

# Weakly Supervised Learning of Rigid 3D Scene Flow — Supplementary Material

Zan Gojic<sup>1,2</sup>

Or Litany<sup>2,3</sup>

Andreas Wieser<sup>1</sup>

Leonidas J. Guibas<sup>2</sup>

Tolga Birdal<sup>2</sup>

<sup>1</sup>ETH Zurich

<sup>2</sup>Stanford University

<sup>3</sup>NVIDIA

[3dsceneflow.github.io](https://github.com/3dsceneflow)

This supplementary material provides additional details, discussion, and results that were omitted from the main paper due to the lack of space. We give a detailed description of the network architecture and the optimization scheme in § 1.1 and § 1.3, explain the datasets and their generation in Sec. 2, and include additional ablation studies, as well as quantitative and qualitative results in Sec. 3

## 1. Implementation details

Here we provide the implementation details of our network architecture, foreground clustering using DBSCAN, and test-time optimization scheme. Additionally, we summarize the Kabsch algorithm and the entropy regularized optimal transport. For a general overview please refer to the main paper.

### 1.1. Network architecture

Our whole network is built upon MinkowskiEngine [4], an auto-differentiation library for sparse tensors. In all experiments, we first randomly sample 8192 points from the source and target point cloud, respectively. We then voxelize the original point clouds into sparse tensors with a voxel size of 0.1 m. If more than 8192 voxels remain, we randomly select 8192 of them. For the evaluation, all the inferred quantities are transferred from the voxels to the originally sampled points using Eq (11). The combined network has 8,078,149 learnable parameters. Fig. 1 shows the detailed architecture, where blue rectangles denote the Minkowski convolutional and transpose convolutional layers, violet rectangles the ResBlocks, and orange rectangles are ReLU activation functions.

**Backbone network.** The detailed architecture of the backbone network is depicted in Fig. 1 (a). We use absolute coordinates instead of the more common binary occupancy as the input feature of the first layer. All convolutional layers are followed by instance normalization and in ResBlocks additionally by the ReLU activation function. The output of the backbone network are 64-dimensional pointwise latent features. Due to the fixed voxel size, the number of points can vary between the point clouds.

**Scene flow head.** The scene flow head, depicted in Fig. 1

(b), takes the initial flow vector field  $\mathbf{V}^{\text{init}}$  as input and computes a residual flow  $\Delta\mathbf{V}^{\text{init}}$ . All layers except the last are followed by the ReLU activation function. Normalization functions are not used in the scene flow head.

**Background segmentation head.** The background segmentation head (Fig. 1 (c)) comprises two sparse convolutional layers, where the first one is followed by the instance normalization and ReLU activation function. The output of the second layer is passed through a sigmoid function to obtain the foreground probabilities  $\mathbf{h}$ .

### 1.2. Foreground clustering

We cluster the foreground points into objects using a simple DBSCAN clustering algorithm. Specifically, we first extract the foreground points  $\mathbf{X}^f$  of the source point cloud using the inferred foreground probabilities  $\mathbf{h}^X$ . We then run a scikit-learn [15] implementation of DBSCAN with  $\text{eps} = 0.75$  m and the minimum number of samples in the neighborhood equal to 5. After clustering, we only retain clusters with at least 10 points. Our clustering algorithm is based on the hypothesis that the objects scattered in the scene, are separated by void space. If two objects (e.g. cars) are too close to each other and get assigned to the same cluster, our method does not have the capacity to recover from this wrong assignment. However, we conduct an ablation study (§ 3.2) and empirically confirm the above-mentioned hypothesis for the autonomous driving datasets.

### 1.3. Test-time optimization

The object-level abstraction of the scene enables us to perform test-time optimization of per-object rigid body transformations. This optimization is performed independently for the background and each of the foreground objects. In the following we assume that our network outputs object-level masks  $\{\mathbf{z}\}_{k=1}^K$  and transformation parameters  $\{\mathbf{T}_k\}_{k=1}^K$ .

**Background transformation.** Let  $\mathbf{z}_1$  and  $\mathbf{T}_1$  denote the inferred background mask and ego-motion transformation parameters, respectively. We use  $\mathbf{z}_1$  to extract the background points  $\mathbf{X}^b$  and  $\mathbf{Y}^b$  of the source and target point cloud, transform  $\mathbf{X}^b$  with the inferred ego-motion  $\mathbf{T}_1$  and

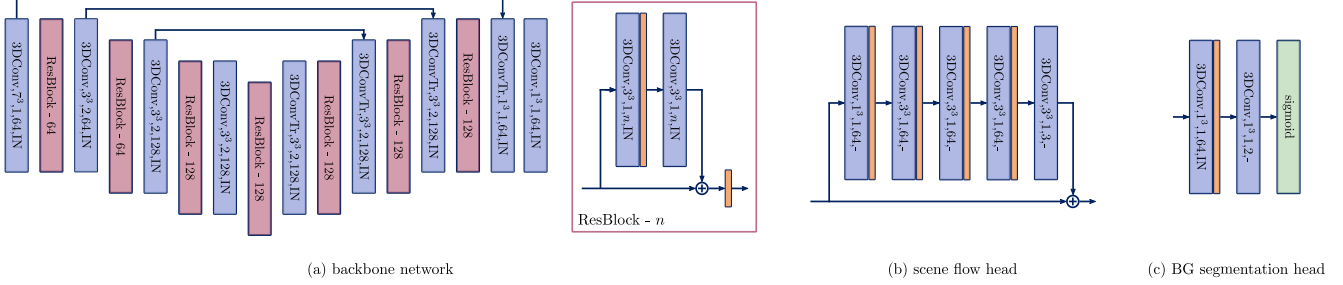


Figure 1: Network architecture of the backbone network (a), scene flow head (b), and background segmentation head (c). Blue blocks denote the convolutional (3DConv) and transpose convolutional (3DConvTr) layers, where  $3DConv,a,b,c,d$  denotes a 3D convolutional layer with the kernel size  $a$ , stride  $b$ , output feature dimension  $c$ , and normalization function  $d$ . Orange rectangle denotes the ReLU activation function [14] and IN is the instance normalization.

use these point clouds as an input to the ICP [2] optimization of the transformation parameters. Specifically, we use the Open3D [22] implementation with the maximum correspondence distance of 0.15 m and a maximum number of iterations equal to 300. On *lidarKITTI* the optimization of the background transformation parameters takes 0.06 s on average for a point cloud pair, using the same computer as in the run time experiments in § 4.6 of the main paper.

**Object level transformations.** For the object level transformations, we follow a similar procedure. We start by extracting the points  $\mathbf{X}^k$  of the object  $k$  in the source point cloud using the object mask  $\mathbf{z}_k$ , and all the foreground points  $\mathbf{Y}^f$  of the target point cloud using the complementary mask of the background mask  $\mathbf{z}_1$ . We then transform the points  $\mathbf{X}^k$  with the inferred transformation parameters  $\mathbf{T}_k$  and again use the point clouds as input to the ICP optimization of the transformation parameters. We use all the foreground points of the target point cloud as we do not have access to the instance level correspondences. The optimization of all foreground transformation parameters for a single *lidarKITTI* point cloud pair takes approximately 0.02 s on average. We perform at most 300 iterations of ICP per object with a maximum correspondences distance equal to 0.25 m.

#### 1.4. Kabsch algorithm

In order to make the paper self contained, we now summarize the weighted Kabsch algorithm, which represents the closed-form differentiable solution to the ego-motion estimation problem:

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

In the following, we omit the superscript  $b$  and define  $\mathbf{q}_l := \phi(\mathbf{x}_l, \mathbf{Y})$  and the resulting point cloud of correspondences

as  $\mathbf{Q} \in \mathbb{R}^{3 \times N} = \{\mathbf{q}_l \in \mathbb{R}^3\}_l$ . Let  $\bar{\mathbf{x}}$  and  $\bar{\mathbf{q}}$ :

$$\bar{\mathbf{x}} := \frac{\sum_{l=1}^N w_l \mathbf{x}_l}{\sum_{l=1}^N w_l}, \quad \bar{\mathbf{q}} := \frac{\sum_{l=1}^N w_l \mathbf{q}_l}{\sum_{l=1}^N w_l} \quad (1)$$

denote the weighted centroids of point clouds  $\mathbf{X}$  and  $\mathbf{Q}$ , respectively. The centered point coordinates can then be computed as  $\tilde{\mathbf{x}}_l := \mathbf{x}_l - \bar{\mathbf{x}}$  and  $\tilde{\mathbf{q}}_l := \mathbf{q}_l - \bar{\mathbf{q}}$ . By arranging them back to the matrix form  $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times 3}$  and  $\tilde{\mathbf{Q}} \in \mathbb{R}^{N \times 3}$ , a weighted covariance matrix  $\mathbf{S} \in \mathbb{R}^{3 \times 3}$  can be computed as

$$\mathbf{S} = \tilde{\mathbf{X}}^T \mathbf{W} \tilde{\mathbf{Q}} \quad (2)$$

where  $\mathbf{W} = \operatorname{diag}(w_1, \dots, w_N)$ . Considering the singular value decomposition  $\mathbf{S} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$ , the optimal rotation matrix is given by

$$\mathbf{R}_{\text{ego}}^* = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{V} \mathbf{U}^T) \end{bmatrix} \mathbf{U}^T \quad (3)$$

where  $\det(\cdot)$  denotes computing the determinant and is used to avoid generating a reflection matrix. Finally, the optimal translation vector is recovered as

$$\mathbf{t}_{\text{ego}}^* = \bar{\mathbf{q}} - \mathbf{R}_{\text{ego}}^* \bar{\mathbf{x}}. \quad (4)$$

Note, when estimating the per-object transformation parameters of the foreground points, we use an unweighted Kabsch algorithm, *i.e.*  $\mathbf{W} = \mathbf{I}$ .

#### 1.5. Entropy-regularized optimal transport

We provide a brief overview of the Sinkhorn algorithm used to approximate the optimal transport following [3]. Consider a discrete probability measure on the simplex  $\Sigma_d \triangleq \{\mathbf{x} \in \mathbb{R}_+^n : \mathbf{x}^T \mathbf{1}_n = 1\}$  with weights  $\mathbf{a} = \{a_i\}$  and locations  $\{x_i\}, i = 1 \dots n$  as:

$$= \sum_{i=1}^n a_i \delta_{x_i}, \quad a_i \geq 0 \wedge \sum_{i=1}^n a_i = 1 \quad (5)$$

where  $\delta_x$  is a Dirac delta function at  $x$ .

For two probability measures  $\mu$  and  $\nu$  in the simplex, let  $U(\mu, \nu)$  denote the polyhedral set of  $n_\mu \times n_\nu$  matrices:

$$U(\mu, \nu) = \{\mathbf{P} \in \mathbb{R}_+^{n_\mu \times n_\nu} : \mathbf{P}\mathbf{1}_{n_\nu} = \mu \wedge \mathbf{P}^\top \mathbf{1}_{n_\mu} = \nu\}$$

where  $\mathbf{1}_d$  is a  $d$ -dimensional vector of ones.  $U(\mu, \nu)$  is also referred to as the *transportation polytope*.

Let  $\mathbf{C}$  be a  $n_\mu \times n_\nu$  cost matrix that is constructed from the *ground cost* function  $c(x_i^\mu, x_j^\nu)$ . Kantorovich’s optimal transport formulation seeks to find the transport plan optimally mapping  $\mu$  to  $\nu$ :

$$\mathcal{W}_c(\mu, \nu) = d_c(\mu, \nu) = \min_{\mathbf{P} \in U(\mu, \nu)} \langle \mathbf{P}, \mathbf{C} \rangle \quad (6)$$

with  $\langle \cdot \rangle$  denoting the Frobenius dot-product. Note that this expression is also known as the *Wasserstein distance* (WD). Due to the computational difficulties in minimizing Eq (6) Cuturi [5] introduced an alternative convex set consisting of joint probabilities with small enough mutual information:

$$U_\alpha(\mu, \nu) = \{\mathbf{P} \in U(\mu, \nu) : KL(\mathbf{P} \parallel \mu \otimes \nu) \leq \alpha\} \quad (7)$$

where  $KL(\cdot)$  refers to the Kullback Leibler divergence [8]. This restricted polytope leads to a tractable distance between discrete probability measures under the cost matrix  $\mathbf{C}$  constructed from the function  $c$ :

$$d_{c,\alpha}(\mu, \nu) = \min_{\mathbf{P} \in U_\alpha(\mu, \nu)} \langle \mathbf{P}, \mathbf{C} \rangle. \quad (8)$$

$d_{c,\alpha}$  can be computed by a modification of simple matrix scaling algorithms, such as Sinkhorn’s algorithm [17, 18]. Note that here the cost matrix is inversely related to the affinity matrix defined in the main paper.

## 2. Datasets

In this work, a total of five datasets are used to train and evaluate the performance of the proposed approach. In the following, we detail the generation of the training and evaluation data. For all datasets, we use the same coordinate system centered at the camera or LiDAR sensor. The  $z$ -axis of the coordinate system points in the viewing direction of the camera or to the front of the car, the positive  $y$ -axis points in the up direction, and  $x$  completes the right-handed coordinate system. All processed point clouds are available under <https://3dsceneflow.github.io/>.

**FlyingThings3D** [11]. FlyingThings3D is a large-scale synthetic dataset of stereo RGB images proposed to train 2D based scene flow networks. We follow [7] and use the subset of the dataset in which the extremely hard examples are omitted. The point clouds are obtained by lifting the stereo images to 3D using the annotated disparity maps, optical flow, and camera parameters. We again follow [7] and remove the points whose disparity and optical flow are occluded (occlusion maps are part of the original dataset).

Dataset	Method	Supervision	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
<i>stereoKITTI</i> (w/o ground)	HPLFlowNet [7]	Full	0.117	0.478	0.778	0.410
	PointPWC-Net [20]	Full	0.069	0.728	0.888	0.265
	FLOT [16]	Full	0.056	0.755	0.908	0.242
	Ours (backbone)	Full	<b>0.042</b>	<b>0.849</b>	<b>0.959</b>	<b>0.208</b>
	Ours	Weak	0.163	0.541	0.658	0.452
Ours++	Weak	0.134	0.709	0.800	0.311	
<i>stereoKITTI</i> (with ground)	HPLFlowNet [7]	Full	0.238	0.194	0.429	0.787
	PointPWC-Net [20]	Full	0.204	0.292	0.556	0.645
	FLOT [16]	Full	0.122	0.480	0.691	0.401
	Ours (backbone)	Full	0.143	0.392	0.660	0.533
	Ours	Weak	0.136	0.470	0.712	0.420
Ours++	Weak	<b>0.068</b>	<b>0.836</b>	<b>0.897</b>	<b>0.263</b>	

Table 1: Evaluation results on *stereoKITTI* dataset. Weakly supervised models are trained on *semanticKITTI* LiDAR point clouds and evaluated directly on stereo point clouds of *stereoKITTI*. Ours and Ours++ denote the output of the network before and after test-time optimization, respectively.

Additionally, points with a depth larger than 35 m are removed [7]. The resulting point clouds are in direct correspondence, i.e.  $\mathbf{Y} = \mathbf{X} + \mathbf{V}$ , where  $\mathbf{X}$  and  $\mathbf{Y}$  are the source and target point cloud, and  $\mathbf{V}$  is the ground truth flow. For training and evaluation, 8192 points are randomly sampled from each point cloud independently.

**stereoKITTI** [12, 13]. This is a dataset based on the KITTI Scene Flow benchmark, which is designed for the evaluation of stereo based scene flow methods. It consists of 200 training and 200 test scenes. Because the ground truth annotations of the test scenes are not available, only the training scenes are used for evaluation. Due to incomplete and noisy annotations, 58 scenes are further removed and the final evaluation set comprises 142 scenes [9, 7]. The stereo images are lifted to point clouds analogously to FlyingThings3D [7]. Again, occluded points and points with a depth larger than 35 m are removed. For evaluation, we sample 8192 points from each point cloud independently. Tab. 1 of the main paper shows the result of the typical evaluation protocol in which the ground points are removed by a naive thresholding the  $y$  coordinate at  $-1.4$  m [9, 7] (the cameras in the KITTI setup are mounted at  $\approx 1.65$  m height above ground). In Tab. 1 we additionally report the results without removing the ground points.

**lidarKITTI** [12, 13]. This dataset comprises the Velodyne 64-beam LiDAR point clouds corresponding to the same 142 scenes used in *stereoKITTI*. The ground truth scene flow vectors are obtained by projecting the points of the source point cloud to the image plane using the provided calibration parameters, and associating them with the scene flow vectors of the corresponding pixels from the *stereoKITTI* dataset [10]. Due to directly using the LiDAR point clouds the points of both frames are not in direct correspondence and exhibit the typical LiDAR sampling pattern with very uneven point density.

**semanticKITTI** [1]. This dataset provides pointwise se-

Dataset	Method	Supervision	scene flow				BG - segmentation				ego motion	
			mean EPE3D [m] ↓	med. EPE3D [m] ↓	med. F-EPE3D [m] ↓	med. B-EPE3D [m] ↓	prec. FG ↑	rec. FG ↑	prec. BG ↑	recall. BG ↑	RRE [ ] ↓	RTE [m] ↓
<i>lidarKITTI</i> (w/o ground)	Ours	Weak	0.150	0.111	0.227	0.104	0.726	0.885	0.978	0.940	0.379	0.130
	Ours+	Weak	0.110	0.064	0.227	0.049	0.726	0.885	0.978	0.940	0.126	0.053
	Ours++	Weak	0.094	0.051	0.164	0.049	0.726	0.885	0.978	0.940	0.126	0.053
<i>lidarKITTI</i> (with ground)	Ours	Weak	0.133	0.109	0.186	0.109	0.734	0.855	0.991	0.980	0.327	0.130
	Ours+	Weak	0.106	0.083	0.186	0.083	0.734	0.855	0.991	0.980	0.142	0.091
	Ours++	Weak	0.103	0.083	0.131	0.083	0.734	0.855	0.991	0.980	0.142	0.091

Table 2: Detailed results of our weakly supervised method on *lidarKITTI* dataset. Ours denotes the direct output of the network. Ours+ and Ours++ are the results with only background and full test-time optimization, respectively

mantic labels annotated in 3D and improved ego-motion information for all sequences of the KITTI odometry benchmark [6]. Different to *stereoKITTI* and *lidarKITTI*, which include only the points that map to the image plane of the front camera, *semanticKITTI* contains full 360 LiDAR sweeps. We therefore process *semanticKITTI* point clouds to make them consistent with *lidarKITTI*. Specifically, we first convert the 3D coordinates of the points to polar coordinates and consider only the points with an azimuth angle in the range  $[-45, 45]$  and elevation angle in the range  $[-24.9, 12.0]$ . This maintains the points that would roughly map to the image plane of the front camera. We then additionally remove the points that are less than 1.5 m or more than 35 m away from the LiDAR sensor. The binary background mask is generated by combining the semantic labels into the background (class labels from 40 to 249) and foreground (other class labels)<sup>1</sup>. We split the point cloud pairs of *semanticKITTI* into 4350 validation (sequences 03 and 05) and 18840 training samples (remaining nine sequences of the training set). Because BG-FG labels of the test dataset are not available, we perform all evaluations on the validation dataset.

**waymo open** [19]. This is a large scale dataset collected by a fleet of waymo self-driving cars in various conditions. It contains 1950 sequences of 20 s duration each, collected with an acquisition rate of 10 Hz. In our experiments we use the training batches 0–16 ( $\approx 50\%$  of the training data) for fine-tuning (§ 3.2) and the validation batches 0–2 ( $\approx 40\%$  of the validation data) for evaluating our approach<sup>2</sup>. Waymo cars are equipped with five LiDAR sensors altogether, one mid-range LiDAR on the top and four short range ones on the sides of the car. In our experiments, we only use the points acquired by the top, mid-range scanner. We then transform these points into a coordinate system centered at the location of the LiDAR sensor in the KITTI setup and follow the processing steps used in *semanticKITTI* dataset to extract only the points that would roughly project to the front camera. Along with the ego-motion information, *waymo-open* also provides the 3D bounding-boxes of vehicles, pedestrians, cyclists, and signs. We use the former

<sup>1</sup><https://github.com/PRBonn/semantic-kitti-api/blob/master/config/semantic-kitti-all.yaml>

<sup>2</sup><https://waymo.com/open/download/>

Dataset	Method	Initialization	EPE3D [m] ↓	Acc3DS ↑	Acc3DR ↑	Outliers ↓
<i>lidarKITTI</i> (w/o ground)	Ours	Pretrained	0.102	0.706	0.833	0.357
	Ours++	Pretrained	<b>0.080</b>	<b>0.834</b>	<b>0.912</b>	<b>0.279</b>
	Ours	Random	0.150	0.521	0.744	0.450
	Ours++	Random	0.094	0.784	0.885	0.314
<i>lidarKITTI</i> (with ground)	Ours	Pretrained	0.091	0.601	0.788	0.445
	Ours++	Pretrained	<b>0.080</b>	<b>0.742</b>	<b>0.850</b>	<b>0.369</b>
	Ours	Random	0.133	0.460	0.746	0.527
	Ours++	Random	0.103	0.686	0.819	0.410

Table 3: Evaluation of our model trained with random initialization of the weights (Random) and with the weights pretrained on *FT3D* (Pretrained), on *lidarKITTI* dataset. Ours and Ours++ denote the direct output of the network and the result after test-time optimization, respectively.

three classes to extract the foreground and consider the remaining points as background.

### 3. Evaluation details and additional results

We start this section by providing additional details and results (§ 3.1) supporting the evaluations presented in Sec. 4 of the main paper, before reporting additional evaluations (§ 3.2) that were omitted from the main paper due to the space constraint.

#### 3.1. Evaluation details

**Additional results on *stereoKITTI*.** In § 4.3 of the main paper, we evaluate the performance of our backbone under full supervision on *FlyingThings3D* and *stereoKITTI*. Tab. 1 supplements that section with the evaluation results on the *stereoKITTI* dataset without removing the ground points. Additionally, we report the generalization results of our weakly supervised model trained on the LiDAR point clouds of *semanticKITTI*.

As expected, the performance of all fully supervised methods drops significantly if the challenging ground points are not removed. Remarkably, our weakly supervised model generalizes from LiDAR to stereo point clouds, and when combined with the test-time optimization even outperforms all methods on *stereoKITTI* with ground points. Tab. 1 also hints that the generalization LiDAR  $\mapsto$  stereo is less challenging than the opposite stereo  $\mapsto$  LiDAR.

**Additional results on *lidarKITTI*.** Tab. 2 supplements Tab. 2 from the main paper and provides detailed results of

Method	Initialization	BG segmentation				ego-motion	
		prec. FG*	rec. FG*	prec. BG*	recall. BG*	RRE [ ] #	RTE [m] #
semanticKITTI(w/o ground)							
Ours	Pretrained	0.950	0.892	0.991	0.996	0.145	0.035
Ours++	Pretrained	0.950	0.892	0.991	0.996	0.127	0.031
Ours	Random	0.971	0.895	0.991	0.998	0.201	0.047
Ours++	Random	0.971	0.895	0.991	0.998	0.133	0.032
semanticKITTI(with ground)							
Ours	Pretrained	0.942	0.909	0.996	0.998	0.177	0.044
Ours++	Pretrained	0.942	0.909	0.996	0.998	0.116	0.029
Ours	Random	0.966	0.904	0.996	0.999	0.249	0.059
Ours++	Random	0.966	0.904	0.996	0.999	0.121	0.032

Table 4: Evaluation of our model trained with random initialization of the weights (Random) and with the weights pretrained on FT3D (Pretrained), on semanticKITTI dataset. Ours and Ours++ denote the direct output of the network and the result after test-time optimization, respectively.

our method on the lidarKITTI dataset. Note, how the test-time optimization improves background (Ours+) as well as foreground (Ours++) scene flow estimates. The results of the BG-segmentation and FG/BG scene flow split should be interpreted with caution, due to the noisy annotation of the lidarKITTI dataset (see below). Further qualitative results are shown in Fig. 2 and failure cases in Fig. 3

Noisy annotations of lidarKITTI. During the evaluation, we have discovered that lidarKITTI annotations are noisy and contain outliers. These GT errors occur especially around the points lying on or close to the object boundaries. Outliers are caused by the projection of the 3D point onto the 2D image plane, where two distant points in 3D can map to the same pixel, as there is no perception of depth. Fig. 4 shows two prominent examples: in some cases, the instance mask and motion of one object are also assigned to the other (top), and in other cases, the background points get assigned the instance label and scene flow of the object in the front (bottom). Because of training semanticKITTI with accurate annotations, our method is still capable of predicting the correct object masks (Fig. 4 (b)). Wrong annotation however result in a lower BG-segmentation performance and cause apparent errors in our scene flow prediction (see Fig. 4 (c)). Unfortunately, this error is reflected in the quantitative evaluations not only for our method but also for many other scene flow algorithms out there. To reduce this effect, we additionally report median EPE3D values in Tab. 2, as well as the FG and BG EPE3D based on our predicted BG-mask, instead of the noisy GT mask.

### 3.2. Additional evaluations and ablation studies

Pretraining vs training from scratch. Evaluations reported in § 4.4 to § 4.6 of the main paper are performed using the weakly supervised model trained on semanticKITTI from scratch. However, recent works show that general 3D backbone networks can benefit from pretraining on large (annotated) datasets [21].

To evaluate this in our setting, we consider a model

Method	Mode	BG segmentation				ego-motion	
		prec. FG*	rec. FG*	prec. BG*	recall. BG*	RRE [ ] #	RTE [m] #
Ours++	generalization	0.960	0.689	0.957	0.996	0.141	0.099
Ours++	ne-tuned	0.945	0.921	0.989	0.992	0.111	0.078

Table 5: Comparison of the model ne-tuned on waymo open with the model trained only on semanticKITTI (generalization), on the waymo open dataset. Fine-tuned model outperforms the directly generalized one in terms of FG precision and ego-motion error.

GT inst. mask	train	test	lidarKITTI (with ground)				
			EPE [m]#	F-EPE [m]#	B-EPE [m]#	RRE [ ] #	RTE [m] #
3	3	3	0.097	0.216	0.085	0.146	0.082
			0.101	0.183	0.094	0.139	0.091
3	3	3	0.097	0.265	0.085	0.146	0.082
			0.102	0.195	0.094	0.139	0.091

Table 6: Performance evaluation of our simple foreground clustering algorithm compared to models using GT instance mask during training and/or testing on lidarKITTI dataset.

whose backbone weights were initialized with weights trained (with full supervision) on FlyingThings3D rather than randomly. The evaluations on lidarKITTI (Tab. 3) and semanticKITTI (Tab. 4) show that, in line with the literature, our backbone network can indeed benefit from pretraining. The improvement in is especially prominent in scene flow estimation (more than 1 cm lower EPE3D in Tab. 3) and in ego-motion estimation (lower rotation and translation errors in Tab. 4). In order to fully adhere to the weakly supervised setting, we use the randomly initialized model in all other evaluations.

Fine-tuning on waymo open In the main paper waymo open dataset is used only to evaluate the direct generalization performance of our model trained on semanticKITTI. However, as briefly discussed in the main paper waymo open also provides all the annotations that our weakly supervised model relies on. We now use these annotations, to evaluate the gain obtained by ne-tuning our model. To this end, we initialize our model with the weights trained on semanticKITTI and ne-tune it for 22k iterations (less than 2 epochs) on waymo open. Tab. 5 shows that the ne-tuned model greatly outperforms the model trained only on semanticKITTI, especially in terms of foreground recall (gain of more than 20 percent points) and relative translation error (improvement of 2 cm). The relative improvement is qualitatively also depicted in Fig. 5

Training and testing with GT instance masks To define the FG instance-level rigidity loss and to perform the test-time optimization of the foreground, we rely on a simple unsupervised clustering of the foreground into individual objects. Arguably, this part could be replaced by a more powerful instance segmentation head, which would however require instance annotations during training. It is of interest to

	lidarKITTI (with ground)		
	EPE [m]#	RRE [ ]#	RTE [m]#
w/o Sinkhorn	0.594	0.539	0.615
with Sinkhorn	0.133	0.327	0.130

Table 7: Ablation study of the Sinkhorn algorithm on darKITTI dataset (evaluation protocol follows Tab. 3 in the main paper).

see how much benefit perfect clustering would bring to our method. We therefore ablate by replacing the output of our clustering with the GT instance labels, which are provided both in semanticKITTI and lidarKITTI. We assess the individual contributions from having GT labels during training, testing or both. Tab. 6 shows that our model with the simple clustering algorithm performs comparably to the models using GT instance masks during training (semanticKITTI) and/or testing (darKITTI). We conclude that our clustering algorithm, which does not require GT instance masks during training or testing, is the preferred option. Due to the noisy instance annotations of the darKITTI dataset, the performance of models using GT instance masks during testing should be interpreted with caution.

Sinkhorn algorithm. The benefit of using the entropy-regularized Sinkhorn algorithm for the ego-motion estimation is two fold: (i) the Sinkhorn distances should yield more accurate correspondences due to the optimality of the transport map, and (ii) in combination with the slack row and column it enables us to down-weight the outliers in a principled manner. We empirically confirm these benefits in an ablation study in which we directly use the affinity matrix  $M$  to compute the soft correspondences  $(x_i^a; Y^b) = Y^b m_{i,j} = \sum_j m_{i,j}$  and estimate the ego-motion with an unweighted Kabsch algorithm (§ 1.4).

Tab. 7 depicts a significant increase of the EPE when Sinkhorn is deactivated. This increase can be accredited to the inferior estimation of the ego-motion parameters, which results in a much higher RTE and RRE.

## References

- [1] 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. *IEEE International Conf. on Computer Vision* 2019. 3
- [2] 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. 2
- [3] Tolga Birdal, Michael Arbel, Umut Simsekli, and Leonidas J Guibas. Synchronizing probability measures on rotations via optimal transport. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* pages 1569–1579, 2020. 2
- [4] 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. 1
- [5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems* pages 2292–2300, 2013. 3
- [6] 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*. 4
- [7] Xiuye Gu, Yijie Wang, Chongruo Wu, Yong Jae Lee, and Panqu Wang. Hpl ownet: Hierarchical permutohedral lattice ownet for scene flow estimation on large-scale point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition* pages 3254–3263, 2019. 3
- [8] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics* 22(1):79–86, 1951. 3
- [9] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3d: Learning scene flow in 3d point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* pages 529–537, 2019. 3
- [10] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteor-net: Deep learning on dynamic 3d point cloud sequences. In *IEEE International Conference on Computer Vision* pages 9246–9255, 2019. 3
- [11] 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. 3
- [12] 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. 3
- [13] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing* 40:60–76, 2018. 3
- [14] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. *ICML*, 2010. 2
- [15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830, 2011. 1
- [16] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene Flow on Point Clouds Guided by Optimal Transport. In *European Conference on Computer Vision* 2020. 3
- [17] Richard Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics* 35(2):876–879, 1964. 3
- [18] Richard Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967. 3
- [19] Pei Sun, Henrik Kretschmar, 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. 4

- [20] 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. 3
- [21] Saining Xie, Jiatao Gu, Demi Guo, Charles R Qi, Leonidas J Guibas, and Or Litany. Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. arXiv preprint arXiv:2007.10985, 2020. 5
- [22] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. arXiv:1801.09847, 2018. 2

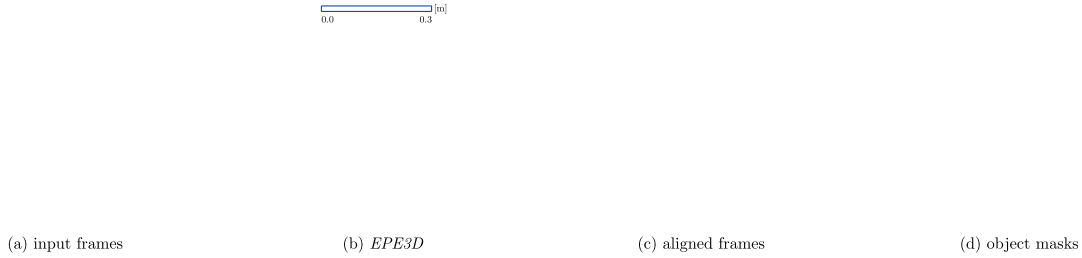


Figure 2: Successful cases of our method on the *lidarKITTI* dataset. By correctly splitting the scene into foreground and background (d), our method estimates the accurate scene flow vectors (b), which align the two frames (c).

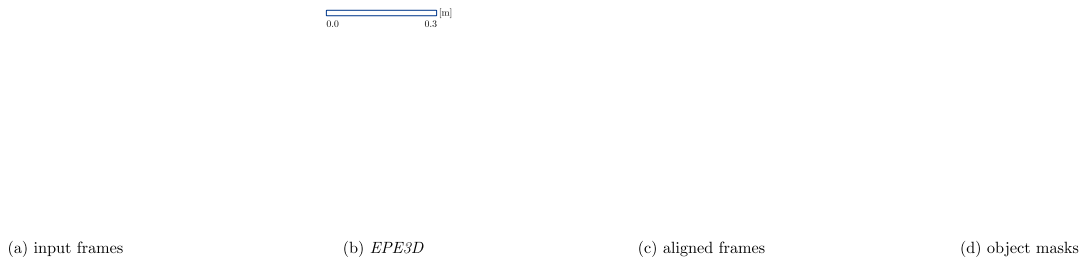


Figure 3: Failure cases of our method on the *lidarKITTI* dataset. **Top:** even though the car’s object mask (d) is correctly predicted, its predicted scene flow vectors yield large end-point-errors (b). **Bottom:** a pillar in the middle of the scene is wrongly predicted as foreground object (d), hence its scene flow does not agree with the background and GT (b).

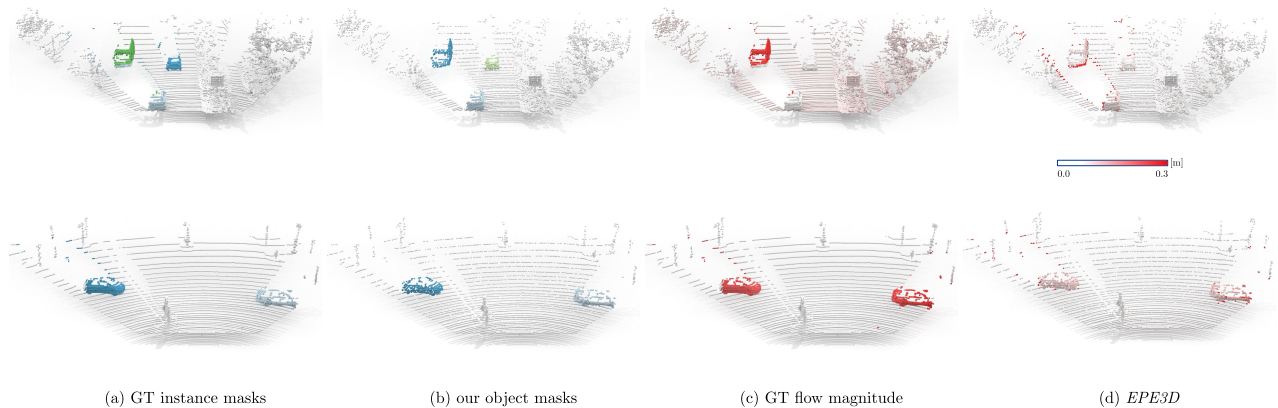


Figure 4: *lidarKITTI* annotations are obtained by projecting 3D points onto the image plane, which results in wrong instance (a) and scene flow (c) annotations for points with azimuth and elevation angles close to the object boundaries (e.g. green car is partially blue in (a) top). On the other hand, our method infers correct object masks (b) and scene flow (d), yet due to the wrong GT annotations, the scene flow appears to be erroneous.

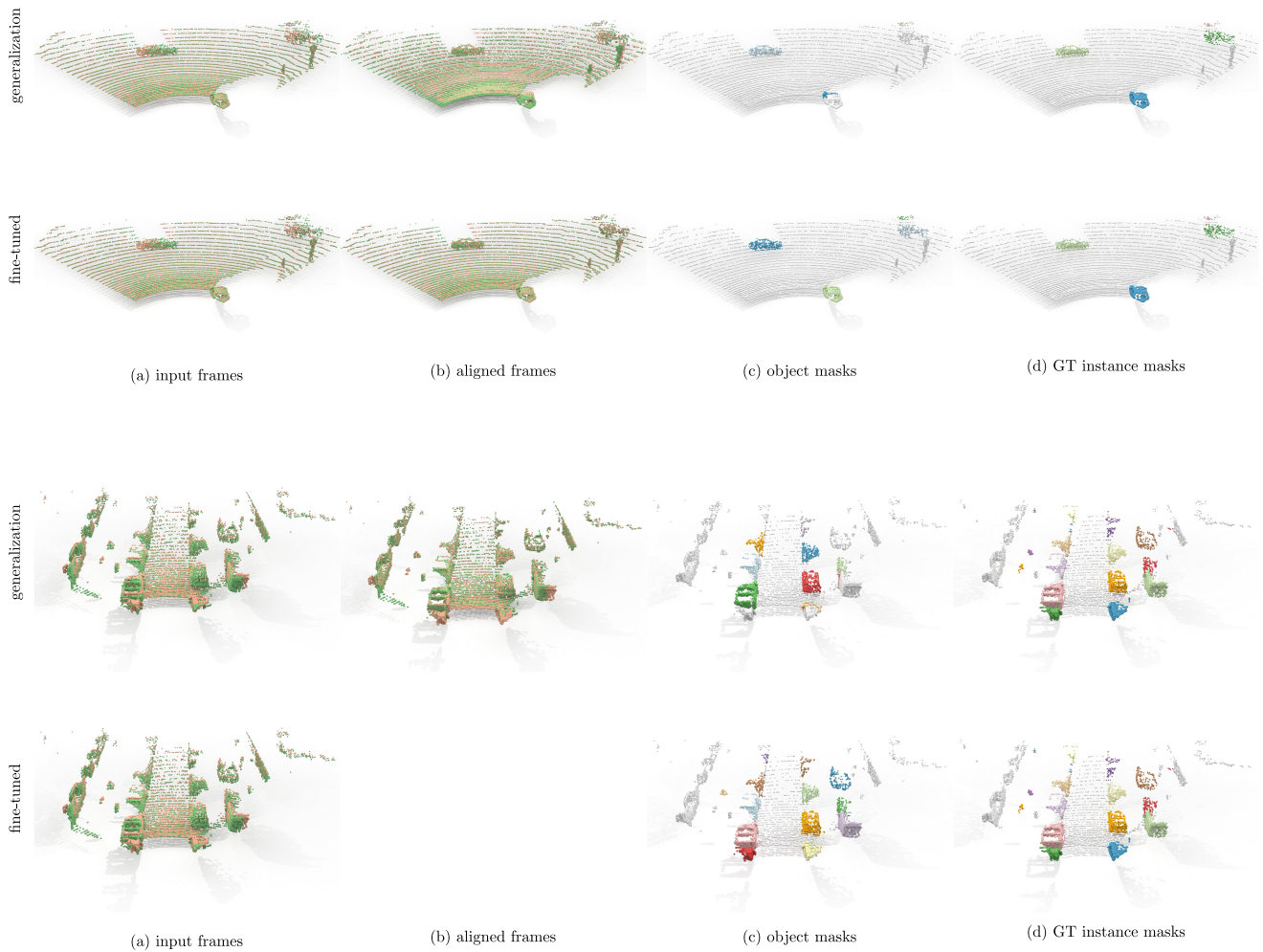


Figure 5: Fine-tuning on *waymo open* improves performance and robustifies our model. **Top:** objects close to the sensor are not common in *semanticKITTI* and hence cannot be detected correctly by the generalized model (c). **Bottom:** a challenging example with 18 foreground objects (much larger than average number in *semanticKITTI*). Note, how more object masks are correctly inferred by our fine-tuned model compared to direct generalization (c-column).

Figure 6: Failure cases on *waymo open* dataset. **Top:** our model is unable to estimate accurate ego-motion and scene flow (b) if the background points consists only of the ground points after foreground removal (c). **Bottom:** rare objects such as trucks (top right corner in c and d) appear ambiguous to our model and cause prediction of the wrong masks (c).