Dynamic 3D Scene Analysis by Point Cloud Accumulation

Shengyu Huang[1] Zan Gojcic[2] Jiahui Huang[3]
Andreas Wieser[1] Konrad Schindler[1]

[1]ETH Zürich [2]NVIDIA [3]BRCist

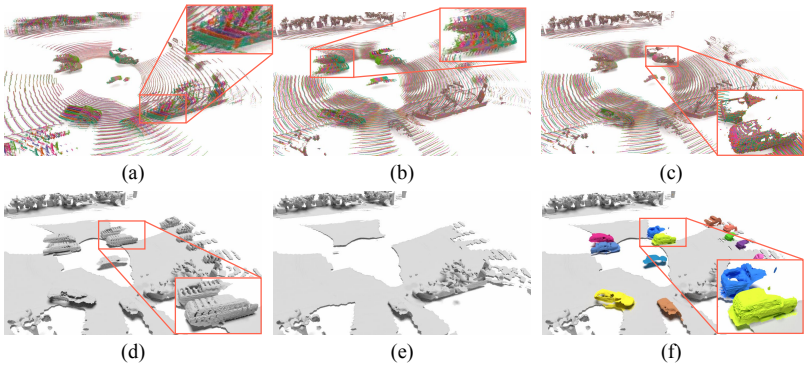

Fig. 1: Points in LiDAR frames acquired over time are not aligned due to the motion of the sensor and of other agents in the scene (a, d). Static background points can be aligned using ego-motion, but this smears the dynamic points across their trajectories (b). While motion segmentation only enables removing the moving points from the scene (e), our method properly disentangles individual moving objects from the static part and accumulates both correctly (c, f).

Abstract. Multi-beam LiDAR sensors, as used on autonomous vehicles and mobile robots, acquire sequences of 3D range scans ("frames"). Each frame covers the scene sparsely, due to limited angular scanning resolution and occlusion. The sparsity restricts the performance of downstream processes like semantic segmentation or surface reconstruction. Luckily, when the sensor moves, frames are captured from a sequence of different viewpoints. This provides complementary information and, when accumulated in a common scene coordinate frame, yields a denser sampling and a more complete coverage of the underlying 3D scene. However, often the scanned scenes contain moving objects. Points on those objects are not correctly aligned by just undoing the scanner's ego-motion. In the present paper, we explore multi-frame point cloud accumulation as a mid-level representation of 3D scan sequences, and develop a method that exploits inductive biases of outdoor street scenes, including their geometric layout and object-level rigidity. Compared to state-of-the-art scene flow estimators, our proposed approach aims to align all 3D points in a common reference frame correctly accumulating the points on the individual objects. Our approach greatly reduces the alignment errors on several benchmark datasets. Moreover, the accumulated point clouds benefit high-level tasks like surface reconstruction. [project page]

1 Introduction

LiDAR point clouds are a primary data source for robot perception in dynamically changing 3D scenes. They play a crucial role in mobile robotics and autonomous driving. To ensure awareness of a large field of view at any point in time, 3D point measurements are acquired as a sequence of sparse scans that each cover a large field-of-view—typically, the full 360°. In each individual scan *(i)* the point density is low, and *(ii)* some scene parts are occluded. Both issues complicate downstream processing. One way to mitigate the problem is to assume the sensor ego-motion is known and to align multiple consecutive scans into a common scene coordinate frame, thus accumulating them into a denser and more complete point cloud. This simple accumulation strategy already boosts performance for perception tasks like object detection [7] and semantic segmentation [4], but it also highlights important problems. First, to obtain sufficiently accurate sensor poses the ego-motion is typically computed in post-processing to enable sensor fusion and loop closures — meaning that it would actually not be available for online perception. Second, compensating the sensor ego-motion only aligns scan points of the static background, while moving foreground objects are smeared out along their motion trajectories (Fig. 1b).

To properly accumulate 3D points across multiple frames, one must disentangle the individual moving objects from the static background and reason about their spatio-temporal properties. Since the grouping of the 3D points into moving objects itself depends on their motion, the task becomes a form of multi-frame 3D scene flow estimation. Traditional scene flow methods [32,58,40] model dynamics in the form of a free-form velocity field from one frame to the next, only constrained by some form of (piece-wise) smoothing [51,52,13].

While this is a very flexible and general approach, it also has two disadvantages in the context of autonomous driving scenes: *(i)* it ignores the geometric structure of the scene, which normally consists of a dominant, static background and a small number of discrete objects that, at least locally, move rigidly; *(ii)* it also ignores the temporal structure and only looks at the minimal setup of two frames. Consequently, one risks physically implausible scene flow estimates [19], and does not benefit from the dense temporal sequence of scans.

Starting from these observations, we propose a novel point cloud accumulation scheme tailored to the autonomous driving setting. To that end, we aim to accumulate point clouds over time while abstracting the scene into a collection of rigidly moving agents [3,19,47] and reasoning about each agent's motion on the basis of a longer sequence of frames [23,22]. Along with the accumulated point cloud (Fig. 1c), our method provides more holistic scene understanding, including foreground/background segmentation, motion segmentation, and per-object parametric motion compensation. As a result, our method can conveniently serve as a common, low-latency preprocessing step for perception tasks including surface reconstruction (Fig. 1f) and semantic segmentation [4].

We carry out extensive evaluations on two autonomous driving datasets *Waymo* [46] and *nuScenes* [7], where our method greatly outperforms prior art. For example, on *Waymo* we reduce the average endpoint error from 12.9 cm to

1.8 cm for the static part and from 23.7 cm to 17.3 cm for the dynamic objects. We observe similar performance gains also on *nuScenes*.

In summary, we present a novel, learnable model for temporal accumulation of 3D point cloud sequences over multiple frames, which disentangles the background from dynamic foreground objects. By decomposing the scene into agents that move rigidly over time, our model is able to learn multi-frame motion and reason about motion in context over longer time sequences. Moreover, our method allows for low-latency processing, as it operates on raw point clouds and requires only their sequence order as further input. It is therefore suitable for use in online scene understanding, for instance as a low-level preprocessor for semantic segmentation or surface reconstruction.

2 Related work

Temporal point cloud processing. Modeling a sequence of point clouds usually starts with estimating accurate correspondences between frames, for which scene flow emerged as a popular representation. Originating from [49], scene flow estimation builds an intuitive and effective dynamic scene representation by computing a flow vector for each source point. While traditional scene flow methods [55,50,51,52] leverage motion smoothness as regularizer within their optimization frameworks, modern learning-based methods learn the preference for smooth motions directly from large-scale datasets [32,58,40,36]. Moreover, manually designed scene priors proved beneficial for structured scenes, for instance supervoxel-based local rigidity constraints [29], or object-level shape priors learned in fully supervised [3] or weakly supervised [19] fashion. Methods like SLIM [2] take a decoupled approach, where they first run motion segmentation before deriving scene flows for each segment separately. Treating the entire point cloud sequence as 4D data and applying a spatio-temporal backbone [33,10,16] demonstrates superior performance and efficiency. Subsequent works enhance such backbones by employing long-range modeling techniques such as Transformers [48,15,60], or by coupling downstream tasks like semantic segmentation [1], object detection [59,42] and multi-modal fusion [38]. While our method employs the representation of multi-frame scene flow, we explicitly model individual dynamic objects, which not only provides a high-level scene decomposition, but also yields markedly higher accuracy.

Motion segmentation. Classification of the points into static and dynamic scene parts serves as an essential component in our pipeline. Conventional geometric approaches either rely on ray casting [8,44] over dense terrestrial laser scans to build clean static maps, or on visibility [39,26] to determine the dynamics of the query point by checking its occlusion state in a dense map. Removert [26] iteratively recovers falsely removed static points from multi-scale range images. Most recently, learning-based methods formulate and solve the segmentation task in a data-driven way: Chen *et al*. [9] propose a deep model over multiple

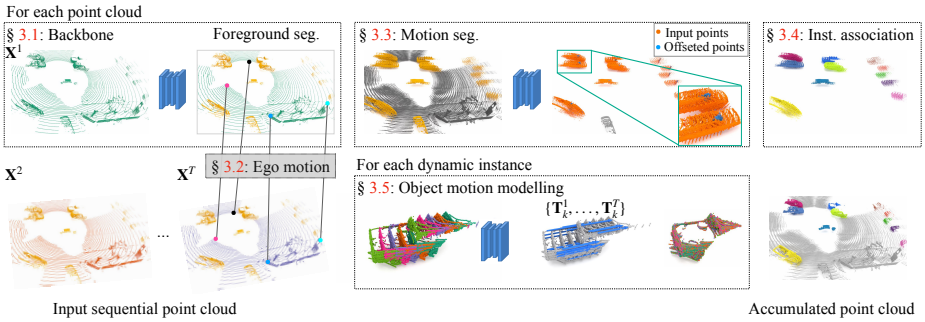

Fig. 2: **Overview**. Our method takes in a point cloud sequence of T frames and starts by extracting foreground points (marked yellow) for each frame. To obtain ego motion, $T-1$ pairwise registrations are performed. Next, points belonging to dynamic foreground object are extracted using our motion segmentation module (marked orange). To boost subsequent spatio-temporal instance association, we additionally predict per-point offset vectors. After instance association, we finally compute the rigid motion separately for each segmented dynamic object.

range image residuals and show SoTA results on a newly-established motion segmentation benchmark [4]. Any Motion Detector [17] first extracts per-frame features from bird's-eye-view projections and then aggregates temporal information from ego-motion compensated per-frame features (in their case with a with convolutional RNN). Our work is similar in spirit, but additionally leverages information from the foreground segmentation task and object clustering in an end-to-end framework.

Dynamic object reconstruction. Given sequential observations of a rigid object, dynamic object reconstruction aims to recover the 3D geometric shape as well as its rigid pose over time. Such a task can be handled either by directly hallucinating the full shape or by registering and accumulating partial observations. Approaches of the former type usually squash partial observations into a global feature vector [62,18,30] and ignore the local geometric structure. [21] go one step further by disentangling shape and pose with a novel supervised loss. However, there is still no guarantee for the fidelity of the completed shape. We instead rely on registration and accumulation. Related works include AlignNet-3D [20] that directly regresses the relative transformation matrix from concatenated global features of the two point clouds. NOCS [53] proposes a category-aware canonical representation that can be used to estimate instance pose w.r.t. its canonical pose. Caspr [43] implicitly accumulates the shapes by mapping a sequence of partial observations to a continuous latent space. [23] and [22] respectively propose multi-way registration methods that accumulate multi-body and non-rigid dynamic point clouds, but do not scale well to large scenes.

3 Method

The network architecture of our multitask model is schematically depicted in Fig. 2. To accumulate the points over time, we make use of the inductive bias that scenes can be decomposed into agents that move as rigid bodies [19]. We start by extracting the latent base features of each individual frame (§ 3.1), which we then use as input to the task-specific heads. To estimate the ego-motion, we employ a differentiable registration module (§ 3.2). Instead of using the ego-motion only to align the static scene parts, we also use it to spatially align the base features, which are reused in the later stages. To explain the motion of the dynamic foreground, we utilize the aligned base features and perform motion segmentation (§ 3.3) as well as spatio-temporal association of dynamic foreground objects (§ 3.4). Finally, we decode the rigid body motion of each foreground object from its spatio-temporal features (§ 3.5). We train the entire model end-to-end with a loss $\mathcal{L}$ composed of five terms:

$$\mathcal{L} = \mathcal{L}_{\text{ego}}^{\bullet} + \mathcal{L}_{\text{FG}}^{\bullet} + \mathcal{L}_{\text{motion}}^{\bullet} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}}^{\bullet} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}}^{\bullet} \,. \tag{1}$$

In the following, we provide a high-level description of each module. Detailed network architectures, including parameters and loss formulations, are given in the supplementary material (unless already elaborated).

Problem setting. Consider an *ordered* point cloud sequence $\mathcal{X} = \{\mathbf{X}^t\}_{t=1}^T$ consisting of T frames $\mathbf{X}^t = [\mathbf{x}_1^t, ..., \mathbf{x}_i^t, ..., \mathbf{x}_{n_t}^t] \in \mathbb{R}^{3\times n_t}$ of variable size, captured by a *single moving observer* at constant time intervals Δt. We denote the first frame $\mathbf{X}^1$ the *target* frame, while the remaining frames $\{\mathbf{X}^t \mid t > 1\}$ are termed *source* frames. Our goal is then to estimate the flow vectors $\{\mathbf{V}^t \in \mathbb{R}^{3\times n_t} | t > 1\}$ that align each of the source frames to the target frame, and hence *accumulate* the point clouds. Instead of predicting unconstrained pointwise flow vectors, we make use of the inductive bias that each frame can be decomposed into a *static* part $\mathbf{X}_{\text{static}}^t$ and K_t rigidly-moving *dynamic* parts $\mathcal{X}_{\text{dynamic}}^t = \{\mathbf{X}_k^t\}_{k=1}^{K_t}$ [19]. For an individual frame, the scene flow vectors $\mathbf{V}_{\text{static}}^t$ of the static part and $\mathbf{V}_k^t$ of the k-th dynamic object can be explained by the rigid ego-motion $\mathbf{T}_{\text{ego}}^t \in \text{SE}(3)$ and the motion of the dynamic object relative to the static background $\mathbf{T}_k^t \in \text{SE}(3)$, respectively as:

$$\mathbf{V}_{\text{static}}^t = \mathbf{T}_{\text{ego}}^t \circ \mathbf{X}_{\text{static}}^t - \mathbf{X}_{\text{static}}^t, \qquad \mathbf{V}_k^t = \mathbf{T}_k^t \mathbf{T}_{\text{ego}}^t \circ \mathbf{X}_k^t - \mathbf{X}_k^t, \tag{2}$$

where $\mathbf{T} \circ \mathbf{X}$ (or $\mathbf{T} \circ \mathbf{x}$) denotes applying the transformation to the point set $\mathbf{X}$ (or point $\mathbf{x}$).

3.1 Backbone network

Similar to [2,24,57], our backbone network converts the 3D point cloud of a single frame into a bird's eye view (BEV) latent feature image. Specifically, we lift the point coordinates to a higher-dimensional latent space using a point-wise MLP and then scatter them into a $H \times W$ feature grid aligned with the gravity axis.

The features per grid cell ("pillar") are aggregated with max-pooling, then fed through a 2D UNet [34] to enlarge their receptive field and strengthen the local context. The output of the backbone network is a 2D latent *base* feature map $\mathbf{F}^t_{\text{base}}$ for each of the T frames.

3.2 Ego-motion estimation

We estimate the ego-motion $\mathbf{T}^t_{\text{ego}}$ using a correspondence-based registration module separately for each source frame. Points belonging to dynamic objects can bias the estimate of ego-motion, especially when using a correspondence-based approach, and should therefore be discarded. However, at an early stage of the pipeline it is challenging to reason about scene dynamics, so we rather follow the conservative approach and classify points into background and foreground, where foreground contains all the *movable* objects (*e.g.*, cars and pedestrians), irrespective of their actual dynamics [19]. The predicted foreground mask is later used to guide motion segmentation in § 3.3.

We start by extracting ego-motion features $\mathbf{F}^t_{\text{ego}}$ and foreground scores $\mathbf{s}^t_{\text{FG}}$ from each $\mathbf{F}^t_{\text{base}}$ using two dedicated heads, each consisting of two convolutional layers separated by a ReLU activation and batch normalization. We then randomly sample N_{ego} background pillars whose $\mathbf{s}^t_{\text{FG}} < \tau$ and compute the pillar centroid coordinates $\mathbf{P}^t = \{\mathbf{p}^t_l\}$. The ego motion $\mathbf{T}^t_{\text{ego}}$ is estimated as:

$$\mathbf{T}^t_{\text{ego}} = \arg\min_{\mathbf{T}} \sum_{l=1}^{N_{\text{ego}}} w^t_l \left\| \mathbf{T} \circ \mathbf{p}^t_l - \phi(\mathbf{p}^t_l, \mathbf{P}^1) \right\|^2 . \tag{3}$$

Here, $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ finds the *soft correspondence* of $\mathbf{p}^t_l$ in $\mathbf{P}^1$, and w^t_l is the weight of the correspondence pair $\big(\mathbf{p}^t_l, \phi(\mathbf{p}^t_l, \mathbf{P}^1)\big)$. Both $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ and w^t_l are estimated using an entropy-regularized Sinkhorn algorithm from $\mathbf{F}^t_{\text{ego}}$ with slack row/column padded [11,61] and the optimal $\mathbf{T}^t_{\text{ego}}$ is computed in closed form via the differentiable Kabsch algorithm [25].

We supervise the ego-motion with an L_1 loss over the transformed pillars $\mathcal{L}_{\text{trans}} = \frac{1}{|\mathbf{P}^t|} \sum_{l=1}^{|\mathbf{P}^t|} \|\mathbf{T} \circ \mathbf{p}^t_l - \overline{\mathbf{T}} \circ \mathbf{p}^t_l\|_1$ and an inlier loss $\mathcal{L}_{\text{inlier}}$ [61] that regularizes the Sinkhorn algorithm, $\mathcal{L}^{\bullet}_{\text{ego}} = \mathcal{L}_{\text{trans}} + \mathcal{L}_{\text{inlier}}$. The foreground score $\mathbf{s}^t_{\text{FG}}$ is supervised using a combination of weighted binary cross-entropy (BCE) loss $\mathcal{L}_{\text{bce}}$ and Lovasz-Softmax loss $\mathcal{L}_{\text{ls}}$ [5]: $\mathcal{L}^{\bullet}_{\text{FG}} = \mathcal{L}_{\text{bce}}(\mathbf{s}^t_{\text{FG}}, \overline{\mathbf{s}}^t_{\text{FG}}) + \mathcal{L}_{\text{ls}}(\mathbf{s}^t_{\text{FG}}, \overline{\mathbf{s}}^t_{\text{FG}})$, with $\overline{\mathbf{s}}^t_{\text{FG}}$ the binary ground truth. The weights in $\mathcal{L}_{\text{bce}}$ are inversely proportional to the square root of elements in each class.

3.3 Motion segmentation

To separate the *moving* objects from the *static* ones we perform motion segmentation, reusing the per-frame base features $\{\mathbf{F}^t_{\text{base}}\}$. Specifically, we apply a differentiable feature warping scheme [45] that warps each $\mathbf{F}^t_{\text{base}}$ using the predicted ego-motion $\mathbf{T}^t_{\text{ego}}$, and obtain a spatio-temporal 3D feature tensor of size

$C \times T \times H \times W$ by stacking the warped feature maps along the channel dimension. This feature tensor is then fed through a series of 3D convolutional layers, followed by max-pooling across the temporal dimension T. Finally, we apply a small 2D UNet to obtain the 2D motion feature map $\mathbf{F}_{\text{motion}}$.

To mitigate discretization error, we bilinearly interpolate grid motion features to all foreground points in each frame.[1] The point-level motion feature for $\mathbf{x}_i^t$ is computed as:

$$\mathbf{f}_{\text{motion},i}^t = \text{MLP}\big(\text{cat}[\psi(\mathbf{x}_i^t, \mathbf{F}_{\text{motion}}), \text{MLP}(\mathbf{x}_i^t)]\big), \tag{4}$$

where $\text{MLP}(\cdot)$ denotes a multi-layer perceptron, $\text{cat}[\cdot]$ concatenation, and $\psi(\mathbf{x}, \mathbf{F})$ a bilinear interpolation from $\mathbf{F}$ to $\mathbf{x}$. The dynamic score $\mathbf{s}_i^t$ of the point $\mathbf{x}_i^t$ is then decoded from the motion feature $\mathbf{f}_{\text{motion},i}^t$ using another MLP, and supervised similar to the foreground segmentation, with a loss $\mathcal{L}_{\text{motion}}^{\bullet} = \mathcal{L}_{\text{bce}}(\mathbf{s}_i^t, \overline{\mathbf{s}}_i^t) + \mathcal{L}_{\text{ls}}(\mathbf{s}_i^t, \overline{\mathbf{s}}_i^t)$, where $\overline{\mathbf{s}}_i^t$ denotes the ground-truth motion label of point $\mathbf{x}_i^t$.

3.4 Spatio-temporal instance association

To segment the dynamic points (extracted by thresholding the $\mathbf{s}_i^t$) into individual objects and associate them over time, we perform spatio-temporal instance association. Different from the common tracking-by-detection [12,56] paradigm, we propose to directly *cluster the spatio-temporal point cloud*, which simultaneously provides instance masks and the corresponding associations. However, naive clustering of the ego-motion aligned point clouds often fails due to LiDAR sparsity and fast object motions, hence we predict a per-point offset vector $\boldsymbol{\delta}_i^t$ pointing towards the (motion-compensated) instance center:

$$\boldsymbol{\delta}_i^t = \text{MLP}\big(\text{cat}[\psi(\mathbf{x}_i^t, \mathbf{F}_{\text{motion}}), \text{MLP}(\mathbf{x}_i^t)]\big). \tag{5}$$

The DBSCAN [14] algorithm is subsequently applied over the deformed point set $\{\mathbf{x}_i^t + \boldsymbol{\delta}_i^t \mid \forall i, \forall t\}$ to obtain an instance index for each point. This association scheme is simple yet robust, and can seamlessly handle occlusions and misdetections. Similar to 3DIS [28] we supervise the offset predictions $\boldsymbol{\delta}_i^t$ with both an L_1-distance loss and a directional loss:

$$\mathcal{L}_{\text{offset}}^{\bullet} = \frac{1}{n} \sum_{\{i,t\}}^{n} \left(\left\| \boldsymbol{\delta}_i^t - \overline{\boldsymbol{\delta}}_i^t \right\|_1 + 1 - \langle \frac{\boldsymbol{\delta}_i^t}{\|\boldsymbol{\delta}_i^t\|}, \frac{\overline{\boldsymbol{\delta}}_i^t}{\|\overline{\boldsymbol{\delta}}_i^t\|} \rangle \right), \tag{6}$$

where $\overline{\boldsymbol{\delta}}$ is the ground truth offset $\mathbf{o} - \mathbf{x}$ from the associated instance centroid $\mathbf{o}$ in the target frame, and $\langle \cdot \rangle$ is the inner product.

3.5 Dynamic object motion modelling

Once we have spatio-temporally segmented objects, we must recover their motions at each frame. As LiDAR points belonging to a single object are sparse

[1] We predict motion labels only for foreground and treat background points as static.

and explicit inter-frame correspondences are hard to find, we take a different approach from the one used in the ego-motion head and construct a novel TubeNet to directly regress the transformations. Specifically, TubeNet takes T frames of the same instance $\mathbf{X}_k$ as input, and regresses its rigid motion parameters $\mathbf{T}_k^t$ as:

$$\mathbf{T}_k^t = \text{MLP}\left(\text{cat}[\tilde{\mathbf{f}}_{\text{motion}}, \tilde{\mathbf{f}}_{\text{ego}}, \tilde{\mathbf{f}}_{\text{pos}}^t, \tilde{\mathbf{f}}_{\text{pos}}^1]\right), \tag{7}$$

where $\tilde{\mathbf{f}}_{\text{motion}}$ and $\tilde{\mathbf{f}}_{\text{ego}}$ are instance-level global features obtained by applying PointNet [41] to the respective point-level features of that instance, $\tilde{\mathbf{f}}_* = \text{PN}(\{\mathbf{f}_{*,i}^t \mid \mathbf{x}_i^t \in \mathbf{X}_k^t\})$. Recall that point-level features $\mathbf{f}_{*,i}^t$ are computed from $\mathbf{F}_*$ via the interpolation scheme described in Eq (4). Here, $\tilde{\mathbf{f}}_{\text{motion}}$ encodes the overall instance motion while $\tilde{\mathbf{f}}_{\text{ego}}$ supplements additional geometric cues. The feature $\tilde{\mathbf{f}}_{\text{pos}}^t = \text{PN}(\mathbf{X}_k^t)$ is a summarized encoding over individual frames and provides direct positional information for accurate transformation estimation. The transformations are initialised to identity and TubeNet is applied in iterative fashion to regress residual transformations relative to the last iteration, similar to RPMNet [61].

For the loss function, we choose to parameterise each $\mathbf{T}$ as an un-normalised quaternion $\mathbf{q} \in \mathbb{R}^4$ and translation vector $\mathbf{t} \in \mathbb{R}^3$, and supervise it with:

$$\mathcal{L}_{\text{obj}}^{\bullet} = \mathcal{L}_{\text{trans}} + \frac{1}{T-1}\sum_{t=2}^{T}\left(\left\|\bar{\mathbf{t}}_k^t - \mathbf{t}_k^t\right\|_2 + \lambda\left\|\bar{\mathbf{q}}_k^t - \frac{\mathbf{q}_k^t}{\|\mathbf{q}_k^t\|}\right\|_2\right), \tag{8}$$

where $\bar{\mathbf{t}}_k^t$ and $\bar{\mathbf{q}}_k^t$ are the ground truth transformation, and λ is a constant weight, set to 50 in our experiments. $\mathcal{L}_{\text{trans}}$ is the same as in the ego-motion (§ 3.2).

3.6 Comparison to related work

WsRSF [19]. Our proposed method differs from WsRSF in several ways: *(i)* WsRSF is a pair-wise scene flow estimation method, while we can handle multiple frames; *(ii)* unlike WsRSF we perform motion segmentation for a more complete understanding of the scene dynamics; *(iii)* our method outputs instance-level associations, while WsRSF simply connects each instance to the complete foreground of the other point cloud.

MotionNet [57]. Similar to our method, MotionNet also deals with sequential point clouds and uses a BEV representation. However, MotionNet *(i)* assumes that ground truth ego-motion is available, while we estimate it within our network; and *(ii)* does not provide object-level understanding, rather it only separates the scene into a static and a dynamic part.

3.7 Implementation details

Our model is implemented in pytorch [37] and can be trained on a single RTX 3090 GPU. During training we minimize Eq (1) with the Adam [27] optimiser, with an initial learning rate 0.0005 that exponentially decays at a rate

of 0.98 per epoch. For both *Waymo* and *nuScenes*, the size of the pillars is $(\delta_x, \delta_y, \delta_z) = (0.25, 0.25, 8)$ m. We sample $N_{\text{ego}} = 1024$ points for ego-motion estimation and set $\tau = 0.5$ for foreground/background segmentation. The feature dimensions of $\mathbf{F}^t_{\text{base}}$, $\mathbf{F}^t_{\text{ego}}$, $\mathbf{F}_{\text{motion}}$ are 32, 64, 64 respectively. During inference we additionally use ICP [6] to perform test-time optimisation of the ego-motion as well as the transformation parameters of each dynamic object. ICP thresholds for ego-motion and dynamic object motion are 0.1/0.2 and 0.15/0.25 m for *Waymo* / *nuScenes*.

4 Experimental Evaluation

In this section, we first describe the datasets and the evaluation setting for our experiments (§ 4.1). We then proceed with a quantitative evaluation of our method and showcase its applicability to downstream tasks with qualitative results for surface reconstruction (§ 4.2). Finally, we validate our design choices in an ablation study (§ 4.3).

4.1 Datasets and evaluation setting

Waymo. The Waymo Open Dataset [46] includes 798/202 scenes for training/validation, where each scene is a 20-second clip captured by a 64-beam LiDAR at 10 Hz. We randomly sample 573/201 scenes for training/validation from the training split, and treat the whole validation split as a held-out test set. We consider every 5 consecutive frames as a *sample* and extract 20 *samples* from each clip, for a total of 11440/4013/4031 samples for training/validation/test.

nuScenes. The nuScenes dataset [7] consists of 700 training and 150 validation scenes, where each scene is a 20-second clip captured by a 32-beam LiDAR at 20 Hz. We use all validation scenes for testing and randomly hold out 150 training scenes for validation. We consider every 11 consecutive frames as a *sample*, resulting in a total of 10921/2973/2973 samples for training/validation/testing.

Ground truth. We follow [24] to construct pseudo ground-truth from the detection and tracking annotations. Specifically, the flow vectors of the background part are obtained from ground truth ego-motion. For foreground objects, we use the unique instance IDs of the tracking annotations and recover their rigid motion parameters by registering the bounding boxes. Notably, for *nuScenes* the bounding boxes are only annotated every 10 frames. To obtain pseudo ground truth for the remaining frames, we linearly interpolate the boxes, which may introduce a small amount of label noise especially for fast-moving objects.

Metrics. We use standard *scene flow* evaluation metrics [32,2] to compare the performance of our approach to the selected baselines. These metric include: *(i)* 3D end-point-error (*EPE* [m]) which denotes the mean L_2-error of all flow vectors averaged over all frames; *(ii)* strict/relaxed accuracy (*AccS* [%] / *AccR* [%]). *i.e.*,

Dataset	Method	Static part				Dynamic foreground				
		EPE avg.↓	AccS↑	AccR↑	ROutlier↓	EPE avg. ↓	EPE med.↓	AccS↑	AccR↑	ROutliers ↓
Waymo	PPWC-Net [58]	0.414	17.6	40.2	12.1	0.475	0.201	9.0	29.3	22.4
	FLOT [40]	0.129	65.2	85.0	2.8	0.625	0.231	9.8	27.4	33.8
	WsRSF [19]	0.063	87.3	96.6	0.6	0.381	0.094	31.3	64.0	10.1
	NSFPrior [31]	0.187	79.8	89.1	4.7	0.237	0.077	44.7	68.6	11.5
	Ours	0.028	97.5	99.5	0.1	0.197	0.062	53.3	77.5	5.9
	Ours+	**0.018**	**99.0**	**99.7**	**0.1**	**0.173**	**0.043**	**69.1**	**86.9**	**5.1**
	Ours (w. ground)	0.042	91.9	98.8	0.1	0.219	0.071	47.1	72.8	8.5
nuScenes	PPWC-Net [58]	0.316	16.1	37.0	8.7	0.661	0.307	7.6	24.2	31.9
	FLOT [40]	0.153	51.7	78.3	4.3	1.216	0.710	3.0	10.3	63.9
	WsRSF [19]	0.195	57.4	82.6	4.8	0.539	0.204	17.9	37.4	22.9
	NSFPrior [31]	0.584	38.9	56.7	26.9	0.707	0.222	19.3	37.8	32.0
	Ours	0.111	65.4	88.6	1.1	0.301	0.146	26.6	53.4	**12.1**
	Ours+	**0.091**	**72.8**	**91.9**	**0.9**	**0.301**	**0.135**	**32.7**	**56.7**	13.7
	Ours (w. ground)	0.134	55.3	83.8	1.9	0.37	0.182	18.2	43.8	17.5

Table 1: Scene flow results on *Waymo* and *nuScenes* datasets.

the fraction of points with $EPE < 0.05/0.10$m or relative error $< 0.05/0.10$; *(iii) Outliers [%]* which denotes the ratio of points with $EPE > 0.30$m or relative error > 0.10; and *(iv) ROutliers* [%], the fraction of points whose $EPE > 0.30$m and relative error > 0.30. We evaluate these metrics for the static and dynamic parts of the scene separately.[2] Following [57,35], we evaluate the performance of all methods only on the points that lie within the square of size 64×64 m^2 centered at the ego-car location in the target frame. Additionally we remove ground points by thresholding along the z-axis.[3] Ablations studies additionally report the quality of *motion segmentation* in terms of recall and precision of *dynamic* parts, and the quality of *spatio-temporal instance association* in terms of mean weighted coverage ($mWCov$, the IoU of recovered instances [54]). For further details, see the supplementary material.

Baselines. We compare our method to 4 baseline methods. PPWC [58] and FLOT [40] are based on dense matching and are trained in a fully supervised manner; WsRSF [19] assumes a *multi-body* scene and can be trained with weak supervision; NSFPrior [31] is an optimisation-based method without pre-training. For PPWC [58], FLOT [40] and WsRSF [19], we sample at most 8192 points from each frame due to memory constraints, and use the k-nn graph to up-sample flow vectors to full resolution at inference time. For NSFPrior [31], we use the full point clouds and take the default hyper-parameter settings given by the authors, except for the early-stopping patience, which we set to 50 to make it computationally tractable on our large-scale dataset. For all baseline methods, we directly estimate flow vectors from any *source* frame to the *target* frame.

[2] A point is labelled as *dynamic* if its ground-truth velocity is > 0.5 m/s.

[3] This setting turns out to better suit the baseline methods [40,58,31]. However, we keep ground points in our dynamic point cloud accumulation task, as thresholding could falsely remove points that are of interest for reconstruction or mapping.

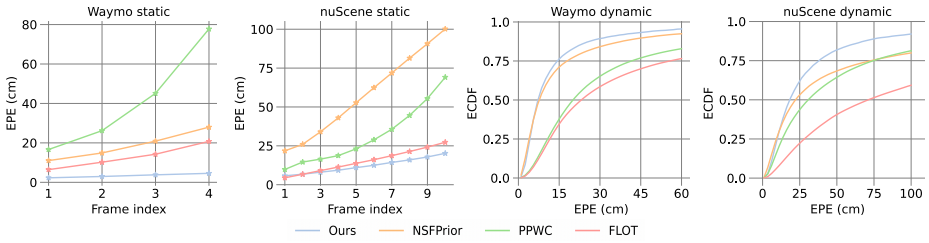

Fig. 3: Our method scales better to more frames. For the challenging dynamic parts, it also has smaller errors as well as fewer extreme outliers.

4.2 Main results

The detailed comparison on the *Waymo* and *nuScenes* data is given in Tab. 1. *Ours* denotes the direct output of our model, while *Ours+* describes the results after test-time refinement with ICP. Many downstream tasks (e.g., surface reconstruction) rely on the accumulation of full point clouds and require also ground points. We therefore also train a variant of our method on point clouds with ground points and denote it *Ours (w. ground)*. To facilitate a fair comparison, we use full point clouds as input during training and inference, but only compute the evaluation metrics on points that do not belong to ground.

Comparison to state of the art. On the static part of the ***Waymo*** dataset, FLOT [40] reaches the best performance among the baselines, but still has an average EPE error of 12.9 cm, which is more than 4 times larger than that of *Ours* (2.8 cm). This result corroborates our motivation for decomposing the scene into a static background and dynamic foreground. Modeling the motion of the background with a single set of transformation parameters also enables us to run ICP at test time (*Ours+*), which further reduces the EPE of the static part to 1.8 cm. On the dynamic foreground points, NSFPrior [31] reaches a comparable performance to *Ours*. However, based on our spatio-temporal association of foreground points we can again run ICP at test-time, which reduces the median EPE error to 4.3 cm, ≈40 % lower than that of NSFPrior. Furthermore, NSFPrior is an optimization method and not amenable to online processing (see Tab. 3). The results for ***nuScenes*** follow a similar trend as the ones for *Waymo*, but are larger in absolute terms due to the lower point density. Our method achieves the best performance on both static and dynamic parts. The gap to the closest competitors is even larger, which auggests that our method is more robust to low point density.

To further understand the error distribution of the dynamic parts, and the evolution of the errors of the static part as the gap between the *source* and *target* frames increases, we plot detailed results in Fig. 3. On both *Waymo* and *nuScenes* our method degrades gracefully, and slower than the baselines. The ECDF curve of the EPE error for foreground points also shows that our method performs best at all thresholds.

	3	4	5	6	7	8	9	10
static EPE avg.	**0.022**	0.025	0.028	0.032	0.037	0.044	0.054	0.066
dynamic EPE avg.	0.199	**0.168**	0.190	0.218	0.250	0.294	0.348	0.412

Table 2: Scene flow results on *Waymo* dataset w.r.t. input length.

	Waymo	*nuScenes*
PPWC-Net [58]	0.608	0.990
FLOT [40]	1.028	2.010
WsRSF [19]	1.252	1.460
NSFPrior [31]	212.256	63.460
Ours	**0.174**	**0.250**

	Waymo	*nuScenes*
ego-motion estimation	0.100	0.188
motion segmentation	0.024	0.040
instance association	0.036	0.009
TubeNet	0.014	0.013

Table 3: Runtimes for the *Waymo* and *nuScenes* datasets. All numbers are seconds per 5-frame (*Waymo*), respectively 11-frame (*nuScenes*) *sample*.

Breakdown of the performance gain. Overall, our method improves over baseline methods by a large margin on two datasets. The gains are a direct result of our design choices: *(i)* by modelling the flow of the background as rigid motion instead of unconstrained flow, we can greatly reduce *ROutlier* and improve the accuracy; *(ii)* different from [19] we perform motion segmentation, and can thus assign ego-motion flow to points on movable, but *static* objects ($\approx 75\%$ of the foreground). This further improves the results (see EPE of static FG in Tab. 4); *(iii)* reasoning on the object level, combined with spatio-temporal association and modelling, improves flow estimates for the *dynamic* foreground.

Generalisation to variable input length T. When trained with a fixed input length ($T = 5$ on *Waymo*), our model is able to generalize to different input lengths (see Tab. 2). The performance degrades moderately with increasing T, as the motions become larger than seen during training. Also, larger displacements make the correspondence problem inherently harder.

Runtime. We report runtimes for our model and several baseline methods on both datasets in Tab. 3 (left). Our method is significantly faster than all baselines under the multi-frame scene flow setting. We also report detailed runtimes of our model for individual steps in Tab. 3 (right). As we can see, backbone feature extraction and pairwise registration (*ego-motion estimation*) account for the majority of the runtime, 57.5% on *Waymo* and 75.2% on *nuScenes*. Note that this runtime is calculated over all frames, while under a data streaming setting, we only need to run the first part for a single incoming frame, then re-use the features of the previous frames at later stages, which will greatly reduce runtime: after initialisation, the runtime for every new sample decreases to around 0.094 s for *Waymo*, respectively 0.079 s on *nuScenes*.

Qualitative results. In Fig. 4 we show qualitative examples of scene and object reconstruction with our approach. By jointly estimating the ego-motion of the static part and the moving object motions, our method accumulates the corresponding points into a common, motion-compensated frame. It thus provides an excellent basis for 3D surface reconstruction.

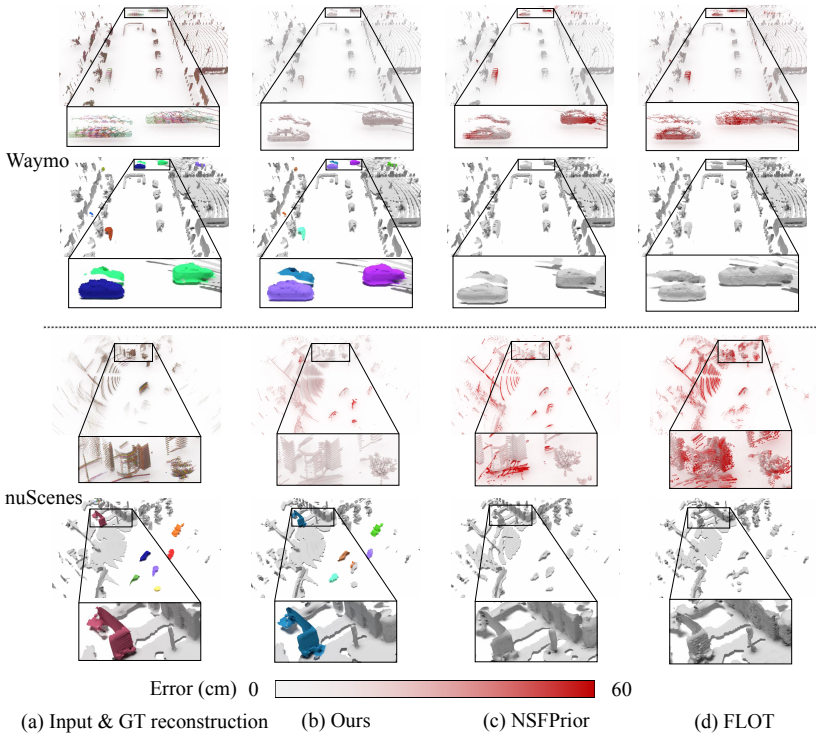

Fig. 4: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for two example scenes from *Waymo* and *nuScenes*.

4.3 Ablation Study

Sequential model. We evaluate the individual modules of our sequential model, namely the foreground segmentation (*FG*), motion segmentation (*MOS*) and offset compensation (*Offset*). We train two models with and without foreground segmentation. For variants without *MOS* or *Offset*, we take the trained full model but remove *MOS* or *Offset* at inference time. The detailed results are summarised in Tab. 4. *FG* enables us to exclude dynamic foreground objects during pairwise registration. On *Waymo*, this reduces EPE of the static parts by 30% from 4.1 to 2.9 cm, and as a result also reduces EPE of dynamic parts from 28.6 to 19.7 cm. By additionally extracting the static foreground parts with *MOS*, the model can recover more accurate ego-motion for them, which reduces EPE from 19.0/19.9 to 2.1/7.4 cm on *Waymo*/*nuScenes*. *Offset* robustifies the instance association against low point density and fast object motion (+3.1 *pp* in *WCov* on *nuScenes*), this further reduces the EPE of dynamic parts by 3.7 cm.

Ego-motion estimation strategy. By default, we directly estimate the ego-motion from any *source* frame to the *target* frame. We compare to an alternative which estimates the ego-motion relative to the previous frame. Although that

	Modules			Motion seg.		Association	Static BG	Static FG	Dynamic FG				
	FG	MOS	Offset	recall↑	precision↑	WCov↑	EPE avg.↓	EPE avg.↓	EPE avg.↓	EPE med.↓	AccS↑	AccR↑	ROutliers↓
Waymo		✓	✓	92.7	94.6	82.2	0.041	0.028	0.286	0.071	45.5	71.3	10.8
	✓		✓	-	-	**83.0**	0.031	0.190	0.198	0.062	**53.5**	77.4	6.3
	✓	✓		92.2	96.5	79.1	0.029	0.021	0.202	0.064	52.5	76.1	6.1
	✓	✓	✓	92.2	**96.8**	80.4	**0.029**	**0.021**	0.197	**0.062**	53.3	**77.5**	5.9
nuScenes		✓	✓	87.8	**92.5**	65.1	0.115	0.076	0.333	0.153	25.0	50.9	13.8
	✓		✓	-	-	**66.8**	0.118	0.199	**0.296**	**0.143**	**26.9**	**53.9**	**11.7**
	✓	✓		89.2	90.7	60.2	**0.113**	0.075	0.348	0.149	25.8	51.9	13.9
	✓	✓	✓	**89.3**	90.8	63.2	0.114	**0.074**	0.301	0.146	26.6	53.4	12.1
Waymo	Kalman tracker			92.3	96.6	77.1	0.030	0.027	0.586	0.099	36.3	61.6	11.7
	chained poses			**93.2**	94.9	81.9	0.044	0.030	**0.171**	0.068	48.0	77.0	**4.8**
nuScenes	Kalman tracker			89.4	90.8	42.9	0.114	0.092	1.238	0.364	10.7	25.1	44.0
	chained poses			88.1	90.2	62.1	0.225	0.151	0.315	0.155	23.4	51.8	13.4

Table 4: Ablation studies on *Waymo* and *nuScenes* datasets.

achieves smaller pairwise errors, after chaining the estimated poses the errors w.r.t. the *target* frame explode, resulting in inferior scene flow estimates (*chained poses* in Tab. 4).

Comparison to tracking-based method. Instead of running spatio-temporal association followed by TubeNet to model the motion of each moving object, an alternative would be to apply Kalman tracker so as to simultaneously solve association and motion. We compare to the modified AB3DMOT [56], which is based on a constant velocity model. That method first clusters moving points for each frame independently to obtain instances, then associates instances by greedy matching of instance centroids based on L_2 distances. The results in Tab. 4 (*Kalman tracker*) show clearly weaker performance, due to the less robust proximity metric based on distances between noisy centroid estimates.

5 Conclusion

We have looked at the analysis of 3D point cloud sequences from a fresh viewpoint, as point cloud accumulation across time. In that view we integrate point cloud registration, motion segmentation, instance segmentation, and piecewise rigid scene flow estimation into a complete multi-frame 4D scene analysis method. By jointly considering sequences of frames, our model is able to disentangle scene dynamics and recover accurate instance-level rigid-body motions. The model processes (ordered) raw point clouds, and can operate online with low latency. A major *limitation* is that our approach is fully supervised and heavily relies on annotated data: it requires instance-level segmentations as well as ground truth motions, although we demonstrate some robustness to label noise from interpolated pseudo ground truth. Also, our system consists of multiple processing stages and cannot fully recover from mistakes in early stages, like incorrect motion segmentation.

In *future work* we hope to explore our method's potential for downstream scene understanding tasks. We also plan to extend it to an incremental setting, where longer sequences of frames can be summarized into our holistic, dynamic scene representation in an online fashion.

References

1. Aygun, M., Osep, A., Weber, M., Maximov, M., Stachniss, C., Behley, J., Leal-Taixé, L.: 4d panoptic LiDAR segmentation. In: Proc. CVPR (2021) 3
2. Baur, S.A., Emmerichs, D.J., Moosmann, F., Pinggera, P., Ommer, B., Geiger, A.: SLIM: Self-supervised LiDAR scene flow and motion segmentation. In: Proc. ICCV (2021) 3, 5, 9
3. Behl, A., Paschalidou, D., Donné, S., Geiger, A.: PointFlowNet: Learning representations for rigid motion estimation from point clouds. In: Proc. CVPR (2019) 2, 3
4. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In: Proc. ICCV (2019) 2, 4
5. Berman, M., Triki, A.R., Blaschko, M.B.: The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: Proc. CVPR (2018) 6
6. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. International Society for Optics and Photonics (1992) 9
7. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: Proc. CVPR (2020) 2, 9
8. Chen, C., Yang, B.: Dynamic occlusion detection and inpainting of in situ captured terrestrial laser scanning point clouds sequence. ISPRS Journal of Photogrammetry and Remote Sensing **119**, 90–107 (2016) 3
9. Chen, X., Li, S., Mersch, B., Wiesmann, L., Gall, J., Behley, J., Stachniss, C.: Moving object segmentation in 3D LiDAR data: A learning-based approach exploiting sequential data. IEEE RA-L (2021) 3
10. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proc. CVPR (2019) 3
11. Cuturi, M.: Sinkhorn distances: Lightspeed computation of optimal transport. Proc. NeurIPS (2013) 6
12. Dendorfer, P., Osep, A., Milan, A., Schindler, K., Cremers, D., Reid, I., Roth, S., Leal-Taixé, L.: MOTchallenge: A benchmark for single-camera multiple target tracking. IJCV (2021) 7
13. Dewan, A., Caselitz, T., Tipaldi, G.D., Burgard, W.: Rigid scene flow for 3d lidar scans. In: Proc. IROS (2016) 2
14. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proc. KDD (1996) 7
15. Fan, H., Yang, Y., Kankanhalli, M.: Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In: Proc. CVPR (2021) 3
16. Fan, H., Yu, X., Ding, Y., Yang, Y., Kankanhalli, M.: PSTNet: Point spatio-temporal convolution on point cloud sequences. In: Proc. ICLR (2020) 3
17. Filatov, A., Rykov, A., Murashkin, V.: Any motion detector: Learning class-agnostic scene dynamics from a sequence of lidar point clouds. In: Proc. ICRA (2020) 4
18. Giancola, S., Zarzar, J., Ghanem, B.: Leveraging shape completion for 3d siamese tracking. In: Proc. CVPR (2019) 4
19. Gojcic, Z., Litany, O., Wieser, A., Guibas, L.J., Birdal, T.: Weakly supervised learning of rigid 3d scene flow. In: Proc. CVPR (2021) 2, 3, 5, 6, 8, 10, 12

20. Groß, J., Ošep, A., Leibe, B.: AlignNet-3D: Fast point cloud registration of partially observed objects. In: Proc. 3DV (2019) 4
21. Gu, J., Ma, W.C., Manivasagam, S., Zeng, W., Wang, Z., Xiong, Y., Su, H., Urtasun, R.: Weakly-supervised 3d shape completion in the wild. In: Proc. ECCV (2020) 4
22. Huang, J., Birdal, T., Gojcic, Z., Guibas, L.J., Hu, S.M.: Multiway non-rigid point cloud registration via learned functional map synchronization. IEEE T-PAMI (2022) 2, 4
23. Huang, J., Wang, H., Birdal, T., Sung, M., Arrigoni, F., Hu, S.M., Guibas, L.J.: MultiBodySync: Multi-body segmentation and motion estimation via 3d scan synchronization. In: Proc. CVPR (2021) 2, 4
24. Jund, P., Sweeney, C., Abdo, N., Chen, Z., Shlens, J.: Scalable scene flow from point clouds in the real world. arXiv preprint arXiv:2103.01306 (2021) 5, 9
25. Kabsch, W.: 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) 6
26. Kim, G., Kim, A.: Remove, then revert: Static point cloud map construction using multiresolution range images. In: Proc. IROS (2020) 3
27. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 8
28. Lahoud, J., Ghanem, B., Pollefeys, M., Oswald, M.R.: 3d instance segmentation via multi-task metric learning. In: Proc. ICCV (2019) 7
29. Li, R., Lin, G., He, T., Liu, F., Shen, C.: HCRF-Flow: Scene flow from point clouds with continuous high-order crfs and position-aware flow embedding. In: Proc. CVPR (2021) 3
30. Li, R., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: PU-GAN: a point cloud upsampling adversarial network. In: Proc. ICCV (2019) 4
31. Li, X., Kaesemodel Pontes, J., Lucey, S.: Neural scene flow prior. Proc. NeurIPS (2021) 10, 11, 12
32. Liu, X., Qi, C.R., Guibas, L.J.: FlowNet3D: Learning scene flow in 3d point clouds. In: Proc. CVPR (2019) 2, 3, 9
33. Liu, X., Yan, M., Bohg, J.: Meteornet: Deep learning on dynamic 3d point cloud sequences. In: Proc. ICCV (2019) 3
34. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proc. CVPR (2015) 6
35. Luo, C., Yang, X., Yuille, A.: Self-supervised pillar motion learning for autonomous driving. In: Proc. CVPR (2021) 10
36. Ouyang, B., Raviv, D.: Occlusion guided self-supervised scene flow estimation on 3d point clouds. arXiv preprint arXiv:2104.04724 (2021) 3
37. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. Proc. NeurIPS (2021) 8
38. Piergiovanni, A., Casser, V., Ryoo, M.S., Angelova, A.: 4d-net for learned multi-modal alignment. arXiv preprint arXiv:2109.01066 (2021) 3
39. Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R.: Long-term 3d map maintenance in dynamic environments. In: Proc. ICRA (2014) 3
40. Puy, G., Boulch, A., Marlet, R.: FLOT: Scene flow on point clouds guided by optimal transport. In: Proc. ECCV (2020) 2, 3, 10, 11, 12

41. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proc. CVPR (2017) 8
42. Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D.: Offboard 3d object detection from point cloud sequences. In: Proc. CVPR (2021) 3
43. Rempe, D., Birdal, T., Zhao, Y., Gojcic, Z., Sridhar, S., Guibas, L.J.: CaSPR: Learning canonical spatiotemporal point cloud representations. In: Proc. NeurIPS (2020) 4
44. Schauer, J., Nüchter, A.: The peopleremover—removing dynamic objects from 3-d point cloud data by traversing a voxel occupancy grid. IEEE RA-L **3**(3), 1679–1686 (2018) 3
45. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-net: Cnns for optical flow using pyramid, warping, and cost volume. In: Proc. CVPR (2018) 6
46. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proc. CVPR (2020) 2, 9
47. Teed, Z., Deng, J.: RAFT-3D: Scene flow using rigid-motion embeddings. In: Proc. CVPR (2021) 2
48. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Proc. NeurIPS (2017) 3
49. Vedula, S., Baker, S., Rander, P., Collins, R., Kanade, T.: Three-dimensional scene flow. In: Proc. ICCV (1999) 3
50. Vogel, C., Schindler, K., Roth, S.: 3d scene flow estimation with a rigid motion prior. In: Proc. ICCV (2011) 3
51. Vogel, C., Schindler, K., Roth, S.: Piecewise rigid scene flow. In: Proc. ICCV (2013) 2, 3
52. Vogel, C., Schindler, K., Roth, S.: 3d scene flow estimation with a piecewise rigid scene model. International Journal of Computer Vision **115**(1), 1–28 (2015) 2, 3
53. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: Proc. CVPR (2019) 4
54. Wang, X., Liu, S., Shen, X., Shen, C., Jia, J.: Associatively segmenting instances and semantics in point clouds. In: Proc. CVPR (2019) 10
55. Wedel, A., Rabe, C., Vaudrey, T., Brox, T., Franke, U., Cremers, D.: Efficient dense scene flow from sparse or dense stereo data. In: Proc. ECCV (2008) 3
56. Weng, X., Wang, J., Held, D., Kitani, K.: 3d multi-object tracking: A baseline and new evaluation metrics. In: Proc. IROS (2020) 7, 14
57. Wu, P., Chen, S., Metaxas, D.N.: MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In: Proc. CVPR (2020) 5, 8, 10
58. Wu, W., Wang, Z.Y., Li, Z., Liu, W., Fuxin, L.: Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In: Proc. ECCV (2020) 2, 3, 10, 12
59. Yang, B., Bai, M., Liang, M., Zeng, W., Urtasun, R.: Auto4d: Learning to label 4d objects from sequential point clouds. arXiv preprint arXiv:2101.06586 (2021) 3
60. Yang, Z., Zhou, Y., Chen, Z., Ngiam, J.: 3D-MAN: 3d multi-frame attention network for object detection. In: Proc. CVPR (2021) 3
61. Yew, Z.J., Lee, G.H.: RPM-Net: Robust point matching using learned features. In: Proc. CVPR (2020) 6, 8
62. Yuan, W., Khot, T., Held, D., Mertz, C., Hebert, M.: PCN: Point completion network. In: Proc. 3DV (2018) 4