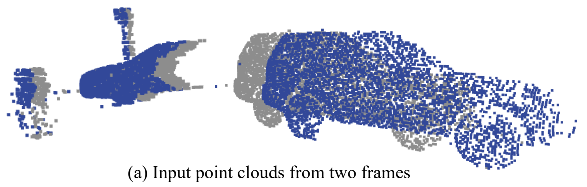
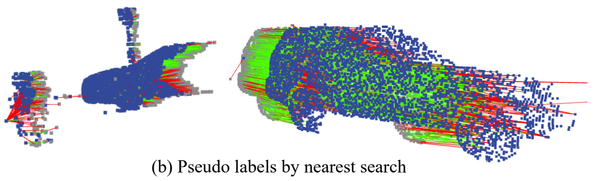
RigidFlow: Self-Supervised Scene Flow Learning on Point Clouds by Local Rigidity Prior

Ruibo Li[1,2], Chi Zhang[2], Guosheng Lin[1,2]*, Zhe Wang[3], Chunhua Shen[4]
[1]S-Lab for Advanced Intelligence, Nanyang Technological University
[2]School of Computer Science and Engineering, Nanyang Technological University
[3]SenseTime Research [4]Zhejiang University
E-mail: ruibo001@e.ntu.edu.sg , gslin@ntu.edu.sg

Abstract

In this work, we focus on scene flow learning on point clouds in a self-supervised manner. A real-world scene can be well modeled as a collection of rigidly moving parts, therefore its scene flow can be represented as a combination of rigid motion of each part. Inspired by this observation, we propose to generate pseudo scene flow for self-supervised learning based on piecewise rigid motion estimation, in which the source point cloud is decomposed into a set of local regions and each region is treated as rigid. By rigidly aligning each region with its potential counterpart in the target point cloud, we obtain a region-specific rigid transformation to represent the flow, which together constitutes the pseudo scene flow labels of the entire scene to enable network training. Compared with most existing approaches relying on point-wise similarities for scene flow approximation, our method explicitly enforces region-wise rigid alignments, yielding locally rigid pseudo scene flow labels. We demonstrate the effectiveness of our self-supervised learning method on FlyingThings3D and KITTI datasets. Comprehensive experiments show that our method achieves new state-of-the-art performance in self-supervised scene flow learning, without any ground truth scene flow for supervision, even outperforming some supervised counterparts.


(a) Input point clouds from two frames


(b) Pseudo labels by nearest search

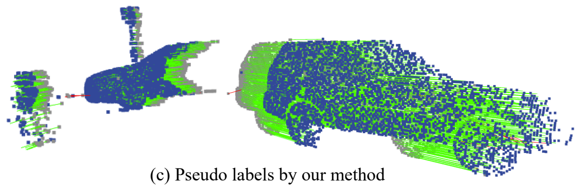

(c) Pseudo labels by our method

Figure 1. Comparison of the pseudo scene flow labels produced by nearest neighbor search and our proposed method. (a) Input point clouds from two consecutive frames; (b) Pseudo labels generated by the point matching with per-point nearness as a measure; (c) Pseudo labels generated by our proposed method. Green line represents the correct pseudo label with absolute error less than $0.1m$ or relative error less than 10%. Red line represents the incorrect pseudo label.

1. Introduction

Scene flow [35] is a 3D motion field to describe the motion of every point between two time steps. As an essential representation of dynamics, scene flow can be applied in numerous downstream applications, such as robotics and autonomous driving, for dynamic scene understanding. Due to the wide applications of 3D sensors, scene flow estimation with point clouds as input has attracted broad interest. However, scene flow data is difficult to collect [25], which makes supervised learning approaches suffer from a shortage of real-world training samples. To overcome the reliance on scene flow data, scene flow learning in a self-supervised manner is proposed as a promising solution, which is the focus of our work here.

To enable deep network training under the self-supervised setting, it is crucial to approximate the scene flow labels given unlabelled point cloud data. To do so,

*Corresponding author: G. Lin. (e-mail: gslin@ntu.edu.sg) most previous approaches [1, 13, 16, 27, 29, 34, 42] perform point matching between two consecutive point clouds to establish point correspondences and treat the 3D coordinate difference between each matching pair as the pseudo scene flow label. However, the point matching strategies applied in these approaches mainly consider point-wise similarities, but often fail to capture potential structured motions of points, leading to inconsistent pseudo scene flow labels. Fig. 1 (b) shows an example of the estimated pseudo scene flow labels based on point matching with per-point nearness as a measure.

For a real-world scene, most of the structures in this scene are rigid or almost so [23]. This allows us to decompose a non-rigid scene into a collection of rigidly moving parts, such that the entire scene flow can be approximated by estimating the rigid motion of individual parts. Based on this intuition, in this work, we propose to generate the pseudo scene flows from point clouds by piecewise rigid motion estimations.

To achieve this goal, an over-segmentation approach is employed to decompose the source point cloud into supervoxels and these supervoxels are treated as rigid during the pseudo label generation. By solving an independent rigid registration for each supervoxel, we find a rigid transformation that rigidly aligns this supervoxel with its potential counterpart in the target point cloud. Based on the rigid transformation estimate, we produce the rigid flow of each supervoxel, thereby generating the entire scene flow approximation as pseudo labels for self-supervised learning. Different from most existing self-supervised approaches that neglect potential structured motions of points, our method explicitly enforces region-wise rigid alignments and generates locally rigid pseudo scene flow labels, where the pseudo labels of points in the same supervoxel obey the same rigid motion pattern.

Main contributions of this paper are listed as follows:

- We present a new self-supervised scene flow learning approach (RigidFlow) that solves the pseudo scene flow label generation as a piecewise rigid motion estimation task;
- By decomposing the source point cloud into a set of local regions, we propose a piecewise pseudo label generation module that explicitly enforces region-wise rigid alignments and produces rigid pseudo flow for each local region.
- Our proposed RigidFlow achieves state-of-the-art performance in self-supervised scene flow learning, without any ground truth scene flow for supervision, even outperforming some supervised counterparts.

2. Related Work

Scene flow estimation on images. Scene flow estimation [35] aims to predict a 3D field in a scene to represent the motion of each point. Nowadays, the local rigidity assumption has been widely used in numerous advanced approaches [2, 9–11, 14, 18, 21, 22, 25, 33, 36–38] to estimate scene flow from images. Especially, UnRigidFlow [18] and EffiScene [11] address unsupervised scene flow estimation from images by jointly learning rigidity masks to constrain scene flow predictions. Unlike these methods that use highly regular 2D images as input and employ photometric error as the major loss function, we focus on scene flow estimation from sparse 3D point clouds and explore the application of the local rigidity to pseudo scene flow label generation, so that self-supervised scene flow learning can be achieved by any supervised loss functions with our generated pseudo labels.

Supervised scene flow estimation on point clouds. With the development of 3D sensors, estimating scene flow from point clouds has been drawing a lot of attention. Various approaches [3, 6–8, 15, 19, 20, 30, 41] have been proposed to estimate scene flow from point clouds by supervised learning. Particularly, the local rigidity has also been applied in some of them [3, 6, 15]. PointFlowNet [3] and Rigid3DSceneFlow [6] propose to directly estimate rigid motion for each 3D object. HCRF-Flow [15] employs the local rigidity to refine scene flow predictions. Different from these fully-supervised approaches learning to refine or constrain predicted flow by the local rigidity, without any ground truth data, we explore how to produce pseudo labels with the guidance of the local rigidity assumption to achieve self-supervised scene flow learning.

Self-supervised scene flow estimation on point clouds. In self-supervised scene flow learning, without ground truth scene flow, most previous methods [1, 13, 16, 27, 29, 34, 42] propose to establish point correspondences between source and target point clouds by point matching, and treat the 3D coordinate difference between each matching pair as a pseudo scene flow label. Specifically, [1, 27, 34] adopt a nearest neighbor loss and [13, 29, 42] adopt a Chamfer loss for self-supervision. In the two loss functions, point correspondences are built by the nearest search. And, [16] builds point correspondences by solving an optimal transport. However, the two point matching strategies mainly consider point-wise similarities but fail to capture potential structured motions of points, leading to unsatisfactory scene flow approximation. Motivated by the local rigidity assumption, we propose to generate pseudo labels by a piecewise rigid motion estimation. By explicitly enforcing region-wise rigid alignments between the source and the target, our method generate locally rigid pseudo scene flow labels, where the pseudo labels of points in the same supervoxel obey the same rigid motion pattern. Although the local rigidity has been considered in some recent self-supervised works [1, 29], the clue of rigidity is limited to smooth or constrain the flow predictions as a regularization. Its ability to improve pseudo label generation is far from being explored.

3. Preliminaries: Rigid registration and ICP

As a crucial task in computer vision, point cloud registration has been well studied in the literature [4, 5, 31, 32, 39, 40]. Given two point clouds, $\boldsymbol{X} = \{\boldsymbol{x}_i \in \mathbb{R}^3\}_{i=1}^{N_x}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_i \in \mathbb{R}^3\}_{i=1}^{N_y}$, point cloud rigid registration aims to predict a rigid transformation that aligns $\boldsymbol{X}$ to $\boldsymbol{Y}$. The rigid transformation can be written as $[\boldsymbol{R}_{XY}, \boldsymbol{t}_{XY}]$, where the rotation matrix $\boldsymbol{R}_{XY} \in \mathrm{SO}(3)$ and the translation vector $\boldsymbol{t}_{XY} \in \mathbb{R}^3$. The objective function of point cloud registration can be expressed as:

$$E(\boldsymbol{R}_{XY}, \boldsymbol{t}_{XY}; \boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{R}_{XY}\boldsymbol{x}_i + \boldsymbol{t}_{XY} - \boldsymbol{y}_{m(i)}\|_2^2, \quad (1)$$

where $\boldsymbol{m}$ is the point mapping from points in $\boldsymbol{X}$ to their corresponding points in $\boldsymbol{Y}$.

Since the point mapping $\boldsymbol{m}$ is unknown, the iterative closest point (ICP) [4] is widely employed to address this problem by alternating between estimating the rigid transformation and finding the point mapping. In each iteration, based on the previous point mapping estimate, the current rigid transformation is updated by solving the least-square problem in Eq. (1). And then, based on the current rigid transformation estimate, the point mapping of each point is updated to its closest match in another point cloud:

$$m(i) = \arg\min_{j} \|\boldsymbol{R}_{XY}\boldsymbol{x}_i + \boldsymbol{t}_{XY} - \boldsymbol{y}_j\|_2^2. \quad (2)$$

Although the ICP is efficient, the performance depends heavily on the initialization of rigid transformation and point matching.

4. Method

Given a temporal sequence of point clouds, $\boldsymbol{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ at frame t and $\boldsymbol{Q} = \{\boldsymbol{q}_i \in \mathbb{R}^3\}_{i=1}^{N}$ at frame $t+1$, scene flow estimation aims to estimate the 3D motion field $\boldsymbol{F} = \{\boldsymbol{f}_i \in \mathbb{R}^3\}_{i=1}^{N}$ in a scene. In this paper, we target at self-supervised point cloud scene flow estimation, where no ground truth scene flow annotations $\boldsymbol{D}$ are provided. To enable network training without ground truth labels, we focus on effective pseudo scene flow label generation, such that the scene flow learning can be achieved using the pseudo labels $\widehat{\boldsymbol{D}}$ as supervision.

In this section, we first introduce how to produce pseudo scene flow for a real-world scene by a piecewise rigid motion estimation. After that, we present the details of our algorithm and explain how to use the generated pseudo labels to achieve self-supervised training. Fig. 2 is the overview of our method.

4.1. Generating Pseudo Labels by Piecewise Rigid Motion Estimation

Scene flow is the 3D motion field of objects in a scene. If the scene contains only a rigidly moving object, the scene flow from $\boldsymbol{P}_{rigid}$ to $\boldsymbol{Q}_{rigid}$ follows the rigid transformation $[\boldsymbol{R}_{PQ}, \boldsymbol{t}_{PQ}]$ between the two point clouds:

$$\boldsymbol{D} = \boldsymbol{R}_{PQ}\boldsymbol{P}_{rigid} + \boldsymbol{t}_{PQ} - \boldsymbol{P}_{rigid}. \quad (3)$$

Thus, for a rigidly moving object, when the ground truth scene flow is unavailable, we can produce the scene flow by finding its optimal rigid transformation between $\boldsymbol{P}_{rigid}$ and $\boldsymbol{Q}_{rigid}$.

For a complex real-world scene, although it is not rigid, most of the structures in this scene are rigid or almost rigid, which makes it possible to approximate a non-rigid scene into a set of rigidly moving regions. Therefore, we can estimate the flow of each rigidly moving region by finding its optimal rigid motion, thereby generating the entire scene flow. In other words, we can generate pseudo scene flows by a piecewise rigid motion estimation.

Decomposing the point cloud $\boldsymbol{P}$ into K rigid regions $\{\boldsymbol{P}^{(1)}, \boldsymbol{P}^{(2)}, ..., \boldsymbol{P}^{(K)}\}$, the piecewise rigid motion estimation from $\boldsymbol{P}$ to $\boldsymbol{Q}$ for the region $\boldsymbol{P}^{(k)}$ can be considered as an independent rigid body registration from $\boldsymbol{P}^{(k)}$ to $\boldsymbol{Q}$:

$$[\boldsymbol{R}_k^*, \boldsymbol{t}_k^*] = \arg\min_{[\boldsymbol{R}_k, \boldsymbol{t}_k]} E(\boldsymbol{R}_k, \boldsymbol{t}_k; \boldsymbol{P}^{(k)}, \boldsymbol{Q}), \quad (4)$$

$$E(\boldsymbol{R}_k, \boldsymbol{t}_k; \boldsymbol{P}^{(k)}, \boldsymbol{Q}) = \frac{1}{N_k} \sum_{i=1}^{N_k} \|\boldsymbol{R}_k\boldsymbol{p}_i^{(k)} + \boldsymbol{t}_k - \boldsymbol{q}_{m^{(k)}(i)}\|_2^2, \quad (5)$$

where N_k is the point number in the rigid region $\boldsymbol{P}^{(k)}$, $m^{(k)}(i)$ is the point mapping from the i-th point in $\boldsymbol{P}^{(k)}$ to its correspondence in $\boldsymbol{Q}$, and $[\boldsymbol{R}_k^*, \boldsymbol{t}_k^*]$ is the optimal rigid transformation for $\boldsymbol{P}^{(k)}$.

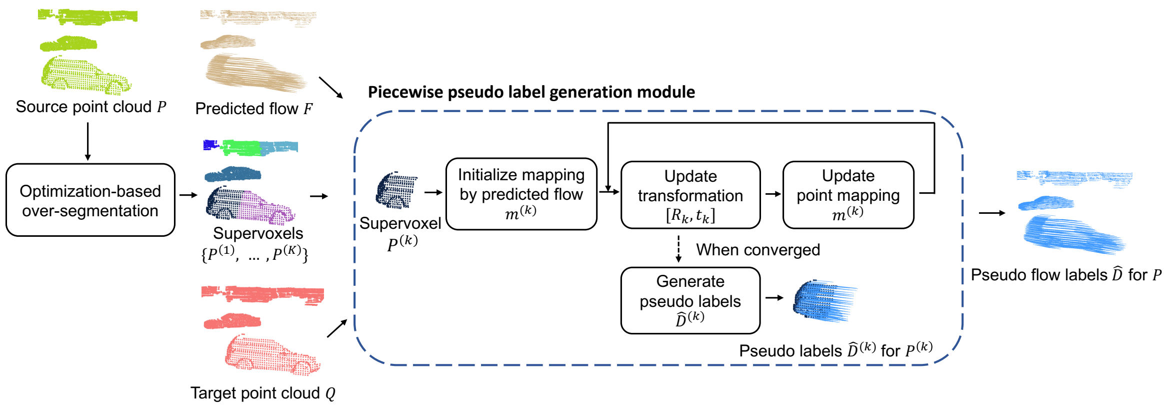
Following Eq. (3), with $[\boldsymbol{R}_k^*, \boldsymbol{t}_k^*]$, the pseudo rigid scene flow estimate $\widehat{\boldsymbol{D}}^{(k)}$ for this region can be computed by:

$$\widehat{\boldsymbol{D}}^{(k)} = \boldsymbol{R}_k^*\boldsymbol{P}^{(k)} + \boldsymbol{t}_k^* - \boldsymbol{P}^{(k)}. \quad (6)$$

Combining the pseudo rigid scene flow estimates for all K rigid regions $\{\widehat{\boldsymbol{D}}^{(1)}, \widehat{\boldsymbol{D}}^{(2)}, ..., \widehat{\boldsymbol{D}}^{(K)}\}$, we obtain the final pseudo scene flow labels $\widehat{\boldsymbol{D}}$ for self-supervised training.

4.2. Piecewise pseudo label generation module

In order to convert pseudo label generation into a piecewise rigid motion estimation, we first employ an oversegmentation method [17] to split the point cloud $\boldsymbol{P}$ into supervoxels and treat these supervoxels as rigid moving regions. After obtaining the supervoxels, we employ a piecewise pseudo label generation module to generate scene flow labels for each supervoxel via finding its rigid transformation from $\boldsymbol{P}^{(k)}$ to $\boldsymbol{Q}$. In this module, we follow the principle of ICP algorithm [4], i.e., alternately estimating point mapping and rigid transformation, such



Source point cloud P
Predicted flow F
Optimization-based over-segmentation
Supervoxels $\{P^{(1)}, \ldots, P^{(K)}\}$
Target point cloud Q
Piecewise pseudo label generation module
Supervoxel $P^{(k)}$
Initialize mapping by predicted flow $m^{(k)}$
Update transformation $[R_k, t_k]$
Update point mapping $m^{(k)}$
When converged
Generate pseudo labels $\widehat{D}^{(k)}$
Pseudo labels $\widehat{D}^{(k)}$ for $P^{(k)}$
Pseudo flow labels $\widehat{D}$ for P

Figure 2. Overview of our pseudo scene flow label generation approach. In our approach, we employ an optimization-based over-segmentation method [17] to split the point cloud in the first frame into a set of supervoxels and estimate the rigid transformation of each supervoxel. Based on the rigid transformation estimate, we generate the pseudo labels for each supervoxel, thereby obtaining the pseudo labels for the entire scene.

Algorithm 1 Pseudo scene flow label generation by piecewise rigid motion estimation

```
Input: Source point cloud $\boldsymbol{P}$, Target point cloud $\boldsymbol{Q}$,
Predicted scene flow from NNs being trained $\boldsymbol{F}$;
Output: Pseudo scene flow labels $\widehat{\boldsymbol{D}}$ ;
Procedure:
 1: Split $\boldsymbol{P}$ into a set of supervoxels $\{\boldsymbol{P}^{(1)}, \ldots, \boldsymbol{P}^{(K)}\}$;
                                        ▶ Oversegmentation
 2: for $k = 1, \ldots, K$ do
 3:     $m^{(k)}(i) \leftarrow \arg\min_j \|\boldsymbol{p}_i^{(k)} + \boldsymbol{f}_i^{(k)} - \boldsymbol{q}_j\|_2^2$
                                        ▶ Initializing by predicted flow
 4:     while not converged do
 5:         Find the matches $\boldsymbol{Q}^{(k)}$ for $\boldsymbol{P}^{(k)}$ based on $\boldsymbol{m}^{(k)}$
 6:         $\boldsymbol{H}^{(k)} \leftarrow \sum_{i=1}^{N_k} (\boldsymbol{p}_i^{(k)} - \overline{\boldsymbol{p}}^{(k)})(\boldsymbol{q}_i^{(k)} - \overline{\boldsymbol{q}}^{(k)})^\top$
 7:         $\boldsymbol{R}_k \leftarrow \boldsymbol{V}^{(k)} \boldsymbol{U}^{(k)\top}$,  $\boldsymbol{t}_k \leftarrow -\boldsymbol{R}_k \overline{\boldsymbol{p}}^{(k)} + \overline{\boldsymbol{q}}^{(k)}$
                                        ▶ Updating rigid transformation
 8:         $m^{(k)}(i) \leftarrow \arg\min_j \|\boldsymbol{R}_k \boldsymbol{p}_i^{(k)} + \boldsymbol{t}_k - \boldsymbol{q}_j\|_2^2$
                                        ▶ Updating point mapping
 9:     end while
10:     $\widehat{\boldsymbol{D}}^{(k)} \leftarrow \boldsymbol{R}_k^* \boldsymbol{P}^{(k)} + \boldsymbol{t}_k^* - \boldsymbol{P}^{(k)}$
                                        ▶ Generating pseudo labels
11: end for
12: $\widehat{\boldsymbol{D}} \leftarrow \{\widehat{\boldsymbol{D}}^{(1)}, \ldots, \widehat{\boldsymbol{D}}^{(K)}\}$
```

that the region-wise rigid alignment between this region with its counterpart is explicitly enforced. Next, we present the details of this module.

Initializing point mapping by predicted flow. The performance of ICP relies greatly on the initialization of rigid transformation and point mapping. When solving the registration from $\boldsymbol{P}^{(k)}$ to $\boldsymbol{Q}$, for each point $\boldsymbol{p}_i^{(k)}$ in $\boldsymbol{P}^{(k)}$, a straightforward way of initialization is to set its closest point in $\boldsymbol{Q}$ as the initial correspondence. Inspired by [16], we establish the initial point mapping based on the predicted scene flow $\boldsymbol{F}$ from neural networks being trained. Specifically, we warp the point $\boldsymbol{p}_i^{(k)}$ by its predicted flow $\boldsymbol{f}_i^{(k)}$, and then take the closest point of this warped point as the initial match:

$$m_{inital}^{(k)}(i) = \arg\min_j \|\boldsymbol{p}_i^{(k)} + \boldsymbol{f}_i^{(k)} - \boldsymbol{q}_j\|_2^2. \tag{7}$$

As the training progresses, the accuracy of the predicted scene flow will be gradually improved, making the closest search of the warped points more likely to find the correct matches and establish good initial point correspondences.

Updating rigid transformation estimate. Based on the previous point mapping estimate, we update the rigid transformation for each supervoxel by solving the least-square problem shown in Eq. (4) and Eq. (5) with the point mapping fixed. Specifically, following [4, 39, 40], we apply the singular value decomposition (SVD) to it.

For the points in supervoxel $\boldsymbol{P}^{(k)}$, we first select their matches $\boldsymbol{Q}^{(k)}$ from $\boldsymbol{Q}$ according to the point mapping estimate $\boldsymbol{m}^{(k)}$. The centers of $\boldsymbol{P}^{(k)}$ and $\boldsymbol{Q}^{(k)}$ are defined as $\overline{\boldsymbol{p}}^{(k)} = \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{p}_i^{(k)}$ and $\overline{\boldsymbol{q}}^{(k)} = \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{q}_i^{(k)}$. The cross-covariance matrix for supervoxel $\boldsymbol{P}^{(k)}$ can be written as:

$$\boldsymbol{H}^{(k)} = \sum_{i=1}^{N_k} (\boldsymbol{p}_i^{(k)} - \overline{\boldsymbol{p}}^{(k)})(\boldsymbol{q}_i^{(k)} - \overline{\boldsymbol{q}}^{(k)})^\top. \tag{8}$$

Using SVD to decompose $\boldsymbol{H}^{(k)}$, we have $\boldsymbol{H}^{(k)} = \boldsymbol{U}^{(k)} \boldsymbol{S}^{(k)} \boldsymbol{V}^{(k)\top}$. The rigid transformation estimate for supervoxel $\boldsymbol{P}^{(k)}$ can be updated in closed-form as:

$$\begin{aligned} \boldsymbol{R}_k &= \boldsymbol{V}^{(k)} \boldsymbol{U}^{(k)\top}, \\ \boldsymbol{t}_k &= -\boldsymbol{R}_k \overline{\boldsymbol{p}}^{(k)} + \overline{\boldsymbol{q}}^{(k)}. \end{aligned} \tag{9}$$

Updating point mapping estimate. Based on the current rigid transformation estimate, the point mapping of each point in supervoxel $\boldsymbol{P}^{(k)}$ is updated to its closest point in $\boldsymbol{Q}$:

$$m^{(k)}(i) = \arg\min_{j} \|\boldsymbol{R}_k \boldsymbol{p}_i^{(k)} + \boldsymbol{t}_k - \boldsymbol{q}_j\|_2^2. \tag{10}$$

Generating pseudo labels for each supervoxel. After several alternating iterations between Eq. (9) and Eq. (10), we obtain the final rigid transformation estimates for each supervoxel as the optimal rigid transformation. Following Eq. (6), we generate pseudo rigid scene flow for each supervoxel. The method of our pseudo label generation is sketched in Algorithm 1.

4.3. Self-supervised training with pseudo labels

Using the generated pseudo labels for supervision, we can achieve the self-supervised training of scene flow estimation networks with supervised loss functions. In this paper, we choose FLOT [30] as the default estimation network and use the l_1-norm loss function for self-supervised learning:

$$L = \frac{1}{3N} \sum_{i=1}^{N} \|\boldsymbol{f}_i - \widehat{\boldsymbol{d}}_i\|_1, \tag{11}$$

where $\boldsymbol{f}_i$ is the predicted scene flow for point i and $\widehat{\boldsymbol{d}}_i$ is our generated pseudo label.

5. Experiment

To validate the effectiveness of our self-supervised learning method, we compare our method with the state-of-the-art fully-supervised and self-supervised approaches. Then, we conduct various ablation experiments to analyze the contribution of different components. Finally, we design some quantitative and qualitative experiments to evaluate the generated pseudo labels for further analysis. All experiments are performed on the large-scale synthetic FlyingThings3D [24] dataset and the real-world KITTI 2015 [25, 26] dataset.

Datasets. 3D data are not directly provided by the two original datasets, thus the point clouds need to be extracted from the original data. Following [30], we denote the two point cloud datasets prepared by HPLFlowNet [7] as $\mathbf{FT3D_s}$ and $\mathbf{KITTI_s}$, respectively. For $\mathrm{FT3D_s}$ and $\mathrm{KITTI_s}$, there are no occluded points in the processed point clouds. We denote the two datasets prepared by FlowNet3D [19] as $\mathbf{FT3D_o}$ and $\mathbf{KITTI_o}$, respectively, where occluded points are preserved. Specially, FlowNet3D [19] also splits the $\mathrm{KITTI_o}$ data to use the first 100 pairs for finetuning and the rest 50 pairs for testing. Here, we denote the finetuning part as $\mathbf{KITTI_f}$ and the rest testing data as $\mathbf{KITTI_t}$. Following the raw data sampling strategy used in [16], we extract some raw point clouds from KITTI dataset as training samples (6026 pairs) and denote them as $\mathbf{KITTI_r}$. There is no overlap between $\mathrm{KITTI_r}$ and $\mathrm{KITTI_o}$.

Implementation details. During evaluation, we conduct two main experiments, one for point clouds without occlusions and one for point clouds with occlusions.

For the experiment on point clouds without occlusions, we follow the experimental setting in [7, 13, 42]. Specifically, we train a FLOT [30] model by our self-supervised approach on $\mathrm{FT3D_s}$ training set and test it on $\mathrm{FT3D_s}$ testing set and $\mathrm{KITTI_s}$. For a pair of point clouds, we randomly sample 8192 points in each point cloud as input. In the pseudo label generation phase, we decompose the source point cloud into 40 supervoxels with an over-segmentation method [17] and set the iteration number in our piecewise pseudo label generation module to 4. We set the batchsize to 1 and use Adam optimizer [12] with an initial learning rate of 0.001.

For the experiment on point clouds with occlusions, we train a FLOT model on $\mathrm{KITTI_r}$ using our self-supervised method and evaluate it on $\mathrm{KITTI_o}$ and $\mathrm{KITTI_t}$. For each training sample, we randomly sample 2048 points in each point cloud as input. In the pseudo label generation phase, we split the source point cloud into 40 supervoxels and set the iteration number to 2. We set the batchsize to 4 and use Adam with an initial learning rate of 0.001.

Our code is implemented based on PyTorch [28], FLOT [30] and S3DPC [17].

Evaluation metrics. We evaluate our method on four evaluation metrics adopted in [7, 30]. We denote the ground truth scene flow and predicted scene flow as $\boldsymbol{D}$ and $\boldsymbol{F}$, respectively. The metrics are defined as follows: **EPE**(m): $\|\boldsymbol{D}-\boldsymbol{F}\|_2$, end point error, averaged over all points; **AS**(%): the ratio of points with EPE < 0.05m or relative error $< 5\%$; **AR**(%): the ratio of points with EPE < 0.1m or relative error $< 10\%$; **Out**(%): the ratio of points with EPE > 0.3m or relative error $> 10\%$.

5.1. Comparison with State-of-the-art Methods

5.1.1 Results on $\mathrm{FT3D_s}$ and $\mathrm{KITTI_s}$

We evaluate our self-supervised learning method on $\mathrm{FT3D_s}$ testing set and $\mathrm{KITTI_s}$ data, following the experimental setting in [7, 13, 42]. We compare our method with five advanced self-supervised approaches: Ego-motion [34], PointPWC-Net [42], SLIM [1], Self-Point-Flow [16] and FlowStep3D [13]. In Table 1, our approach achieves the best performance on **EPE**, **AS**, and **AR**, and obtains the second best performance on **Out**, which demonstrates the effectiveness and the generalization ability of our self-supervised learning algorithm. Especially, our method is the only self-supervised approach that achieves an **EPE** metric below $7cm$. For this metric, our method outperforms the recent FlowStep3D by 17.6% and 38.3% on $\mathrm{FT3D_s}$ and $\mathrm{KITTI_s}$ respectively, despite the model capacity of our used FLOT is significantly less than that of their FlowStep3D (0.11 M $v.s.$ 0.68 M).

Table 1. Quantitative results on FT3D$_{\mathbf{s}}$ testing set and KITTI$_{\mathbf{s}}$. Full. represents fully-supervised training, Self. represents self-supervised training. Compared with self-supervised approaches, our approach achieve the state-of-the-art performance on three metrics. Especially, our approach is the only self-supervised one that achieves an EPE metric below $7cm$. Without any ground truth scene flow for supervision, our approach even outperforms some supervised ones.

Data	Method	Sup.	EPE↓	AS↑	AR↑	Out↓
FT3D$_{s}$	FlowNet3D [19]	Full.	0.0864	47.89	83.99	54.64
	HPLFlowNet [7]	Full.	0.0804	61.44	85.55	42.87
	PointPWC-Net [42]	Full.	0.0588	73.79	92.76	34.24
	FLOT [30]	Full.	0.0520	73.20	92.70	35.70
	FlowStep3D [13]	Full.	0.0455	81.62	96.14	21.65
	Ego-motion [34]	Self.	0.1696	25.32	55.01	80.46
	PointPWC-Net [42]	Self.	0.1213	32.39	67.42	68.78
	Self-Point-Flow [16]	Self.	0.1009	42.31	77.47	60.58
	FlowStep3D [13]	Self.	0.0852	53.63	82.62	**41.98**
	Ours	Self.	**0.0692**	**59.62**	**87.10**	46.42
KITTI$_{s}$	FlowNet3D [19]	Full.	0.1064	50.65	80.11	40.03
	HPLFlowNet [7]	Full.	0.1169	47.83	77.76	41.03
	PointPWC-Net [42]	Full.	0.0694	72.81	88.84	26.48
	FLOT [30]	Full.	0.0560	75.50	90.80	24.20
	FlowStep3D [13]	Full.	0.0546	80.51	92.54	14.92
	Ego-motion [34]	Self.	0.4154	22.09	37.21	80.96
	PointPWC-Net [42]	Self.	0.2549	23.79	49.57	68.63
	SLIM(8192 point) [1]	Self.	0.1207	51.78	79.56	40.24
	Self-Point-Flow [16]	Self.	0.1120	52.76	79.36	40.86
	FlowStep3D [13]	Self.	0.1021	70.80	83.94	**24.53**
	Ours	Self.	**0.0619**	**72.37**	**89.23**	26.18

We also compare our self-supervised method with some supervised approaches that are trained on FT3D$_{s}$ training set. As shown in Table 1, without any ground truth for supervision, our self-supervised method outperforms FlowNet3D [19] and performs on par with HPLFlowNet [7] on FT3D$_{s}$. Evaluated on KITTI$_{s}$ without fine-tuning, our self-supervised method has better generalization ability than FlowNet3D and HPLFlowNet. Qualitative results on FT3D$_{s}$ are shown in Fig. 3.

5.1.2 Results on KITTI$_{o}$ and KITTI$_{t}$

We evaluate our self-supervised learning method on KITTI$_{o}$ and KITTI$_{r}$. In KITTI$_{o}$ data, following [30], we remove all points with depth larger than $35m$ for evaluation. As shown in Table 2, our method outperforms Self-Point-Flow [16] on all metrics. The FLOT trained on KITTI$_{r}$ via our self-supervised method achieves better performance than the fully-supervised FLOT [30] trained on FT3D$_{o}$, which demonstrates the advantage of our self-supervised learning strategy. That is, compared with learning from labeled synthetic data, the proposed self-supervised learning allows models to learn better representations from unannotated realistic data directly. Some qualitative results on KITTI$_{o}$ are shown in Fig. 3.

Table 2. Quantitative results on KITTI$_{o}$. Without any ground truth scene flow for supervision, our method outperforms Self-Point-Flow [16], and even performs better than supervised FlowNet3D [19] and FLOT [30].

Method	Sup.	Train. data	EPE↓	AS↑	AR↑	Out↓
FlowNet3D [19]	Full.	FT3D$_{o}$	0.173	27.6	60.9	64.9
FLOT [30]	Full.	FT3D$_{o}$	0.107	45.1	74.0	46.3
Self-Point-Flow [16]	Self.	KITTI$_{r}$	0.105	41.7	72.5	50.1
Ours	Self.	KITTI$_{r}$	**0.102**	**48.4**	**75.6**	**44.2**

Table 3. Quantitative results on KITTI$_{t}$ following the test settings in [29]. Without pre-trained on FT3D$_{o}$ with supervision, our self-supervised FLOT and FlowNet3D still achieve better performance.

Method	Pre-train	Training data	EPE↓	AS↑	AR↑
JGF [27]	✓	FT3D$_{o}$ + KITTI$_{f}$	0.218	10.17	34.38
WWL [29]	✓	FT3D$_{o}$ + KITTI$_{f}$	0.169	21.71	47.75
Ours (FlowNet3D)		KITTI$_{r}$	0.152	30.17	61.14
Ours (FLOT)		KITTI$_{r}$	**0.117**	**38.75**	**69.73**

Table 4. Ablation study for piecewise pseudo label generation module (**PPLG**). **NN**: nearest neighbor search. **RCA**: region-wise center alignment. **RRA**: region-wise rigid alignment. Δ denotes the difference in metric with respect to the baseline method.

Method	NN	RCA	RRA	EPE↓	ΔEPE
nearest neighbor search	✓			0.217	0.000
+ region-wise center alignment	✓	✓		0.089	-0.128
+ region-wise rigid align. (**Ours**)	✓		✓	**0.071**	-0.146

Then, we compare the performance of our method with WWL [29] and JGF [27] in a self-supervised learning manner. Following WWL [29], the experiment is conducted on KITTI$_{t}$. As shown in Table 3, without pre-trained on FT3D$_{o}$ with full-supervision, the FLOT and the FlowNet3D trained via our proposed self-supervised method still perform better than theirs.

5.2. Ablation study

We validate the effectiveness of each component in our method. Models are trained on FT3D$_{s}$ training set and tested on FT3D$_{s}$ testing set.

Piecewise pseudo label generation module. At the core of our design is the piecewise pseudo label generation module (**PPLG**). For each supervoxel, this module generates pseudo labels by alternately estimating point mapping and rigid transformation to explicitly enforce region-wise rigid alignments. The module contains three key steps: point mapping initialization, rigid transformation update, and point mapping update.

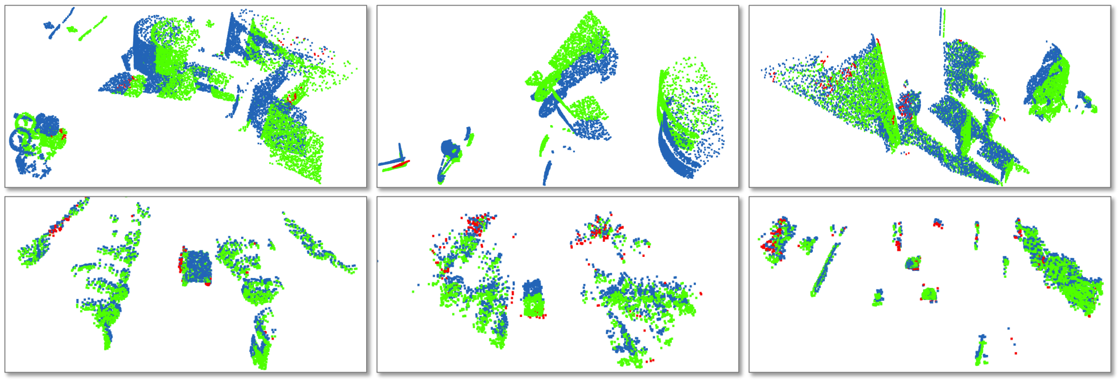


Figure 3. Qualitative results on **FT3D**$_s$ (top) and **KITTI**$_o$ (bottom). Bule points represent the source point cloud. Green points represent the points translated by the correct scene flow predictions. Red points represent the points translated by the incorrect predictions. The scene flow predictions are measured by **AR**.

Table 5. Comparison of different strategies in pseudo label generation module. **IwoF**: point mapping initialization without predicted flow. **IwF**: point mapping initialization with predicted flow. **GC**: label generation from correspondence. **GR**: label generation from rigid motion.

Method	**IwoF**	**IwF**	**GC**	**GR**	**EPE** ↓
Strategy I	✓			✓	0.356
Strategy II		✓	✓		0.080
Ours		✓		✓	**0.071**

To demonstrate the advantage of region-wise rigid alignment in pseudo scene flow generation, we design two baseline methods for comparison: nearest neighbor search, denoted by **NN**, and region-wise center alignment, denoted by **RCA**. In nearest neighbor search, for each point, we directly treat the initial match derived from our point mapping initialization (Eq. 7) as the corresponding point to produce a pseudo label without considering any region-wise constraints. As shown in Table 4, compared with the method of nearest neighbor search (**NN**), by enforcing region-wise alignments, our module significantly reduces the EPE metric by 14.6cm. For the method of region-wise center alignment, when updating transformation, we only encourage the center of each supervoxel to coincide with that of its counterpart rather than enforcing region-wise rigid alignments. Specifically, we fix the rotation matrix $\boldsymbol{R}_k$ in Eq. (6), Eq. (9) and Eq. (10) to an identity matrix. As shown in Table 4, by enforcing region-wise rigid alignments, our module outperforms the method of region-wise center alignment (**RCA**) by 22.8% on EPE metric, which demonstrates the effectiveness of our region-wise rigid alignment approach.

We next study the initialization strategy and the label generation strategy in **PPLG**. In our module, we initialize the point mapping by the predicted flow from neural networks being trained, denoted as **IwF**, and produce pseudo rigid scene flow from the optimal rigid transformation, denoted as **GR**. To verify the advantages of our design, we include two strategies for comparison: **Strategy I** initializes the point mapping without using predicted flow, denoted as **IwoF**, and **Strategy II** produces pseudo scene flow based on the coordinate difference between each optimal correspondence, denoted as **GC**. As shown in Table 5, our module outperforms Strategy I and Strategy II by around 80% and 14% on the EPE metric, respectively, which demonstrates the effectiveness of our design. And Table 6 shows the impact of different update iteration numbers on our method.

Table 6. The impact of different update iteration numbers.

Iteration No.	1	2	3	4
EPE ↓	0.077	0.073	0.071	**0.069**

Table 7. The impact of different supervoxel numbers.

Desired supervoxel No.	80	60	40	20
EPE ↓	0.074	0.073	**0.071**	0.077

Table 8. Time consumption of our method to generate pseudo labels for a training sample with 8,192 points in each point cloud.

Component	Over-segmentation	Label generation	Total
Time (ms)	76.1	96.4	172.5

Impact of supervoxel number. When generating pseudo labels, we decompose a scene into a set of supervoxels and find the rigid motion of each supervoxel. As shown in Table 7, the model achieves the best performance when our self-supervised method decomposes each scene into 40 supervoxels for label generation.

Time consumption. We evaluate the running time of our pseudo label generation method for a training sample with 8192 points in each point cloud. In Table 8, the total time

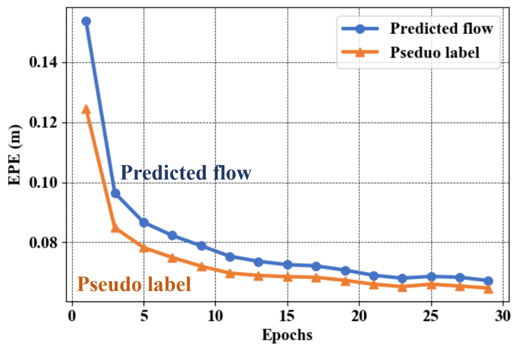


Figure 4. The end point error (**EPE**, the lower, the better) of model predictions (**Blue curve**) and generated pseudo labels (**Orange curve**) on some training samples. During the training, the accuracy of pseudo labels is consistently higher than that of predicted flow. This allows us to apply the pseudo labels as supervision.

consumption for a training sample is 172.5ms on a single 1080ti GPU.

5.3. Analysis on pseudo labels

We conduct some quantitative and qualitative experiments to evaluate the generated pseudo labels for further analysis. In Fig. 4, using part of training samples (197 samples) as testing data, we compare the error of our generated pseudo labels and the predictions of the trained network. We can make the following observations from Fig. 4. (1) During the training, the quality of our generated pseudo labels is gradually improved. (2) The accuracy of pseudo labels is consistently higher than that of predicted flow. This allows us to apply the pseudo labels as supervision. (3) The performance gap between pseudo labels and network predictions is gradually reduced. As illustrated in Fig. 5 (c)-(e), the quality of our generated pseudo labels for the airplane and the chair are gradually improved along with training iterations, which demonstrates the effectiveness of our pseudo label generation method. More quantitative and qualitative results are in supplementary.

6. Conclusion

In this paper, we propose to produce pseudo scene flow labels by a piecewise rigid motion estimation. By solving an independent rigid registration for each region, we find a region-specific rigid transformation to represent the flow, thereby generating the entire scene flow approximation as pseudo labels for self-supervised learning. Comprehensive experiments on FlyingThings3D and KITTI datasets demonstrate that our proposed approach achieves state-of-the-art performance in self-supervised scene flow learning, without any ground truth scene flow for supervision, even outperforming some supervised counterparts.

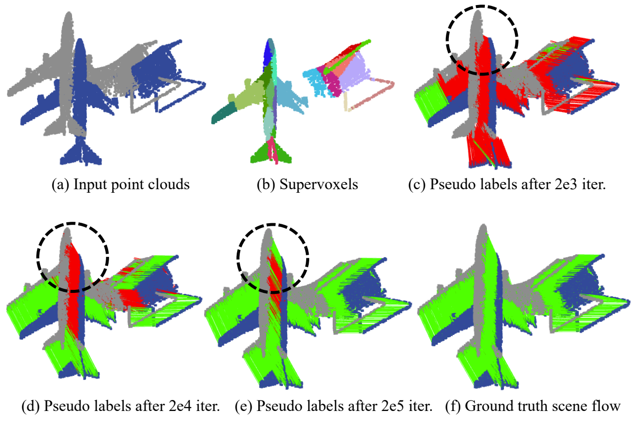

(a) Input point clouds
(b) Supervoxels
(c) Pseudo labels after 2e3 iter.
(d) Pseudo labels after 2e4 iter.
(e) Pseudo labels after 2e5 iter.
(f) Ground truth scene flow

Figure 5. Our generated pseudo labels after different training iterations for the same scene. (a) Input point clouds. Blue points represent the source point cloud and gray points represent the target. (b) Supervoxel results of the source point cloud. Different colors indicate different supervoxels. (c) - (e) Our generated pseudo labels after different training iterations. **Green line** indicates the **correct** pseudo label measured by **AR**. **Red line** indicates the **incorrect** pseudo label. (f) Ground truth scene flow. The quality of our generated pseudo labels for the airplane and the chair are gradually improved along with training iterations.

7. Limitations

Although achieving remarkable results, our method still has some limitations. Firstly, our pseudo label generation method is based on the local rigidity assumption. For scenes with strongly non-rigid motion, the local rigidity assumption may be not suitable. In practice, these scenes are not common. Secondly, our method generates pseudo labels by enforcing region-wise alignments. Occluded regions without valid counterparts will affect our method. To evaluate the impact of occlusion issues on our method, in Sec. 5.1.2, we conduct experiments on real-world point cloud data with occluded regions. As shown in Table 2 and Table 3, compared with existing approaches, our method achieve the best results. Dealing with non-rigid motions and occlusion issues is worthy of in-depth exploration in our future work.

Acknowledgments

This study is supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). This research is also supported by the National Research Foundation, Singapore, under its AI Singapore Programme (AISG Award No: AISG-RP-2018-003), and the Ministry of Education, Singapore, under its Academic Research Fund (Tier-2 research grant: MOE-T2EP20220-0007, Tier-1 research grant: RG95/20).

References

[1] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13126–13136, 2021. 2, 3, 5, 6

[2] Aseem Behl, Omid Hosseini Jafari, Siva Karthik Mustikovela, Hassan Abu Alhaija, Carsten Rother, and Andreas Geiger. Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2574–2583, 2017. 2

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

[4] PJ Besl and Neil D McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 14(02):239–256, 1992. 3, 4

[5] Kent Fujiwara, Ko Nishino, Jun Takamatsu, Bo Zheng, and Katsushi Ikeuchi. Locally rigid globally non-rigid surface registration. In *2011 International Conference on Computer Vision*, pages 1527–1534. IEEE, 2011. 3

[6] 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*, pages 5692–5703, 2021. 2

[7] 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*, pages 3254–3263, 2019. 2, 5, 6

[8] Pan He, Patrick Emami, Sanjay Ranka, and Anand Rangarajan. Learning scene dynamics from point cloud sequences. *International Journal of Computer Vision*, pages 1–27, 2022. 2

[9] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. Sphereflow: 6 dof scene flow from rgb-d pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3526–3533, 2014. 2

[10] Mariano Jaimez, Mohamed Souiai, Jörg Stückler, Javier Gonzalez-Jimenez, and Daniel Cremers. Motion cooperation: Smooth piece-wise rigid scene flow from rgb-d images. In *2015 International Conference on 3D Vision*, pages 64–72. IEEE, 2015. 2

[11] Yang Jiao, Trac D Tran, and Guangming Shi. Effiscene: Efficient per-pixel rigidity inference for unsupervised joint learning of optical flow, depth, camera pose and motion segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5538–5547, 2021. 2

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[13] 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. 2, 5, 6

[14] Suryansh Kumar, Yuchao Dai, and Hongdong Li. Monocular dense 3d reconstruction of a complex dynamic scene from two perspective frames. In *Proceedings of the IEEE international conference on computer vision*, pages 4649–4657, 2017. 2

[15] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 364–373, 2021. 2

[16] 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. 2, 4, 5, 6

[17] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3d point clouds. *ISPRS journal of photogrammetry and remote sensing*, 143:39–47, 2018. 3, 4, 5

[18] Liang Liu, Guangyao Zhai, Wenlong Ye, and Yong Liu. Unsupervised learning of scene flow estimation fusing with local rigidity. In *IJCAI*, pages 876–882, 2019. 2

[19] 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*, pages 529–537, 2019. 2, 5, 6

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

[21] Zhaoyang Lv, Kihwan Kim, Alejandro Troccoli, Deqing Sun, James M Rehg, and Jan Kautz. Learning rigidity in dynamic scenes with a moving camera for 3d motion field estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 468–484, 2018. 2

[22] 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*, pages 3614–3622, 2019. 2

[23] D Man and A Vision. A computational investigation into the human representation and processing of visual information, 1982. 2

[24] 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. 5

[25] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference*

on computer vision and pattern recognition, pages 3061–3070, 2015. 1, 2, 5

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

[27] 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*, pages 11177–11185, 2020. 2, 6

[28] 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

[29] 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, 6

[30] 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 16*, pages 527–544. Springer, 2020. 2, 5, 6

[31] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 3

[32] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: science and systems*, volume 2, page 435. Seattle, WA, 2009. 3

[33] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8375–8384, 2021. 2

[34] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. In *2020 International Conference on 3D Vision (3DV)*, pages 150–159. IEEE, 2020. 2, 5, 6

[35] 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. 1, 2

[36] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a rigid motion prior. In *2011 International Conference on Computer Vision*, pages 1291–1298. IEEE, 2011. 2

[37] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, 2013. 2

[38] 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. 2

[39] Yue Wang and Justin Solomon. Prnet: self-supervised learning for partial-to-partial registration. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 8814–8826, 2019. 3, 4

[40] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019. 3, 4

[41] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6954–6963, 2021. 2

[42] 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, 5, 6