SCOOP: Self-Supervised Correspondence and Optimization-Based Scene Flow

Itai Lang[1,2*] Dror Aiger[2] Forrester Cole[2] Shai Avidan[1] Michael Rubinstein[2]
[1]Tel Aviv University [2]Google Research
{itailang@mail, avidan@eng}.tau.ac.il {aigerd, fcole, mrub}@google.com

Abstract

Scene flow estimation is a long-standing problem in computer vision, where the goal is to find the 3D motion of a scene from its consecutive observations. Recently, there have been efforts to compute the scene flow from 3D point clouds. A common approach is to train a regression model that consumes source and target point clouds and outputs the per-point translation vector. An alternative is to learn point matches between the point clouds concurrently with regressing a refinement of the initial correspondence flow. In both cases, the learning task is very challenging since the flow regression is done in the free 3D space, and a typical solution is to resort to a large annotated synthetic dataset.

We introduce SCOOP, a new method for scene flow estimation that can be learned on a small amount of data without employing ground-truth flow supervision. In contrast to previous work, we train a pure correspondence model focused on learning point feature representation and initialize the flow as the difference between a source point and its softly corresponding target point. Then, in the run-time phase, we directly optimize a flow refinement component with a self-supervised objective, which leads to a coherent and accurate flow field between the point clouds. Experiments on widespread datasets demonstrate the performance gains achieved by our method compared to existing leading techniques while using a fraction of the training data. Our code is publicly available[1].

1. Introduction

Scene flow estimation [27] is a fundamental problem in computer vision with various use-cases, such as autonomous driving, scene parsing, pose estimation, and object tracking, to name a few. Given two consecutive observations of a 3D scene, the aim is to compute the dynamics of the scene between the observations. Scene flow prediction based on 2D images has been thoroughly investigated in the literature [17, 19, 28, 32, 33]. However, in light of the recent proliferation of 3D sensors, such as LiDAR, there is a surge of interest in scene flow methods that operate directly on the 3D data [10, 13, 16, 21, 34].

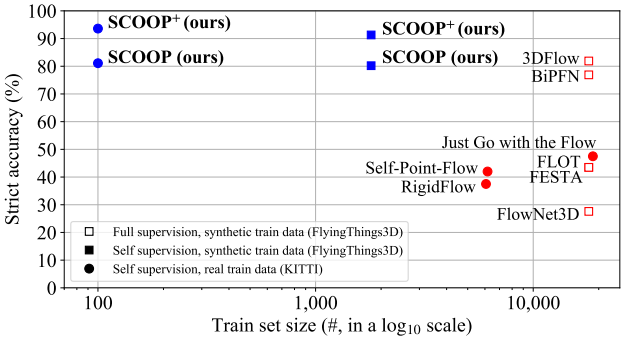

Figure 1. **Flow accuracy on the KITTI benchmark *vs*. the train set size.** Our method is trained on one or two orders of magnitude less data while surpassing the performance of the competing techniques [4, 13, 14, 16, 21, 24, 30, 31] by a large margin. Please see Table 1 for the complete details of the evaluation settings.

Liu *et al*. [16] were among the first to pursue this research avenue. They proposed FlowNet3D, a fully-supervised neural network that learned to regress the flow between 3D point clouds and showed remarkable performance improvement over image-based techniques [1, 19, 29]. Since their method required ground-truth flow annotations, which are scarce for real-world data, they turned to training on a large synthetic dataset that compromised the generalization capability to real-world LiDAR data.

Follow-up works devised self-supervised learning schemes [13, 21] and narrowed the domain gap by training on unannotated LiDAR point cloud pairs. However, similar to Liu *et al*. [16], they used a regression approach in which the model should learn to compute the flow in the *free 3D space*. This task is extremely challenging, given the irregular nature of point clouds, and requires a large amount of training data for the network to converge.

In another line of work [8, 11, 24], researchers leveraged point cloud correspondence for scene flow prediction. In this approach, the flow is computed as the translation of a point in the first point cloud (source) to its softly corre-

[1]https://github.com/itailang/SCOOP
*The work was done during an internship at Google Research.

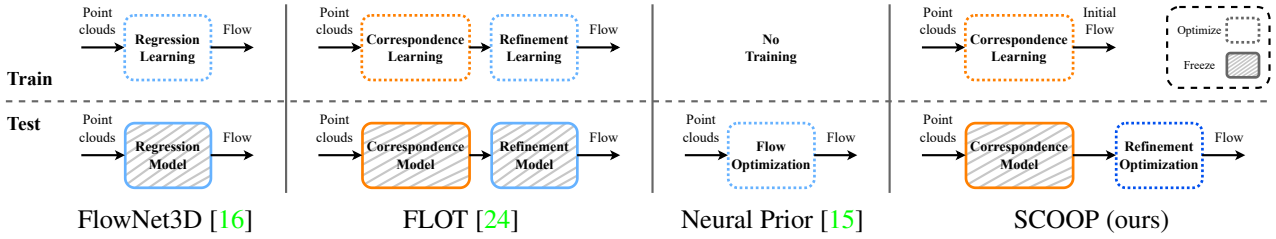

Figure 2. **Comparison of scene flow approaches.** Given a pair of point clouds, FlowNet3D [16] learns to regress the flow in the free 3D space, and the trained model is frozen for testing. FLOT [24] concurrently trains two network components: one that computes point correspondence and another that regresses a correction to the resulting correspondence flow. Neural Prior [15] optimizes the flow between the point clouds from scratch without learning. In contrast to previous work, we take a hybrid approach. We train a *pure correspondence model without flow regression*, which serves for flow initialization. Then, we *directly optimize only the flow refinement* at the test-time.

sponding point in the second one (target). The softly corresponding point is a weighted sum of target points based on point similarity in a learned latent space. Thus, rather than the challenging regression problem in the 3D ambient space, the flow estimation task boils down to point feature learning and is reduced to the convex combination space [26] of existing target points. However, to relax this constraint, another network component is trained to regress flow corrections. The joint training of point representation and flow refinement burdens the learning process and retains the reliance on large datasets with flow supervision.

Another emerging approach is an optimization-only flow computation [15, 23]. In this case, no training data is involved, and the flow is optimized at run-time for each scene separately. Despite the high accuracy such a dedicated optimization achieves, it requires a long processing time.

We present SCOOP, a hybrid flow estimation method that can be learned from a small amount of training data. SCOOP consists of two parts: a self-supervised neural network for point cloud correspondence and a direct flow refinement optimization module. During the training phase, the network learns to extract point features for soft point matches, which initialize the flow between the point clouds. In contrast to previous work, our network is focused on learning just the point embeddings, allowing its training on a very small dataset, as shown in Figure 1. Additionally, we consider the confidence of the network in the computed correspondences to guide the learning process better.

Then, instead of training another network for regressing flow updates, we define an optimization problem and directly optimize residual flow refinement vectors at runtime. The optimization objective encourages a coherent flow field while retaining the translated source points close to the target point cloud. Our design choices improve the accuracy compared to learning-based methods and reduce the processing time with respect to the optimization-only approach [15, 23]. For both correspondence learning and refinement optimization, we use a self-supervised distance objective and a smoothness prior instead of ground-truth flow labels. Figure 2 presents the difference between our approach and leading previous ones.

In summary, we propose a hybrid flow prediction approach for point clouds based on self-supervised correspondence learning and direct run-time residual flow optimization. Using well-established datasets in the scene flow literature, we show that our approach yields clear performance improvement over existing state-of-the-art methods while using a fraction of the training data and without employing any ground-truth flow supervision.

2. Related Work

Flow regression. A common approach for scene flow estimation on point clouds is to train a flow regression model [4, 10, 16, 31, 34]. It is a neural network that computes the flow vectors between the point clouds in the ambient 3D space. Liu *et al*. [16] proposed FlowNet3D, which encoded the point clouds into a latent space, mixed point features with a flow embedding layer, and regressed the scene flow by decoding the mixed point features. FlowNet3D was trained in a fully-supervised manner, using an l_2 loss with respect to ground-truth flow annotations.

Liu *et al*. [16] inspired a line of follow-up works [4, 13, 21, 30, 31]. Wang *et al*. [31] added spatial and temporal attention layers to FlowNet3D's architecture. In Bi-PointFlowNet [4], the authors propagated features from each point cloud bidirectionally, augmenting the point feature representation. Mittal *et al*. [21] discarded flow supervision by utilizing a self-supervised nearest neighbor loss and cycle consistency between the forward and reveres scene flows, and Li *et al*. [13, 14] extracted flow labels for training from the data itself. Similar to the latter methods, we also refrain from ground-truth flow supervision in our training scheme. However, rather than flow regression, we base our technique on soft point matches in the scene, which simplifies the flow estimation problem.

Point cloud correspondence. Finding correspondences is widely applied to various vision tasks [12, 24, 35, 37].

Several methods have been proposed for dense mapping between non-rigid point cloud shapes [7, 9, 12, 36]. Recently, Lang *et al*. [12] suggested constructing one point cloud by the other using latent space similarity and the point coordinates themselves rather than regressing the corresponding point cloud [9, 36]. Inspired by Lang's work, we do not use flow regression in our model and concentrate the learning process on point feature representation. However, while Lang *et al*. operated on complete shapes with one-to-one correspondence, our method accommodates scenes with partial objects where a perfect match may not exist.

Researchers have taken the correspondence approach to the scene flow problem as well [8, 11, 24]. FLOT [24] computed an optimal transport plan that served for an initial flow between the point clouds and further regressed flow refinement with a series of learned convolutions. Our work builds on FLOT but differs from it in three main aspects. First, we exclude flow regression from our training scheme and instead apply direct run-time optimization to refine the initial correspondence-based flow. Second, we use the model's confidence in the computed point matches to improve the point feature learning. Third, we do not use any ground-truth flow annotations, neither for the correspondence training nor for the refinement optimization, whereas FLOT relies on fully-supervised scene flow data.

Optimization-based scene flow. Pontes *et al*. [23] suggested a scene flow estimation technique that does not involve learning. Instead, the flow was optimized completely at run-time, such that the warped source is close to the target point cloud while demanding the flow to be "as-rigid-as-possible". Pontes *et al*. encoded this prior by minimizing the graph Laplacian defined over the source points. In follow-up work [15], the explicit graph was replaced by a neural prior, which implicitly regularized the optimized flow field. In contrast to these papers, we initialize the flow with a learned correspondence model and optimize only the residual flow refinement at run-time.

3. Method

A point cloud is a set of unordered 3D points $X \in \mathbb{R}^{n\times 3}$, where n is the number of points. Given a pair of point clouds of a scene, denoted as $X, Y \in \mathbb{R}^{n\times 3}$ and referred to as source and target, respectively, our goal is to estimate a flow field $F^* \in \mathbb{R}^{n\times 3}$ describing the per-point motion from X to Y.

We tackle this problem via self-supervised soft correspondence learning between the two point clouds and a direct flow refinement optimization. An overview of the method is shown in Figure 3. First, a deep neural network is used to extract point features. Then, we calculate a matching cost between points in the learned feature space. Based on this cost, we solve an optimal transport problem to compute a softly matched target point for each source point, where the difference between the two is regarded as the correspondence-based flow. Finally, we refine the flow field by demanding its consistency across neighboring source points and obtain our estimated scene flow. In both correspondence learning and flow refinement, no ground-truth flow labels are employed.

3.1. Matching Cost

The cost of matching a point $x_i \in X$ to a point $y_j \in Y$ is determined based on the point representation learned by a deep neural network. The network consumes the raw point clouds X, Y and computes point features $\Phi_X, \Phi_Y \in \mathbb{R}^{n\times d}$, where d is the per-point feature dimension. The network's architecture is based on PointNet++ [25]. Its details are given in the supplemental material.

Inspired by previous work [12, 13, 24], we first compute the cosine similarity in the learned feature space:

$$S_{ij} = \frac{\Phi_X^i \cdot (\Phi_Y^j)^\top}{||\Phi_X^i||_2 ||\Phi_Y^j||_2}, \tag{1}$$

where $\Phi_X^i, \Phi_Y^j \in \mathbb{R}^d$ are the i'th and j'th rows of Φ_X and Φ_Y, respectively. Then, the cost is set to

$$C_{ij} = 1 - S_{ij} \tag{2}$$

for points with a Euclidean distance less than 10 meters and to ∞ otherwise to avoid flow between points too far apart.

3.2. Soft Correspondence

Finding correspondence between the source and target point clouds can be modeled as an optimal transport problem, where each source point is assigned with a mass $\frac{1}{n}$ that is transported to the target points [13, 24]. Similar to FLOT [24], we use the relaxed transport problem:

$$\begin{aligned} T^* = &\underset{T\in\mathbb{R}_+^{n\times n}}{\operatorname{argmin}} \sum_{ij} (C_{ij}T_{ij} + \epsilon T_{ij}(\log T_{ij} - 1)) \\ &+ \lambda(\mathsf{KL}(T1_n, \frac{1}{n}1_n) + \mathsf{KL}(T^\top 1_n, \frac{1}{n}1_n)), \end{aligned} \tag{3}$$

where $C_{ij} \geq 0$ is the matching cost from Equation 2 and $T_{ij} \geq 0$ is the amount of mass transported between points. The parameters $\epsilon, \lambda \geq 0$ control the relaxation of the problem. $1_n \in \mathbb{R}^n$ is a vector with all entries equal 1. KL is the Kullback-Leibler divergence used for soft preservation of the transported mass between the point clouds.

The second term in the summation operation in Equation 3 is an entropic regularization, which enables solving the problem efficiently by the Sinkhorn algorithm [5, 6]. We use this algorithm to estimate the optimal transport matrix T^* from C to represent the soft correspondence between the point clouds. The complete derivation of the transport problem and the Sinkhorn algorithm's details are given in the supplementary material.

3.3. Correspondence-Based Flow

We leverage the optimal transport plan T^* to compute correspondence weights for the source and target points for an initial estimate of the scene flow. Different from FLOT [24], which includes all the target points as candidates for each source point, we consider only target points with maximal transport amount from the source point. This design choice focuses our flow estimation pipeline on the most relevant target candidates and improves the method's results.

For a point $x_i \in X$, the matching weights are calculated as follows:

$$w_{ij} = \frac{e^{T^*_{ij}}}{\sum_{l \in \mathcal{N}_Y(x_i)} e^{T^*_{il}}}, \tag{4}$$

where $\mathcal{N}_Y(x_i)$ is a neighborhood containing the k_s indices of the $\{y_j\}$ points with the top mass transport $\{T^*_{ij}\}$. The softly corresponding point $\hat{y}_{x_i}$ to x_i is:

$$\hat{y}_{x_i} = \sum_{j \in \mathcal{N}_Y(x_i)} w_{ij} y_j, \tag{5}$$

and the initial estimated flow for the point x_i is:

$$f_i = \hat{y}_{x_i} - x_i. \tag{6}$$

Note that if we define $\widehat{T}^*_{ij} = w_{ij}$ for $j \in \mathcal{N}_Y(x_i)$ and 0 otherwise, we get the initial flow field as:

$$F = \widehat{T}^* Y - X = \widehat{Y} - X, \tag{7}$$

where $\widehat{Y} \in \mathbb{R}^{n \times 3}$ contains the points $\{\hat{y}_{x_i}\}$.

3.4. Training Objective

To learn point representation suitable for scene flow without ground-truth supervision, we apply the flowing loss terms. First, for a tractable flow estimation, we would like each softly corresponding point $\hat{y}_{x_i}$ to have a nearby target point y_j. It may be achieved by the nearest-neighbor distance term, as done by Mittal *et al*. [21]:

$$\mathcal{D} = \frac{1}{|X|} \sum_{x_i \in X} \min_{y_j \in Y} ||\hat{y}_{x_i} - y_j||_2^2. \tag{8}$$

However, the correspondence quality for the source points can vary. For example, points on a flat region will have less distinctive correspondences than points with geometrically unique features. Thus, we augment the distance term in Equation 8 with the matching confidence of each point.

The confidence measure is based on the correspondence similarity that we define as:

$$s_{x_i} = \sum_{j \in \mathcal{N}_Y(x_i)} w_{ij} S_{ij}. \tag{9}$$

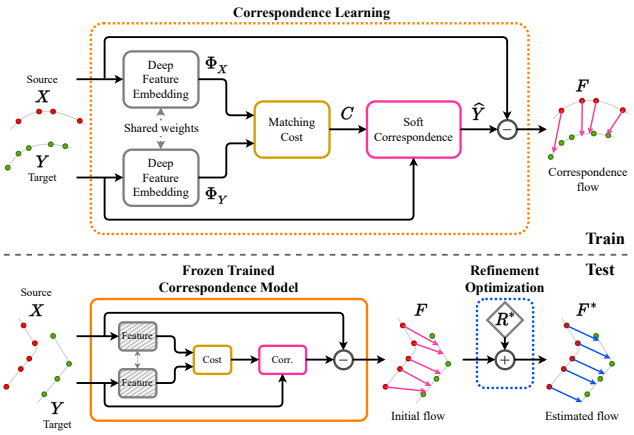

Figure 3. **The proposed method.** SCOOP includes two components: a learned point cloud correspondence model and a flow refinement module. The model learns deep point embeddings Φ_X, Φ_Y to establish soft point matches based on a matching cost C in the latent space. The initial flow F from the training phase is the difference between the softly corresponding point cloud $\widehat{Y}$ and the source point cloud X. At the test-time, we freeze the trained model and optimize a residual flow refinement R^* to produce a smooth and consistent scene flow F^* between the point clouds.

The value of s_{x_i} is in the range $[-1, 1]$. To get a confidence value between 0 and 1, we trim the negative values, set the matching confidence of x_i to be $p_{x_i} = \max(s_{x_i}, 0)$, and use p_{x_i} to define our confidence-aware distance loss:

$$\mathcal{L}_{dist} = \frac{1}{|X|} \sum_{x_i \in X} p_{x_i} \min_{y_j \in Y} ||\hat{y}_{x_i} - y_j||_2^2. \tag{10}$$

The loss term $\mathcal{L}_{dist}$ can be minimized by either minimizing p_{x_i} or the distance between $\hat{y}_{x_i}$ and its nearest neighbor $y_j \in Y$. To avoid the degenerate solution of $p_{x_i} = 0$ for all $x_i \in X$, we add a confidence loss term:

$$\mathcal{L}_{conf} = \frac{1}{|X|} \sum_{x_i \in X} 1 - p_{x_i}, \tag{11}$$

which penalizes the degenerate solution.

Additionally, to preserve the geometric structure of the source point cloud, we would like the flow field to be smooth. That is, neighboring source points should have a similar flow prediction. Thus, we regularize the learning process with a flow smoothness loss [11]:

$$\mathcal{L}_{flow} = \frac{1}{|X| k_f} \sum_{x_i \in X} \sum_{l \in N_X(x_i)} ||f_i - f_l||_1, \tag{12}$$

where $N_X(x_i)$ is the Euclidean neighborhood of x_i in $X \backslash x_i$ of size k_f. The overall training objective is:

$$\mathcal{L}_{total} = \mathcal{L}_{dist} + \alpha_{conf} \mathcal{L}_{conf} + \alpha_{flow} \mathcal{L}_{flow}, \tag{13}$$

where α_{conf} and α_{flow} are hyperparameters, balancing the contribution of the different loss terms.

3.5. Flow Refinement Optimization

The advantage of the correspondence-based flow, presented in Equation 7, is that the softly matching points are in the vicinity of the surface of objects in the target scene. However, it limits the flow to the convex hull [26] of points in the target point cloud. We enable the flow to deviate from this constraint by a flow refinement optimization step at runtime.

Instead of training an additional neural network part to regress flow corrections, as done by Puy *et al*. [24], we *directly* optimize a flow refinement component $R^* \in \mathbb{R}^{n\times 3}$ using the self-supervised distance and smoothness losses defined in Equations 10 and 12, respectively. An illustration of these losses is depicted in Figure 4.

The optimization problem for the flow refinement takes the form:

$$R^* = \underset{R\in\mathbb{R}^{n\times 3}}{\operatorname{argmin}} \frac{1}{|X|}\sum_{x_i\in X} \min_{y_j\in Y} p_{x_i}||x_i + (f_i + r_i) - y_j||_2^2 + \lambda_{flow}\frac{1}{|X|k_f}\sum_{x_i\in X}\sum_{l\in N_X(x_i)} ||(f_i + r_i) - (f_l + r_l)||_1, \tag{14}$$

where $r_i \in R$ is the flow refinement for point x_i, and the refined scene flow is $F^* = F + R^*$. Our flow refinement module further preserves the structure of the source point cloud, where the target points $\{y_j\}$ are used as anchors to guide the refined flow and keep the proximity to the underlying target surface.

4. Experiments

In this section, we evaluate SCOOP's performance using widely spread datasets and compare it with recent state-of-the-art (SOTA) works on scene flow estimation. Additionally, we demonstrate the influence of the flow refinement module, analyze the performance and run-time duration, and verify our design choices with an ablation study.

4.1. Experimental Setup

Datasets. We adopt two common datasets in the scene flow literature, FlyingThings3D [18] and KITTI [19, 20]. Originally, these benchmarks did not include point cloud data. They were processed to a point cloud format by Liu *et al*. [16] and denoted as FT3D_o and KITTI_o, respectively.

FT3D_o is a large-scale synthetic dataset with 18,000/2,000 train/validation scene examples of randomly moving objects from the ShapeNet collection [3]. Each example contains a pair of point clouds and ground-truth flow vectors. Since the objects' motion is randomized, they may appear or disappear from the view of the scene and create occlusions. The dataset also includes a mask for points whose flow is invalid due to occlusions.

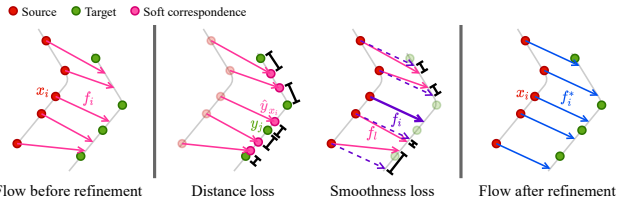

Figure 4. **Illustration of the flow refinement objective.** The initial flow $\{f_i\}$ stems from the translation of the source points $\{x_i\}$ (red) to their softly corresponding ones $\{\hat{y}_{x_i}\}$ (magenta). The flow is refined with a distance loss that keeps the proximity of the translated points to the target points $\{y_j\}$ (green) and a smoothness loss that encourages similar flow vectors (dashed purple) for neighboring points. The optimization process results in a flow field $\{f_i^*\}$ (blue) that preserves the structure of the source point cloud and warps it close to the implicit surface of the target point cloud.

The KITTI_o dataset contains 150 real-world LiDAR scenes. Every scene includes source and target point clouds with flow annotations for the source points. Ground points are removed, and the source points are considered to have a valid flow [16]. KITTI_o was further split by Mittal *et al*. [21] into sets of 100 and 50 examples, marked as KITTI_v and KITTI_t, respectively, for fine-tuning experiments. Li *et al*. [13] also built a large unlabeled LiDAR dataset for self-supervised learning on real-world data. They took raw LiDAR scans from the KITTI scenes [19,20], disjoint from the KITTI_o data, and created a training set of 6,068 instances denoted as KITTI_r.

Evaluation metrics. We use well-established evaluation metrics from previous works [16, 21, 24]: End-Point-Error $EPE\,[m]$, Strict Accuracy $AS\,[\%]$, Relaxed Accuracy $AR\,[\%]$, and Outliers $Out.\,[\%]$. These metrics are based on the point error e_i and the relative error e_i^{rel}:

$$e_i = ||f_i^* - f_i^{gt}||_2, \quad e_i^{rel} = \frac{||f_i^* - f_i^{gt}||_2}{||f_i^{gt}||_2}, \tag{15}$$

where f_i^* and f_i^{gt} are the predicted and ground-truth flow for point x_i, respectively. The EPE is the average point error, measured in meters; AS is the percentage of points whose $e_i < 0.05\,[m]$ or $e_i^{rel} < 5\%$; AR is the percentage of points for which $e_i < 0.1\,[m]$ or $e_i^{rel} < 10\%$; and $Out.$ is the percentage of points with $e_i > 0.3\,[m]$ or $e_i^{rel} > 10\%$.

Implementation details. SCOOP is implemented in PyTorch [22], where the publicly available PointNet++ [25] implementation is adapted for our point feature embedding. The model is trained on $n = 2{,}048$ points, sampled at random from the source and target point clouds of the scene examples. Only the 3D coordinates of the points are used as input to the model. The parameters ϵ and λ from Equation 3 are defined as learnable variables and optimized as part of the learning process. The point feature

Method	Supervision	Train data	Test data	$EPE\downarrow$	$AS\uparrow$	$AR\uparrow$	$Out.\downarrow$
FlowNet3D [16]	*Full*	$FT3D_o$ (18,000)	$KITTI_o$	0.173	27.6	60.9	64.9
FLOT [24]	*Full*	$FT3D_o$ (18,000)	$KITTI_o$	0.107	45.1	74.0	46.3
FESTA [31]	*Full*	$FT3D_o$ (18,000)	$KITTI_o$	0.094	44.9	83.4	-
3DFlow [30]	*Full*	$FT3D_o$ (18,000)	$KITTI_o$	0.073	81.9	89.0	26.1
BiPFN [4]	*Full*	$FT3D_o$ (18,000)	$KITTI_o$	0.065	76.9	90.6	26.4
SCOOP (ours)	*Self*	$FT3D_o$ (1,800)	$KITTI_o$	0.063	79.7	91.0	24.4
SCOOP$^+$ (ours)	*Self*	$FT3D_o$ (1,800)	$KITTI_o$	**0.047**	**91.3**	**95.0**	**18.6**
JGF [21]	*Full* + *Self* + *Self*	$FT3D_o$ (18,000) + nuScenes (700) + $KITTI_v$ (100)	$KITTI_t$	0.105	46.5	79.4	-
SPF [13]	*Self* + *Self*	$KITTI_r$ (6,068) + $KITTI_v$ (100)	$KITTI_t$	0.089	41.7	75.0	-
RigidFlow [14]	*Self*	$KITTI_r$ (6,068)	$KITTI_t$	0.117	38.8	69.7	-
SCOOP (ours)	*Self*	$KITTI_v$ (100)	$KITTI_t$	**0.052**	**80.6**	**92.9**	**19.7**
Graph Prior [23]	*Self*	N/A (optimization-only)	$KITTI_t$	0.082	84.0	88.5	-
Neural Prior [15]	*Self*	N/A (optimization-only)	$KITTI_t$	**0.036**	92.3	96.2	-
SCOOP$^+$ (ours)	*Self*	$KITTI_v$ (100)	$KITTI_t$	0.039	**93.6**	**96.5**	**15.2**

Table 1. **Quantitative comparison.** We compare scene flow evaluation metrics for different supervision settings, train data, and test data. The number of training examples is indicated in parentheses. EPE, AS, AR, and $Out.$ stand for End-Point-Error, Strict Accuracy, Relaxed Accuracy, and Outliers, respectively. The symbol $^+$ indicates an evaluation using all the points in the test point clouds, as done for the optimization-only methods [15, 23]. While other baselines apply fully-supervised training, our method yields better performance without employing ground-truth flow labels. Besides, SCOOP can be trained *only* on $KITTI_v$, with as few as 100 training instances. In contrast, alternative learning-based methods use additional training data, such as nuScenes, or a large dataset, such as $KITTI_r$. Please see further details in subsections 4.1 and 4.2.

dimension is $d = 128$. For the neighborhood sizes we use $k_s = 64$, $k_f = 32$, and the losses' hyperparameters are set to $\alpha_{conf} = 0.1$, $\alpha_{flow} = 10$.

As in previous work [14, 16, 21, 24], we evaluate SCOOP on point clouds of 2,048 points randomly sampled from the source and target. However, the full point clouds of $KITTI_o$ and $KITTI_t$ are an order of magnitude larger and have different cardinality. Thus, for a complete evaluation of the entire scene flow, we also utilize our method (denoted as SCOOP$^+$ for this case) to exploit the whole point cloud information and test the performance for the original resolution. Additional implementation details appear in the supplementary.

Baseline methods. Our method is contrasted with the recent methods FlowNet3D [16], FLOT [24], FESTA [31], 3DFlow [30], and BiPFN [4]. These methods require ground-truth flow supervision. Additionally, we compare our results with the recent self-supervised flow models of Mittal *et al*. [21] and Li *et al*. [13, 14], and the optimization-based techniques Graph Prior [23] and Neural Prior [15].

4.2. Scene Flow Results

Cross-dataset evaluation. We demonstrate the generalization power of SCOOP by training it on the $FT3D_o$ and testing its performance on $KITTI_o$. Table 1 summarises the results. The alternative methods [4, 16, 24, 30, 31] are trained on $FT3D_o$ in a fully-supervised fashion: their models are learned with the ground-truth flow information, and the points with an occluded flow are excluded from the training objective using the mask provided in the dataset.

In contrast, our model is trained in a *completely* self-supervised manner. We assume no knowledge of the flow annotations nor the occlusion mask and do not use them in our losses. Additionally, we use *only* 1,800 randomly selected examples from $FT3D_o$, while the competitors employ all 18,000 scene instances. Still, SCOOP improves over the SOTA method BiPFN [4] in all the evaluation metrics. Moreover, utilizing the entire point cloud data further increases our performance.

FlowNet3D [16], FESTA [31], 3DFlow [30], and BiPFN [4] are regression-based networks that predict the flow in the 3D ambient space. The models adapt to the characteristics of the synthetic training set, and the generalization to the real-world test data is limited. FLOT [24] leverages point cloud correspondence based on learned point features, which eases the flow prediction problem. However, it also jointly learns to regress a flow correction component that burdens the point representation training process.

SCOOP, on the other hand, is focused only on learning point embeddings suitable for scene flow estimation, guided by our self-supervised losses. It extracts discriminative features, which transfer well across the $FT3D_o$ and $KITTI_o$ datasets, and enables to compute the correspondence-based flow between the point clouds. In contrast to FLOT, we delegate the flow refinement process to the test phase, directly optimize it in a self-supervised fashion, and surpass their flow estimation performance.

Figure 5 shows a visual comparison between the results

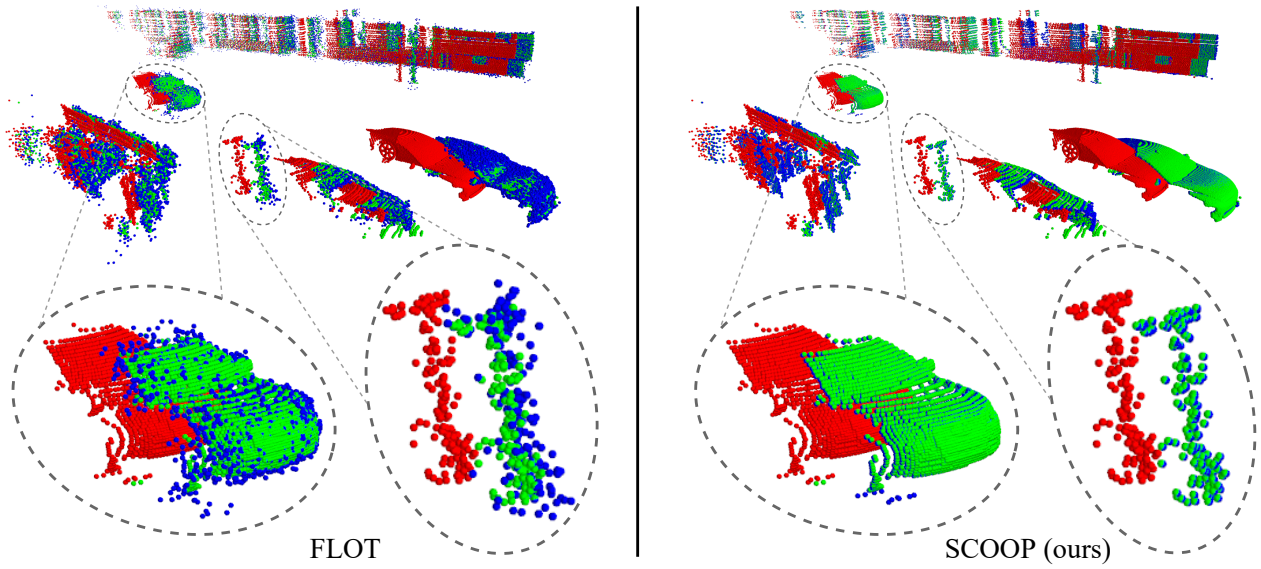

Figure 5. **Visual comparison of scene flow results for a KITTI$_o$ example scene.** The training was done on the FT3D$_o$ dataset. The source and target point clouds are shown in red and green, respectively. The warped source point cloud by FLOT [24] (left) and by our method (right) is presented in blue. While the result of FLOT deviates from the surface of the target, SCOOP preserves the source point cloud's structure and computes its accurate flow.

of FLOT and our method. The warped source point cloud by FLOT is noisy, and the structure of objects in the scene is compromised. On the contrary, SCOOP produces a coherent flow field across neighboring points, preserves their local geometry, and accurately predicts the scene flow. Additional visualizations are presented in the supplementary.

Training on a small dataset. Since our model does not include a flow regression component and has to learn only point features, it can be trained on a very limited amount of data. To demonstrate this ability, we train it *from scratch* on the 100 point cloud pairs of KITTI$_v$ and use KITTI$_t$ for testing. The results of this experiment are presented in Table 1.

Different from our work, the competing methods of Mittal *et al*. [21] and Li *et al*. [13, 14] are based on flow regression and require a large amount of training data. Mittal *et al*. utilize a fully-supervised pre-training on FT3D$_o$, and the additional outdoor flow dataset nuScenes [2], before fine-tuning on KITTI$_v$. Li *et al*. [13, 14] train their model on the large KITTI$_r$ dataset. Our SCOOP outperforms these other methods while being trained *only* on KITTI$_v$, which is almost two orders of magnitudes smaller than KITTI$_r$.

The pure optimization methods [15, 23] find the solution per scene separately, which might lead to sub-optimal local minima. In contrast, we leverage the correspondence statistics learned from the data and adapt the initial flow to the scene at hand by our residual run-time optimization. The initial correspondence flow serves as a good starting point for the optimization phase, yielding a similar or better final result compared to the optimization-only alternatives.

The influence of flow refinement. Our flow results before and after refinement are presented in Figure 6. Posing the learning part of SCOOP as a correspondence problem enables its effective training on a small dataset. However, the flow predictions from the training phase are confined to a linear combination of existing target points, which may not represent the exact flow of the scene. Moreover, wrong matches for the source points can occur and cause flow errors. In such cases, our refinement module comes into play.

Given the output flow from the trained correspondence model, the refinement module optimizes correction vectors subject to two objectives: a warped source point should be close to a target point; neighboring source points should have a similar flow. These objectives help fixing inconsistencies in the flow field and increase the flow accuracy. As seen in Figure 6, our refinement step improves the initial flow estimation and results in an accurate flow field, which is similar to the ground-truth scene flow.

4.3. Performance and Time Analysis

We analyze the performance-time trade-off in Figure 7 by recording the EPE and inference time for different methods. The measurements were done on an Nvidia Titan Xp GPU for computing the flow for complete point clouds of the KITTI$_t$ dataset.

Network-only methods [13, 14, 21] tend to be fast but with limited accuracy. Optimizing the flow prediction separately for each scene [15] results in a low EPE. However, it takes a long time. Our hybrid method bridges the trade-off gap between these two approaches. SCOOP$^+$ offers a work-

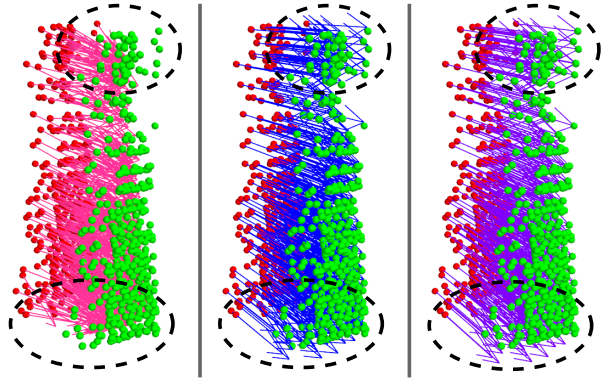

Figure 6. **The flow refinement effect.** We demonstrate the effect on data from KITTI_t. The source point cloud is in red, and the target is in green. Our correspondence model was trained on the KITTI_v dataset, and its flow estimation before refinement is shown in magenta (left). The optimized refined flow is presented in blue (center). We also show the ground-truth scene flow in purple for reference (right). The refined flow better covers the target point cloud (top center ellipse). It also breaches the convex hull of the given target points and enables computing the correct flow for source points whose target is missing (bottom center ellipse).

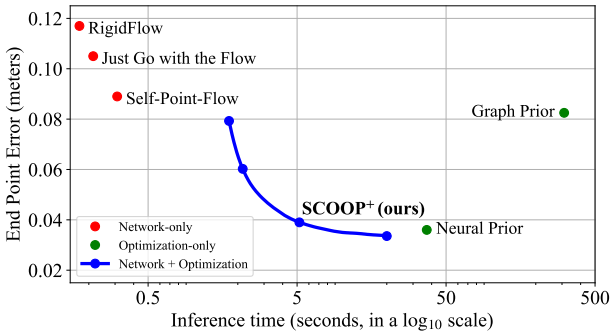

Figure 7. **Flow estimation error *vs*. inference time for the KITTI_t dataset.** SCOOP$^+$ has a lower error than the network-only models and a shorter inference time than the optimization-only methods. It also allows different balances along the error and time trade-off, as presented by the blue curve.

Setting	$EPE\downarrow$	$AS\uparrow$
(a) All target points as candidates ($k_s = n$)	0.047	91.1
(b) W/O confidence ($p_{x_i} = 1$, $\alpha_{conf} = 0$)	0.044	90.3
(c) W/O smoothness loss term ($\alpha_{flow} = 0$)	0.056	86.6
(d) W/O flow refinement ($R^* = 0$)	0.115	43.8
Our complete method	**0.039**	**93.6**

Table 2. **Component ablative settings.** SCOOP was trained on KITTI_v and evaluated on KITTI_t. The results show that the best performance is obtained with our complete method. Additional details about the ablation experiments are given in subsection 4.4.

ing point with more than 50% error reduction over the feed-forward models and about $8\times$ faster inference time than the optimization-only Neural Prior work. SCOOP$^+$ also enables a different balance between time and performance, as seen in Figure 7. By reducing the number of run-time optimization steps, the user can shorten the inference time, achieving a working point closer to that of the network-only models.

4.4. Ablation Study

The design choices in our method are verified by ablation experiments presented in Table 2. We change one element each time and keep all the others the same. The following ablative settings were examined: (a) use all target points for soft correspondence instead of the ones with the highest transport amount (Equation 5); (b) ignore the point matching confidence by setting $p_{x_i} = 1$ in Equation 10 and $\alpha_{conf} = 0$ in Equation 13; (c) exclude the smoothness flow loss $\mathcal{L}_{flow}$ from Equation 13; and (d) turn off the flow refinement module.

The ablation study validates the contribution of the proposed components to the method's performance. Considering a subset of target points for correspondence enables the model to concentrate on the most relevant candidates for flow estimation. The matching confidence emphasizes the influence of the more confident points in our confidence-aware distance loss $\mathcal{L}_{dist}$. The smoothness loss term is important for regularizing the point representation learning to obtain similar features across neighboring points. Lastly, our flow refinement optimization improves the consistency of the flow field and reduces the EPE substantially. In the supplementary material, we provide an ablation study on the FT3D_o train set size and find that a 10% fraction of the data suffices for our method to realize its potential.

5. Conclusions

This paper presented SCOOP, a novel self-supervised scene flow estimation method for 3D point clouds based on correspondence learning and flow refinement optimization. Previous works suggested learning a flow regression model, training a neural network that jointly learned point cloud correspondence and flow refinement, or optimizing the flow completely at run-time without learning.

In contrast, we split the flow prediction process into two simpler problems. Our correspondence model is focused only on learning point features to initialize the flow from soft matches between the point clouds. Then, we directly optimize a residual flow refinement at run-time. This approach enables SCOOP to be trained on a small set of point cloud scenes without utilizing ground-truth supervision while outperforming state-of-the-art fully-supervised and self-supervised learning methods, as well as optimization-based alternative techniques.

References

[1] Thomas Brox and Jitendra Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011. 1

[2] 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 (CVPR)*, pages 11621–11631, 2020. 7

[3] Angel X. Chang, Thomas Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[4] Wencan Cheng and Jong Hwan Ko. Bi-PointFlowNet: Bidirectional Learning for Point Cloud Based Scene Flow Estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 108–124, 2022. 1, 2, 6

[5] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling Algorithms for Unbalanced Transport Problems. *Mathematics of Computation*, 87:2563–2609, 2018. 3

[6] Marco Cuturi. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2292–2300, 2013. 3

[7] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. Learning Elementary Structures for 3D Shape Generation and Matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7433–7443, 2019. 3

[8] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J. Guibas, and Tolga Birdal. Weakly Supervised Learning of Rigid 3D Scene Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5692–5703, 2021. 1, 3

[9] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. 3D-CODED: 3D Correspondences by Deep Deformation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 230–246, 2018. 3

[10] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. HPLFlowNet: Hierarchical Permutohedral Lattice FlowNet for Scene Flow Estimation on Large-Scale Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019. 1, 2

[11] Yair Kittenplon, Yonina C. Eldar, and Dan Raviv. FlowStep3D: Model Unrolling for Self-Supervised Scene Flow Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4114–4123, 2021. 1, 3, 4

[12] Itai Lang, Dvir Ginzburg, Shai Avidan, and Dan Raviv. DPC: Unsupervised Deep Point Correspondence via Cross and Self Construction. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 1442–1451, 2021. 2, 3

[13] 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 Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15577–15586, 2021. 1, 2, 3, 5, 6, 7

[14] 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, 2022. 1, 2, 6, 7

[15] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural Scene Flow Prior. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7838–7851, 2021. 2, 3, 6, 7

[16] Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. FlowNet3D: Learning Scene Flow in 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. 1, 2, 5, 6

[17] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep Rigid Instance Scene Flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3614–3622, 2019. 1

[18] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. 5

[19] Moritz Menze and Andreas Geiger. Object Scene Flow for Autonomous Vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015. 1, 5

[20] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2:427–434, 2015. 5

[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)*, pages 11177–11185, 2020. 1, 2, 4, 5, 6, 7

[22] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic Differentiation in PyTorch. 2017. 5

[23] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene Flow from Point Clouds with or without Learning. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 261–270, 2020. 2, 3, 6, 7

[24] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–544, 2020. 1, 2, 3, 4, 5, 6, 7

[25] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in

a Metric Space. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5099–5108, 2017. 3, 5

[26] R. Tyrrell Rockafellar. *Convex Analysis*, volume 28. Princeton University Press, 1970. 2, 5

[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 (ICCV)*, volume 2, pages 722–729, 1999. 1

[28] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise Rigid Scene Flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1377–1384, 2013. 1

[29] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D Scene Flow Estimation with a Piecewise Rigid Scene Model. *International Journal of Computer Vision*, 115(1):1–28, 2015. 1

[30] Guangming Wang, Yunzhe Hu, Zhe Liu, Yiyang Zhou, Masayoshi Tomizuka, Wei Zhan, and Hesheng Wang. What Matters for 3D Scene Flow Network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 38–55, 2022. 1, 2, 6

[31] 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 (CVPR)*, pages 14173–14182, 2021. 1, 2, 6

[32] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic Scene Flow Computation for 3D Motion Understanding. *International Journal of Computer Vision*, 95:29–51, 2011. 1

[33] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient Dense Scene Flow from Sparse or Dense Stereo Data. In David Forsyth, Philip Torr, and Andrew Zisserman, editors, *Proceedings of the Seventh IEEE International Conference on Computer Vision (ECCV)*, pages 739–751, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 1

[34] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 88–107, 2020. 1, 2

[35] Xin Wu, Hao Zhao, Shunkai Li, Yingdian Cao, and Hongbin Zha. SC-wLS: Towards Interpretable Feed-forward Camera Re-localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 585–601, 2022. 2

[36] Yiming Zeng, Yue Qian, Zhiyu Zhu, Junhui Hou, Hui Yuan, and Ying He. CorrNet3D: Unsupervised End-to-end Learning of Dense Correspondence for 3D Point Clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6052–6061, 2021. 3

[37] Chengliang Zhong, Peixing You, Xiaoxue Chen, Hao Zhao, Fuchun Sun, Guyue Zhou, Xiaodong Mu, Chuang Gan, and Wenbing Huang. SNAKE: Shape-aware Neural 3D Keypoint Field. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2