OptFlow: Fast Optimization-based Scene Flow Estimation without Supervision

Rahul Ahuja Chris Baker Wilko Schwarting
ISEE AI
{rahulahuja, chrisbaker, wilko}@isee.ai

Abstract

Scene flow estimation is a crucial component in the development of autonomous driving and 3D robotics, providing valuable information for environment perception and navigation. Despite the advantages of learning-based scene flow estimation techniques, their domain specificity and limited generalizability across varied scenarios pose challenges. In contrast, non-learning optimization-based methods, incorporating robust priors or regularization, offer competitive scene flow estimation performance, require no training, and show extensive applicability across datasets, but suffer from lengthy inference times.

In this paper, we present OptFlow, a fast optimization-based scene flow estimation method. Without relying on learning or any labeled datasets, OptFlow achieves state-of-the-art performance for scene flow estimation on popular autonomous driving benchmarks. It integrates a local correlation weight matrix for correspondence matching, an adaptive correspondence threshold limit for nearest-neighbor search, and graph prior rigidity constraints, resulting in expedited convergence and improved point correspondence identification. Moreover, we demonstrate how integrating a point cloud registration function within our objective function bolsters accuracy and differentiates between static and dynamic points without relying on external odometry data. Consequently, OptFlow outperforms the baseline graph-prior method by approximately 20% and the Neural Scene Flow Prior method by 5%-7% in accuracy, all while offering the fastest inference time among all non-learning scene flow estimation methods.

1. Introduction

Estimating 3D motion fields from dynamic scenes is a fundamental problem in computer vision with wide-ranging applications, from autonomous driving to scene parsing and object tracking. Scene flow estimation plays a vital role in many of these applications, enabling machines to perceive and navigate through their environments. For example, in autonomous driving, scene flow estimation helps vehicles understand the 3D structure and motion of the surrounding environment, which is essential for making safe and informed decisions. Similarly, in robotics, scene flow estimation assists robots in navigating through complex environments by providing a 3D understanding of the scene.

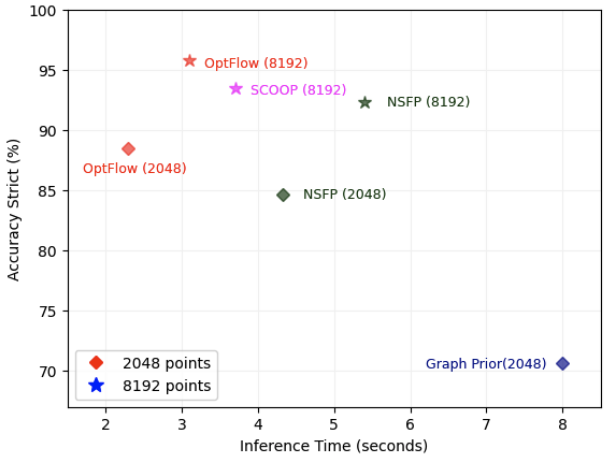

Figure 1. **Graph depicting flow accuracy Acc_5 *vs*. inference time in seconds with 2048 and 8192 points used**. OptFlow is the fastest algorithm while achieving state-of-the-art results among all the non-learning-based methods. The experiments were run on an NVIDIA Tesla T4 GPU.

Traditionally, state-of-the-art methods for scene flow estimation have relied heavily on image-based training methods using semantic depth. However, in recent years, interest in using lidar-based methods has grown. Learning-based methods [19, 30] pioneered this area, but these methods require large annotated datasets for training, which can be difficult and expensive to obtain in real-life scenarios. Even though many driving datasets [4, 5, 10] are available, obtaining annotations for ground truth flow vectors can be a challenge.

This challenge has led to the development of large synthetic datasets, such as FlyingThings3D [20], which have emerged as an alternative means of training or pre-training models. However, a significant domain gap often exists

between synthetic and real-world datasets. This gap has given rise to self-supervised scene flow estimation models [21, 31], which reduce the domain gap by training the models on non-annotated datasets. However, these models often suffer from slow convergence and require a considerable amount of training time.

Optimization-based methods for scene flow estimation [17,22] optimize flow for each point cloud pair without using training data. However, their high accuracy is often accompanied by long processing times, which limits their practicality in some applications.

To address this issue, we present **OptFlow**, a fast non-learning optimization-based method. We introduce novel concepts such as the local correlation weight matrix, integrated ego-motion compensation, and adaptive max correspondence threshold limit, which significantly improves the convergence speed of our optimization method. Our method improves accuracy on real-world autonomous driving benchmark datasets by at least 20% over the baseline method [22] and is competitive with the current state-of-the-art methods.

Our work makes the following contributions:

1. Point correspondence matching by incorporating a local correlation weight matrix for the target point cloud in the objective function. This helps with better aligning associated points and producing more accurate results.

2. An adaptive maximum correspondence threshold, which reduces noisy correspondences and further improves the quality of the estimates.

3. An intrinsic point cloud matching transformation function based on ICP. This improves our flow estimates, increases the convergence speed, and helps distinguish static points from dynamic points.

2. Related Work

Non-learning-based methods A method revolving around the analysis of a sequence of stereo images [27] first pioneered the field of scene flow estimation. They first calculated the 2D optical flow between each pair of stereo images, which gives the motion of pixels in the image plane. They then used the epipolar geometry between the stereo pairs to back-project the 2D optical flow to the 3D scene flow. Another non-learning-based iterative method is the Non-rigid Iterative Closest Point (NICP) work by Amberg *et al*. [1]. The goal of this method was to register the point cloud representation of the scene to the current frame. However, this method was sensitive to initialization and not suitable for large-scale differences between the template and the scanned mesh.

Supervised learning-based methods The most prevalent algorithms in learning-based approaches for scene flow estimation involve training a flow regressor model, typically a neural network, to compute flow vectors between point clouds in the ambient 3D space [18, 19, 24, 25, 28–30]. The introduction of deep learning architectures for point cloud processing, such as PointNet [24] and its hierarchical extension PointNet++ [25] by Qi *et al*., has inspired numerous works on scene flow estimation. Learning based state-of-the-art models have proposed innovative network architectures that build on these foundational models to estimate scene flow directly from point clouds, such as FlowNet3D [19].

FlowNet3D [19] employed PointNet++ for feature encoding of points and introduced a flow embedding layer that learned to aggregate geometric similarities and spatial relations of points for motion estimation. Wang *et al*. [30] subsequently enhanced FlowNet3D by incorporating geometric constraints in the form of point-to-plane distance and angular alignment, resulting in FlowNet3D++. Numerous models have since emerged, focusing on improving point cloud feature extraction and adding a flow refinement module on top of it.

One such model is FESTA [29], which introduced spatial and temporal features through an attention mechanism that effectively captures the temporal relationships between consecutive point clouds and the spatial relationships. BiPointFlowNet [7], another notable model, leverages both forward and backward correspondence information in a bidirectional approach to better handle occlusions and out-of-view regions. The FLOT3D [23] method introduced a novel frustum-based optimization method for scene flow estimation, leveraging learned optimization techniques and applying a series of transformations in a coarse-to-fine manner. Lastly, FH-Net [8] presents a fast hierarchical network that addresses the computational challenges associated with dense 3D scene flow estimation, utilizing a hierarchical structure and lightweight building blocks to exploit both local and global features.

Self-supervised learning-based methods Recent research has seen a surge of interest in self-supervised learning as a potential solution for enhancing scene flow estimation from point cloud data [3, 13, 14, 16, 21, 26, 31] and monocular images [6, 11, 32]. PointPwcNet [31] introduced the concept of cycle consistency loss, which in turn inspired Mittal et al. [21] to apply a similar approach for point cloud correspondence identification. They innovatively merged cycle consistency loss with nearest neighbor loss to tackle the challenge of scene flow estimation. PointPwcNet [31] made use of the Chamfer Distance [9], smoothness constraints, and Laplacian regularization to train scene flow in a self-supervised way. SLIM [3] solved scene flow esti-

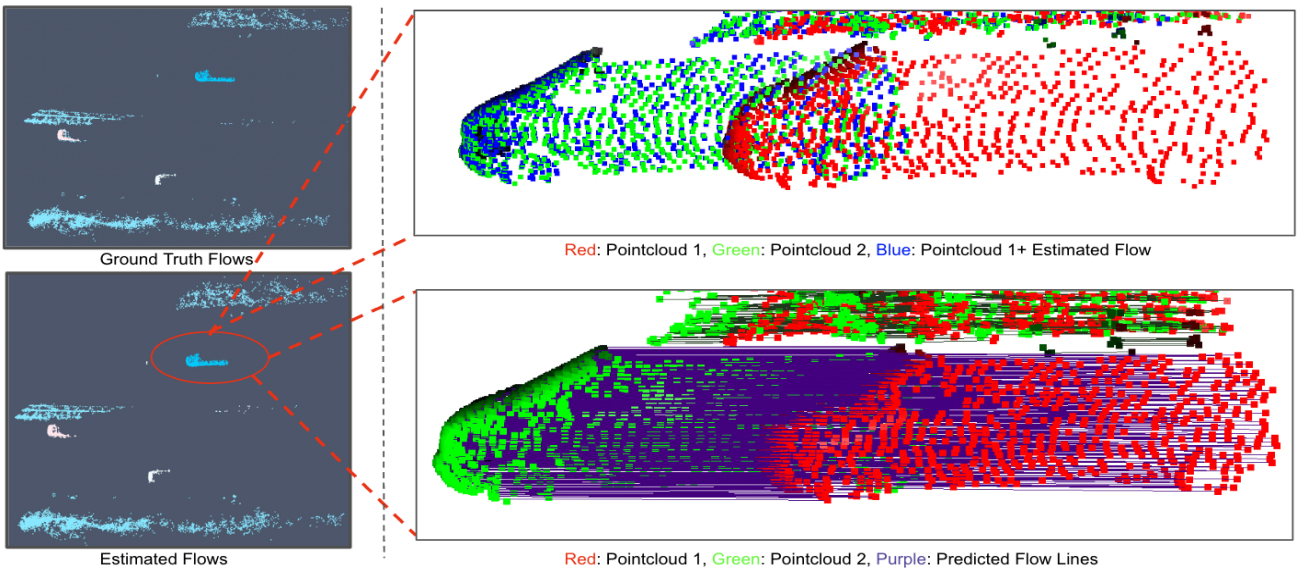

Figure 2. **Visualization of predicted flows on the KITTI Dataset. Left:** The color-coded map illustrates a comparison between ground truth flow vectors and our predicted flow values. **Top Right**: The point cloud P_{T-1} is depicted in red, point cloud P_T in green, and the translated point cloud $(P_{T-1}+F)$ in blue. Note the proximity of the blue points to the ground truth green points, indicating high prediction accuracy. **Bottom Right**: The visualization of predicted flow lines is presented.

mation while simultaneously classifying motion segmentation. Flowstep3d [12], on the other hand, employed a soft point matching module to calculate pairwise matches between points in the source and target point clouds. Based on this, we introduced a local correlation weight matrix in our algorithm to improve soft correspondences within our objective function. These strategies have demonstrated considerable promise in augmenting scene flow estimation and can be customized to various datasets while preserving generalizability. Nevertheless, self-supervised learning-based methods still demand considerable training data to reach satisfactory learning outcomes, and the associated training cost can often be exorbitantly high.

Optimization based methods Optimization-based methods present a distinct class of non-learning-based approaches for estimating scene flow. Uniquely, these methods circumvent the need for model training, opting instead for complete runtime flow optimization. Such an approach was notably employed by authors in [22], where they encoded the prior of the flow to be as rigid as possible by minimizing the graph laplacian defined over the source points. A subsequent work by Argo AI [17] replaced the explicit graph with a neural prior using a coordinate-based MLP, thereby implicitly regularizing the optimized flow field. Recently, the scene flow estimation method, SCOOP [13], was introduced, which innovatively combines pre-training on a subset of data to learn soft correspondences and secure initial flows, followed by optimization-based flow refinement steps. This hybrid approach has allowed SCOOP to deliver competitive results, using considerably less training data.

3. Method

Problem definition: Given two sets of 3D point clouds, $P_{t-1} \in \mathbb{R}^{n_1 \times 3}$ and $P_t \in \mathbb{R}^{n_2 \times 3}$, representing a dynamic scene at two different times $t-1$ and t, respectively, the task is to compute the 3D motion vector for each point in P_{t-1}.

Since the number of points in each set may be different and there may not be a one-to-one correspondence between points, we model the motion of each point in P_{t-1} using a flow vector f in R^3, and the collection of these points is called a flow field $F \in \mathbb{R}^{n_1 \times 3}$.

In our approach, we predict flow f_i for each point p_i in P_{t-1} such that the total distance between 2 sets of point clouds is minimum. Thus, we want to find an optimal set of flow vectors such that:

$$F^* = \arg\min_F \sum_{p_i \in P_{t-1}} dist(p_i + f_i, P_t), \qquad (1)$$

where $dist$ is the function which computes distance between $p_i + f_i$ and its corresponding nearest point in P_t.

3.1. Integrated Ego-motion Compensation

In real-world scenarios, LiDAR data is often captured from a moving vehicle, making ego-motion capture criti-

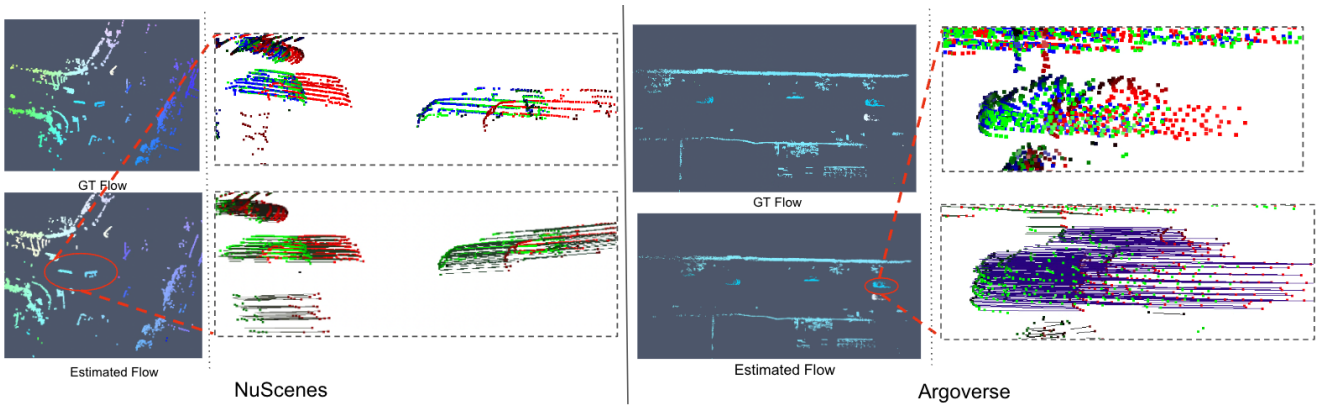

Figure 3. **Visualization of predicted flows on the nuScenes and Argoverse datasets. Left:** A color-coded map illustrates a comparison between the ground truth flow vectors and our predicted flow values. **Top Right**: The point cloud P_{T-1} is depicted in red, point cloud P_T in green, and the transformed point cloud $P_{T-1} + F$ in blue. **Bottom Right**: The visualization of predicted flow lines is presented. The nuScenes example is particularly intricate, as the ego-vehicle is positioned at a turn, resulting in angled flows. Additionally, the color-coded map illustrates the variance in flow values associated with each point.

cal for localization and motion estimation. Moreover, with the majority of points in autonomous driving datasets being static, the incorporation of ego-motion can simplify scene flow estimation by aligning the input point cloud pairs and focusing on dynamic objects. Though standard datasets such as Argoverse [5] and nuScenes [4] provide ego-motion information, this data hasn't been tapped in previous scene flow estimation research. Furthermore, errors can occur due to the absence or inaccuracy of ego-motion sensor data in certain scenarios. To diminish dependence on external odometry data, we integrate an Iterative Closest Point (ICP) [2] based transformation function, denoted as T, into our optimization process, enabling its estimation alongside flow vector estimation. At each step, we transform the current point cloud into the next coordinate frame, minimising the distance between corresponding static points.

The updated objective function is defined as:

$$F^* = \arg\min_{F,T} \sum_{p_i \in P_{t-1}} dist(Tp_i + f_i, P_t). \tag{2}$$

In addition to point cloud flow F, we also estimate the transformation T from the frame of point cloud 1 to the frame of point cloud 2. This is to delineate the transformation of static parts of the scene due to vehicle motion from the flow of dynamic objects and their associated points. In section 5.4, we demonstrate the effectiveness of this transformation in enhancing accuracy.

3.2. Local Correlation Weight Matrix

To establish the flow field, F, between two point clouds, it is essential to find correspondences between points in the source and target point clouds. Previous methods, such as [21, 22], use the 1 nearest neighbor approach to find correspondences, which can lead to sub-optimal solutions due to initial inaccuracies.

Building on the global correlation unit in FlowStep3D [12], which employs cosine similarity to find soft correlations between features, we introduce a local correlation weight matrix. For each point, p_i, in the source point cloud, P_{t-1}, transformed as $(Tp_i + f_i)$, we identify the k_{local} nearest points in the target point cloud, P_t. We then compute a similarity score, sim_{ij}, based on the exponential of the negative squared distance, d_{ij}, between point pairs:

$$sim_{ij} = e^{-d_{ij}^2}. \tag{3}$$

The correlation weight, M_{ij}, representing the confidence in the similarity between the source and target points, is calculated using sim_{ij}:

$$M_{ij} = \exp\left(\frac{sim_{ij} - 1}{\epsilon}\right). \tag{4}$$

Using these weights, we compute a weighted average of the target points $q_j \in P_t$ to determine the optimal correspondence, q_{avg_i}, for each source point, p_i:

$$q_{avg_i} = \frac{\sum_{j=1}^{k_{local}} M_{i,j} q_j}{\sum_{j=1}^{k_{local}} M_{i,j}}. \tag{5}$$

This leads to a new objective function:

$$E_{fit} = \sum_{i=1}^{n_1} ||Tp_i + f_i - q_{avg_i}||_2^2. \tag{6}$$

We use our objective function in a bidirectional manner to both sets of point clouds, to effectively align it with the

Method	Supervision	Learning Based	#points	$EPE\downarrow$	$\%_5\uparrow$	$\%_{10}\uparrow$	$Out.\downarrow$
Just Go With The Flow (JGF) [21]	*Full + Self + Self*	True	2048	0.105	46.5	79.4	-
Self Point-Flow (SPF) [15]	*Self + Self*	True	2048	0.089	41.7	75.0	-
RigidFlow [16]	*Self*	True	2048	0.117	38.8	69.7	-
SCOOP [13]	*Self*	Partial	2048	0.052	80.6	92.9	19.7
Neural Prior [17]	*Self*	False	2048	0.052	84.8	94.3	**16.4**
OptFlow (Ours)	*Self*	False	2048	**0.049**	**88.25**	**95.1**	18.2
Graph Prior [22]	*Self*	False	Full Pointcloud	0.082	84.0	88.5	-
Neural Prior [17]	*Self*	False	Full Pointcloud	0.034	92.3	96.4	**12.0**
SCOOP$^+$ [13]	*Self*	Partial	Full Pointcloud	0.039	93.6	96.5	15.2
OptFlow (Ours)	*Self*	False	Full Pointcloud	**0.028**	**96.1**	**97.8**	13.2

Table 1. **Quantitative comparison on 2048 points and full pointcloud on KITTI Dataset with 35m range set on the point cloud.** Comparison with prior work as reported in the [13]. Here EPE denotes the end-point error, $\%_5$ denotes strict accuracy,$\%_{10}$ denotes relaxed accuracy and $Out.$ denotes the percentage of points that are outliers (i.e.$EPE \geq$0.3m)

principles of Chamfer Distance [9], ensuring a more symmetrical and comprehensive evaluation of correspondences.

Note: ϵ and K_{local} are hyperparameters. While ϵ is fixed at 0.03, K_{local} depends on the density of the dataset, with larger values for sparse datasets and smaller values for dense datasets.

3.3. Adaptive Distance Threshold

While constructing the local correlation weight matrix by finding the K nearest neighbors, it is critical to eliminate outliers that could adversely affect point correspondences. Although some studies employ a fixed distance limit of 2.0m, we introduce an adaptive distance threshold that evolves during optimization.

This adaptive threshold, d_{thresh}, decreases at regular intervals of n steps, with the rationale that the flow vectors' accuracy improves with each iteration, necessitating a more restrictive threshold for better matching precision. The threshold is halved at each interval until it reaches a lower limit of 0.2m. Correlations between points exceeding this distance threshold are assigned a weight of zero in the weight matrix M, ensuring that only those within the threshold contribute to the final flow field.

In the current implementation, the interval is empirically set to 100 steps, which means that the threshold is halved after every 100 iterations.

3.4. Rigidity Constraint

Inspired by [22] and building upon the principles presented in it, we add a rigidity constraint in our optimization objective to maintain geometric coherence in the source point cloud. Specifically, we enforce local rigidity among points in close proximity within a subgraph of K_{rigid} points, mimicking the characteristics of rigid body motion.

Our rigidity constraint is enforced by minimizing the difference between the flows of each pair of points within the subgraph. The formulation leverages the graph laplacian, which implicitly captures the topological structure of the point cloud, to regularize the scene flow. Mathematically, the rigidity function E_{rigid} is expressed as:

$$E_{rigid}(f_i) = \sum_{i,j \in S} W^{ij}_{rigid} \left\| f_i - f_j \right\|_2^2, \tag{7}$$

where S is the set of edges of a subgraph G containing K_{rigid} points in the source point cloud P_{T-1}, and W^{ij}_{rigid} is a weight defined similar to sim_{ij}:

$$W^{ij}_{rigid} = e^{-d^2_{ij}}, \tag{8}$$

where d_{ij} is the distance between points p_i and p_j in subgraph G. The weight W^{ij}_{rigid} assigns higher importance to pairs of points that are closer, promoting rigidity within the local region.

Note: We use $K_{rigid} = 50$, as proposed in [22].

Combining all the proposed methods above, our final objective function becomes:

$$E_{obj} = E_{fit} + \alpha_{rigid} E_{rigid} \tag{9}$$

$$= \sum_{i=1}^{n_1} ||Tp_i + f_i - q_{avg}||_2^2 + \alpha_{rigid} \sum_{i,j \in S} W^{ij}_{rigid} \left\| f_i - f_j \right\|_2^2 \tag{10}$$

where n_1 represents all points in point cloud P_{t-1} and α_{rigid} is the weight of the rigidity loss.

4. Experiments

In this section, we evaluate OptFlow's performance on synthetic and real-life autonomous driving datasets and compare it with recent state-of-the-art methods for scene

	FlyingThings3D [20] *#Test: 2,000*				KITTI Scene Flow [10] *#Test: 50*				Argoverse Scene Flow [5] *#Test: 212*				nuScenes Scene Flow [4] *#Test: 310*			
	EPE (m)	$\%_5 \uparrow$ (%)	$\%_{10} \uparrow$ (%)	$\theta_\epsilon \downarrow$ (rad)	EPE (m)	$\%_5 \uparrow$ (%)	$\%_{10} \uparrow$ (%)	$\theta_\epsilon \downarrow$ (rad)	EPE (m)	$\%_5 \uparrow$ (%)	$\%_{10} \uparrow$ (%)	$\theta_\epsilon \downarrow$ (rad)	EPE (m)	$\%_5 \uparrow$ (%)	$\%_{10} \uparrow$ (%)	$\theta_\epsilon \downarrow$ (rad)
FlowNet3D [19]	0.134	22.64	54.17	0.305	0.199	10.44	38.89	0.386	0.455	1.34	6.12	0.736	0.505	2.12	10.81	0.620
PointPWC-Net [31]	**0.121**	29.09	**61.70**	**0.229**	0.142	29.91	59.83	0.239	0.405	8.25	25.47	0.674	0.442	7.64	22.32	0.497
JGF [21]		—			0.218	10.17	34.38	0.254	0.542	8.80	20.28	0.715	0.625	6.09	0.139	0.432
PointPWC-Net [31]		—			0.177	13.29	42.15	0.272	0.409	9.79	29.31	0.643	0.431	6.87	22.42	0.406
Non-rigid ICP [1]	0.339	14.05	35.68	0.480	0.338	22.06	43.03	0.460	0.461	4.27	13.90	0.741	0.402	6.99	21.01	0.492
Graph Prior [22]	0.259	16.30	41.60	0.369	0.093	64.76	82.13	0.137	0.257	25.26	47.50	0.467	0.288	20.19	43.59	0.337
NSFP [17]	0.234	19.16	46.74	0.341	0.050	81.68	93.19	0.133	**0.159**	38.43	63.08	**0.374**	**0.175**	35.18	63.45	0.279
OptFlow (ours)	0.224	**31.73**	57.1	0.340	0.052	**84.3**	**93.2**	**0.130**	0.20	**43.85**	**65.5**	0.39	0.216	**40.85**	**65.97**	**0.271**

Table 2. **Results on 2048 points with no point cloud range limit set.** Comparison with prior work as reported in the NSFP [17]. Top section between shows *off-the-shelf* **supervised learning methods**; the middle section shows **self-supervised learning methods**; and the bottom section shows **non-learning-based methods**. Here EPE denotes the end-point error, $\%_5$ denotes strict accuracy,$\%_{10}$ denotes relaxed accuracy and θ_ϵ denotes angle error.

flow estimation. Additionally, we compare their generalizability, speed of execution, and model complexity. Our less complex method achieves competitive performance without any annotations and training data available.

Datasets: These are the four major datasets that we evaluated our models on:

1. **FlyingThings3D**: FlyingThings3D is a large-scale synthetic dataset of random objects from the ShapeNet collection. Similar to [17, 22], we use a pre-processed dataset released by [19]. The dataset is evaluated on 2000 test samples.

2. **KITTI Scene Flow:** KITTI was designed to evaluate scene flow methods on real-world self-driving scenarios. In our experiments, we use a pre-processed dataset that was released by [22]. The dataset is split into 100 train and 50 test sample sets, with ground points filtered out.

3. **nuScenes:** nuScenes dataset is a large-scale autonomous driving dataset in urban environments. It is challenging due to the presence of a lot of dynamic objects in the scene and also because of the presence of occlusions in the LiDAR point clouds. As there are no official scene flow annotations, we use the ego-vehicle poses and 3D object tracks to create pseudo-labels as done in [17, 22]. The ground points below a certain threshold are also filtered out. The results are evaluated on 310 test samples from 150 test scenes, same as [17, 22].

4. **Argoverse:** Argoverse is another large-scale challenging autonomous driving dataset released by Argo AI. Similar to nuScenes, scene flow annotation are not provided for Argoverse and the same process has been followed to derive the pseudo labels. The results are evaluated on 212 test samples, same as [17, 22].

Metrics: To assess the effectiveness of our method, we utilized commonly used metrics (such as those in previous works [17, 19, 21, 22, 31]) which include: EPE ε (end-point error), which measures the mean absolute distance between two point clouds; Acc_5 (Accuracy Strict), which calculates the percentage of estimated flows where the EPE is less than 0.05m or the relative error E_0 is less than 5%; Acc_{10} (Accuracy Relax), which measures the percentage of estimated flows where the EPE is less than 0.1m or the relative error E_0 is less than 10%; and θ, the mean angle error between the estimated and ground-truth scene flows.

Implementation Details: Our implementation leverages PyTorch, utilizing its automatic differentiation library to optimize our objective function with the AdamW optimizer. We initialize our flow parameters, $F \in \mathbb{R}^{n_1 \times 3}$, as empty tensors. Additionally, we initialize a rotation vector, $r \in \mathbb{R}^{1\times 3}$, and a translation vector, $t \in \mathbb{R}^{1\times 3}$—components of the transformation matrix $T \in \mathbf{SE}(3)$ defined in (6)—using values derived from ICP registration. The learning rate is set at 4e-3, and the optimization process is run for 600 iterations, incorporating early stopping based on loss value. To compare with leading architectures, we sample our dataset using 2048 points, 8192 points, and the full point cloud. All experiments are executed on an NVIDIA T4 GPU.

5. Results

5.1. Comparison with different methods

As demonstrated in Table 1, we assess our method using both 2048 points and the entire point cloud limited to a depth of 35m, aligning our evaluation with parameters typically reported in the current literature. The results depicted in Table 1 are either obtained from the recently published study [13] or independently reproduced by our team.

In Table 2, our experimental setup follows the methodologies adopted by [17,22]. Here, we consider the full point

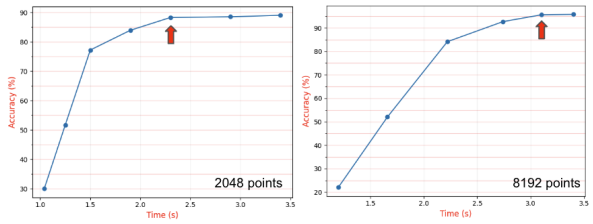

Figure 4. **Performance of OptFlow algorithm on KITTI with different iterations and time taken for them. The red arrow shows the most optimal performance.**

cloud without imposing the range constraints applied in the previous evaluation. These evaluation metrics are sourced from [17].

A thorough examination of both tables reveals that our method consistently outperforms alternative techniques across all primary datasets. Specifically focusing on non-learning-based approaches, OptFlow surpasses the performance of all comparable methods across every dataset, while also delivering the fastest inference time, as demonstrated in Figure 1.

Figures 2 and 3 provide qualitative results of our algorithm applied to the KITTI, Argoverse, and nuScenes datasets.

5.2. Inference Time analysis and tradeoff

Figure 4 presents a detailed analysis of our OptFlow model's performance over a range of timesteps on the KITTI dataset. Optimum performance is observed at the 600th iteration, with inference times of approximately 2.3 seconds for 2048 points and 3.1 seconds for 8192 points.

5.3. Evaluation on high-density point clouds

As depicted in Tables 1 and 2, we predominantly evaluated our model's performance using 2048 points. Nonetheless, it's important to note that real-world LiDAR sensor data often contain tens of thousands of points, necessitating the testing of scene flow methods on such high-density point clouds.

Our model's performance on high-density point clouds is evaluated in Figure 5, which illustrates the trade-off between accuracy and inference time as we scale the number of points within the KITTI Scene Flow dataset. For benchmarking, we've compared our method with Neural Scene Flow Prior (NSFP) [17], presently the best-performing algorithm for non-learning-based approaches.

The results clearly demonstrate that our method substantially outperforms Neural Scene Flow Prior [17] in terms of accuracy, even while operating at a faster inference speed and utilizing fewer points.

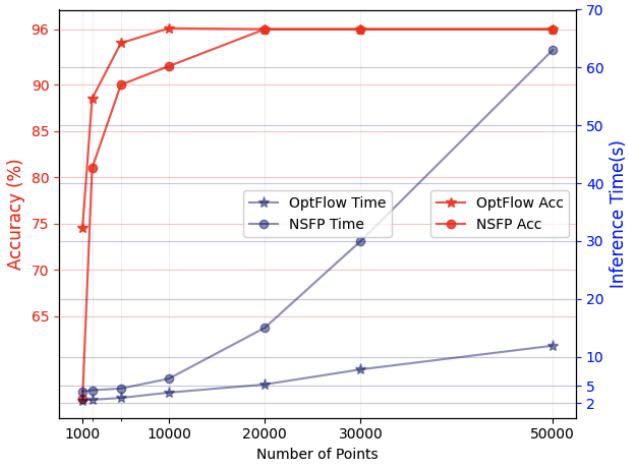

Figure 5. **Performance(Acc_5) comparison of our algorithm and NSFP on the KITTI dataset for varying point cloud densities. This shows we get around 7x speedup over NSFP [17] as the point cloud density increases.** The point clouds are processed parallelly after 8k points as discussed in sec. 1 of supplementary.

5.4. Ablation Studies

Experiment	$EPE\downarrow$	$\%_5\uparrow$
(a) W/O integrated ego-motion transformation	0.081	91.6
(b) W/O adaptive distance threshold (thresh = 2m)	0.035	91.9
(c) W/O correlation weight matrix	0.029	95.6
Our complete method (a + b + c)	**0.028**	**95.8**

Table 3. **Ablation experiment with 8192 points** evaluated on $KITTI_t$.

Tables 3 and 4 demonstrate the influence of each component presented in our paper. Table 3 quantifies the percentage change each element induces in the complete algorithm. For evaluation, we used a certain setup: considering our full method as a baseline and illustrating the impact of each constituent on the EPE and Acc_5 metrics. We carried out experiments which entailed a) the omission of integrated ego-motion transformation, thus forgoing the transformation matrix and assuming the identity matrix for T in equation 6; b) keeping a fixed distance threshold of 2m, with no adaptive adjustments; c) disregarding the correlation weight matrix, hence only the nearest neighbor in the target point-cloud was contemplated as a corresponding point for each source point; d) implementing our comprehensive algorithm with all components included.

Table 3 highlights the fact that every component makes a significant contribution, though the correlation weight matrix has a minor variance from the full method. In contrast, Table 4 shows that when we decrease the number of points,

Experiment	$EPE\downarrow$	$\%_5\uparrow$
(a) W/O integrated ego-motion transformation	0.095	84.12
(b) W/O adaptive distance threshold (thresh = 2m)	0.063	79.56
(c) W/O correlation weight matrix	0.050	84.90
Our complete method (a + b + c)	**0.049**	**88.25**

Table 4. **Ablation experiment with 2048 points** evaluated on KITTI_t.

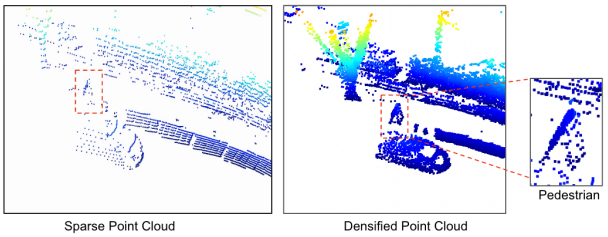

Figure 6. **Point Cloud Densification:** One example from nuScenes depicting the application of point cloud densification. Our method was able to densify a pedestrian in a sparse point cloud.

the impact of each component amplifies considerably, especially the correlation weight matrix, which increases Acc_5 by a significant 4%.

6. Applications

Densification: Our scene-flow estimation methods demonstrate such precision that they can be employed to densify successive point-cloud frames. By computing pairwise scene-flow estimates, we can project the current point-clouds - augmented with estimated flows - onto the succeeding point-cloud, and perpetuate this densification process through 'n' frames. As illustrated in Figure 6, we apply our method on the nuScenes dataset to successfully densify a pedestrian situated within an extremely sparse point cloud.

Object Annotation/Motion Segmentation: The incorporation of ego-motion compensation into our objective function facilitates accurate alignment between two input pairs of point clouds, assisting in closely aligning static objects. This alignment process diminishes the flow values of static objects, thus simplifying the differentiation between dynamic and static objects. As shown in Figure 7, motion segmentation is achieved after the ego-motion compensation transformation is applied to an input pair of point clouds. Furthermore, our algorithm lends itself to the annotation of dynamic objects in new datasets, extending its utility beyond scene flow estimation.

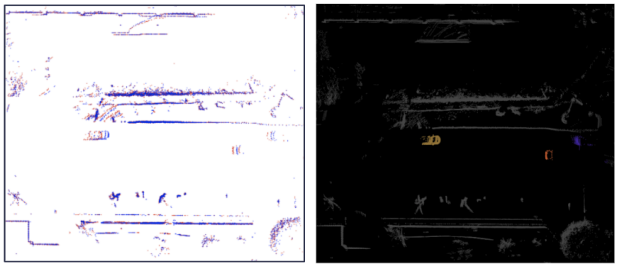

Figure 7. **Motion Segmentation(BeV):** On the left, a pair of sparse point clouds is shown in red and blue, after applying ego-motion compensation. On the right, motion is segmented out and dynamic objects are color-coded, just from the scene flow result. Points exceeding a set threshold are then marked as dynamic.

7. Limitations

As reflected in Table 1 and Figure 1, our method achieves state-of-the-art results with better inference speed amongst all non-learning-based techniques. However, the execution time may not suffice for real-time applications, such as for use in autonomous vehicles.

Another challenge lies in the need for precision in hyperparameter settings for each specific dataset. When handling a new dataset, our method requires a hyperparameter tuning algorithm to obtain peak performance. Hyperparameters, such as $\alpha_{rigidity}$ in the final objective function (9) and K_{local} in the local correlation weight matrix, are pivotal in influencing the performance across each dataset.

8. Conclusion

To conclude, our research introduces an innovative and efficient non-learning scene flow estimation method. Our method incorporates a local correlation weight matrix and an adaptive distance threshold, both instrumental in accelerating flow field convergence and enhancing correspondence accuracy.

Additionally, we proposed an intrinsic ego-motion compensation function that bolsters precision and curbs computational complexity by reducing flow value computations for static points and primarily concentrating on dynamic points. This approach culminates in state-of-the-art results in the Accuracy Strict (Acc_5) metric for all key autonomous driving datasets, and the fastest inference time among all non-learning methods.

Finally, we demonstrate the versatility of our algorithm with its applications in dynamic/static object annotation and point cloud densification.

References

[1] Brian Amberg, Sami Romdhani, and Thomas Vetter. Optimal step nonrigid icp algorithms for surface registration. 06 2007. 2, 6

[2] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(5):698–700, 1987. 4

[3] Stefan Baur, David Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1, 4, 6

[5] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8748–8757, 2019. 1, 4, 6

[6] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. pages 7062–7071, 10 2019. 2

[7] Wencan Cheng and Jong Hwan Ko. Bi-pointflownet: Bidirectional learning for point cloud based scene flow estimation. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVIII*, pages 108–124. Springer, 2022. 2

[8] Lihe Ding, Shaocong Dong, Tingfa Xu, Xinli Xu, Jie Wang, and Jianan Li. Fh-net: A fast hierarchical network for scene flow estimation on real-world point clouds. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIX*, pages 213–229. Springer, 2022. 2

[9] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2, 5

[10] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1, 6

[11] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. In *CVPR*, 2020. 2

[12] Yair Kittenplon, Yonina C Eldar, and Dan Raviv. Flowstep3d: Model unrolling for self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4114–4123, 2021. 3, 4

[13] Itai Lang, Dror Aiger, Forrester Cole, Shai Avidan, and Michael Rubinstein. SCOOP: Self-Supervised Correspondence and Optimization-Based Scene Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 5, 6

[14] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15577–15586, June 2021. 2

[15] Ruibo Li, Guosheng Lin, and Lihua Xie. Self-point-flow: Self-supervised scene flow estimation from point clouds with optimal transport and random walk. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15577–15586, 2021. 5

[16] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16959–16968, June 2022. 2, 5

[17] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34:7838–7851, 2021. 2, 3, 5, 6, 7

[18] Haisong Liu, Tao Lu, Yihui Xu, Jia Liu, Wenjie Li, and Lijun Chen. Camliflow: bidirectional camera-lidar fusion for joint optical flow and scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5791–5801, 2022. 2

[19] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds, 2019. 1, 2, 6

[20] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4040–4048, 2016. 1, 6

[21] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 4, 5, 6

[22] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. In *2020 international conference on 3D vision (3DV)*, pages 261–270. IEEE, 2020. 2, 3, 4, 5, 6

[23] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Flot: Scene flow on point clouds guided by optimal transport. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pages 527–544. Springer, 2020. 2

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2

[25] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[26] Yaqi Shen, Le Hui, Jin Xie, and Jian Yang. Self-supervised 3d scene flow estimation guided by superpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5271–5280, June 2023. 2

[27] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 722–729. IEEE, 1999. 2

[28] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What matters for 3d scene flow network. In *European Conference on Computer Vision*. Springer, 2022. 2

[29] Haiyan Wang, Jiahao Pang, Muhammad A Lodhi, Yingli Tian, and Dong Tian. Festa: Flow estimation via spatial-temporal attention for scene point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14173–14182, 2021. 2

[30] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020. 1, 2

[31] 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 *European Conference on Computer Vision*, pages 88–107. Springer, 2020. 2, 6

[32] Yuliang Zou, Zelun Luo, and Jia-Bin Huang. Df-net: Unsupervised joint learning of depth and flow using cross-task consistency. In *European Conference on Computer Vision*, 2018. 2