RSF: Optimizing Rigid Scene Flow From 3D Point Clouds Without Labels

David Deng
UC Berkeley
davezdeng8@berkeley.edu

Avideh Zakhor
UC Berkeley
avz@berkeley.edu

Abstract

We present a method for optimizing object-level rigid 3D scene flow over two successive point clouds without any annotated labels in autonomous driving settings. Rather than using pointwise flow vectors, our approach represents scene flow as the composition a global ego-motion and a set of bounding boxes with their own rigid motions, exploiting the multi-body rigidity commonly present in dynamic scenes. We jointly optimize these parameters over a novel loss function based on the nearest neighbor distance using a differentiable bounding box formulation. Our approach achieves state-of-the-art accuracy on KITTI Scene Flow and nuScenes without requiring any annotations, outperforming even supervised methods. Additionally, we demonstrate the effectiveness of our approach on motion segmentation and ego-motion estimation. Lastly, we visualize our predictions and validate our loss function design with an ablation study.

1. Introduction

Understanding the 3D motion in a scene is an essential problem in computer vision and robotics. 3D scene flow is a low-level motion representation, characterized as a 3D motion field over all points in the scene. With the increasing prevalence of LiDAR and depth sensors, scene flow estimation from point clouds has become an increasingly important problem. Advances in deep feature learning on point sets [37, 38, 6, 55, 22] have recently given rise to deep learning approaches to scene flow estimation [27, 48, 13, 50, 54, 12, 51, 32, 1, 33]. However, because real-world scene flow annotations are expensive to acquire, many of these approaches rely on synthetic datasets [29] and often do not generalize well to real world data, especially LiDAR scans. Additionally, most of these works fail to exploit the rigidity present in most scenes *i.e.* the dynamics of most scenes can be decomposed into the motion of multiple rigidly moving objects. They predict unconstrained, pointwise motion vectors, resulting in inaccurate and physically inconsistent predictions.

In light of these shortcomings, we propose a new *object-level* scene flow estimation approach that jointly optimizes a global ego-motion and a set of bounding boxes with their own rigid motions, without using any annotated labels. To optimize this new scene flow parameterization using gradient methods, we develop a novel loss function and a differentiable bounding box formulation. We demonstrate the effectiveness of our approach on the KITTI [30, 31, 11] and nuScenes[4] datasets. In particular, our method achieves 2x lower end-point-error than the current state-of-the-art on the KITTI Scene Flow dataset.

In summary, our main contributions are that:

- We propose a novel objective function to optimize object-level rigid scene flow without any annotated labels in autonomous driving settings.
- We develop a differentiable 3D bounding box formulation to optimize our scene flow parameters.
- Our method produces physically plausible and interpretable scene flow by constraining the predicted motion to be rigid.
- Our approach accurately detects moving objects at an instance level without labeled supervision.
- Our approach significantly outperforms the state-of-the-art on KITTI Scene Flow and nuScenes.

2. Related Work

Supervised Scene Flow The term scene flow was first introduced in [44]. Traditional methods primarily predicted scene flow from stereo and RGB-D [15, 49, 20, 14, 40, 45, 46, 47], although some used LiDAR [43]. Following the rise of deep learning, neural networks became a prevalent tool for scene flow estimation on images [19, 52, 53, 16, 17]. [27] pioneered deep scene flow estimation on point clouds by combining the FlowNet architecture [9, 18] with advances in deep point cloud feature learning [37, 38]. They trained their model in a supervised manner on synthetic data [29] and generalize it to real world data [10]. Subsequent works on supervised scene flow estimation from point

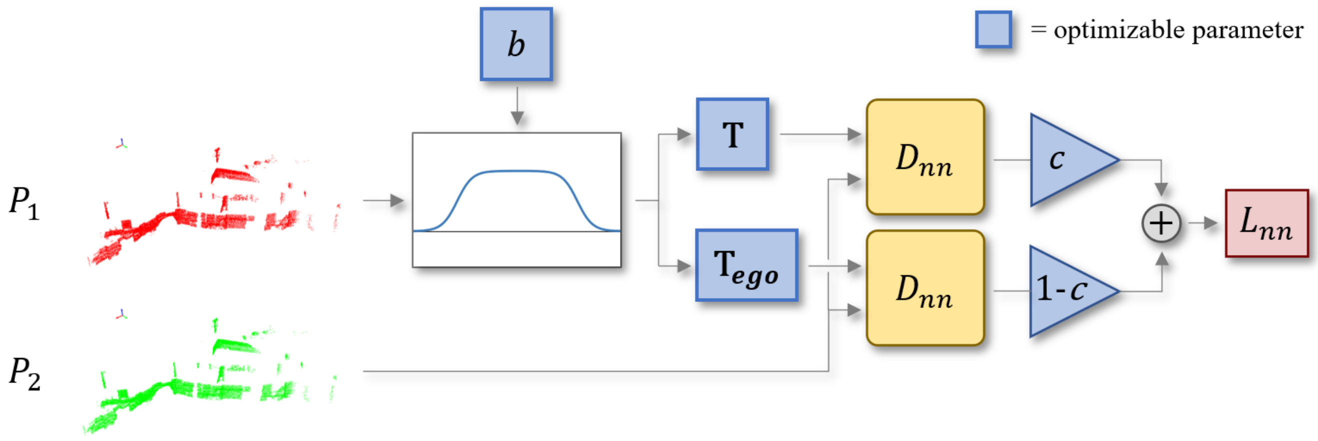


Figure 1: Overview of our loss function for a single bounding box. Terms in blue blocks are optimizable scene flow parameters b: bounding box parameters, $\mathbf{T}$: bounding box's rigid transformation, $\mathbf{T}_{ego}$: ego-motion transformation, c: box's confidence score, and the plot refers to our differentiable bounding box approximation. From P_1, we differentiably select the points inside the bounding box and transform them using $\mathbf{T}$ and $\mathbf{T}_{ego}$. Then we compute the nearest neighbor distance between the two transformed point sets and P_2. Lastly, we weigh the two nearest neighbor distance by c and $1-c$ respectively and sum them to compute the loss.

clouds followed this training framework. [13] and [50] developed faster and more accurate network architectures using sparse permutohedral lattices and cost volumes respectively. [54] predicted scene flow as a global ego-motion and a set of residual flow vectors. [36] framed scene flow estimation as a correspondence problem by borrowing ideas from optimal transport. [39] utilized high-order CRFs to constrain the scene flow output to be locally smooth and rigid. Lastly, [51] used a recurrent network to iteratively refine scene flow predictions.

Self-Supervised Scene Flow Because of the domain shift from synthetic to real world datasets, as well as the difficulty acquiring scene flow annotations for real LiDAR scans, recent works on scene flow estimation from point clouds have explored self-supervision. [33] was the first, utilizing a nearest neighbor and cycle-consistency loss. [41] and [35] developed loss functions that use Chamfer Distance in conjunction with smoothness and shape constraints. [51, 54, 1] use losses similar to these, but [51] uses a recurrent network architecture, while [54, 1] predict ego-motion in addition to flow vectors. None of these approaches exploit the multi-body rigid property of dynamic scenes. Recently, [24] proposed a method that over-segments point clouds and computes a rigid transform for each segment, but does not incorporate any notion of objects.

Scene Flow Optimization Rather than predicting scene flow in real time, recently a few works have used offline optimization approaches to estimate scene flow. In particular, [35] directly optimizes its self-supervised loss function over each point cloud at inference time, and [25] trains a neural network over a single pair of point clouds to predict scene flow, using the network as an implicit smoothness regularizer. Our approach falls within this category, but unlike the previous works, we optimize scene flow at the object level.

Object Scene Flow While multi-body rigidity is a commonly used prior in scene flow estimation, most existing works require some form of annotation. [34, 26, 28, 53] predicted object-level scene flow from stereo, images, or RGB-D. [26, 28, 53] required annotated segmentation labels, and [34] focused on object detection. [32] was the first work to predict object level scene flow from point clouds, but it required scene flow, object detection, and ego-motion labels. More recently, [12] proposed a weakly supervised approach to object-level scene flow estimation that only requires ego-motion and foreground/background labels, and [8] built on this by utilizing a recurrent update network. All of the aforementioned approaches except [34] require annotated labels at training time. Our approach is the first to leverage the rigidity of dynamic scenes in scene flow estimation while also requiring no labelled supervision.

3. Method

The objective of 3D scene flow estimation is, given a pair of point clouds $P_1 \in R^{3\times N_1}, P_2 \in R^{3\times N_2}$, to predict the scene flow between them, defined as a set of vectors $F \in R^{3\times N_1}$ that indicate the motion of the points in P_1 to their corresponding location in P_2. Note that there may not be direct correspondences between the two point clouds.

3.1. Motion Representation

Previous works usually predict scene flow directly, parameterized as a set of pointwise motion vectors. However, this parameterization is highly unconstrained and fails to exploit the underlying rigidity present in most scenes. We

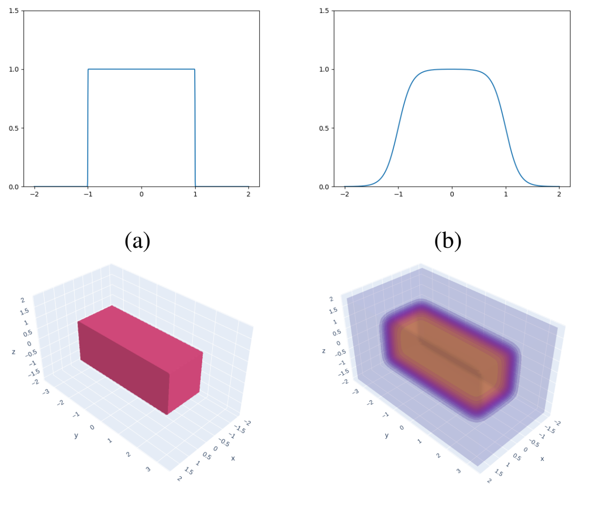

(a) (b) (c) (d)

Figure 2: Visualizations of non-differentiable vs differentiable bounding boxes in 1 and 3 dimensions: (a) non-differentiable 1D bounding line; (b) differentiable 1D bounding line; (c) non-differentiable 3D bounding box; (d) differentiable 3D bounding box.

instead parameterize scene flow as the composition of a global ego-motion and a set of bounding boxes containing moving objects, each with their own rigid motions. Specifically, our objective is to compute a global ego-motion $\mathbf{T}_{ego} = \{\mathbf{R}_{ego} \in SO(3), \mathbf{t}_{ego} \in R^3\} \in SE(3)$; a set of k bounding boxes $B = \{\mathbf{b}_i\}^k$ parameterized as $\mathbf{b} = (c, x, y, z, w, l, h, \theta)$ where c is a confidence score indicating whether or not the box contains a moving object, x, y, z is the center of the bounding box, w, l, h are the box's dimensions, and θ is a heading angle; and a set of k rigid transformations of the form $\mathbf{T}_i = \{\mathbf{R}_i \in SO(3), \mathbf{t}_i \in R^3\} \in SE(3)$, one for each bounding box. Because we focus specifically on autonomous driving settings, we assume the boxes have zero pitch and roll. Refer to the supplement for additional details on our motion parameterization.

In the context of autonomous driving, bounding boxes are usually sufficient to capture the shapes of moving objects. We choose to use bounding boxes to parameterize objects rather than segmentation masks as in previous works because similar to pointwise flow vectors, segmentation masks are highly unconstrained and can represent incoherent objects e.g. an object with two detached points on opposite sides of the scene. On the other hand, optimizing bounding boxes constrains the objects to physically plausible point sets, significantly reducing the dimensionality of the optimization space.

3.2. Overview

An overview of our proposed objective function is shown in Figure 1. To compute the scene flow parameters without any labelled supervision, we optimize them over a novel loss function that characterizes the nearest neighbor distance (NND) from P_1 to P_2 after applying the scene flow parameters. Given a bounding box b_i, we select the points inside it using a differentiable bounding box approximation. We then transform these points using $\mathbf{T}_i$ and $\mathbf{T}_{ego}$ and compute the NND between the two transformed point sets and P_2. We call these terms the foreground and background loss, respectively. Finally, we multiply the foreground and background loss by c_i and $1 - c_i$ respectively and add these together to compute the total loss for b_i. The final loss is the sum of all per-box losses. To compute scene flow parameters, we simply minimize this loss function using gradient-based optimization. As a clarifying note, we do not use any neural networks and are only interested in directly optimizing the scene flow parameters. In the following sections, we explain each component of the loss function in detail.

3.3. Differentiable Bounding Boxes

Our proposed loss function takes the NND of transformed points in P_1 to P_2, where pointwise NND is defined as:

$$D_{nn}(P_1, P_2)[i] = \min_{p_2 \in P_2} ||p_2 - P_1[i]||_2^2 \tag{1}$$

where $[i]$ denotes the ith point. However, the NND of points inside a bounding box b_i is a step function with respect to b_i and therefore non-differentiable. In order to optimize b_i over the NND, we propose a novel, differentiable bounding box approximation.

Fundamentally, a bounding box is a membership function in 3D space. Each point in the domain of the function is mapped to either 1 or 0 depending on whether it is inside or outside the box. In the simplified case of a 1 dimensional bounding box or "bounding line", the membership function would be the difference of two shifted unit step functions, as shown in Figure 2a. Inspired by [7], we can replace the step functions with sigmoid approximations to make them differentiable, shown in Figure 2b, resulting in a 1D differentiable bounding box:

$$box_{1D}(x) = \frac{1}{1 + e^{k(x+\frac{l}{2})}} - \frac{1}{1 + e^{k(x-\frac{l}{2})}} \tag{2}$$

where l is the width of the box and k is a parameter controlling the sharpness of the sigmoid slope.

To generalize to 3D, we take the product of 1D bounding lines along each of the three axes of the bounding box, resulting in a 3D bounding box membership density field shown in Figure 2d. This results in:

$$box_{3D}(x, y, z) = \prod_{d \in \{x_b, y_b, z_b\}} \left(\frac{1}{1 + e^{k(d+\frac{l_d}{2})}} - \frac{1}{1 + e^{k(d-\frac{l_d}{2})}}\right) \tag{3}$$

where x_b, y_b, z_b are x, y, z transformed into the local coordinate frame of the bounding box, and l_d is the width of the box along dimension d.

The boxes are initialized in a grid over the ground plane. We initialize our box shapes to approximate that of cars, using the anchor dimensions from [22]. The heading angle, rotation, and translation of each box are initialized to 0, the identity, and $\vec{0}$, respectively. For more details, please refer to the supplement.

3.4. Loss Function

Our proposed loss function is the sum of independently computed per-box losses, so for simplicity we will focus on the loss of a single box b, written as:

$$\begin{aligned} L_{nn} &= c \sum_{i=1}^{N_1} \hat{w}_i * (D_{nn}(\mathbf{R}\mathbf{p}_i + \mathbf{t}, P_2) + \epsilon) \\ &+ (1-c) \sum_{i=1}^{N_1} \hat{w}_i * D_{nn}(\mathbf{R}_{ego}\mathbf{p}_i + \mathbf{t}_{ego}, P_2) \\ &= cL_{fg} + (1-c)L_{bg} \end{aligned} \tag{4}$$

where c is the confidence score of the box. $\hat{w}_i$ is the i$^{\text{th}}$ element of $\hat{\mathbf{w}}$, a normalized vector of weights produced by the smooth bounding box approximation for b computed using Equation 3 over P_1, $\mathbf{p}_i$ is the i$^{\text{th}}$ point in P_1, $\mathbf{R}$ and $\mathbf{t}$ are b's associated rotation and translation respectively, N_1 and N_2 indicated the number of points in P_1 and P_2, and ϵ is a constant penalty, to be described shortly. Lastly, L_{fg} and L_{bg} denote foreground and background loss, respectively.

The foreground loss is the differentiable NND of points in b_i after P_1 is transformed by $\mathbf{T}_i$, and the background loss is the NND after P_1 is transformed by $\mathbf{T}_{ego}$, describing the NND under the assumption that the points in b_i belong to a dynamic rigid object or a static object. During optimization, it is unknown whether a given box contains a moving or static object, so to model this uncertainty, we write the final loss as the sum of these two terms weighted by c and $1-c$, where $c \in (0,1)$. If the foreground loss is lower than the background loss, the box likely contains a dynamic object, and to minimize the overall loss, c will converge towards 1. Likewise, c will converge towards 0 if the background loss is smaller. Concurrently, if the confidence is high, the loss from this box will primarily propagate gradients towards optimizing $\mathbf{T}$, and if it is low, the loss will optimize $\mathbf{T}_{ego}$, ensuring that primarily static objects are used for ego-motion optimization. During inference, we threshold c to determine which boxes contain moving objects.

Since the ego-motion is shared among all bounding boxes, it is more constrained than the per-box rigid motions. When a bounding box contains a static part of the scene, we find empirically the foreground rigid motion to be nearly identical to the ego-motion, but converging to a barely smaller loss. This results in many static background objects erroneously having a high confidence. To address this, we add a small constant penalty ϵ to the foreground term. This value can be interpreted as the minimum distance an object needs to traverse to be considered dynamic.

3.5. Auxillary Terms

In addition to the main loss function described above, we also incorporate a few auxillary terms. Since we narrow our use case specifically to autonomous driving datasets, we broadly assume that the moving objects are cars, and construct certain auxillary terms based on this assumption.

Box Dimension Regularization We apply a small penalty L_{shape} to the bounding box shape that constrains it to be about the size of an average car:

$$L_{shape} = \Delta_w^2 + \Delta_l^2 + \Delta_h^2 \tag{5}$$

where $\Delta_w, \Delta_l, \Delta_h$ are the underlying parameters to the width, length, and height of the boxes, described in the supplement.

Heading Term Because cars face the direction they are moving, we apply a consistency loss $L_{heading}$ that forces the heading of bounding boxes to point in the same direction as their motion.

$$L_{heading} = ||\theta_{xy} - \mathbf{t}_{xy}||_2^2 \tag{6}$$

where θ_{xy} is a 2D vector parameterizing the heading, further described in the supplement, and $\mathbf{t}_{xy}$ are the x and y components of $\mathbf{t}$ on the ground plane.

Angle Term Moving cars do not experience large rotations, so we apply a small penalty L_{angle} on the magnitude of the rotation angle θ due to $\mathbf{R}$.

$$L_{angle} = \theta^2 \tag{7}$$

Mass Term We also apply a mass term L_{mass} that encourages the bounding boxes to have more points inside them. If a box contains part of a moving object, this term encourages it to converge around it entirely. If it does not contain a moving object, it is still helpful for the boxes to contain many background points in order to make the ego-motion estimation more robust.

$$L_{mass} = -\sum_{i=1}^{N_1} w_i \tag{8}$$

where w_i is the i$^{\text{th}}$ element of $\mathbf{w}$, the unnormalized vector of membership weights from the differentiable bounding box.

Combining these terms, our final per-box loss is

$$\begin{aligned} L = L_{nn} &+ \lambda_{shape}L_{shape} + \lambda_{heading}L_{heading} \\ &+ \lambda_{angle}L_{angle} + \lambda_{mass}L_{mass} \end{aligned} \tag{9}$$

Table 1: Scene flow evaluation.

Dataset	Method	Supervision/ Approach	Training Data	EPE3D ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
StereoKITTI	FlowNet3D [27]	Full	FT3D	0.177	0.374	0.668	0.527
	HPLFlowNet [13]	Full	FT3D	0.117	0.478	0.778	0.410
	PointPWCNet [50]	Full	FT3D	0.069	0.728	0.888	0.265
	FLOT [36]	Full	FT3D	0.056	0.755	0.908	0.242
	EgoFlow [54]	Full	FT3D	0.069	0.670	0.879	0.404
	FlowStep3D [51]	Full	FT3D	0.055	0.805	0.925	0.149
	HCRF-Flow [39]	Full	FT3D	0.053	0.863	0.944	0.180
	WeaklyRigidFlow [12]	Full	FT3D	0.042	0.849	0.959	0.208
	PointPWCNet [50]	Self	FT3D	0.255	0.238	0.496	0.686
	EgoFlow [54]	Self	FT3D	0.415	0.221	0.372	0.810
	FlowStep3D [51]	Self	FT3D	0.102	0.708	0.839	0.246
	SLIM [1]	Self	RawKITTI	0.121	0.518	0.796	0.402
	SLIM*[1]	Self	RawKITTI	0.067	0.77	0.934	0.249
	RigidFlow [24]	Self	FT3D	0.062	0.724	0.892	0.262
	Chamfer*	Optimization	-	0.991	0.056	0.071	0.942
	PointPWCNet [50]	Optimization	-	0.657	0.357	0.405	0.72
	NSFP [25]	Optimization	-	0.036	0.912	0.961	0.154
	NSFP*[25]	Optimization	-	0.034	0.914	0.962	0.151
	Ours	Optimization	-	0.035	0.932	0.971	0.146
	Ours*	Optimization	-	**0.017**	**0.973**	**0.989**	**0.096**
LidarKITTI	PointPWCNet [50]	Full	FT3D	0.390	0.387	0.550	0.653
	FLOT [36]	Full	FT3D	0.653	0.155	0.313	0.837
	WeaklyRigidFlow [12]	Weak	SemKITTI	0.094	0.784	0.885	0.314
	ExploitingRigidity [8]	Weak	SemKITTI	**0.071**	0.824	0.913	0.295
	Chamfer*	Optimization	-	0.944	0.022	0.057	0.992
	PointPWCNet [50]	Optimization	-	0.734	0.248	0.347	0.845
	NSFP*[25]	Optimization	-	0.142	0.688	0.826	0.385
	Ours*	Optimization	-	0.085	**0.883**	**0.929**	**0.239**
nuScenes	Chamfer*	Optimization	-	0.879	0.035	0.082	0.976
	PointPWCNet*[50]	Optimization	-	0.615	0.199	0.328	0.86
	NSFP *[25]	Optimization	-	0.177	0.374	0.668	0.527
	Ours *	Optimization	-	**0.107**	**0.717**	**0.862**	**0.321**

* methods that use the entire point cloud. All other methods downsample to 8,192 points.

where each λ is a hyperparameter that controls the influence of the corresponding loss. The total loss is then the sum of L over all boxes.

3.6. Inference

To select the boxes containing moving objects, we first filter out any bounding boxes that contain fewer than n_{min} points, where n_{min} is a threshold that depends on the point cloud density of the given dataset. Then we apply non-maximum-suppression and keep boxes with a score of 0.85 or above. We classify these boxes as dynamic. With the dynamic boxes, we segment P_1 by assigning each point to the most confident box it is in, as there may still be some overlap between boxes. Finally, we apply each box's rigid transformation to its points, and the ego-motion to the remaining background points to compute the scene flow as

$$\mathbf{f}_i = \mathbf{R}_i\mathbf{p}_i + \mathbf{t}_i - \mathbf{p}_i \tag{10}$$

where $\mathbf{p}_i$ is the i$^{\text{th}}$ point in P_1, $\mathbf{f}_i$ is its scene flow prediction, and $\mathbf{R}_i$ and $\mathbf{t}_i$ are its associated rotation and translation.

4. Results

We evaluate our method both quantitatively and visually on various datasets for scene flow estimation, moving object segmentation, and ego-motion estimation, comparing them with the current state-of-the-art. Additionally, we explore the effects of our design choices using an ablation study.

4.1. Datasets

KITTI Scene Flow [30, 31, 11] We evaluate our scene flow and motion segmentation predictions on the KITTI Scene Flow Dataset, a real world autonomous driving dataset containing 142 pairs of point clouds annotated with scene flow vectors and instance segmentation masks. The authors removed pedestrians and cyclists, so the only moving ob-

Table 2: Motion segmentation results on StereoKITTI.

Method	mIoU ↑	Accuracy ↑
SLIM [1]	42.9	60.1
Ours	**86.6**	**92.9**

jects in the dataset are cars. The dataset has two settings: StereoKITTI and LidarKITTI. StereoKITTI is the traditional setting in which the point clouds are generated from stereo disparity maps and therefore have correspondences. Most of the existing methods evaluate on this setting. LidarKITTI is a more challenging setting in which the point clouds are captured by a Velodyne 64-beam LiDAR. They are sparser and do not have direct correspondences. The ground truth scene flow vectors are assigned by projecting the LiDAR points onto the StereoKITTI disparity map and using the associated scene flow annotations.

For a fair comparison, we adopt the common data preprocessing step introduced by [13] of removing ground points via naive thresholding at 1.4 m below the sensor, and cropping any points further than 35 m from the sensor. We use this step on all our datasets and experiments. Due to memory constraints during training, most prior learning-based approaches also downsample their point clouds to 8,192 points. Our efficient implementation enables us to use the entire point cloud, so on StereoKITTI, we report our performance using both 8,192 points, and all points. The typical size of undownsampled point clouds in each dataset is detailed in the supplement.

On LidarKITTI, we optimize our scene flow parameters over the entire LiDAR scan and evaluate it on the points in front of the vehicle with scene flow annotations. We found that only optimizing over points in front of the vehicle in LiDAR scans results in false positive detections at the point cloud boundaries due to cropping. This problem is somewhat mitigated when using the entire 360°point cloud.

nuScenes [4] nuScenes is a more challenging dataset than KITTI, consisting of sparser LiDAR sweeps and more complex driving scenarios. We use a subset of nuScenes released by [25], consisting of 310 point cloud pairs with ground points removed using RANSAC. The ground truth flow vectors are drawn using track annotations.

SemanticKITTI [2] We evaluate our ego-motion predictions on SemanticKITTI, a large scale autonomous driving dataset curated from the KITTI odometry dataset. It consists of 21 sequences of LiDAR frames annotated with ground truth poses. Due to the size of the dataset, we only evaluate on a random subset of 500 point cloud pairs. We also use SemanticKITTI to evaluate our method's motion segmentation accuracy; refer to the supplement for details.

FlyingThings3D (FT3D) [29] FlyingThings3D is a large synthetic dataset consisting of stereo pairs with scene flow annotations generated from CAD models moving randomly in space. While our approach does not make use of FT3D, previous supervised approaches use it for training.

Table 3: Ego-Motion Estimation Evaluation on SemanticKITTI.

Method	Rotation Error ($\circ$) ↓	Translation Error (m) ↓	Rotation Accuracy ↑	Translation Accuracy ↑
ICP [3]	0.244	0.122	0.906	0.878
Ours	**0.235**	**0.107**	**0.916**	**0.94**

RawKITTI [10] RawKITTI consists of the approximately 38,000 raw LiDAR scans from the KITTI dataset. [1] uses this for self-supervised training.

4.2. Evaluation Metrics

To evaluate scene flow, we use the standard metric of 3D end-point error (EPE3D), which is the average $l2$ distance between the predicted and ground truth scene flow. We also adopt the following metrics from [27, 13]: strict accuracy (Acc3DS): the percentage of points with EPE3D < 0.05 m or relative error $< 5\%$; relaxed accuracy (Acc3DR): the percentage of points with EPE3D < 0.1 m or relative error $< 10\%$; and Outliers: the percentage of points with EPE3D > 0.3 m or relative error $> 10\%$.

To evaluate motion segmentation, we group all moving points into a single class and report the mean intersection-over-union (mIoU) and the segmentation accuracy:

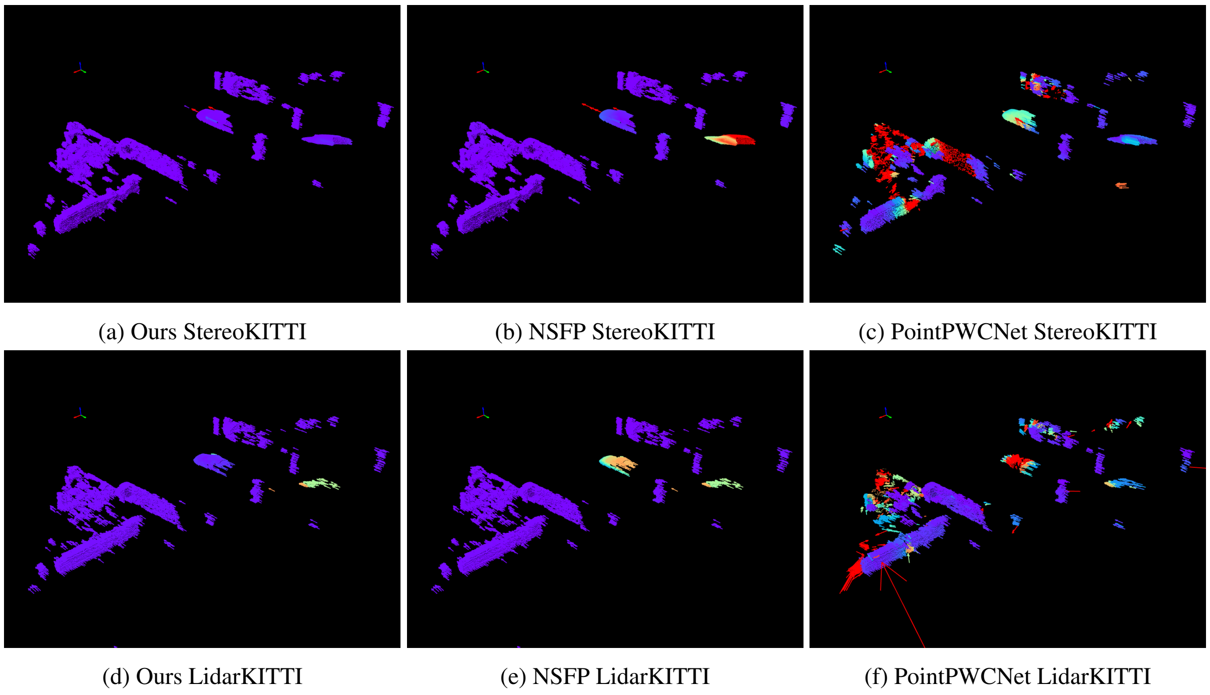
$$mIoU = 0.5(\frac{TP}{TP+FP+FN} + \frac{TN}{TN+FP+FN}) \tag{11}$$

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{12}$$

For ego-motion estimation, we report the average rotation and translation errors of our predictions in degrees and meters, as well as the rotation and translation accuracy, defined as the percentage of scenes where the rotation error $<$ 0.5° and the translation error < 0.1 m.

4.3. Baselines

We compare our work against the following supervised, self-supervised, and optimization-based scene flow estimation approaches: FlowNet3D [27], HPLFlownet [13], PointPWCNet [50], FLOT [36], EgoFlow [54], FlowStep3D [51], HCRF-Flow [39], WeaklyRigidFlow [12], SLIM [1], NSFP [25], RigidFlow [24], and ExploitingRigidity [8]. Additionally, we directly optimize scene flow over two representative self-supervised loss functions: (i) the Chamfer Distance, used in [35, 51] (ii) the self-supervised loss from [50], which adds a smoothness and laplacian term to the Chamfer Distance. We compare motion segmentation results against SLIM [1], a state-of-the-art self-supervised ap-


(a) Ours StereoKITTI (b) NSFP StereoKITTI (c) PointPWCNet StereoKITTI

(d) Ours LidarKITTI (e) NSFP LidarKITTI (f) PointPWCNet LidarKITTI

(g) Error Colorbar

Figure 3: Visualization of scene flow predictions for our method, NSFP, and the PointPWCNet loss function under direct optimization on a scene in KITTI. Color indicates the EPE3D of the prediction, with red indicating high error and purple indicating low error. For StereoKITTI, the colorscale ranges from 0-0.5 m error, while for LidarKITTI, it ranges from 0-1 m. In this scene, the ego vehicle is moving forward as two cars approach from the opposite direction. Our approach is able to accurately predict the flow on both cars in the stereo setting, and the closer one in the LiDAR setting. NSFP struggles on moving objects, while PointPWC predicts locally smooth, but incoherent flow.

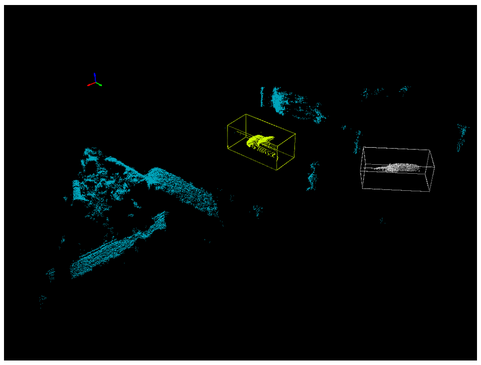


Figure 4: Visualization of bounding box/segmentation. Detected moving objects are shown inside bounding boxes and colored according to the resulting segmentation. Background points are cyan.

proach. Unlike our method, SLIM only predicts binary motion segmentation masks, and for dynamic points it predicts pointwise flow vectors instead of rigid motions. For egomotion estimation, we use ICP [3] as a baseline.

4.4. Scene Flow Evaluation

We report our results and baselines on StereoKITTI, LidarKITTI, and nuScenes in Table 1. Without relying on any annotated data, our method significantly outperforms the state-of-the-art supervised, self-supervised, and optimization baselines in most metrics on all datasets. In particular, when utilizing all points on StereoKITTI, our approach achieves 2x lower EPE3D than the previous state-of-the-art, and when using 8,192 points, our approach still outperforms all other downsampled methods on all metrics. In the LidarKITTI setting, [8] achieves lower EPE3D than our method, but we achieve better accuracy. Compared to other label-free optimization-based approaches, our method performs the best by far, achieving 1.7x lower EPE3D than [25]. Similarly, despite being a more challenging dataset, our method performs well on nuScenes, achieving an average error of 0.107 m, significantly outperforming [25].

Relative to the baselines, our method performs better on StereoKITTI than LidarKITTI. Because there does not exist same-domain training data for StereoKITTI, StereoKITTI baselines must always overcome a domain shift during in-

Table 4: Ablation study over various loss terms and design choices on StereoKITTI. ∇b refers to whether we use differentiable or non-differentiable bounding boxes.

∇b	L_{shape}	L_{mass}	$L_{heading}$	L_{angle}	EPE3D ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
	✓		✓	✓	0.284	0.74	0.759	0.319
✓		✓	✓	✓	0.218	0.473	0.678	0.457
✓	✓		✓	✓	0.401	0.66	0.663	0.406
✓	✓	✓		✓	0.024	0.959	0.984	0.116
✓	✓	✓	✓		**0.017**	**0.974**	**0.989**	**0.096**
✓	✓	✓	✓	✓	**0.017**	0.973	**0.989**	**0.096**

ference. On the other hand, LidarKITTI approaches exploit the abundant LiDAR data and labels in KITTI, which our method does not require. This explains why our method outperforms the baselines by a larger margin on StereoKITTI than LidarKITTI.

4.5. Motion Segmentation Evaluation

Our motion segmentation results are shown in Table 2. Our approach significantly outperforms [1] in both metrics. Qualitatively, we found that [1] often only segments parts of moving objects, while our method reliably segments entire objects. In the supplement, we also evaluate on SemanticKITTI and compare against a state-of-the-art supervised baseline [5]. While the baseline achieves better accuracy than our method, it requires per-point annotated supervision. Refer to the supplement for more details.

4.6. Ego-Motion Evaluation

Our ego-motion results on SemanticKITTI are shown in Table 3. We outperform ICP on all metrics, achieving an average of 0.235° rotation error and 0.107 m translation error. As a note, we observed that ICP performs unusually well on SemanticKITTI due to the already close alignment of the LiDAR scans, as well as the absence of moving objects in several of the scenes.

4.7. Visualizations

We qualitatively compare our method against NSFP [25] and the PointPWCNet [50] loss function, two state-of-the-art optimization methods, by visualizing the predictions on a scene from KITTI in Figure 3. As shown, our method produces the most accurate predictions. In particular, NSFP struggles to predict dynamic objects, while PointPWCNet generates locally smooth predictions but fails to exhibit object level rigidity. Despite the fact that the primary supervisory signal for all three approaches is the NND, our approach achieves more accurate predictions by directly constraining the scene flow to be rigid.

Additionally, we visualize our predicted bounding boxes and segmentation masks in Figure 4. Our approach is able to accurately detect the moving objects in the scenes. From the visualization, one can note the effect of our heading and mass losses, as the boxes are centered around and oriented with the moving objects. For a more exhaustive visualization, refer to the supplement.

4.8. Ablation Study

To evaluate our design choices, we conduct an ablation study, shown in Table 4. We find that the differentiable bounding boxes, shape regularization, and mass term contribute significantly to our performance. Without differentiable bounding boxes, the shape and position of the boxes are not updated. Without the shape regularizer, the mass term causes boxes to grow too large. Without the mass term, the boxes converge to empty regions without points. The heading and angle terms also provide slight increases in accuracy. On StereoKITTI, the angle term actually slightly decreases the accuracy, but generally, and especially in LiDAR settings, it improves optimization stability.

4.9. Limitations

Our approach faces a few limitations. As an optimization framework, our method is unsuitable for real-time inference. Additionally, our current formulation is only applicable to autonomous driving settings where objects can be reasonably parameterized using boxes. Lastly, despite our method's state-of-the-art performance, we still struggle with sparse, occluded, and featureless portions of the input point clouds, especially in the LiDAR domain. See the supplement for more details.

5. Conclusion

We introduce the first approach that exploits the multi-body rigidity of dynamic scenes without requiring annotated scene flow or segmentation labels. Our approach achieves state-of-the-art performance on the KITTI Scene Flow Benchmark and nuScenes. With the growing prevalence of depth sensors and LiDAR, our work will be useful for processing the raw, unlabelled point clouds they generate. In the future we hope to combine our approach with learning for real-time inference, and to generalize to arbitrary 3D scenes.

References

[1] Stefan Baur, David Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. Slim: Self-

supervised lidar scene flow and motion segmentation. In *In International Conference on Computer Vision (ICCV)*, 2021.

[2] 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 *In International Conference on Computer Vision (ICCV)*, 2019.

[3] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *In Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss. Moving Object Segmentation in 3D LiDAR Data: A Learning-based Approach Exploiting Sequential Data. *IEEE Robotics and Automation Letters (RA-L)*, 6:6529–6536, 2021.

[6] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019.

[7] József Dániel Dombi and Zsolt Gera. The approximation of piecewise linear membership functions and lukasiewicz operators. *Fuzzy Sets Syst.*, 154:275–286, 2005.

[8] Guanting Dong, Yueyi Zhang, Hanlin Li, Xiaoyan Sun, and Zhiwei Xiong. Exploiting rigidity constraints for lidar scene flow estimation. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 12776–12785, June 2022.

[9] 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 *In International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

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

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

[12] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly Supervised Learning of Rigid 3D Scene Flow, 2021.

[13] 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 *In Computer Vision and Pattern Recognition (CVPR)*, pages 3254–3263, 2019.

[14] Frederic Devernay Diana Mateus Matthieu Guilbert. Multi-camera scene flow by tracking 3-d points and surfels. *In Computer Vision and Pattern Recognition (CVPR)*, 2006.

[15] Frédéric Huguet and Frédéric Devernay. A variational method for scene flow estimation from stereo sequences. In *In International Conference on Computer Vision (ICCV)*, pages 1–7. IEEE, 2007.

[16] Junhwa Hur and Stefan Roth. Self-supervised monocular scene flow estimation. *In Computer Vision and Pattern Recognition (CVPR)*, 2020.

[17] Junhwa Hur and Stefan Roth. Self-supervised multi-frame monocular scene flow. In *In Computer Vision and Pattern Recognition (CVPR)*, 2021.

[18] 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 *In Computer Vision and Pattern Recognition (CVPR)*, volume 2, page 6, 2017.

[19] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. *ECCV*, 2018.

[20] Mariano Jaimez, Mohamed Souiai, Javier Gonzalez-Jimenez, and Daniel Cremers. A primal-dual framework for real-time dense rgb-d scene flow. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 98–104. IEEE, 2015.

[21] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[22] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. 2019.

[23] Jake Levinson, Carlos Esteves, Kefan Chen, Noah Snavely, Angjoo Kanazawa, Afshin Rostamizadeh, and Ameesh Makadia. An analysis of svd for deep rotation estimation. *Neural Information Processing Systems (NeurIPS)*, 2020.

[24] Ruibo Li, Chi Zhang, Guosheng Lin, Zhe Wang, and Chunhua Shen. Rigidflow: Self-supervised scene flow learning on point clouds by local rigidity prior. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 16959–16968, June 2022.

[25] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Neural scene flow prior. *Advances in Neural Information Processing Systems*, 34, 2021.

[26] Vikranth Dwaracherla Lin Shao, Parth Shah and Jeannette Bohg. Motion-based object segmentation based on dense rgb-d scene flow. *IEEE Robotics and Automation Letters*, 2018.

[27] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019.

[28] Wei-Chiu Ma, Shenlong Wang, Rui Hu, Yuwen Xiong, and Raquel Urtasun. Deep rigid instance scene flow. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 3614–3622, 2019.

[29] 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 *In Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[30] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015.

[31] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *Proc. of the ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[32] Himangi Mittal, Brian Okorn, and David Held. Pointflownet: Learning representations for rigid motion estimation from point clouds. *In Computer Vision and Pattern Recognition (CVPR)*, 2019.

[33] Himangi Mittal, Brian Okorn, and David Held. Just go with the flow: Self-supervised scene flow estimation. *In Computer Vision and Pattern Recognition (CVPR)*, 2020.

[34] Andreas Geiger Philip Lenz, Julius Ziegler and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[35] Jhony Kaesemodel Pontes, James Hays, and Simon Lucey. Scene flow from point clouds with or without learning. *International Conference on 3D Vision (3DV)*, 2020.

[36] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *In European Conference on Computer Vision (ECCV)*, 2020.

[37] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *In Computer Vision and Pattern Recognition (CVPR)*, 2017.

[38] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Neural Information Processing Systems (NeurIPS)*, 2017.

[39] Tong He Fayao Liu Chunhua Shen Ruibo Li, Guosheng Lin. Hcrf-flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding. *In Computer Vision and Pattern Recognition (CVPR)*, 2021.

[40] René Schuster, Christian Bailer, Oliver Wasenmüller, and Didier Stricker. Combining stereo disparity and optical flow for basic scene flow. *CoRR*, abs/1801.04720, 2018.

[41] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *In Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018.

[42] Salti Tombari and Di Stefano. Unique signatures of histograms for local surface description. *In European Conference on Computer Vision (ECCV)*, 2010.

[43] 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, 2017.

[44] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *In Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 722–729. IEEE, 1999.

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

[46] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *In International Conference on Computer Vision (ICCV)*, pages 1377–1384, 2013.

[47] 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.

[48] Zirui Wang, Shuda Li, Henry Howard-Jenkins, Victor Prisacariu, and Min Chen. Flownet3d++: Geometric losses for deep scene flow estimation. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 91–98, 2020.

[49] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *In European Conference on Computer Vision (ECCV)*, pages 739–751. Springer, 2008.

[50] 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. *In European Conference on Computer Vision (ECCV)*, 2020.

[51] Dan Raviv Yair Kittenplon, Yonina C. Eldar. Flowstep3d: Model unrolling for self-supervised scene flow estimation. *In Computer Vision and Pattern Recognition (CVPR)*, 2021.

[52] Gengshan Yang and Deva Ramanan. Upgrading optical flow to 3d scene flow through optical expansion. In *In Computer Vision and Pattern Recognition (CVPR)*, 2020.

[53] Gengshan Yang and Deva Ramanan. Learning to segment rigid motions from two frames. In *In Computer Vision and Pattern Recognition (CVPR)*, 2021.

[54] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. *arXiv preprint arXiv:1704.07813*, 2017.

[55] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. 2018.