Weakly Supervised Learning of Rigid 3D Scene Flow

Zan Gojcic[1,2] Or Litany [2,3] Andreas Wieser[1] Leonidas J. Guibas[2] Tolga Birdal[2]
[1]ETH Zurich [2]Stanford University [3]NVIDIA
3dsceneflow.github.io

Abstract

We propose a data-driven scene flow estimation algorithm exploiting the observation that many 3D scenes can be explained by a collection of agents moving as rigid bodies. At the core of our method lies a deep architecture able to reason at the ***object-level*** *by considering 3D scene flow in conjunction with other 3D tasks. This object level abstraction enables us to relax the requirement for dense scene flow supervision with simpler binary background segmentation mask and ego-motion annotations. Our mild supervision requirements make our method well suited for recently released massive data collections for autonomous driving, which do not contain dense scene flow annotations. As output, our model provides low-level cues like pointwise flow and higher-level cues such as holistic scene understanding at the level of rigid objects. We further propose a test-time optimization refining the predicted rigid scene flow. We showcase the effectiveness and generalization capacity of our method on four different autonomous driving datasets. We release our source code and pre-trained models under* github.com/zgojcic/Rigid3DSceneFlow.

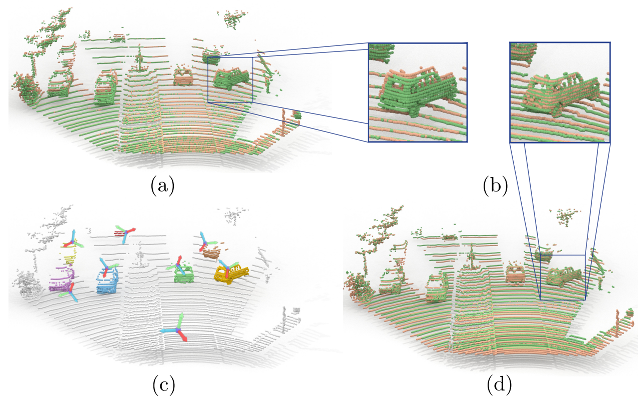


Figure 1: Our network takes two successive frames as input (a), and outputs a set of transformation parameters for each segmented rigid agent (c) which are used to recover per-point rigid scene flow. After applying the predicted flow to the first point cloud, the two frames are aligned (b, d).

1. Introduction

Understanding dynamic 3D environments is a core challenge in computer vision and robotics. In particular, applications such as self-driving and robot navigation rely upon a robust perception of dynamically changing 3D scenes. To equip autonomous agents with the ability to infer spatiotemporal geometric properties, there has recently been an increased interest in *3D scene flow* as a form of low-level dynamic scene representation [37, 67, 73, 51, 49, 54]. Scene flow is the 3D motion field of points in the scene [69] and is a generalization of 2D optical flow. In fact, optical flow [3, 24] can be understood as the projection of the scene flow onto a camera image plane [14]. Such dense motion fields can serve bottom-up approaches for high-level dynamic scene understanding tasks like semantic segmentation [38] or motion perception. However, representing dynamics via a free form velocity field has two major disadvantages. First, in most applications of interest, dynamics are attributed to rigid object motion [9, 42]. This notion has been extensively exploited in robotics [8, 13, 7, 6] and holds especially for vehicles in autonomous driving. Predicting unconstrained per-point flow may lead to non-viable results, e.g. parts of the same car might move in different directions. Second, accurately learning direct flow estimation necessitates dense supervision that is expensive to acquire and prone to annotation errors. As a result, many methods have resorted to training on simulated data [41, 73, 51], yet this comes at the price of a non-negligible domain gap. Other methods have attempted to solve the problem in a completely unsupervised manner [67, 75, 44], however they fail to provide competitive performance. In Fig. 2 we illustrate these two extremes of no- and full-supervision while spanning the many intermediate possibilities, sorted according to the annotation effort[1].

Based on this observation, we seek a sweet spot between supervision effort and performance. To this end, we propose a scene abstraction approach that uses rigid objects as the basic components. More specifically, by splitting the scene into foreground (movable objects) and background (static objects), we explain the background (BG) flow as

[1]Note that no prior work exists at different points on the spectrum given in Fig. 2. This leaves ample room for future exploration.

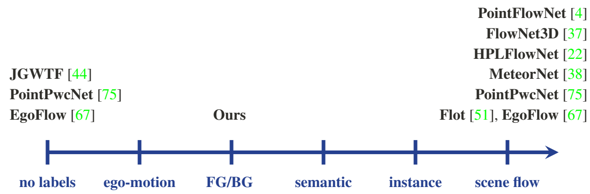


Figure 2: Recent scene flow methods either use full supervision (and suffer from domain gap) or no-supervision (and suffer from reduced performance). Instead, our method uses weak supervision and benefits from the best of both worlds.

the sensor ego-motion and the foreground (FG) flow as clusters of rigidly moving entities. As a result, we tackle both aforementioned challenges at the same time: (1) we enforce a rigidity constraint to get meaningful and more accurate (foreground and background) flow, (2) we can relax the requirement for dense flow supervision with a much simpler binary mask annotation and ego-motion that can often be extracted directly from the agent's IMU. The result is a weakly supervised method for accurate flow estimation that, unlike completely unsupervised approaches, outperforms the previous state of the art (SoTA) by a significant margin. For example, reducing the end-point-error on the *lidarKITTI* dataset by more than 30 cm relative to the SoTA. At the same time, our result provides an interpretable and readily usable object-level scene representation. In brief, our contributions are:

- We exploit the geometry of the rigid scene flow problem to introduce an inductive bias into our network. This allows us to learn from weak supervision signals: background masks and ego-motion.
- Our data-driven method decomposes the scene into rigidly moving agents enabling us to reason on the level of objects rather than points. We use this notion to propose a new test-time optimization, further refining the flow predictions.
- Our method is backed by a novel, flexible, scene flow backbone, which can be adapted to solve various tasks.

As a result of these contributions, our method greatly outperforms the SoTA on several benchmarks: *FT3D* [41], *stereoKITTI* [42], *lidarKITTI* [20], and *semanticKITTI* [5], while generalizing to the *waymo-open* dataset [64] without additional fine-tuning.

2. Related Work

Data Driven 3D Scene Flow. While there is extensive literature on traditional 3D scene flow [69, 27, 74, 31, 65, 68, 48, 60, 61, 29, 17, 14, 9, 50], we focus our attention on recent data-driven methods that emerged based on advances in deep learning on unordered point sets [52, 82, 1].

Early methods for 3D scene flow estimation mimicked their 2D counterparts. SceneFlowNet [41] used *2D optical flow and disparity maps* to estimate the 3D scene flow. FlowNet3D [37] successfully adopted the ideas from FlowNet [28, 16]. FlowNet3D++ [73] extended FlowNet3D to incorporate additional geometric constraints. MeteorNet [38] used multiple temporally ordered frames to improve the accuracy of the inferred flow. Wang *et al.* [71] incorporated a continuous convolution into a 3D-FCN [66] to undo both the ego-motion and object-motion in two consecutive LiDAR frames. PointRNN [19] used recurrent neural networks to model temporal point sets, which yields 3D scene flow as a by-product. HPLFlowNet [22] ordered the points into a permutohedral lattice of SplatNet [63] to apply bilateral convolution layers. This allowed efficient and robust non-rigid 3D flow computation. Both OccFlow [49] and CaSPR [54] introduced spatio-temporal representations to continuously and densely estimate the scene flow. Mustafa and Hilton used semantic coherence between multiple frames to improve 4D scene flow estimation, cosegmentation, and reconstruction [46]. Mittal *et al.* [44] and PointPWCNet [75] proposed self-supervised losses to infer the scene flow in an end-to-end manner. Finally, FLOT [51] proposed a simple correspondence-based end-to-end scene flow network. While our backbone also estimates correspondences, decomposing the scene into rigid agents provides us further higher-level scene understanding and enables test-time optimization while requiring less supervision.

Local Rigidity and Multi-body Motion. Flow estimation has also been tackled by imposing physical priors such as multi-body rigidity. Initial attempts involved factorization [11, 35] to separate independently moving objects. Golyanik *et al.* [21] used rigidity constraints on over-segmentation of RGBD-frames. [45] used detection & tracking to constrain the flow. Vogel *et al.* [70] modeled scene flow using piecewise rigidly moving planar patches. Dewan *et al.* [15] used 3D descriptors to enforce local geometric constancy on a factor graph. GraphFlow [2] and SphereFlow [25] considered large motions and relied on sparse keypoints that are not repeatable in 3D [57, 76]. Similar to us, Jamiez *et al.* [30] and recent Dynamic-SLAM pipelines [26, 58, 62] assumed that clustering would yield motion segmentation. In fact, MaskFusion [56] and EMFusion [62] explicitly used Mask-RCNN [23] to this end.

On a data driven front, considering the rigidity in the scenes [39], Ma *et al.* [40] made use of depth and flow estimates from a stereo RGB-D setup within an optimization framework to obtain the 3D motion of each instance. This method relied upon a given instance segmentation, a difficult problem to solve even for the SoTA approaches [72, 81, 77]. Based on VoxelNet [85], PointFlowNet [4] jointly predicted 3D scene flow, bounding boxes, and rigid motion of objects in the scene. Yi *et al.* [80] used a PointNet++ [53] based flow estimator for piecewise rigid 3D part induction.

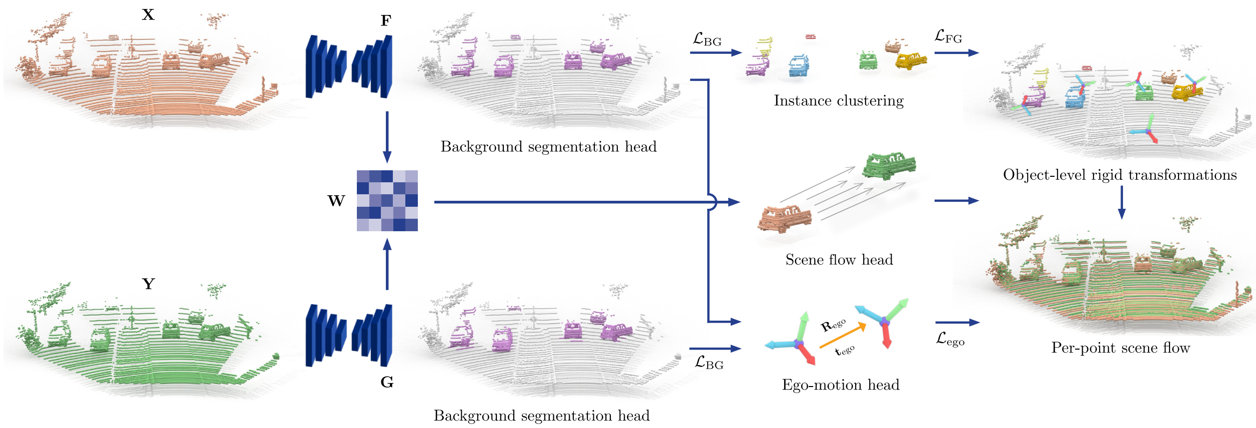


Figure 3: Architecture of our weakly-supervised scene flow estimation pipeline. Our module consumes point clouds $\mathbf{X}$ and $\mathbf{Y}$ of two consecutive frames and estimates per-object transformation parameters $\{\mathbf{T}\}_{k=1}^{K-1}$, ego-motion $\mathbf{T}_{\text{ego}}$, and object masks $\{\mathbf{z}\}_{k=1}^{K}$. These outputs can be combined into an object-level scene abstraction and pointwise rigid scene flow.

3. Method

Problem setting. Suppose that we observe a pair of 3D scenes $\mathbf{X}$ and $\mathbf{Y}$ acquired by a *single moving observer* in two consecutive instants t_0 and t_1, respectively. Here, $\mathbf{X} \in \mathbb{R}^{3\times N} = \{\mathbf{x}_i \in \mathbb{R}^3\}_i$ denotes a point cloud (so does $\mathbf{Y}$) and $\mathbf{V} \in \mathbb{R}^{3\times N} = \{\mathbf{v}_i \in \mathbb{R}^3\}_i$ its corresponding vector field in 3D s.t. $\mathbf{X} + \mathbf{V} \approx \mathbf{Y}$ relates frame t_0 to t_1. We further assume that $\mathbf{X}$, $\mathbf{Y}$, and also $\mathbf{V}$ are *multi-body i.e.* composed of multiple objects. Hence, $\mathbf{V}$ can be clustered into K objects $\mathcal{V} = \{\mathbf{V}_k \in \mathbb{R}^{3\times N_k}\}_{k=1}^{K}$ each of which follows rigid dynamics, *i.e.* $\mathbf{V}$ can be summarized by a set of K rigid transformations $\mathcal{T} \equiv \{\mathbf{T}_k \in SE(3)\}_{k=1}^{K}$ such that $\mathbf{V} \approx \{\mathbf{T}_k \circ \mathbf{X}_k - \mathbf{X}_k\}_{k=1}^{K}$ where $K \ll N$ and:

$$SE(3) = \left\{ \mathbf{T} \in \mathbb{R}^{4\times 4} \colon \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \right\}, \tag{1}$$

$\mathbf{R} \in SO(3)$, $\mathbf{t} \in \mathbb{R}^3$.[2] The motion of the immobile *background*, determines the *ego-motion* $\mathbf{T}_{\text{ego}} \subset \mathcal{T}$.

Summary. We refrain from directly predicting unconstrained pointwise flow vectors and rather aim to estimate $\mathcal{T} \equiv \{\mathbf{T}_k\}_{k=1}^{K}$ for all rigid bodies from which the entire scene flow $\mathcal{V} = \{\mathbf{V}_k\}_{k=1}^{K}$ can be recovered. To this end, we propose to *learn* the task of rigid flow estimation by solving an optimization problem composed of a set of loss functions as illustrated in Fig. 3. To reason on the level of objects, we use both instance masking and motion. To obtain the masks, we use a FG / BG prediction module in conjunction with an FG clustering. To estimate ego-motion, we run a differentiable registration on the BGs extracted from both point sets. The motions of the individual objects in the FGs are obtained similarly under the assumption of local-rigidity. We avoid flow-level or instance-level supervision altogether and only assume the availability of the *binary FG/BG annotations* and *ego-motion information* as a much weaker supervision signal than dense scene flow. In the sequel, we first describe our formulation of the individual objectives (§ 3.1) before proceeding to our network architecture (§ 3.2). Additional details are available in our supplement.

[2] We denote the action of $\mathbf{T}$ as $\mathbf{X}' = \mathbf{T} \circ \mathbf{X}$ and $\hat{\mathbf{X}}' = \mathbf{T}\hat{\mathbf{X}}$ where $\hat{\mathbf{X}} \in \mathbb{R}^{4\times N}$ is the *homogenized* $\mathbf{X}$.

3.1. Energy Formulation

Our solution to the 3D scene flow estimation is attained as the minimum of a non-convex energy composed of a BG segmentation loss $\mathcal{L}_{\text{BG}}$, an ego-motion loss $\mathcal{L}_{\text{ego}}$, and an FG loss $\mathcal{L}_{\text{FG}}$:

$$\mathbf{\Gamma}^\star = \operatorname*{argmin}_{\mathbf{\Gamma}} \mathcal{L}_{\text{BG}} + \mathcal{L}_{\text{ego}} + \mathcal{L}_{\text{FG}} \tag{2}$$

where the optimal rigid scene flow $(\mathcal{V}^\star, \mathcal{T}^\star)$ results from the output of a *deep neural network* φ with learnable parameters $\mathbf{\Gamma}$: $(\mathcal{V}^\star, \mathcal{T}^\star) = \varphi_{\mathbf{\Gamma}^\star}(\mathbf{X}, \mathbf{Y})$. Next, we detail the individual loss terms; each involves an *unknown* or *latent variable* obtained as a network prediction as specified in § 3.2.

Background segmentation error ($\mathcal{L}_{\text{BG}}$). To decompose the scene into agents that move as rigid bodies, we follow a coarse-to-fine approach. In the first step we aim to split the background and foreground points, where the foreground represents all points belonging to the movable objects (e.g., cars, cyclists, people, …). In order to learn this binary segmentation of a point cloud, we minimize the loss $\mathcal{L}_{\text{BG}} = \frac{1}{2}(\mathcal{L}_{\text{BG}}^{\mathbf{X}} + \mathcal{L}_{\text{BG}}^{\mathbf{Y}})$ where:

$$\mathcal{L}_{\text{BG}}^{\mathbf{X}} = \frac{1}{N} \sum_{i=1}^{N} \text{BCE}(h_i^{\mathbf{X}}, \bar{h}_i^{\mathbf{X}}). \tag{3}$$

$\bar{\mathbf{h}}^{\mathbf{X}}$, $\bar{\mathbf{h}}^{\mathbf{Y}}$ denote the GT binary masks of point clouds $\mathbf{X}$ and $\mathbf{Y}$, respectively. $\mathbf{h}^{\mathbf{X}} = \{h_i^{\mathbf{X}}\}_{i=1}^{N}$ and $\mathbf{h}^{\mathbf{Y}} = \{h_i^{\mathbf{Y}}\}_{i=1}^{N}$ are the inferred foreground probabilities of points in $\mathbf{X}$ and $\mathbf{Y}$ yet to be clarified in § 3.2, and $\mathrm{BCE}(h_i, \bar{h}_i) = \bar{h}_i \log(h_i) + (1-\bar{h}_i)\log(1-h_i)$.

Ego-motion error ($\mathcal{L}_{\text{ego}}$). The scene flow of the physically static background can be fully explained by its transformation parameters, the ego-motion. We estimate these parameters by first extracting the background points of both the source and target point cloud[3]. To reduce the computational complexity, we then randomly sample $N^b = 1024$ points therefrom such that $\mathbf{X}^b \in \mathbb{R}^{3\times N^b} \subset \mathbf{X}$ and $\mathbf{Y}^b \in \mathbb{R}^{3\times N^b} \subset \mathbf{Y}$. The goal of ego-motion estimation is to compute the optimal $\mathbf{R}_{\text{ego}} \in \mathrm{SO}(3)$ and $\mathbf{t}_{\text{ego}} \in \mathbb{R}^3$ in the weighted least-squares sense

$$\mathbf{R}^\star_{\text{ego}}, \mathbf{t}^\star_{\text{ego}} = \underset{\mathbf{R}_{\text{ego}},\mathbf{t}_{\text{ego}}}{\operatorname{argmin}} \sum_{l=1}^{N^b} w_l \|\mathbf{R}_{\text{ego}}\mathbf{x}_l^b + \mathbf{t}_{\text{ego}} - \phi(\mathbf{x}_l^b, \mathbf{Y}^b)\|^2,$$

where $\phi(\mathbf{x}, \mathbf{Y})$ is a *soft assignment function* returning a point from $\mathbf{Y}$ that corresponds to $\mathbf{x} \in \mathbf{X}$. The weights w_l will be clarified below.

We approximate the optimal soft assignment ϕ via the entropy-regularized Sinkhorn algorithm [59, 12, 78]. To this end, we momentarily assume the availability of an affinity matrix $\mathbf{M} \in \mathbb{R}^{N^b \times N^b}$ describing the similarity of the background points in $\mathbf{X}^b$ and $\mathbf{Y}^b$. Prediction of the currently unknown $\mathbf{M}$ will be made precise in § 3.2. Given $\mathbf{M}$, we perform an alternating row and column normalization on it for $k_S \triangleq 3$ iterations, which yields $\mathbf{A} \in \mathbb{R}_+^{N^b \times N^b}$, a *doubly stochastic* (DS) assignment matrix.

In practice, due to the occlusions and sampling pattern, not all background points will have correspondences. We therefore add a slack row and column to $\mathbf{M}$ (hence to $\mathbf{A}$), which enable down-weighting the outliers, while still returning a DS-matrix. The soft correspondence function then reads $\phi(\mathbf{x}_i^b, \mathbf{Y}^b) = \mathbf{Y}^b \mathbf{a}_i / \|\mathbf{a}_i\|_1$ where $\mathbf{a}_i$ is the i-th column of $\mathbf{A}$ after removing the slack row. Given the correspondences, the ego-motion can be recovered in closed-form using a (differentiable) *weighted* Kabsch algorithm [34] where the weights w_i are obtained as the total contribution $w_i = \sum_{j=1}^{N^b} a_{ij}$. Our ego-motion penalty measures the $l1$-discrepancy between the points transformed with the estimated $(\mathbf{R}_{\text{ego}}, \mathbf{t}_{\text{ego}})$ and the GT parameters $(\bar{\mathbf{R}}_{\text{ego}}, \bar{\mathbf{t}}_{\text{ego}})$:

$$\mathcal{L}_{\text{trans}} = \frac{1}{B}\sum_{i=1}^{B} \|(\bar{\mathbf{R}}_{\text{ego}}\mathbf{x}_i^b + \bar{\mathbf{t}}_{\text{ego}}) - (\mathbf{R}_{\text{ego}}\mathbf{x}_i^b + \mathbf{t}_{\text{ego}})\|_1,$$

where B denotes the number of all background points in point cloud $\mathbf{X}$. To stabilize the training, we further add a regularizer loss that discourages the assignment of large values to the slack rows and columns [78]:

$$\mathcal{L}_{\text{inlier}} = \frac{1}{N^b}\sum_{i=1}^{N^b}\Big(1 - \sum_{j=1}^{N^b} a_{ij}\Big) + \frac{1}{N^b}\sum_{j=1}^{N^b}\Big(1 - \sum_{i=1}^{N^b} a_{ij}\Big).$$

The overall ego-motion loss is then the weighted sum:

$$\mathcal{L}_{\text{ego}} = \mathcal{L}_{\text{trans}} + \lambda_{\text{inlier}}\mathcal{L}_{\text{inlier}} \tag{4}$$

where $\lambda_{\text{inlier}} := 0.005$ in all our experiments.

FG instance-level rigidity error ($\mathcal{L}_{\text{FG}}$). To define our *per-instance rigidity loss*, we assume the availability of the following entities: **(i)** $\mathbf{X}^f$, the points of the source frame belonging to the FG; **(ii)** $\mathbf{V}^f$, the flow vectors associated to $\mathbf{X}^f$; and **(iii)** foreground clusters $\mathcal{C} = \{\mathbf{C}^k \in \mathbb{R}^{3\times N_k} = \{\mathbf{c}_j^k \in \mathbb{R}^3\}_j\}_{k=1}^{N^C}$ aggregating the individual rigid entities. (i) is a by-product of BG segmentation, *i.e.* during training, the indices of these points are obtained from the GT mask and during inference by thresholding the *inferred* FG probabilities. The flow vectors in (ii) are the result of the *scene flow module* (see § 3.2). Finally, (iii) is computed by a simple DBSCAN clustering [18] of the 3D coordinates in $\mathbf{X}^f$. The DBSCAN clustering is based on the hypothesis that the foreground objects scattered across the scene are naturally separated by *void space* [33]. We refrain from using a data driven instance segmentation module because our simple approach alleviates the need for instance segmentation labels.

Our rigidity loss $\mathcal{L}_{\text{rigid}}$ encourages the predicted flow vectors $\mathbf{V}_k^f$ of each cluster k to be *congruent*, *i.e.* $\mathbf{V}_k^f$ can be well approximated by a rigid transformation $\mathbf{T}_k$ composed of the rotation $\mathbf{R}_k$ and translation $\mathbf{t}_k$:

$$\mathcal{L}_{\text{rigid}} = \frac{1}{N^c}\sum_{k=1}^{N^c}\frac{1}{N^k}\sum_{j=1}^{N^k}\|\mathbf{R}_k\mathbf{c}_j^k + \mathbf{t}_k - (\mathbf{c}_j^k + \mathbf{v}_j^k)\|_1 \tag{5}$$

The supervision signals $\mathbf{R}_k$ and $\mathbf{t}_k$ are computed on the fly, such that they best explain the underlying flow $\mathbf{V}^f$:

$$\mathbf{T}_k^\star = \underset{\mathbf{T}_k}{\operatorname{argmin}} \|\mathbf{T}_k \circ \mathbf{C}_k - (\mathbf{C}_k + \mathbf{V}_k^f)\|. \tag{6}$$

We solve Eq (6) for each individual cluster once again using the Kabsch algorithm [34]. We additionally complement the per-cluster rigidity objective with a two way Chamfer distance (CD) computed across all the foreground points:

$$\mathcal{L}_{\text{CD}} = \sum_{\mathbf{x}\in\mathbf{X}_v^f}\min_{\mathbf{y}\in\mathbf{Y}^f}\|\mathbf{x}-\mathbf{y}\|_2 + \sum_{\mathbf{y}\in\mathbf{Y}^f}\min_{\mathbf{x}\in\mathbf{X}_v^f}\|\mathbf{x}-\mathbf{y}\|_2 \tag{7}$$

where $\mathbf{X}_v^f := \mathbf{X}^f + \mathbf{V}^f$. The overall FG loss is then a weighted sum of the above objectives:

$$\mathcal{L}_{\text{FG}} = \mathcal{L}_{\text{rigid}} + \lambda_{\text{CD}}\mathcal{L}_{\text{CD}} \tag{8}$$

where $\lambda_{\text{CD}} := 0.5$ in all our experiments.

[3]During training we use the GT BG segmentation mask $1 - \bar{\mathbf{h}}$, while during inference we threshold the inferred FG probabilities $1 - \mathbf{h}$.

3.2. Network implementation

Provided a large dataset with *FG-BG mask* annotations as well as *ego-motion*, we learn to minimize Eq (2) using a deep neural network φ_{Γ} as shown in Fig. 3. In the sequel, we describe the individual modules of our network and the full inference of rigid scene flow. We refer the reader to the supplement for more details.

Backbone. Our formulation (§ 3.1) involves the estimation of different entities, requiring our network to solve *multiple tasks* similar to [32]. While it would be possible to deploy a specialized network for each task, this would increase the memory footprint and would not encourage tasks to reinforce each other [84, 83]. Instead, we propose a flexible backbone suited for solving multiple tasks through specialized heads. Specifically, our backbone network is based on Minkowski-Net [10] and follows a U-Net [55]-like encoder-decoder architecture with skip connections. Its input is a *sparsely* voxelized point cloud $\mathbf{X}^v \in \mathbb{R}^{3\times N^v}$ and its outputs are per-point latent features $\mathbf{F}^v \in \mathbb{R}^{64\times N^v}$. The same backbone with shared weights is also applied to $\mathbf{Y}^v \in \mathbb{R}^{3\times M^v}$ to obtain the latent features $\mathbf{G}^v \in \mathbb{R}^{64\times M^v}$. In the following, we omit the superscript v for clarity and unless specified differently, use $\mathbf{X}$ and $\mathbf{Y}$ to refer to the voxelized point clouds and $\mathbf{F}$ and $\mathbf{G}$ to refer to their associated latent features.

Background segmentation head. Our background segmentation head consists of two sparse convolutional layers with instance normalization and the ReLU [47] activation function after the first one. It takes the latent features $\mathbf{F}$ and $\mathbf{G}$ as input and outputs per-point foreground probabilities $\mathbf{h}^{\mathbf{X}} \in \mathbb{R}^{N^v}$ and $\mathbf{h}^{\mathbf{Y}} \in \mathbb{R}^{M^v}$.

Ego-motion head. Given the latent features $\mathbf{F}^b$ and $\mathbf{G}^b$ of the background points $\mathbf{X}^b$ and $\mathbf{Y}^b$, the ego-motion head computes the affinity matrix $\mathbf{M} \in \mathbb{R}^{N^b\times N^b}$ s.t.

$$M_{ij} = \exp\big(-\|\mathbf{f}_i^b - \mathbf{g}_j^b\|/\tau_{\text{ego}}\big), \tag{9}$$

where τ_{ego} controls the *softness* of the correspondences.

Scene flow head. Based on the notion that scene flow is tightly coupled to correspondences [51], we now devise our scene flow head. To assure differentiability, we estimate *soft* correspondences. To allow for large motion, we measure the similarity in the latent space of *features* $\mathbf{F}$ and $\mathbf{G}$ instead of in the physical space. Hence, we find the corresponding points in $\mathbf{X}$ and $\mathbf{Y}$ as:

$$\mathbf{X}_c := \mathbf{Y}\mathbf{D}, \;\; d_{ij} := \text{softmax}(-\frac{1}{\tau_{\text{flow}}}\|\mathbf{f}_i - \mathbf{g}_j\|_2) \tag{10}$$

where τ_{flow} is again a learnable temperature that controls the *softness* of the correspondences in the same manner as τ_{ego} above. Soft correspondences can be used to compute the initial flow estimate as $\mathbf{V}^{\text{init}} := \mathbf{X}_c - \mathbf{X}$. However, this initial flow vector field is likely to be noisy due to large motions, sampling, and imperfect latent features and thus still has to be refined [51]. Our *refinement* module takes $\mathbf{V}^{\text{init}}$ as input and locally smoothens it by estimating a residual flow $\Delta\mathbf{V}^{\text{init}}$ thorough a series of sparse convolutional layers. The refined scene flow $\mathbf{V} \in \mathbb{R}^{3\times N^v}$ is then obtained as: $\mathbf{V} = \mathbf{V}^{\text{init}} + \Delta\mathbf{V}^{\text{init}}$. The detailed architecture of the scene flow head is given in the supplement.

From transformations to per-point rigid scene flow. The output of our multi-task network comprises of: (i) transformation parameters of the ego-motion $\mathbf{T}_{\text{ego}}$ and individual clusters $\{\mathbf{T}_k\}_{k=1}^{K-1}$; (ii) object level masks $\{\mathbf{z}_k\}_{k=1}^{K}$; and (iii) unconstrained pointwise scene flow estimates $\mathbf{V}$. The *pointwise rigid scene flow* $\mathbf{V}^{\text{rigid}}$ can then be recovered as $\mathbf{V}^{\text{rigid}} \approxeq \{\mathbf{T}_k \circ \mathbf{X}_k - \mathbf{X}_k\}_{k=1}^{K}$, where $\mathbf{X}_k$ denotes the points of $\mathbf{X}$ belonging to the cluster k according to the inferred object masks $\mathbf{z}_k$. For the points that are neither assigned to the background nor to any of the foreground rigid bodies, we use the unconstrained scene flow predictions $\mathbf{V}$.

Training and implementation details. Our method is implemented in PyTorch using the *MinkowskiEngine* [10]. Unless specified differently, we train our network in an end-to-end manner for 40 epochs (or until convergence), by minimizing Eq (2). We train on a single NVIDIA GTX2080Ti with batch size 8. We use the Adam [36] optimizer with an initial learning rate 10^{-3}, which is decayed every epoch according to an exponential schedule with $\gamma = 0.98$. The whole training takes about two and a half days. The detailed parameters of the Sinkhorn algorithm and DBSCAN clustering are available in the supplement.

Inference. The abstraction of the scene into a collection of rigid bodies enables us to run test-time optimization of their inferred transformation parameters. Specifically, we run an optimization scheme in which we iteratively minimize the closest point distance to the target points. For ego-motion, we index the background points, and for individual clusters, all the foreground points of $\mathbf{Y}$. The indexing is performed using the inferred object masks $\mathbf{z}_k^{\mathbf{X}}$ and $\mathbf{z}_k^{\mathbf{Y}}$. Given the final transformation estimates $\{\mathbf{T}_k^\star\}_{k=1}^{K}$ *pointwise rigid scene flow* can be recomputed as $\mathbf{V}^{\text{rigid}} \approxeq \{\mathbf{T}_k^\star \circ \mathbf{X}_k - \mathbf{X}_k\}_{k=1}^{K}$. A detailed description of our optimization scheme including its run-time is available in the supplement.

In the following, we reintroduce the superscript v to denote the voxelized point clouds. Note that $\mathbf{V}^{\text{rigid}}$ represents the flow for the voxel centers $\mathbf{X}^v$ and during inference, still has to be *transferred* to the points $\mathbf{X}$. We perform this transfer by a simple inverse-distance weighted interpolation:

$$\mathbf{v}_i^\star = \frac{\sum_{j:\mathbf{x}_j^v \in \mathcal{E}(\mathbf{x}_i)} \mathbf{v}_j^{\text{rigid}} \|\mathbf{x}_i - \mathbf{x}_j^v\|_2^{-1}}{\sum_{j:\mathbf{x}_j^v \in \mathcal{E}(\mathbf{x}_i)} \|\mathbf{x}_i - \mathbf{x}_j^v\|_2^{-1}}, \tag{11}$$

where $\mathcal{E}(\cdot)$ returns the set of k-NN in the Euclidean sense.

Dataset	Method	Supervision	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
FT3D	FlowNet3D [37]	Full	0.114	0.412	0.771	0.602
	HPLFlowNet [22]	Full	0.080	0.614	0.855	0.429
	PointPWC-Net [75]	Full	0.059	0.738	0.928	**0.342**
	FLOT [51]	Full	**0.052**	0.732	0.927	0.357
	EgoFlow [67]	Full	0.069	0.670	0.879	0.404
	Ours	Full	**0.052**	**0.746**	**0.936**	0.361
stereoKITTI	Flownet3D [37]	Full	0.177	0.374	0.668	0.527
	HPLFlowNet [22]	Full	0.117	0.478	0.778	0.410
	PointPWC-Net [75]	Full	0.069	0.728	0.888	0.265
	FLOT [51]	Full	0.056	0.755	0.908	0.242
	EgoFlow [67]	Full	0.103	0.488	0.822	0.394
	Ours	Full	**0.042**	**0.849**	**0.959**	**0.208**

Table 1: Evaluation results in a fully supervised setting on *FT3D* and *stereoKITTI* datasets.

Dataset	Method	Supervision	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
lidarKITTI (w/o ground)	PointPWC-Net [75]	Full	0.390	0.387	0.550	0.653
	FLOT [51]	Full	0.653	0.155	0.313	0.837
	Ours (backbone)	Full	0.535	0.262	0.437	0.742
	Ours	Weak	0.150	0.521	0.744	0.450
	Ours+	Weak	0.110	0.745	0.844	0.353
	Ours++	Weak	**0.094**	**0.784**	**0.885**	**0.314**
lidarKITTI (with ground)	PointPWC-Net [75]	Full	0.710	0.114	0.219	0.932
	FLOT [51]	Full	0.773	0.084	0.177	0.943
	MeteorNet [38] [1]	Full	0.277	/	/	/
	Ours (backbone)	Full	0.820	0.102	0.190	0.934
	Ours	Weak	0.133	0.460	0.746	0.527
	Ours+	Weak	0.106	0.673	0.808	0.421
	Ours++	Weak	**0.102**	**0.686**	**0.819**	**0.410**

[1] MeteorNet uses three tine frames and was trained on 100 and evaluated on 42 scenes of this dataset.

Table 2: Evaluation results on *lidarKITTI*. Ours (backbone) denotes our model from § 4.3 trained with full supervision on *FT3D*. Ours are the direct estimates of our pipeline. Ours+ and Ours++ additionally denote test-time optimization of only ego-motion and all rigid bodies, respectively.

4. Experimental Evaluation

In this section, we first describe the datasets (§ 4.1) and evaluation metrics (§ 4.2) used in our experiments. We start the evaluation, by assessing the performance of our backbone flow estimation network under full supervision on point clouds lifted from stereo images (§ 4.3). We then proceed to evaluate our full pipeline in a weakly supervised setting on real LiDAR scans (§ 4.4). Finally, we showcase the generalization capability of our method (§ 4.5) and justify our design choices in an ablation study (§ 4.6).

4.1. Datasets

For all datasets, we follow a common preprocessing step [37, 22] and remove points whose depth or distance to the sensor is larger than 35 m. For training and evaluation, we randomly sample 8192 points from both frames independently. A detailed description of the datasets and preprocessing steps is available in the supplement.

FlyingThings3D (FT3D). [41] is a large-scale stereo dataset of synthetic man-made objects that are scattered in space and move randomly between the two frames. We generate the point clouds and GT scene flow in accordance with [22]. FT3D consists of 19640 training examples (from which we use 3928 for validation) and 3824 test examples. Note, FT3D is only used for training in the fully supervised evaluation of our backbone and scene flow head (§ 4.3).

stereoKITTI. [42, 43] is a real world scene flow dataset with 142 point cloud pairs, which are all used for testing. The point clouds and GT scene flow are obtained by lifting the annotated disparity maps and optical flow to 3D [22]. As a consequence, the points of the two frames are under direct correspondence. We remove the ground points by naive thresholding of the height coordinate [37, 22].

lidarKITTI. [20] is a real world dataset acquired with a Velodyne 64-beam LiDAR. It consists of the same 142 pairs as *stereoKITTI*. GT is obtained by projecting the point clouds to the image plane and assigning them the annotated 3D flow vectors. In this dataset, the points of the two input frames are not in direct correspondence and have a typical sampling pattern of a LiDAR sensor.

semanticKITTI. [5] provides per point semantic labels and accurate ego-motion for 21 LiDAR sequences of the KITTI odometry dataset. It is split into eleven (00-10) LiDAR sequences for training and eleven (11-21) for testing. We use sequences 03 and 05 for validation and the remaining nine for training. SemanticKITTI is used to train our method in a weakly supervised manner (§ 4.4 to § 4.6). Note that this dataset does not contain dense scene flow annotations.

4.2. Evaluation metrics

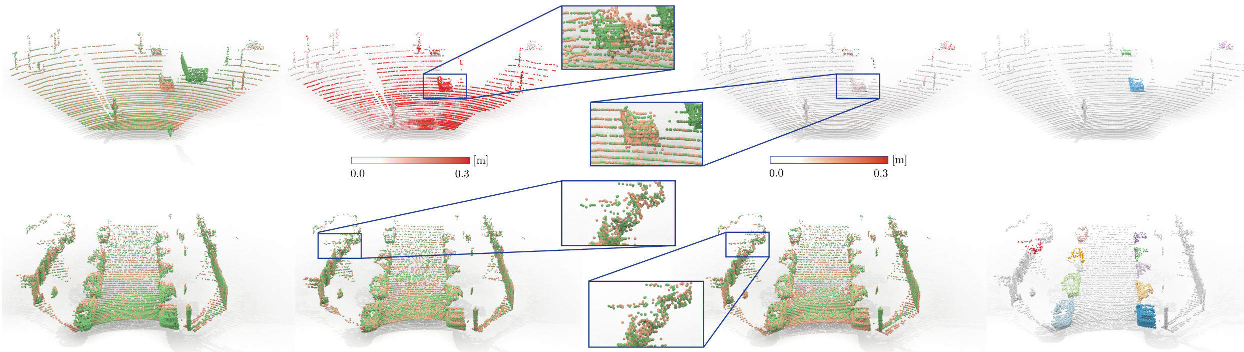
We use standard evaluation metrics to assess the performance of our approach and compare it with SoTA methods, *FlowNet3D* [37], *HPLFlowNet* [22], *PointPWCNet* [75], FLOT [51], and EgoFlow [67]. Our main evaluation metric is the 3D end-point-error (*EPE3D*), defined as the mean $l2$ distance between the predicted and GT scene flow.

Additionally, we follow [37, 22] and also report: (i) strict accuracy (*Acc3DS*), defined as the percentage of points whose *EPE3D* < 0.05 m or relative error < 0.05, (ii) relaxed accuracy (*Acc3DR*) that denotes the ratio of points whose *EPE3D* < 0.10 m or relative error < 0.10, and (iii) *Outliers*, i.e. the ratio of points whose *EPE3D* > 0.30 m or relative error > 0.10.

For the experiments in a weakly supervised setting, we also report the relative angular error (*RAE*) and the relative translation error (*RTE*) of the estimated ego-motion.

4.3. Our backbone under full supervision

A core module of our proposed pipeline is the backbone network described in § 3.2. It is therefore valuable to first assess its performance in conjunction with the scene flow prediction head, before turning to evaluate the performance of our entire weakly-supervised pipeline. To this end, we follow the traditional setting used by our competitors and train in a fully supervised manner on *FT3D* by minimizing



(a) Input point clouds (b) FLOT (c) Zoom in (d) Ours (rigid flow) (e) Ours (rigid objects)

Figure 4: Qualitative results of our weakly supervised method on *lidarKITTI* (top) and *waymo open* (bottom). For improved visibility, the *EPE3D* (top row b,c) is clipped to the range between 0.0 m (white) at 0.3m (red). As a result of predicting an unconstrained pointwise sceneflow, the rigid objects (car) in the results of FLOT might get deformed (c).

the $l1$ distance between the predicted and GT scene flow. We then evaluate our model on both *FT3D* and *stereoKITTI*.

When evaluated on *FT3D*, our method performs on par with FLOT [51] in terms of *EPE3D* and outperforms all methods in terms of *Acc3DS* and *Acc3DR* (Tab. 1). More importantly, it achieves superior generalization performance on *stereoKITTI*, where it consistently outperforms SoTA in all evaluation metrics, setting a new SoTA with 0.042 m *EPE3D* ($\approx$ 1.5 cm better than the closest competitor). Based on these results, we conclude that sparse convolutions are an effective backbone for scene-flow estimation and that our simple scene flow head can match the performance of SoTA while enabling better generalization.

4.4. Our pipeline under weak supervision

Setting. Point clouds in both *FT3D* and *stereoKITTI* are obtained in the same manner: by lifting stereo images to 3D. Hence, their domain gap is relatively small. On the other hand, in LiDAR-based autonomous driving scenarios, point clouds are much sparser and assume a very different sampling pattern, resulting in a much more challenging setting for scene flow estimation.

We evaluate our entire weakly-supervised pipeline in this challenging setting by using *lidarKITTI* dataset. Specifically, we consider two scenarios: 1) we remove ground points by naively thresholding the vertical coordinate, 2) we use the “raw” point clouds that also include the ground points for which the flow estimation is especially difficult.

We train a joint model for both scenarios in a weakly supervised manner using the point clouds from *semanticKITTI*. Unlike *semanticKITTI*, *lidarKITTI* only includes the points and annotations of the objects that are visible within the front camera images. We, therefore, process semanticKITTI in the same manner. Since there are no LiDAR datasets available with scene flow annotations, we use the models trained on *FT3D* for all the baselines.

Evaluation. Fig. 4 and Tab. 2 show that the domain gap between the stereo and LiDAR point clouds is too big for the traditional fully supervised methods to generalize effectively. Indeed, the performance of SoTA methods is up to 10 times worse when compared to the results on *stereoKITTI*. On the other hand, our weakly supervised model predicts accurate rigid scene flow with an *EPE3D* ≈ 0.1 m for both scenarios (after test-time optimization), while also providing an object-level abstraction (Fig. 4). Since our fully supervised backbone model also fails to generalize, we conclude that the crucial advantage of our method does not lie in a stronger backbone, but rather in the ability to train on the same domain. Additional qualitative and quantitative results are available in the supplement.

4.5. Generalization to other datasets.

Waymo open [64] is a recently introduced large-scale autonomous driving dataset that would ideally be used for supervision of 3D scene flow methods. However, it does not provide dense flow annotations. While it does include all the annotations that our weakly supervised approach relies on, we are more interested in using it to evaluate the generalization capability of our method. To this end, we use the first three sequences[4] of the *waymo open* validation set and quantitatively evaluate our weakly supervised model in terms of ego-motion estimation and background segmentation. We provide qualitative results of the rigid scene flow estimation and object-level scene abstraction in Fig. 4.

Remarkably, our model that was trained only on *semanticKITTI* can seamlessly generalize to *waymo open*. When evaluated on the task of ego-motion estimation, it achieves an *RRE* of $0.141°$ and *RTE* of 0.099 m. In the BG-segmentation task, its performance on the foreground points drops to 0.960 precision and 0.689 recall, while on the background points it remains high with 0.957 and 0.996

[4]This results in more than 14k point cloud pairs, which is almost the same size as the whole *semanticKITTI* dataset

$\mathcal{L}_{\text{ego}}$	$\mathcal{L}_{\text{CD}}$	$\mathcal{L}_{\text{rigid}}$	*lidarKITTI* EPE [m] ↓	Acc3DS ↑	Acc3DR ↑	RRE [°] ↓	RTE [m] ↓
	✓	✓	0.721	0.044	0.093	0.476	0.750
✓		✓	0.363	0.044	0.163	0.610	0.342
✓	✓		0.136	0.409	0.712	0.380	0.146
✓	✓	✓	**0.134**	**0.460**	**0.746**	**0.320**	**0.130**

Table 3: Ablation study of the proposed training objective. All models are trained on *semanticKITTI* and evaluated without test-time optimization on *lidarKITTI (with ground)* dataset.

precision and recall, respectively. The drop in foreground performance can be accredited to the domain gap between the datasets [79]; *waymo open* includes many more foreground objects, especially pedestrians, than *semanticKITTI*.

4.6. Ablation Studies

Influence of different loss terms. We compare our model trained with the full loss function to multiple ablations in Tab. 3. Each model is trained on *semanticKITTI* and evaluated *without* test-time optimization on *lidarKITTI* with ground points. Tab. 3 shows that the terms $\mathcal{L}_{\text{ego}}$ and $\mathcal{L}_{\text{CD}}$ are crucial for the performance of our model. Adding $\mathcal{L}_{\text{rigid}}$ further regularizes the performance and leads to an improvement in all evaluation metrics. Note how the terms that are applied only on foreground points (e.g. $\mathcal{L}_{\text{rigid}}$) also improve the ego-motion estimation Tab. 3.

Task-specific networks. Instead of training a single network capable of solving multiple tasks, one could also devise a combination of task-specific networks. We ablate this design choice in Tab. 4 in which we compare our full model to BG segmentation and ego-motion specific networks. Both task specific networks comprise of our backbone with the corresponding head and are trained with the $\mathcal{L}_{\text{BG}}$ and $\mathcal{L}_{\text{ego}}$ objective function, respectively. Tab. 4 shows that the performance of our full model is slightly inferior to the task-specific one when compared on BG segmentation. This is expected, since in our full pipeline the BG segmentation head is also trained nearly in isolation, with a single loss function. On the other hand, our full model outperforms the task-specific ego-motion model, even though the task-specific model is combined with the GT background mask. This shows that individual tasks (*e.g.* flow and ego-motion estimation) can indeed reinforce each other, which leads to better downstream performance.

Pretraining the backbone with full supervision. We analyze the effect of initializing our weakly supervised model with pretrained backbone weights. To this end, we use the backbone weights from the model trained with full supervision in § 4.3. Initializing the backbone with the pretrained model leads to 1.4 cm and 2.3 cm improvement in terms of *EPE3D* on *lidarKITTI* without and with ground points, respectively. Further details and additional metrics of this ablation study are available in the supplement. Note that

Task: BG seg.	Task: ego-motion	BG segmentation: prec. FG ↑	rec. FG ↑	prec. BG ↑	recall. BG ↑	ego-motion: RRE [°] ↓	RTE [m] ↓
semanticKITTI (w/o ground)							
✓		**0.977**	**0.901**	**0.992**	**0.998**	-	-
	✓	-	-	-	-	0.245	0.054
✓	✓	0.971	0.895	0.991	**0.998**	**0.201**	**0.047**
semanticKITTI (with ground)							
✓		**0.970**	**0.911**	**0.996**	**0.999**	-	-
	✓	-	-	-	-	0.307	0.071
✓	✓	0.966	0.904	**0.996**	**0.999**	**0.249**	**0.059**

Table 4: Comparison of our full pipeline with specialized networks for BG segmentation and ego-motion estimation, respectively. Note, we provide GT background masks to the ego-motion specialized network also in the test phase.

in all evaluations presented in Sec. 4 we use the inferior model with randomly initialized weights trained only with weak supervision.

Run time. We now compare our method to FLOT [51] and PointPWC-Net [75] in terms of run-time and number of parameters[5]. We perform the evaluation on a standalone computer with Intel Xeon E5-1650, 32GB RAM, and a single NVIDIA Titan V. For FLOT [51] and PointPWC-Net [75] we use the official implementation provided by the authors. FLOT has the lowest number of trainable parameters (0.11 million) but, with 0.395 seconds on average, also the highest run time per point cloud pair. PointPWC-Net has a larger model with approximately 7.7 million parameters but performs one inference step in 0.147 seconds on average. Finally, our method contains about 8 million parameters and requires 0.154 seconds on average for a single point cloud pair. With added test-time optimization our run time increases to 0.234 seconds on average.

5. Conclusion

Scene flow is the lowest level in a hierarchy of dynamic scene perception. As such, while providing a useful cue to higher-level tasks, it is also the most demanding to supervise. Based on this observation, in this work, we have introduced a novel method that relaxes the dense supervision by integrating flow into a higher-level scene abstraction in the form of multi rigid-body motion. The result is a state-of-the-art flow estimation network that additionally outputs a concise dynamic scene representation. In particular, our mild supervision requirements are well suited for utilizing the annotation level of recently released massive data collections for autonomous driving. In future work, we plan to incorporate cues from multiple frames further seeking temporal consistency as well as increased accuracy.

Acknowledgements. This work is sponsored by Stanford-Ford Alliance, the Samsung GRO program, NSF grant IIS-1763268, the Vannevar Bush Faculty fellowship, and the NVIDIA GPU grant. We thank Barbara Verbost for her help with the visualizations and Davis Rempe for proofreading the article.

[5]The evaluation is performed with 8192 randomly sampled points on the *lidarKITTI* dataset.

References

[1] Eman Ahmed, Alexandre Saint, Abd El Rahman Shabayek, Kseniya Cherenkova, Rig Das, Gleb Gusev, Djamila Aouada, and Björn Ottersten. Deep learning advances on different 3d data representations: A survey. *arXiv preprint arXiv:1808.01462*, 1, 2018. 2

[2] Hassan Abu Alhaija, Anita Sellent, Daniel Kondermann, and Carsten Rother. Graphflow–6d large displacement scene flow via graph matching. In *German Conference on Pattern Recognition*, pages 285–296. Springer, 2015. 2

[3] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995. 1

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

[5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *IEEE International Conf. on Computer Vision*, 2019. 2, 6

[6] Tolga Birdal, Umut Simsekli, Mustafa Onur Eken, and Slobodan Ilic. Bayesian pose graph optimization via bingham distributions and tempered geodesic mcmc. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 308–319. Curran Associates, Inc., 2018. 1

[7] Mai Bui, Tolga Birdal, Haowen Deng, Shadi Albarqouni, Leonidas Guibas, Slobodan Ilic, and Nassir Navab. 6d camera relocalization in ambiguous scenes via continuous multimodal inference. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 139–157, Cham, 2020. Springer International Publishing. 1

[8] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017. 1

[9] Rodrigo L Carceroni and Kiriakos N Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *International Journal of Computer Vision*, 49(2-3):175–214, 2002. 1, 2

[10] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 5

[11] João Paulo Costeira and Takeo Kanade. A multibody factorization method for independently moving objects. *International Journal of Computer Vision*, 29(3):159–179, 1998. 2

[12] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013. 4

[13] Haowen Deng, Mai Bui, Nassir Navab, Leonidas Guibas, Slobodan Ilic, and Tolga Birdal. Deep bingham networks: Dealing with uncertainty and ambiguity in pose estimation. *arXiv preprint arXiv:2012.11002*, 2020. 1

[14] Frederic Devernay, Diana Mateus, and Matthieu Guilbert. Multi-camera scene flow by tracking 3-d points and surfels. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2203–2212. IEEE, 2006. 1, 2

[15] Ayush Dewan, Tim Caselitz, Gian Diego Tipaldi, and Wolfram Burgard. Rigid scene flow for 3d lidar scans. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1765–1770. IEEE, 2016. 2

[16] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015. 2

[17] Mingsong Dou, Sameh Khamis, Yury Degtyarev, Philip Davidson, Sean Ryan Fanello, Adarsh Kowdle, Sergio Orts Escolano, Christoph Rhemann, David Kim, Jonathan Taylor, et al. Fusion4d: Real-time performance capture of challenging scenes. *ACM Transactions on Graphics (TOG)*, 35(4):1–13, 2016. 2

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996. 4

[19] Hehe Fan and Yi Yang. Pointrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019. 2

[20] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2, 6

[21] Vladislav Golyanik, Kihwan Kim, Robert Maier, Matthias Nießner, Didier Stricker, and Jan Kautz. Multiframe scene flow with piecewise rigid motion. In *2017 International Conference on 3D Vision (3DV)*, pages 273–281. IEEE, 2017. 2

[22] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hplflownet: Hierarchical permutohedral lattice flownet for scene flow estimation on large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3254–3263, 2019. 2, 6

[23] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[24] Berthold KP Horn and Brian G Schunck. Determining optical flow. In *Techniques and Applications of Image Understanding*, volume 281, pages 319–331. International Society for Optics and Photonics, 1981. 1

[25] 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

[26] Jiahui Huang, Sheng Yang, Zishuo Zhao, Yu-Kun Lai, and Shi-Min Hu. Clusterslam: A slam backend for simultaneous rigid body clustering and motion estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5875–5884, 2019. 2

[27] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007. 2

[28] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2462–2470, 2017. 2

[29] Matthias Innmann, Michael Zollhöfer, Matthias Nießner, Christian Theobalt, and Marc Stamminger. Volumedeform: Real-time volumetric non-rigid reconstruction. In *European Conference on Computer Vision*, pages 362–379. Springer, 2016. 2

[30] Mariano Jaimez, Christian Kerl, Javier Gonzalez-Jimenez, and Daniel Cremers. Fast odometry and scene flow from rgb-d cameras based on geometric clustering. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3992–3999. IEEE, 2017. 2

[31] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015. 2

[32] Huaizu Jiang, Deqing Sun, Varun Jampani, Zhaoyang Lv, Erik Learned-Miller, and Jan Kautz. Sense: A shared encoder network for scene-flow estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3195–3204, 2019. 5

[33] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4867–4876, 2020. 4

[34] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976. 4

[35] Ken-ichi Kanatani. Motion segmentation by subspace separation and model selection. In *Proceedings Eighth IEEE International Conference on computer Vision. ICCV 2001*, volume 2, pages 586–591. IEEE, 2001. 2

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

[37] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 1, 2, 6

[38] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *IEEE International Conference on Computer Vision*, pages 9246–9255, 2019. 1, 2, 6

[39] 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

[40] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3614–3622, 2019. 2

[41] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. arXiv:1512.02134. 1, 2, 6

[42] 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. 1, 2, 6

[43] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 6

[44] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. *arXiv preprint arXiv:1912.00497*, 2019. 1, 2

[45] Frank Moosmann and Christoph Stiller. Joint self-localization and tracking of generic objects in 3d range data. In *2013 IEEE International Conference on Robotics and Automation*, pages 1146–1152. IEEE, 2013. 2

[46] Armin Mustafa and Adrian Hilton. Semantically coherent 4d scene flow of dynamic scenes. *International Journal of Computer Vision*, pages 1–17, 2019. 2

[47] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5

[48] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 343–352, 2015. 2

[49] Michael Niemeyer, Lars Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5379–5389, 2019. 1, 2

[50] J-P Pons, Renaud Keriven, and Olivier Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 822–827. IEEE, 2005. 2

[51] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *European Conference on Computer Vision*, 2020. 1, 2, 5, 6, 7, 8

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

[53] Charles Ruizhongtai 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*, pages 5099–5108, 2017. 2

[54] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. Caspr: Learning

canonical spatiotemporal point cloud representations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 2

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[56] Martin Runz, Maud Buffier, and Lourdes Agapito. Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects. In *2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 10–20. IEEE, 2018. 2

[57] Samuele Salti, Federico Tombari, and Luigi Di Stefano. A performance evaluation of 3d keypoint detectors. In *2011 International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission*, pages 236–243. IEEE, 2011. 2

[58] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual slam and structure from motion in dynamic environments: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–36, 2018. 2

[59] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, 35(2):876–879, 1964. 4

[60] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1395, 2017. 2

[61] Miroslava Slavcheva, Maximilian Baust, and Slobodan Ilic. Sobolevfusion: 3d reconstruction of scenes undergoing free non-rigid motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2646–2655, 2018. 2

[62] Michael Strecke and Jorg Stuckler. Em-fusion: Dynamic object-level slam with probabilistic data association. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5865–5874, 2019. 2

[63] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 2

[64] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: An open dataset benchmark. *arXiv preprint arXiv:1912.04838*, 2019. 2, 7

[65] Georg Tanzmeister, Julian Thomas, Dirk Wollherr, and Martin Buss. Grid-based mapping and tracking in dynamic environments using a uniform evidential environment representation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6090–6095. IEEE, 2014. 2

[66] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017. 2

[67] Ivan Tishchenko, Sandro Lombardi, Martin R Oswald, and Marc Pollefeys. Self-supervised learning of non-rigid residual flow and ego-motion. *arXiv preprint arXiv:2009.10467*, 2020. 1, 2, 6

[68] Arash K Ushani, Ryan W Wolcott, Jeffrey M Walls, and Ryan M Eustice. A learning approach for real-time temporal scene flow estimation from lidar data. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5666–5673. IEEE, 2017. 2

[69] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *IEEE International Conference on Computer Vision*, pages 722–729. IEEE, 1999. 1, 2

[70] 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

[71] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2

[72] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 2

[73] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Adrian Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. *arXiv preprint arXiv:1912.01438*, 2019. 1, 2

[74] A. Wedel, C. Rabe, T. Vaudrey, T. Brox, U. Franke, and D. Cremers. Efficient dense scene flow from sparse or dense stereo data. In *ECCV*, Marseille, France, October 2008. 2

[75] Wenxuan Wu, Zhiyuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: A coarse-to-fine network for supervised and self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:1911.12408*, 2019. 1, 2, 6, 8

[76] Jiaqi Yang, Yang Xiao, and Zhiguo Cao. Toward the repeatability and robustness of the local reference frame for 3d shape matching: An evaluation. *IEEE Transactions on Image Processing*, 27(8):3766–3781, 2018. 2

[77] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3323–3332, 2019. 2

[78] Zi Jian Yew and Gim Hee Lee. Rpm-net: Robust point matching using learned features. In *CVPR*, 2020. 4

[79] Li Yi, Boqing Gong, and Thomas Funkhouser. Complete & label: A domain adaptation approach to semantic segmentation of lidar point clouds, 2020. 8

[80] Li Yi, Haibin Huang, Difan Liu, Evangelos Kalogerakis, Hao Su, and Leonidas Guibas. Deep part induction from articulated object pairs. *ACM Transactions on Graphics (TOG)*, 37(6):209, 2019. 2

[81] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d

instance segmentation in point cloud. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019. 2

[82] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2018. 2

[83] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020. 5

[84] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 5

[85] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2