.198	0.583	0.010	0.000	0.108	0.002
Gojcic [11]	6087.87±	1690.56	20000	0.083	0.155	0.064	0.032	0.650	0.368
Sim2Real [17]	99.35±	13.88	8192	0.157	0.229	0.106	0.137	0.565	0.254
EgoFlow [38]	2116.34±	292.32	8192	0.205	0.447	0.079	0.090	0.111	0.018
PPWC [46]	79.43±	2.20	8192	0.130	0.168	0.092	0.129	0.556	0.229
FlowStep3D [20]	687.54±	3.13	8192	0.161	0.173	0.132	0.176	0.553	0.248
ICP [5]	523.11±	169.34	8192	0.204	0.557	0.025	0.028	0.112	0.015

Analysis of Results. ZeroFlow runs in real-time (Tables 1–2), matching the state-of-the-art runtime of FastFlow3D, a supervised method with the same feed forward architecture, and running more than 3$\times$ faster than the next fastest label-free method (PPWC [46]) while processing almost 7$\times$ more

Table 2: Quantitative results on Waymo Open using the evaluation protocol from Chodosh et al. [6]. Runtimes are collected on an NVIDIA A6000 with a batch size of 1 [22]; both methods have identical feed forward architectures and thus share the same runtime. FastFlow3D requires human supervision.

	Runtime (ms)	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow (Ours)	$21.66\pm$ 0.48	0.083	0.216	0.016	0.017	0.376	0.149
FastFlow3D [18]		0.075	0.195	0.015	0.016	0.490	0.239

points. But ZeroFlow is not only fast, it produces competitive quality flow: its Threeway EPE is around 1 cm worse than FastFlow3D, and 3.5 cm worse than slow, high quality optimization-based approaches [25, 6]. To contextualize the scale of these errors, the manufacturer of Argoverse 2 lidars only guarantees the depth accuracy of the lidars themselves to ± 3.0 cm of error [27]; see Supplemental B for further discussion of meaningful differences in quality.

Table 1 also demonstrates the interesting phenomena where ZeroFlow's student model can surpasses the performance of the teacher (NSFP [25]); ZeroFlow outperforms NSFP on *Static FG EPE* and *Static BG EPE*. NSFP's error on these stationary points is zero mean, and we posit that the student network learned to recognize non-moving points and regress their expected flow of $\vec{0}$.

4.2 What are the label differences that allow FastFlow3D to outperform ZeroFlow?

In Table 1 and Table 2, the human label supervised FastFlow3D slightly outperforms ZeroFlow; both have identical model architectures at inference and train on the same point clouds, meaning differences must come from human versus machine generated flow vectors and human foreground / background point labeling of those point clouds. We aim to understand: to what degree are various differences in flow vectors and components of loss functions implicated in these performance differences?

4.2.1 Is there systematic error in method scene flow estimates compared to human labels?

NSFP, Chodosh, FastFlow3D, and ZeroFlow all provide high quality flow on full size point clouds (Figure 1); however, no method *perfectly* matches human ground truth annotations. Are these disagreements symmetric and zero mean? Is there systematic bias?

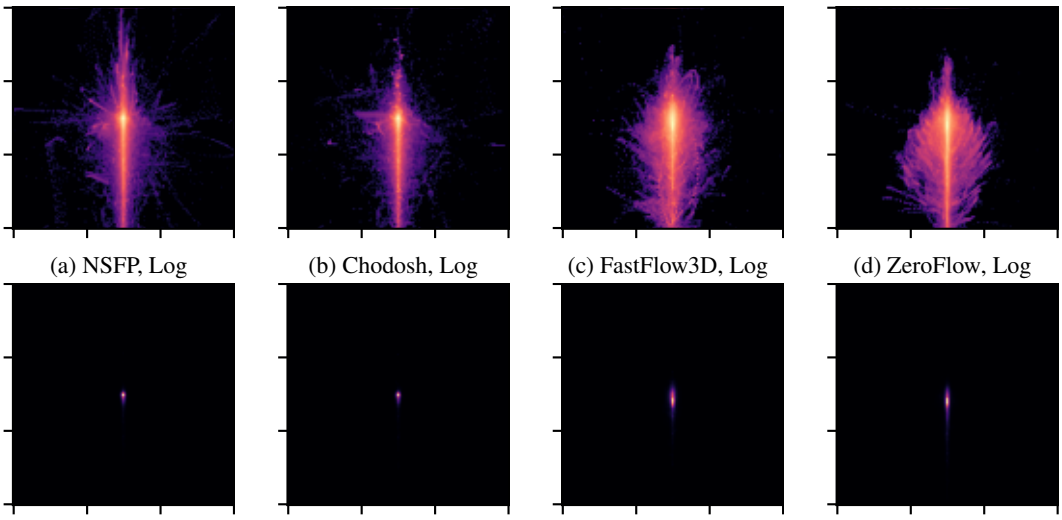
To understand these possible biases, using Argoverse 2 [45] we compute a metric space endpoint residual heatmap of different scene flow estimates for points moving above 0.5 m/s (Figure 4). For each point, the method's estimated flow vector is projected into birds-eye-view and rotated into the coordinate frame of the human annotated ground truth vector; the ground truth vector is made to point vertically, with its endpoint at 0m, 0m and the relative endpoint of the pseudo-label is accumulated on the heatmap. Unrotated versions of these residual plots as well as heatmaps for naïve baselines (e.g. Nearest Neighbors, $\vec{0}$ Flow) are shown in Supplemental D, Figures 6–8. We also compare ZeroFlow trained using NSFP pseudo-labels to ZeroFlow trained using Chodosh pseudo-labels (Section 4.2.4); the resulting residual heatmaps (Supplemental D, Figure 8) are virtually identical.

All method labels have zero mean transverse error (horizontal axis), but varying degrees of flow underestimation along the longitudinal axis. NSFP underestimates flow magnitude by 0.950cm / frame, Chodosh by 0.968cm / frame, FastFlow3D by 1.208cm / frame, and ZeroFlow by 1.913cm / frame. These growing error rates are roughly proportional to the overall performance of these methods on Argoverse 2. See Supplemental B to contextualize the scale of these differences.

4.2.2 How much of ZeroFlow's performance is due to its flow speed weighting?

Unlike human supervised FastFlow3D, ZeroFlow *cannot* use human foreground / background point labels, so it uses pseudo-label point speed as a proxy for importance (Section 3.3, Equation 5). To understand the impact of this weighting on the performance of ZeroFlow, we train ZeroFlow with a modified loss; rather than scaling by speed (Equation 5), we uniformly weight all points ($\sigma(\cdot) = 1$).

Table 3 shows Equation 5's scaling provides a non-trivial (1.4cm) performance improvement over the uniform weighting on *Dynamic FG EPE*; intuitively, this is because upweighting fast-moving objects leads to lower error on these dynamic foreground points. The static point error differences are within 0.1cm (within retrain error; see Supplemental B), resulting in Threeway EPE increase of 0.5cm.



(a) NSFP, Log (b) Chodosh, Log (c) FastFlow3D, Log (d) ZeroFlow, Log

(e) NSFP, Abs. (f) Chodosh, Abs. (g) FastFlow3D, Abs. (h) ZeroFlow, Abs.

Figure 4: Birds-eye-view heatmaps of endpoint residuals for different teacher and student methods on non-background points moving above 0.5m/s, rotated into the coordinate frame of the ground truth labels. Brighter color indicates more points in each bin, and the distance between ticks is 1 meter. Perfect labels would produce a single central dot. The top row shows frequency on a log color scale to show error distribution shape; bottom row shows frequency on an absolute color scale to show relative frequency. All methods slightly underestimate flow vector length, but relative to the scale of other forms of noise in autonomous vehicles (e.g. sensor noise), the magnitude is negligible.

Table 3: Comparison between ZeroFlow trained on Argoverse 2 using the uniform point weighting and speed-based upweighting in Equation 5. Our speed-based upweighting scheme provides non-trivial improvement over the uniform point weighting baseline. Both methods have their results averaged over 3 runs (see Supplemental B for details).

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow*	0.087	0.217	0.023	0.023	0.394	0.164
ZeroFlow ($\sigma(\cdot) = 1$)*	0.092	0.231	0.022	0.022	0.380	0.150

4.2.3 How much of FastFlow3D's performance is due to its semantic point weighting?

Unlike ZeroFlow, FastFlow3D *can* use human foreground / background point labels to upweight the flow importance of foreground points (Section 3.3, Equation 4). To understand the impact of this weighting, we train FastFlow3D with a modified loss; rather than scaling using semantics as described in Equation 4, we uniformly weight all points ($\sigma(\cdot) = 1$).

Table 4: Comparison between ZeroFlow, FastFlow3D, and the ablated FastFlow3D with uniform scaling ($\sigma(\cdot) = 1$) trained on Argoverse 2. The performance of FastFlow3D with Uniform Scaling is nearly identical to ZeroFlow's performance. Methods with an * have performance averaged over 3 training runs (see Supplemental B for details). Underlined methods require human supervision.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow* (Ours)	0.087	0.217	0.023	0.023	0.394	0.164
FastFlow3D ($\sigma(\cdot) = 1$)	0.085	0.220	0.018	0.018	0.434	0.182
FastFlow3D* [18]	0.076	0.186	0.021	0.021	0.474	0.200

As shown in Table 4, the performance of FastFlow3D ($\sigma(\cdot) = 1$) degrades almost completely to ZeroFlow's performance (e.g. $0.076 \rightarrow 0.085$ vs 0.087 for Threeway EPE).

This raises the question: why is the performance improvement of semantic weighting larger than our unsupervised moving point weighting scheme (Section 4.2.2)? Not only does semantic weighting provide increased loss on moving objects, it implicitly teaches the network to recognize the structure of objects themselves. For example, with Equation 4 scaling, end-point error on a stationary pedestrian is significantly higher than static background points, incentivizing the network to learn to detect the point *structure* common to pedestrians, even if immobile, to perfect the predictions on those points.

4.2.4 Can ZeroFlow improve by using an existing, better quality pseudo-labeling method?

Section 4.2.3 shows that FastFlow3D trained with human flow vectors alone degrades to almost the level of ZeroFlow; however the remaining quality gap (8.7 cm Threeway EPE with pseudo-label vectors and speed weighting, versus 8.5 cm with human vectors and no weighting) must be due to the superior quality of human flow vectors. This raises the question: can an existing optimization method that produces better quality flow vectors produce a better version of ZeroFlow?

To understand the impact of better pseudo-label vectors, we train ZeroFlow on Argoverse 2 using superior quality flow vectors from Chodosh et al. [6], which proposes a refinement step to NSFP lablels to provide improvements to flow vector quality.

Table 5: Comparison between ZeroFlow trained on Argoverse 2 using NSFP pseudo-labels and ZeroFlow using Chodosh et al. [6] pseudo-labels. Methods with an * have performance averaged over 3 training runs (see Supplemental B for details). The minor quality improvement of Chodosh pseudo-labels does not lead to a meaningful difference in performance.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow (NSFP pseudo-labels)*	0.087	0.217	0.023	0.023	0.394	0.164
ZeroFlow (Chodosh et al. [6] pseudo-labels)	0.088	0.227	0.019	0.019	0.366	0.139
ZeroFlow ($\sigma(\cdot) = 1$, NSFP pseudo-labels)*	0.092	0.231	0.022	0.022	0.380	0.150
ZeroFlow ($\sigma(\cdot) = 1$, Chodosh et al. [6] pseudo-labels)	0.090	0.234	0.018	0.018	0.364	0.136

As shown in Table 5, ZeroFlow trained on Chodosh refined pseudo-labels provides no meaningful quality improvement over the raw NSFP pseudo-labels (as discussed in Supplemental B, a Threeway EPE difference of 0.2cm is within training variance for ZeroFlow). These results also hold for our ablated version of ZeroFlow with uniform point scaling from Section 4.2.2.

5 Conclusion

Our scene flow approach, Zero-Label Real-Time Scene Flow (ZeroFlow), produces competitive quality scene flow in real-time *without human labels* via our conceptually simple distillation pipeline.

The human annotation free nature of our system provides several technical and social benefits. Due to the fact that human labels are *extremely expensive* to collect, sequences selected for annotation tend to be chosen for their *structural* diversity (e.g. dense urban scenes that tend to have lots of stationary objects) in order to maximally benefit lidar detectors. Our label-free training is orders of magnitude cheaper and is embarrassingly parallel, enabling a training run of the model on the *full* distribution of logs (including logs with significantly more motion but are uninteresting to lidar detectors, e.g. highways), thus directly matching the deployment distribution and consequently leading to greater robustness during deployment.

Breaking the dependence upon human annotations also democratizes scene flow — practitioners only need data and commodity GPUs to get labels, instead of tremendous capital to subcontract with a data labeling company that may have questionable labor ethics [32].

Limitations and Future Work. ZeroFlow inherits the biases of its pseudo-labels. Unsurprisingly, if the pseudo-labels consistently fail to predict scene flow for small objects, our method will also be unable to predict scene flow for these objects; however, further innovation in model architecture, loss functions, and pseudo-labels may yield better performance. In order to enable further work on Scene Flow via Distillation-based methods, we release[4] our code, trained model weights, and NSFP flow pseudo-labels, representing 1.2 GPU months for Argoverse 2 and 3.5 GPU months for Waymo Open.

[4]Links to these materials will be provided after review.

References

[1] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. *Int. J. Comput. Vis.*, 101(1):6–21, 2013.

[2] Ramy Battrawy, René Schuster, Mohammad-Ali Nikouei Mahani, and Didier Stricker. RMS-FlowNet: Efficient and Robust Multi-Scale Scene Flow Estimation for Large-Scale Point Clouds. In *Int. Conf. Rob. Aut.*, pages 883–889. IEEE, 2022.

[3] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. SLIM: Self-supervised LiDAR scene flow and motion segmentation. In *Int. Conf. Comput. Vis.*, pages 13126–13136, 2021.

[4] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Int. Conf. Comput. Vis.*, pages 7962–7971, 2019.

[5] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Img. Vis. Comput.*, 10(3):145–155, 1992.

[6] Nathaniel Chodosh, Deva Ramanan, and Simon Lucey. Re-Evaluating LiDAR Scene Flow for Autonomous Driving. *arXiv preprint*, 2023.

[7] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *Int. Conf. Intel. Rob. Sys.*, pages 1765–1770. IEEE, 2016.

[8] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting Rigidity Constraints for LiDAR Scene Flow Estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12776–12785, 2022.

[9] Marvin Eisenberger, Zorah Lahner, and Daniel Cremers. Smooth shells: Multi-scale shape registration with functional maps. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12265–12274, 2020.

[10] Emeç Erçelik, Ekim Yurtsever, Mingyu Liu, Zhijie Yang, Hanzhen Zhang, Pınar Topçam, Maximilian Listl, Yılmaz Kaan Çaylı, and Alois Knoll. 3D Object Detection with a Self-supervised Lidar Scene Flow Backbone. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 247–265, Cham, 2022. Springer Nature Switzerland.

[11] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5692–5703, 2021.

[12] 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 *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3254–3263, 2019.

[13] Simon Hadfield and Richard Bowden. Kinecting the dots: Particle based scene flow from depth sensors. In *Int. Conf. Comput. Vis.*, pages 2290–2295. IEEE, 2011.

[14] Simon Hadfield and Richard Bowden. Scene particles: Unregularized particle-based scene flow estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(3):564–576, 2013.

[15] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3526–3533, 2014.

[16] Shengyu Huang, Zan Gojcic, Jiahui Huang, and Konrad Schindler Andreas Wieser. Dynamic 3D Scene Analysis by Point Cloud Accumulation. In *European Conference on Computer Vision, ECCV*, 2022.

[17] Zhao Jin, Yinjie Lei, Naveed Akhtar, Haifeng Li, and Munawar Hayat. Deformation and Correspondence Aware Unsupervised Synthetic-to-Real Scene Flow Estimation for Point Clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7233–7243, 2022.

[18] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable Scene Flow From Point Clouds in the Real World. *IEEE Robotics and Automation Letters*, 12 2021.

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

[20] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4114–4123, 2021.

[21] Alex Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019.

[22] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 473–488. Springer, 2020.

[23] Ruibo Li, Guosheng Lin, Tong He, Fayao Liu, and Chunhua Shen. HCRF-Flow: Scene flow from point clouds with continuous high-order CRFs and position-aware flow embedding. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 364–373, 2021.

[24] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. RigidFlow: Self-Supervised Scene Flow Learning on Point Clouds by Local Rigidity Prior. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 16959–16968, 2022.

[25] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural Scene Flow Prior. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[27] Nikola Lopac, Irena Jurdana, Adrian Brnelić, and Tomislav Krljan. Application of Laser Systems for Detection and Ranging in the Modern Road Transportation and Maritime Sector. *Sensors*, 22(16), 2022. ISSN 1424-8220.

[28] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[29] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Int. Conf. Comput. Vis.*, pages 3061–3070, 2015.

[30] Himangi Mittal, Brian Okorn, and David Held. Just Go With the Flow: Self-Supervised Scene Flow Estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020.

[31] Mahyar Najibi, Jingwei Ji, Yin Zhou, Charles R. Qi, Xinchen Yan, Scott Ettinger, and Dragomir Anguelov. Motion Inspired Unsupervised Perception and Prediction in Autonomous Driving. European Conference on Computer Vision (ECCV), 2022.

[32] Billy Perrigo. OpenAI used Kenyan workers on less than $2 per hour, Jan 2023.

[33] J-P Pons, Renaud Keriven, O Faugeras, and Gerardo Hermosillo. Variational stereovision and 3D scene flow estimation with statistical similarity measures. In *Int. Conf. Comput. Vis.*, volume 2, pages 597–597. IEEE Computer Society, 2003.

[34] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *Int. J. Comput. Vis.*, 72 (2):179–193, 2007.

[35] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *Int. Conf. 3D Vis.*, pages 261–270. IEEE, 2020.

[36] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Eur. Conf. Comput. Vis.*, pages 527–544. Springer, 2020.

[37] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[38] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *Int. Conf. 3D Vis.*, pages 150–159. IEEE, 2020.

[39] Kyle Vedder. Current Approaches and Future Directions for Point Cloud Object Detection in Intelligent Agents, 2021.

[40] Kyle Vedder and Eric Eaton. Sparse PointPillars: Maintaining and Exploiting Input Sparsity to Improve Runtime on Embedded Systems. *International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[41] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Int. Conf. Comput. Vis.*, volume 2, pages 722–729. IEEE, 1999.

[42] *Velodyne Lidar Alpha Prime*. Velodyne Lidar, 11 2019.

[43] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D scene flow estimation with a rigid motion prior. In *Int. Conf. Comput. Vis.*, pages 1291–1298. IEEE, 2011.

[44] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Int. Conf. Comput. Vis.*, pages 1377–1384, 2013.

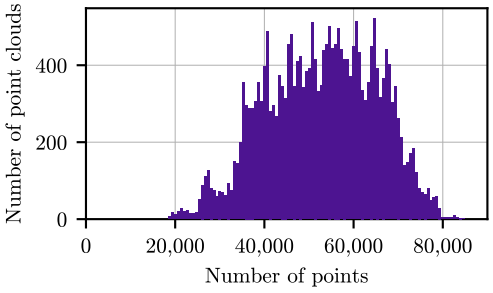
[45] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next Generation Datasets for Self-driving Perception and Forecasting. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021.

[46] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *Eur. Conf. Comput. Vis.*, pages 88–107. Springer, 2020.

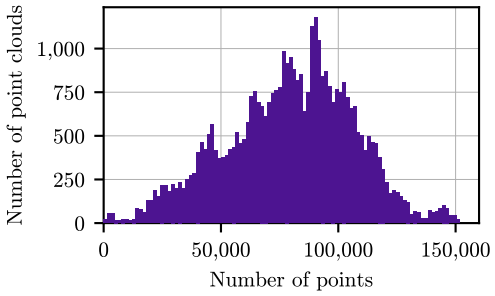
[47] Guangyao Zhai, Xin Kong, Jinhao Cui, Yong Liu, and Zhen Yang. FlowMOT: 3D Multi-Object Tracking by Scene Flow Association. *ArXiv*, abs/2012.07541, 2020.

A Argoverse 2 and Waymo Open point cloud sizes

As shown in Figure 5, Argoverse 2 [45] and Waymo Open [37] are significantly larger than the 8,192 point sampled point clouds of toy evaluation datasets, e.g. FlyingThings3D [28].



(a) Distribution of point cloud sizes in the Argoverse 2 Sensor *val* split: $\mu = 52,871.6; \sigma = 12,227.2$.



(b) Distribution of point cloud sizes in the Waymo Open *val* split: $\mu = 79,327.8; \sigma = 27,182.1$.

Figure 5: Point cloud size distributions for the *val* set of the Argoverse 2 Sensor [45] and Waymo Open [37] datasets after ground removal and clipped to a 102.4m $\times$ 102.4m box around the ego vehicle.

B Characterizing inter-training run final performance variance for ZeroFlow and FastFlow3D

On Argoverse 2, Threeway EPE difference between ZeroFlow and the human supervised FastFlow3D is 1.6cm (Table 1); how much of this gap can be attributed to training variance between runs? To answer this question, we train ZeroFlow and FastFlow3D from scratch 3 times each. ZeroFlow is trained on the same Argoverse 2 NSFP pseudo-labels (Table 7), resulting in a mean Threeway EPE of 0.092m with error of 0.003m (0.3cm) in either direction, and FastFlow3D is trained on the Argoverse 2 human labels (Table 8), resulting in a mean Threeway EPE of 0.092m with error under 0.003m (0.3cm) in either direction.

To contextualize the scale of this variance, the underlying Velodyne VLP-32 sensors used to collect the Argoverse 2 are only certified to ± 3 cm of error [27] (an order of magnitude greater than the deviation from the mean train performance for ZeroFlow), and this entirely neglects additional sources of noise introduced from other real world effects such as empirical ego motion compensation.

Table 6: Performance of ZeroFlow over 3 train runs on the same NSFP pseudo-labels.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow Run #1	0.087	0.214	0.023	0.023	0.409	0.180
ZeroFlow Run #2	0.087	0.215	0.024	0.024	0.397	0.164
ZeroFlow Run #3	0.089	0.222	0.022	0.022	0.377	0.147
ZeroFlow Average	0.087	0.217	0.023	0.023	0.394	0.164

Table 7: Performance of ZeroFlow ablated with uniform point scaling ($\sigma(\cdot) = 1$) over 3 train runs on the same NSFP pseudo-labels.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
ZeroFlow ($\sigma(\cdot) = 1$) Run #1	0.089	0.224	0.021	0.021	0.388	0.151
ZeroFlow ($\sigma(\cdot) = 1$) Run #2	0.092	0.231	0.022	0.022	0.377	0.155
ZeroFlow ($\sigma(\cdot) = 1$) Run #3	0.095	0.240	0.023	0.023	0.375	0.144
ZeroFlow ($\sigma(\cdot) = 1$) Average	0.092	0.231	0.022	0.022	0.380	0.150

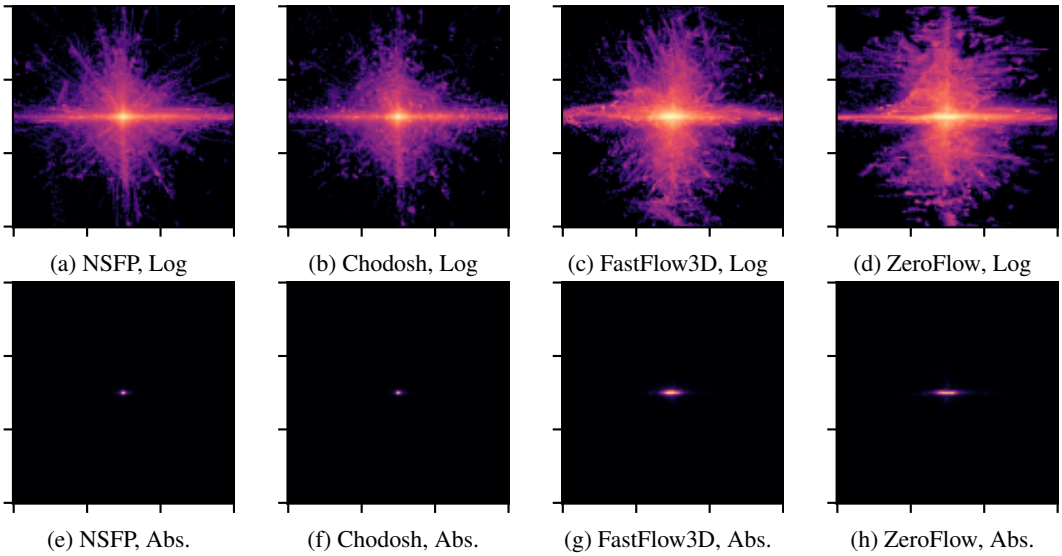
Table 8: Performance of FastFlow3D over 3 train runs on the Argoverse 2 human labels.

	Threeway EPE	Dynamic FG EPE	Static FG EPE	Static BG EPE	Dynamic AccRelax	Dynamic AccStrict
FastFlow3D Run #1	0.074	0.181	0.020	0.020	0.462	0.190
FastFlow3D Run #2	0.076	0.186	0.021	0.021	0.480	0.209
FastFlow3D Run #3	0.079	0.191	0.023	0.023	0.481	0.203
FastFlow3D Average	0.076	0.186	0.021	0.021	0.474	0.200

C Human Labeling versus Pseudo-labeling costs

NSFP pseudolabeling of the Argoverse 2 train split (700 sequences of 150 frames) required a total of 753 hours of NVidia Turing generation GPU time. At May, 2023 Amazon Web Services EC2 prices, a single `g4dn.xlarge`, equipped with a single NVidia Tesla T4, costs $0.526 per hour, for a total cost of $394 to pseudo-label. By comparison, at an estimated $0.10 per frame per cuboid (no public cost statements exist for production quality AV dataset labels), Argoverse 2's train split has an average of 75 cuboids per frame [45], for a total cost on the order of $787,500 to human annotate.

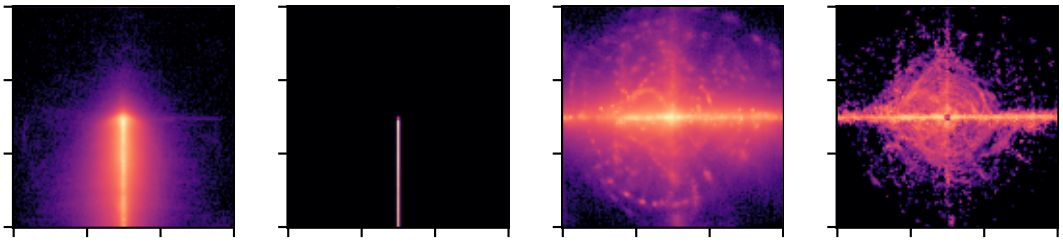
D Neural Scene Flow Prior Endpoint Errors



(a) NSFP, Log (b) Chodosh, Log (c) FastFlow3D, Log (d) ZeroFlow, Log

(e) NSFP, Abs. (f) Chodosh, Abs. (g) FastFlow3D, Abs. (h) ZeroFlow, Abs.

Figure 6: Birds-eye-view heatmap of endpoint residuals for different teacher and student methods on non-background points moving above 0.5m/s in the raw coordinate frame of the ground truth labels. Brighter color indicates more points in each bin. Perfect labels would produce a single central dot. Distance between ticks is 1 meter. Top row shows frequency on a log color scale to display error distribution shape. Bottom row shows frequency on an absolute color scale to display centroid.

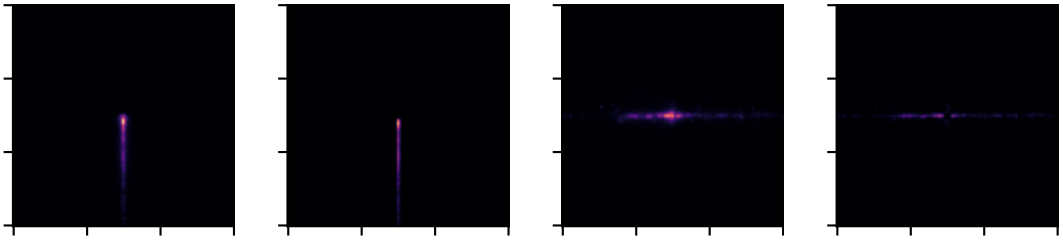


(a) Nearest Neighbor, Log, Rotated

(b) $\vec{0}$ Flow, Log, Rotated

(c) Nearest Neighbor, Log, Unrotated

(d) $\vec{0}$ Flow, Log, Unrotated



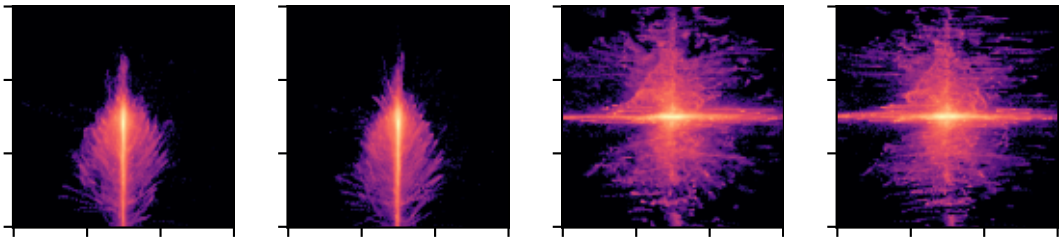
(e) Nearest Neighbor, Abs, Rotated

(f) $\vec{0}$ Flow, Abs, Rotated

(g) Nearest Neighbor, Abs, Unrotated

(h) $\vec{0}$ Flow, Abs, Unrotated

Figure 7: Birds-eye-view heatmap of endpoint residuals for naïve flow methods of predicting flow (Nearest Neighbor and $\vec{0}$ Flow on all points) for non-background points moving above 0.5m/s in the raw coordinate frame of the ground truth labels. Brighter color indicates more points in each bin. Perfect labels would produce a single central dot. Distance between ticks is 1 meter. Top row shows frequency on a log color scale to display error distribution shape. Bottom row shows frequency on an absolute color scale to display centroid. Left half shows results in the rotated coordinate frame of the ground truth labels. Right half shows results in the unrotated coordinate frame of the ground truth labels.

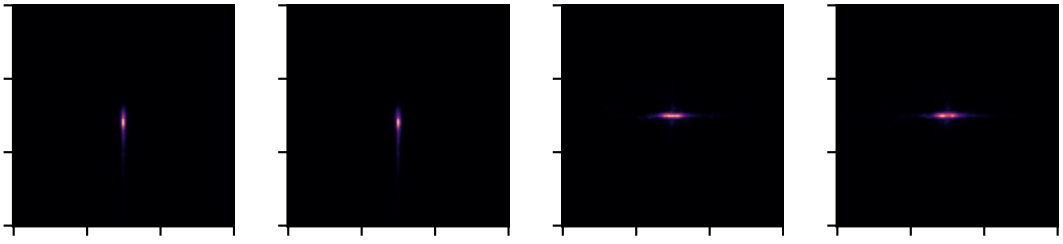


(a) ZeroFlow on NSFP, Log, Rotated

(b) ZeroFlow on Chodosh, Log, Rotated

(c) ZeroFlow on NSFP, Log, Unrotated

(d) ZeroFlow on Chodosh, Log, Unrotated



(e) ZeroFlow on NSFP, Abs, Rotated.

(f) ZeroFlow on Chodosh, Abs, Rotated.

(g) ZeroFlow on NSFP, Abs, Unrotated.

(h) ZeroFlow on Chodosh, Abs, Unrotated.

Figure 8: Birds-eye-view heatmap of endpoint residuals for ZeroFlow trained on NSFP and Chodosh pseudo-labels on non-background points moving above 0.5m/s in the raw coordinate frame of the ground truth labels. Brighter color indicates more points in each bin. Perfect labels would produce a single central dot. Distance between ticks is 1 meter. Top row shows frequency on a log color scale to display error distribution shape. Bottom row shows frequency on an absolute color scale to display centroid. Left half shows results in the rotated coordinate frame of the ground truth labels. Right half shows results in the unrotated coordinate frame of the ground truth labels.