DISS. ETH NO. 27078

Point Cloud to Pose, Pose to Point Cloud

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES of ETH ZURICH

(Dr. sc. ETH Zurich)

presented by

Shengyu Huang

MSc in Geomatik, ETH Zurich

born on 17.11.1995

citizen of China

accepted on the recommendation of

Prof. Dr. Konrad Schindler, examiner
Prof. Dr. Or Litany, co-examiner
Prof. Dr. Angela Dai, co-examiner
Prof. Dr. Simon Lucey, co-examiner
Prof. Dr. Andreas Wieser, co-examiner

2024

Abstract

placeholder for abstract

Kurzfassung

german text

german text

Acknowledgements

Thank you

Konrad, Andreas, Zan, Iro, Or, Sanja, Federico Michael, Michael, Keisuke, Titas, Lukas

Jiahui, Zian

Mikhail, Anton, Nico, Nikolai, Riccardo, Rodrigo, Alex, Nando, Ozgur, Manu, Corrine, Stefano, Bingxin, Binbin, Yujia, Torben,

Liyuan, Yuru, Hanfeng, Tao, Yuanwen,

Jixuan, Ye, Jimeng, Yikui, Lukas.

Lei Ke

Contents

1 Introduction

paper dissertations require an introduction which includes the overriding research question, the methodology used and the relevance of the thesis to science and economy the conclusion should merge the results of the publications and include suggestions for ongoing research

1.1. Outline & Contributions

The work presented in this thesis comprises articles published to journals and conferences of the fields of Computer Vision and Experimental Fluid Dynamics.

1.1.1. Predator

In Chapter ...

Contributions.

- A warping-based variational optical flow approach is adapted to volumetric particle flow estimation. To account also for larger displacements between time steps, a coarse-to-fine pyramid scheme is implemented. An efficient primal-dual optimization approach is used to minimize the energy, which is formulated as a saddle-point problem.
- The energy formulation is derived from the stationary Stokes equations, which model the incompressibility and viscosity of the fluid. Both, a hard constraint and an alternative soft constraint variant on the flow divergence are integrated into the variational framework. The Euler-Lagrange equations of the energy reveal that the viscosity of the fluid can be accounted for via quadratic regularization of the flow gradient.
- To handle also large measurement volumes, a semi-dense formulation is introduced. It takes into account the individual requirements of particle reconstruction and flow estimation. While the data term is evaluated at full resolution, the flow field is regularized at a lower spatial resolution.
- Extensive evaluations are carried out for different data terms and regularizers. Further, ablation studies on the matching window size, seeding density and stepsize of the semi-dense formulation are performed, justifying the chosen parameters and the effectiveness of the presented approach.

1.2. Relevance to Science and Economy

xxx

Open Source.. xxx

Scientific Recognition.. xxx

2 Background

placeholder

2.1. Point cloud register

3 Dynamic 3D Scene Analysis by Point Cloud Accumulation

Shengyu Huang, Zan Gojcic, Jiahui Huang, Andreas Wieser, Konrad Schindler
European Conference on Computer Vision, 2022

(Author version; for typeset version please refer to the original conference paper.)

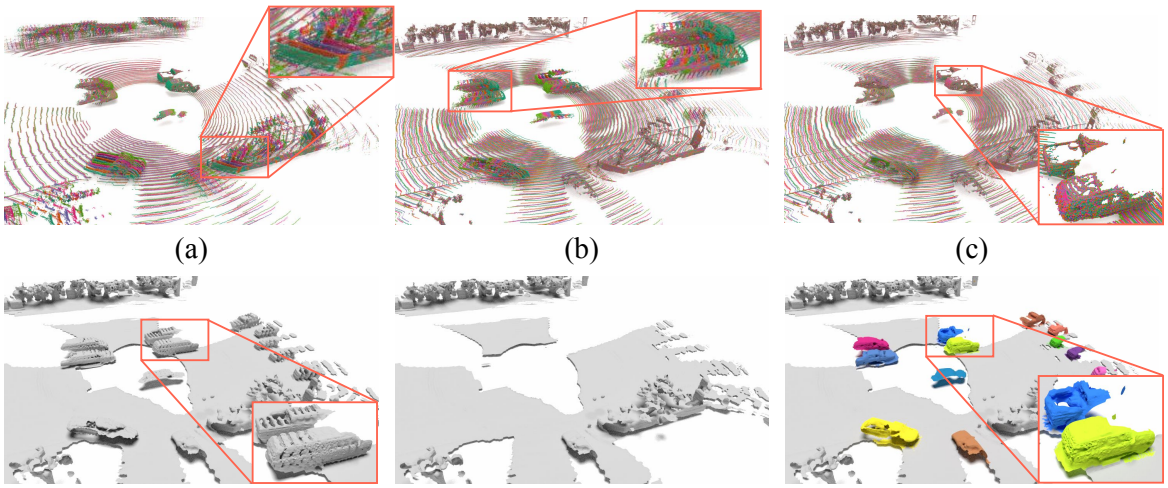

Figure 3.1.: Points in LiDAR frames acquired over time are not aligned due to the motion of the sensor and of other agents in the scene (a, d). Static background points can be aligned using ego-motion, but this smears the dynamic points across their trajectories (b). While motion segmentation only enables 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`]

3.1. 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. 3.1b).

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 [35, 63, 44] 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 [56, 57, 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, 52] and reasoning about each agent's motion on the basis of a longer sequence of frames [24, 23]. Along with the accumulated point cloud (Fig. 3.1c), 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. 3.1f) and semantic segmentation [4].

We carry out extensive evaluations on two autonomous driving datasets *Waymo* [50] 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 fore-

ground 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.

3.2. 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 [54], 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 [60, 55, 56, 57] leverage motion smoothness as regularizer within their optimization frameworks, modern learning-based methods learn the preference for smooth motions directly from large-scale datasets [35, 63, 44, 39]. Moreover, manually designed scene priors proved beneficial for structured scenes, for instance supervoxel-based local rigidity constraints [32], 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 [36, 10, 16] demonstrates superior performance and efficiency. Subsequent works enhance such backbones by employing long-range modeling techniques such as Transformers [53, 15, 65], or by coupling downstream tasks like semantic segmentation [1], object detection [64, 46] and multi-modal fusion [42]. 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, 48] over dense terrestrial laser scans to build clean static maps, or on visibility [43, 29] to determine the dynamics of the query point by checking its occlusion state in a dense map. Removert [29] 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 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

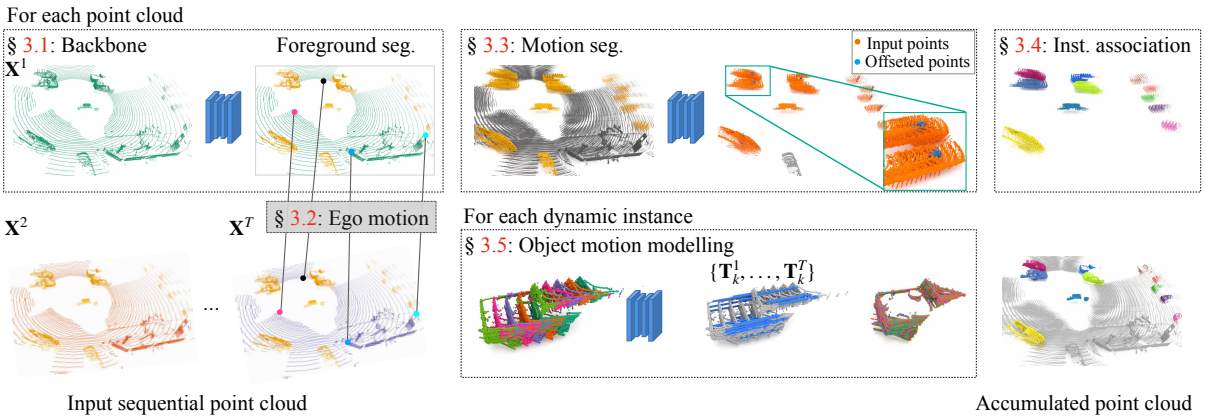

Figure 3.2.: **Overview**. Our method takes in a point cloud sequence of T frames and starts by extracting foreground points (marked yellow) for each frame. To obtain ego motion, $T-1$ pairwise registrations are performed. Next, points belonging to dynamic foreground object are extracted using our motion 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.

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 [67, 18, 33] and ignore the local geometric structure. [22] 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 [21] that directly regresses the relative transformation matrix from concatenated global features of the two point clouds. NOCS [58] proposes a category-aware canonical representation that can be used to estimate instance pose w.r.t. its canonical pose. Caspr [47] implicitly accumulates the shapes by mapping a sequence of partial observations to a continuous latent space. [24] and [23] respectively propose multi-way registration methods that accumulate multi-body and non-rigid dynamic point clouds, but do not scale well to large scenes.

3.3. Method

The network architecture of our multitask model is schematically depicted in Fig. 3.2. 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 (§ 3.3.1), which we then use as input to the task-specific heads. To estimate the ego-motion, we employ a differentiable registration module (§ 3.3.2). 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 (§ 3.3.3) as well as spatio-temporal association of dynamic foreground objects (§ 3.3.4). Finally, we decode the rigid body motion of each foreground object from its spatio-temporal features (§ 3.3.5). We

train the entire model end-to-end with a loss $\mathcal{L}$ composed of five terms:

$$\mathcal{L} = \mathcal{L}_{\text{ego}} + \mathcal{L}_{\text{FG}} + \mathcal{L}_{\text{motion}} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \,. \tag{3.1}$$

$$\mathcal{L} = \mathcal{L}_{\text{ego}} + \mathcal{L}_{\text{FG}} + \mathcal{L}_{\text{motion}} + \lambda_{\text{offset}}\mathcal{L}_{\text{offset}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} \,. \tag{3.2}$$

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 $\mathbf{X}^t_{\text{static}}$ and K_t rigidly-moving *dynamic* parts $\mathcal{X}^t_{\text{dynamic}} = \{\mathbf{X}^t_k\}_{k=1}^{K_t}$ [19]. For an individual frame, the scene flow vectors $\mathbf{V}^t_{\text{static}}$ of the static part and $\mathbf{V}^t_k$ of the k-th dynamic object can be explained by the rigid ego-motion $\mathbf{T}^t_{\text{ego}} \in \text{SE}(3)$ and the motion of the dynamic object relative to the static background $\mathbf{T}^t_k \in \text{SE}(3)$, respectively as:

$$\mathbf{V}^t_{\text{static}} = \mathbf{T}^t_{\text{ego}} \circ \mathbf{X}^t_{\text{static}} - \mathbf{X}^t_{\text{static}}, \qquad \mathbf{V}^t_k = \mathbf{T}^t_k \mathbf{T}^t_{\text{ego}} \circ \mathbf{X}^t_k - \mathbf{X}^t_k, \tag{3.3}$$

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

3.3.1. Backbone network

Similar to [2, 26, 62], 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 [37] 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.

3.3.2. 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 § 3.3.3.

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:

$$\mathbf{T}^t_{\text{ego}} = \text{argmin}_{\mathbf{T}} \sum_{l=1}^{N_{\text{ego}}} w^t_l \left\| \mathbf{T} \circ \mathbf{p}^t_l - \phi(\mathbf{p}^t_l, \mathbf{P}^1) \right\|^2 . \tag{3.4}$$

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, 66] and the optimal $\mathbf{T}^t_{\text{ego}}$ is computed in closed form via the differentiable Kabsch algorithm [27].

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 $\mathcal{L}_{\text{inlier}}$ [66] that regularizes the Sinkhorn algorithm, $\mathcal{L}_{\text{ego}} = \mathcal{L}_{\text{trans}} + \mathcal{L}_{\text{inlier}}$. 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]: $\mathcal{L}_{\text{FG}} = \mathcal{L}_{\text{bce}}(\mathbf{s}^t_{\text{FG}}, \overline{\mathbf{s}}^t_{\text{FG}}) + \mathcal{L}_{\text{ls}}(\mathbf{s}^t_{\text{FG}}, \overline{\mathbf{s}}^t_{\text{FG}})$, 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.

3.3.3. 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 [49] 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 $\mathbf{x}^t_i$ is computed as:

$$\mathbf{f}^t_{\text{motion},i} = \text{MLP}\big(\text{cat}[\psi(\mathbf{x}^t_i, \mathbf{F}_{\text{motion}}), \text{MLP}(\mathbf{x}^t_i)]\big), \tag{3.5}$$

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 $\mathbf{s}^t_i$ of the point $\mathbf{x}^t_i$ is then decoded from the motion feature $\mathbf{f}^t_{\text{motion},i}$ using another MLP, and supervised similar to the foreground segmentation, with a loss $\mathcal{L}_{\text{motion}} = \mathcal{L}_{\text{bce}}(\mathbf{s}^t_i, \overline{\mathbf{s}}^t_i) + \mathcal{L}_{\text{ls}}(\mathbf{s}^t_i, \overline{\mathbf{s}}^t_i)$, where $\overline{\mathbf{s}}^t_i$ denotes the ground-truth motion label of point $\mathbf{x}^t_i$.

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

3.3.4. 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, 61] 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:

$$\boldsymbol{\delta}_i^t = \mathrm{MLP}\big(\mathrm{cat}[\psi(\mathbf{x}_i^t, \mathbf{F}_{\mathrm{motion}}), \mathrm{MLP}(\mathbf{x}_i^t)]\big). \tag{3.6}$$

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 mis-detections. Similar to 3DIS [31] we supervise the offset predictions $\boldsymbol{\delta}_i^t$ with both an L_1-distance loss and a directional loss:

$$\mathcal{L}_{\mathrm{offset}} = \frac{1}{n}\sum_{\{i,t\}}^{n}\left(\left\|\boldsymbol{\delta}_i^t - \overline{\boldsymbol{\delta}}_i^t\right\|_1 + 1 - \langle\frac{\boldsymbol{\delta}_i^t}{\|\boldsymbol{\delta}_i^t\|}, \frac{\overline{\boldsymbol{\delta}}_i^t}{\|\overline{\boldsymbol{\delta}}_i^t\|}\rangle\right), \tag{3.7}$$

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.

3.3.5. 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 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:

$$\mathbf{T}_k^t = \mathrm{MLP}\left(\mathrm{cat}[\tilde{\mathbf{f}}_{\mathrm{motion}}, \tilde{\mathbf{f}}_{\mathrm{ego}}, \tilde{\mathbf{f}}_{\mathrm{pos}}^t, \tilde{\mathbf{f}}_{\mathrm{pos}}^1]\right), \tag{3.8}$$

where $\tilde{\mathbf{f}}_{\mathrm{motion}}$ and $\tilde{\mathbf{f}}_{\mathrm{ego}}$ are instance-level global features obtained by applying PointNet [45] to the respective point-level features of that instance, $\tilde{\mathbf{f}}_* = \mathrm{PN}(\{\mathbf{f}_{*,i}^t \mid \mathbf{x}_i^t \in \mathbf{X}_k^t\})$. Recall that point-level features $\mathbf{f}_{*,i}^t$ are computed from $\mathbf{F}_*$ via the interpolation scheme described in Eq. (3.5). Here, $\tilde{\mathbf{f}}_{\mathrm{motion}}$ encodes the overall instance motion while $\tilde{\mathbf{f}}_{\mathrm{ego}}$ supplements additional geometric cues. The feature $\tilde{\mathbf{f}}_{\mathrm{pos}}^t = \mathrm{PN}(\mathbf{X}_k^t)$ 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 [66].

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:

$$\mathcal{L}_{\text{obj}}^{\bullet} = \mathcal{L}_{\text{trans}} + \frac{1}{T-1} \sum_{t=2}^{T} \left(\left\| \bar{\mathbf{t}}_k^t - \mathbf{t}_k^t \right\|_2 + \lambda \left\| \bar{\mathbf{q}}_k^t - \frac{\mathbf{q}_k^t}{\|\mathbf{q}_k^t\|} \right\|_2 \right), \tag{3.9}$$

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 (§ 3.3.2).

3.3.6. Comparison to related work

WsRSF [19]. Our proposed method differs from WsRSF in several ways: *(i)* WsRSF is a pairwise 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 [62]. 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.

3.3.7. Implementation details

Our model is implemented in pytorch [40] and can be trained on a single RTX 3090 GPU. During training we minimize Eq. (3.1) with the Adam [30] 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 $\mathbf{F}_{\text{base}}^t$, $\mathbf{F}_{\text{ego}}^t$, $\mathbf{F}_{\text{motion}}$ 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*.

3.4. Experimental Evaluation

In this section, we first describe the datasets and the evaluation setting for our experiments (**??**). We then proceed with a quantitative evaluation of our method and showcase its applicability to downstream tasks with qualitative results for surface reconstruction (§ 3.4.2). Finally, we validate our design choices in an ablation study (§ 3.4.3).

Dataset	Method	Static part				Dynamic foreground				
		EPE avg.↓	AccS↑	AccR↑	ROutlier↓	EPE avg. ↓	EPE med.↓	AccS↑	AccR↑	ROutliers ↓
Waymo	PPWC-Net [63]	0.414	17.6	40.2	12.1	0.475	0.201	9.0	29.3	22.4
	FLOT [44]	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 [34]	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 [63]	0.316	16.1	37.0	8.7	0.661	0.307	7.6	24.2	31.9
	FLOT [44]	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 [34]	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 3.1.: Scene flow results on *Waymo* and *nuScenes* datasets.

3.4.1. Datasets and evaluation setting

Waymo. The Waymo Open Dataset [50] 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 [26] 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 [35, 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.*, the fraction of points with *EPE* < 0.05/0.10m or relative error < 0.05/0.10; *(iii) Outliers [%]* which denotes the ratio of points with *EPE* > 0.30m or

relative error > 0.10; and *(iv) ROutliers* [%], the fraction of points whose *EPE* > 0.30m and relative error > 0.30. We evaluate these metrics for the static and dynamic parts of the scene separately.[2] Following [62, 38], 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 [59]). For further details, see the supplementary material.

Baselines. We compare our method to 4 baseline methods. PPWC [63] and FLOT [44] 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 [34] is an optimisation-based method without pre-training. For PPWC [63], FLOT [44] 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 [34], 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.

3.4.2. Main results

The detailed comparison on the *Waymo* and *nuScenes* data is given in Tab. 3.1. *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 [44] 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 [34] 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, $\approx$40 % lower than that of NSFPrior. Furthermore, NSFPrior is an optimization method and not amenable to online processing (see

[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 [44, 63, 34]. 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.

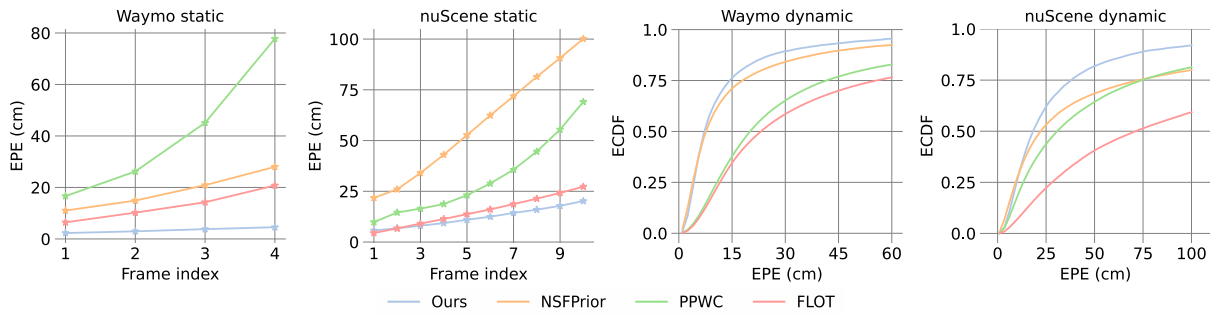

Figure 3.3.: Our method scales better to more frames. For the challenging dynamic parts, it also has smaller errors as well as fewer extreme outliers.

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

Tab. 3.3). 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. 3.3. 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.

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. 3.4); *(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. 3.2). 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. 3.3 (left). Our method is significantly faster than all baselines under the multi-frame

	Waymo	*nuScenes*
PPWC-Net [63]	<u>0.608</u>	<u>0.990</u>
FLOT [44]	1.028	2.010
WsRSF [19]	1.252	1.460
NSFPrior [34]	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 3.3.: Runtimes for the *Waymo* and *nuScenes* datasets. All numbers are seconds per 5-frame (*Waymo*), respectively 11-frame (*nuScenes*) *sample*.

	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		✓	✓	<u>92.7</u>	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 3.4.: Ablation studies on *Waymo* and *nuScenes* datasets.

scene flow setting. We also report detailed runtimes of our model for individual steps in Tab. 3.3 (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. 3.4 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.

3.4.3. 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. 3.4. *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

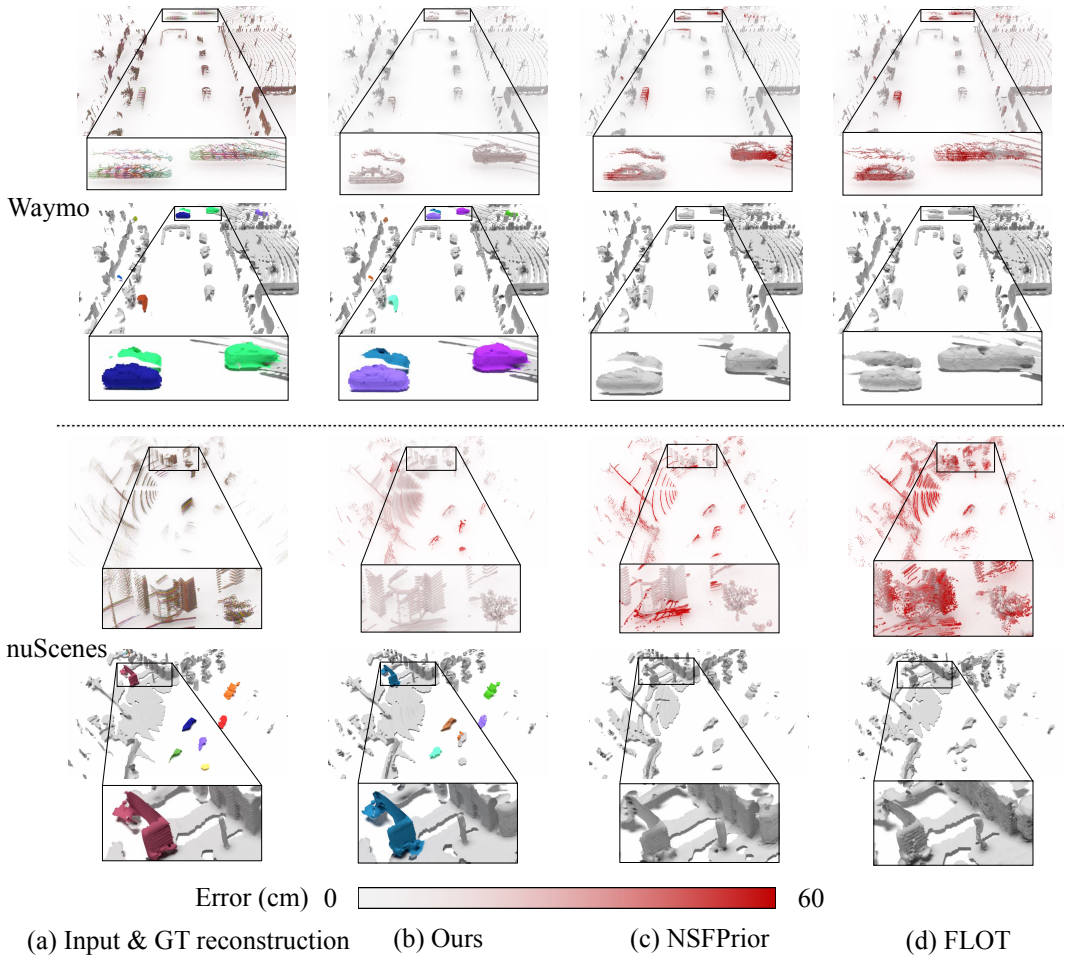

Figure 3.4.: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for two example scenes from *Waymo* and *nuScenes*.

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 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. 3.4).

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 [61], 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. 3.4 (*Kalman*

tracker) show clearly weaker performance, due to the less robust proximity metric based on distances between noisy centroid estimates.

3.5. 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 piece-wise 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.

3.6. Appendix

In this supplementary document, we first present additional information about our network architecture and implementation in § 3.6.1. We then elaborate on technical details of ego-motion estimation, iterative pose refinement, and scene reconstruction in § 3.6.2. Precise definitions of loss functions and evaluation metrics are provided in § 3.6.3. Further analysis of the two datasets, *nuScenes* and *Waymo*, is presented in § 3.6.4, followed by additional quantitative results including scene flow estimation, instance association, and the TubeNet motion model in § 3.6.5. Finally, we show more qualitative results in § 3.6.6.

3.6.1. Network and implementation

Network architecture. The detailed network architecture is depicted in Fig. 3.10. Our network is a sequential model consisting of (i) per-frame feature extraction used to estimate ego-motion, (ii) multi-frame feature extraction to segment dynamic objects and regress offset vectors towards the associated instance center, and (iii) TubeNet to regress the rigid motions of dynamic objects. The Pillar encoder and the two UNets operate without Batch Normalisation [25], this speeds up training and inference without any loss in performance [41]. Our network achieves flexibility w.r.t. the number of input frames via global max-pooling along the temporal dimension in the InitConv3D block (Fig. 3.10).

Implementation details. We use `torch_scatter`[4] to efficiently convert point-wise features to pillar/instance-level global features. To spatially align the backbone features we use `grid_sample`, implemented in PyTorch [40]. Before clustering, we apply voxel down-sampling implemented in TorchSparse [51] to reduce the point density and improve clustering efficiency. The voxel size is set to 15 cm. Instance labels at full resolution are recovered by indexing points to their associated voxel cell.

3.6.2. Methodology

Ego-motion estimation. Given two sets $(\mathbf{P}^1, \mathbf{P}^t)$ of pillar centroid coordinates and associated L_2-normalised features $(\mathbf{F}^1_{\text{ego}}, \mathbf{F}^t_{\text{ego}})$, we first compute the cost matrix $\mathbf{M}^t = 2 - 2\langle \mathbf{F}^t_{\text{ego}}, \mathbf{F}^1_{\text{ego}} \rangle$ and an Euclidean distance matrix $\mathbf{D}^t_{l,m} = \|\mathbf{p}^t_l - \mathbf{p}^1_m\|_2$ from pillar coordinates. We then pad $\mathbf{M}^t$ with a learnable slack row and column to accommodate outliers, before iteratively alternating between row normalisation and column normalisation[5] for five times to approximate a doubly stochastic permutation matrix $\mathbf{S}^t$ that satisfies

$$\sum_{l=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall m = 1, ..., N_{\text{ego}}, \quad \sum_{m=1}^{N_{\text{ego}}+1} \mathbf{S}^t_{l,m} = 1, \forall l = 1, ..., N_{\text{ego}}, \quad \mathbf{S}^t_{l,m} \geq 0. \tag{3.10}$$

[4] `https://github.com/rusty1s/pytorch_scatter`

[5] To improve training stability, row and column normalisations operate in log-space.

Here $\mathbf{S}^t_{l,m}$ represents the probability of $(\mathbf{p}^t_l, \mathbf{p}^1_m)$ being in correspondence. $\mathbf{p}^t_l$ is considered as an outlier and should be ignored during pose estimation if its slack column value $\mathbf{S}^t_{l,-1} \to 1$. We further mask $\mathbf{S}^t$ using a support matrix $\mathbf{I}^t$ computed from $\mathbf{D}^t$ as:

$$\mathbf{I}^t = \left(\mathbf{D}^t < s\right), \quad s = v \cdot \Delta t \ , \tag{3.11}$$

where v is the maximum speed and Δt is the interval between two frames. The final corresponding point $\phi(\mathbf{p}^t_l, \mathbf{P}^1)$ of $\mathbf{p}^t_l$ and its weight w^t_l are computed as

$$\phi(\mathbf{p}^t_l, \mathbf{P}^1) = (\mathbf{I}^t \odot \mathbf{D}^t)_{[l,:-1]}\mathbf{X}^1, \quad w^t_l = \sum_{m=1}^{N_{\text{ego}}} (\mathbf{I}^t \odot \mathbf{D}^t)_{l,m} \ , \tag{3.12}$$

with $\odot$ the Hadamard product. 3.4 is solved with the Kabsch algorithm. For a detailed derivation, please refer to [19]. The value of v is dataset-specific, we set it to 30 m/s for the *Waymo*, respectively 10 m/s for *nuScenes*.

Iterative refinement of TubeNet estimates. To improve the estimation of the transformation parameters for dynamic objects, we unroll TubeNet for two iterations, as often done in point cloud registration [66, 20]. Specifically, for a dynamic object $\mathbf{X}_k$, we first estimate the initial rigid transformation $\mathbf{T}^{0,t}_k$ of the t^{th} frame $\mathbf{X}^t_k$ following 3.8. We then obtain the transformed points $\mathbf{X}^{t'}_k = \mathbf{T}^{0,t}_k \circ \mathbf{X}^t_k$. Next, we update the positional feature $\tilde{\mathbf{f}}^{t'}_{\text{pos}} = \text{PN}(\mathbf{X}^{t'}_k)$ and regress the residual transformation matrix $\mathbf{T}^{t,1}_k$ again, according to 3.8. The final transformation is $\mathbf{T}^{t,1}_k \cdot \mathbf{T}^{t,0}_k$. For better stability during training, the gradients between the two iterations are detached. We assign higher weight to the latter iteration to improve accuracy. The overall loss $\mathcal{L}^{\bullet}_{\text{obj}}$ is:

$$\mathcal{L}^{\bullet}_{\text{obj}} = 0.7 \cdot \mathcal{L}^{\bullet,0}_{\text{obj}} + \mathcal{L}^{\bullet,1}_{\text{obj}} \tag{3.13}$$

Scene reconstruction. To show the benefits of our method for downstream tasks, we use the points accumulated with different methods as a basis for 3D surface reconstruction, see Fig. 3.8 and 3.9. Specifically, we use the accumulated points as input to the Poisson reconstruction [28], implemented in Open3D [68]. To estimate point cloud normals, the neighborhood radius is set to 0.5 m and the maximum number of neighbors is set to 128. The depth for the Poisson method is set to 10, and the reconstructed meshes are filtered by removing vertices with densities below the 15^{th} percentile.

3.6.3. Loss functions and evaluation metrics

Weighted BCE loss. To compensate for class imbalance, we use a weighted BCE and compute the weights of each class on the fly. Specifically, for a mini-batch with N_{pos} positive and N_{neg} negative samples, the associated weights w_{pos} and w_{neg} are computed as:

$$w_{\text{pos}} = \min\left(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{pos}}}}, w_{\text{max}}\right) \qquad w_{\text{neg}} = \min\left(\sqrt{\frac{N_{\text{pos}} + N_{\text{neg}}}{N_{\text{neg}}}}, w_{\text{max}}\right) \ , \tag{3.14}$$

where w_{max} is the maximum weight of a class.[6] The final weighted BCE loss $\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}})$ is

$$\mathcal{L}_{\text{bce}}(\mathbf{x}, \overline{\mathbf{x}}) = \frac{1}{|\mathbf{x}|} \sum_{i=1}^{|\mathbf{x}|} w_i (\overline{\mathbf{x}}_i \log(\mathbf{x}_i) + (1 - \overline{\mathbf{x}}_i) \log(1 - \mathbf{x}_i)) \ , \tag{3.15}$$

with $\mathbf{x}$ and $\overline{\mathbf{x}}$ are predicted and ground truth labels, and w_i the weight of the i^{th} sample, computed as

$$w_i = \begin{cases} w_{\text{pos.}}, & \text{if } \overline{\mathbf{x}}_i = 1 \\ w_{\text{neg.}}, & \text{otherwise} \end{cases} \ . \tag{3.16}$$

Lovász-Softmax loss. The Jaccard index (ratio of Intersection over Union) is commonly used to measure segmentation quality. In the binary classification setting, we can set the ground truth labels as $\overline{\mathbf{x}}_i \in \{-1, 1\}$, then the Jaccard index of the foreground class J_1 is computed as

$$J_1(\overline{\mathbf{x}}, \mathbf{x}) = \frac{|\{\overline{\mathbf{x}} = 1\} \cap \{\text{sign}(\mathbf{x}) = 1\}|}{|\{\overline{\mathbf{x}} = 1\} \cup \{\text{sign}(\mathbf{x}) = 1\}|} \ , \quad J_1(\overline{\mathbf{x}}, \mathbf{x}) \in [0, 1] \ , \tag{3.17}$$

with $\mathbf{x}$ the prediction and $\text{sign}()$ the sign function. The corresponding loss $\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x})$ to minimise the empirical risk is

$$\Delta_{J_1}(\overline{\mathbf{x}}, \mathbf{x}) = 1 - J_1(\overline{\mathbf{x}}, \mathbf{x}) \ . \tag{3.18}$$

However, this is not differentiable and cannot be directly employed as a loss function. The authors of [5] have proposed to optimise it using a Lovász extension. The Lovász extension $\dddot{\Delta}$ of a set function Δ is defined as:

$$\dddot{\Delta}(\mathbf{m}) = \sum_{i=1}^{p} \mathbf{m}_i \, g_i(\mathbf{m}) \ , \tag{3.19}$$

with

$$g_i(\mathbf{m}) = \Delta(\{\pi_1, \ldots, \pi_i\}) - \Delta(\{\pi_1, \ldots, \pi_{i-1}\}), \tag{3.20}$$

where π denotes a permutation that places the components of $\mathbf{m}$ in decreasing order. Considering $\mathbf{m}_i = \max(1 - \mathbf{x}_i \overline{\mathbf{x}}_i, 0)$, the Lovász-Softmax loss $\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x})$ is defined as

$$\mathcal{L}_{ls}(\overline{\mathbf{x}}, \mathbf{x}) = \dddot{\Delta}_{J_1}(\mathbf{m}). \tag{3.21}$$

Inlier loss. Previous works [66, 19] have observed that the entropy-regularized optimal transport [11] has a tendency to label most points as outliers. To alleviate this issue, we follow [66, 19] and use an inlier loss $\mathcal{L}_{\text{inlier}}$ on the matching matrix $\mathbf{D}^t$, designed to encourage inliers. The inlier loss is defined as

$$\mathcal{L}^t_{\text{inlier}} = \frac{1}{2N_{\text{ego}}} (2N_{\text{ego}} - \sum_{l=1}^{N_{\text{ego}}} \sum_{m=1}^{N_{\text{ego}}} \mathbf{D}^t_{l,m}) \ . \tag{3.22}$$

[6] We find that in some extreme cases, there are very few (< 10) positive samples and way more negative samples. We thus bound w_{max} at 50 to ensure stability.

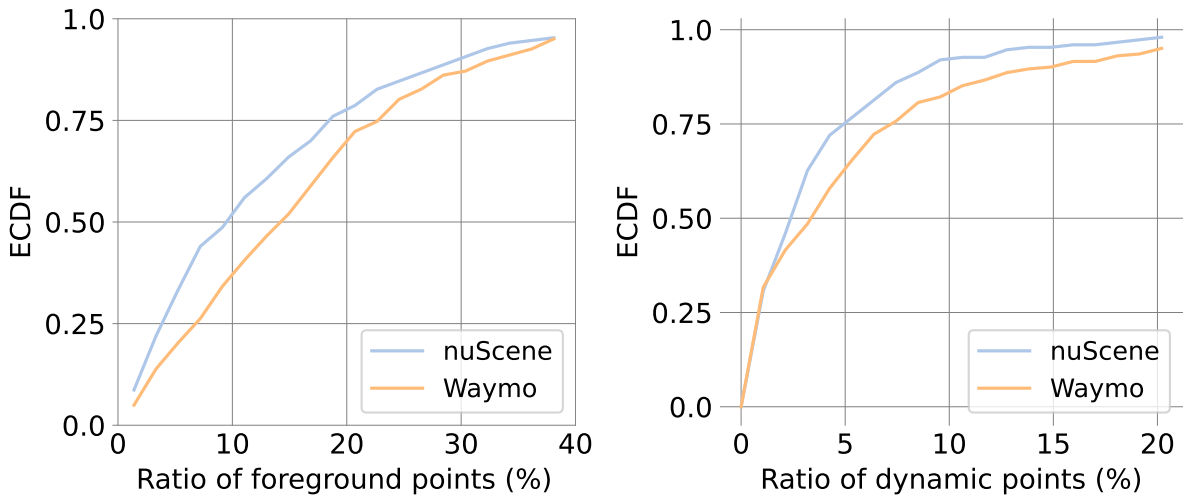

Figure 3.5.: *ECDF* curve of points lying on foreground objects and on dynamic objects, for both the *Waymo* and *nuScenes* datasets.

Instance association metrics. To quantitatively measure the spatio-temporal instance association quality, we report weighted coverage (*WCov*) as well as recall and precision at a certain threshold. Given the ground truth clusters $\mathcal{G}$ and the estimated clusters $\mathcal{O}$, recall measures the ratio of clusters in $\mathcal{G}$ that have an overlap above some threshold with a cluster in $\mathcal{O}$, while precision does the same in the opposite direction. Weighted coverage $WCov(\mathcal{G}, \mathcal{O})$ is computed as

$$WCov(\mathcal{G}, \mathcal{O}) = \sum_{i=1}^{|\mathcal{G}|} \frac{1}{|\mathcal{G}|} w_i \max_j \mathrm{IoU}(r_i^G, r_j^0), \qquad w_i = \frac{|r_i^G|}{\sum_k |r_k^G|}, \tag{3.23}$$

where r_i^G and r_j^O are clusters from $\mathcal{G}$ and $\mathcal{O}$, and $\mathrm{IoU}(r_i^G, r_j^O)$ denotes the overlap between two clusters.

ECDF. The Empirical Cumulative Distribution Function (ECDF) measures the distribution of a set of values:

$$\mathrm{ECDF}(x) = \frac{\big|\{o_i < x\}\big|}{\big|O\big|}, \tag{3.24}$$

where $O = \{o_i\}$ is a set of samples and $x \in [\min\{O\}, \max\{O\}]$.

3.6.4. Dataset analysis

In total, we have 150, respectiverly 202 scenes as held-out test sets in *nuScenes* and *Waymo*. The *ECDF* curve of points belonging to foreground and dynamic objects are shown in Fig. 3.5. As can be seen, the ratios of foreground and dynamic points span a large range (40% and 20%). Recalling that the scene flow estimation performance of the dynamic parts falls far behind that of the static parts (3.1), this large range of ratios of dynamic objects hints at different difficulties across the scenes. The median fractions of foreground points are 16.2%/9.4% in *Waymo*/*nuScenes*, the median fractions of points on moving objects are 3.5%/2.4%. In other words, roughly 75% of all foreground objects are static. This motivates our strategy to start

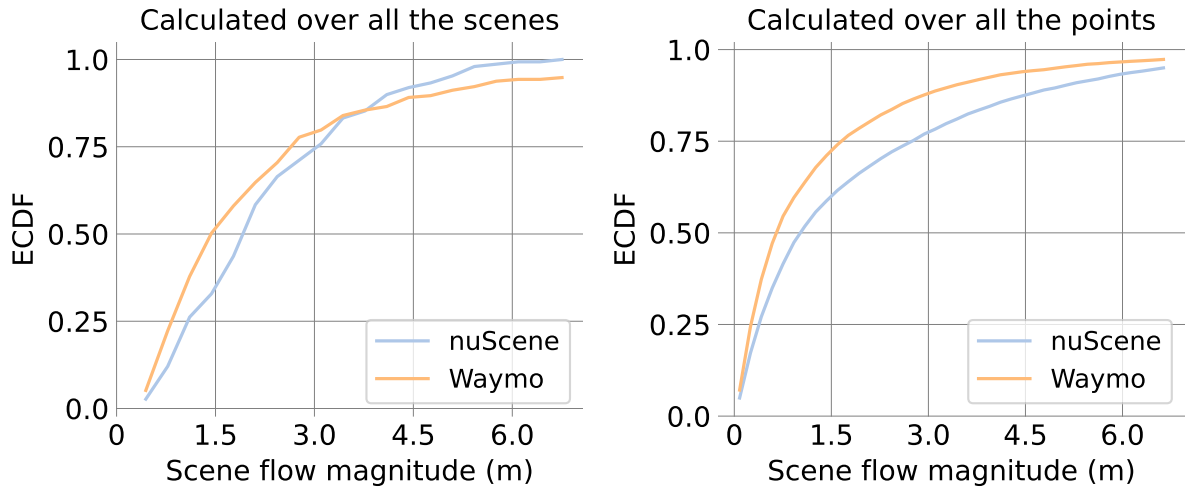

Figure 3.6.: Scene flow magnitudes of dynamic objects in the *Waymo* and *nuScenes* datasets.

		Static part				Dynamic foreground			
Dataset	Method	EPE avg.↓	AccR↑	AccS↑	ROutlier↓	EPE avg. ↓	AccR↑	AccS↑	ROutliers ↓
Waymo	PPWC-Net [63]	0.475 ± 0.543	35.0	14.2	13.5	0.658 ± 0.696	27.1	7.9	22.9
	FLOT [44]	0.381 ± 0.516	68.8	51.8	13.0	0.772 ± 0.711	30.1	11.2	31.9
	WsRSF [19]	1.415 ± 1.352	34.6	23.0	56.9	1.764 ± 1.744	21.0	8.6	61.6
	NSFPrior [34]	0.159 ± 0.231	87.1	73.5	4.3	0.355 ± 0.456	63.7	41.3	14.3
	Ours	**0.088** ± 0.237	**91.6**	**81.9**	**2.3**	**0.169** ± 0.259	**76.8**	**52.9**	**5.3**
nuScenes	PPWC-Net [63]	0.488 ± 0.402	34.2	12.7	17.5	0.784 ± 0.547	22.8	6.9	35.0
	FLOT [44]	0.597 ± 0.582	53.3	35.1	26.6	1.156 ± 0.714	13.2	3.7	56.5
	WsRSF [19]	0.658 ± 0.483	47.5	31.1	31.5	0.925 ± 0.627	29.8	15.0	42.2
	NSFPrior [34]	0.501 ± 0.344	57.8	37.7	21.3	0.743 ± 0.537	39.1	19.9	31.1
	Ours	**0.226** ± 0.206	**72.3**	**46.7**	**7.4**	**0.394** ± 0.26	**47.8**	**22.7**	**17.3**

Table 3.5.: Scene flow estimation results on *Waymo* and *nuScenes* datasets. Numbers are averaged over all test scenes.

with motion segmentation, so as to make explicit the large static component (including many objects that could move) whose scene flow is identical to the ego-motion.

In Fig. 3.6, we show the *ECDF* curve of scene flow magnitudes (L_2-norm of scene flow vectors) for the dynamic portions of the two datasets. The motions span a large range, but 75% of the flow vectors are of moderate magnitude <3 m. *nuScenes* has slightly larger overall flow magnitudes than *Waymo*, but *Waymo* contains more instances of large motions (Fig. 3.6 (left)).

3.6.5. Additional results

Results averaged over scenes.

In 3.1 we report evaluation metrics calculated over all the points in the test set. However, this does not fully reveal the difficulties encountered in different scenes. Here, we first calculate evaluation metrics per scene, then report the average over scenes in Tab. 3.5. For *EPE avg.*, we additionally report the standard deviations. We can see that for both static and dynamic parts,

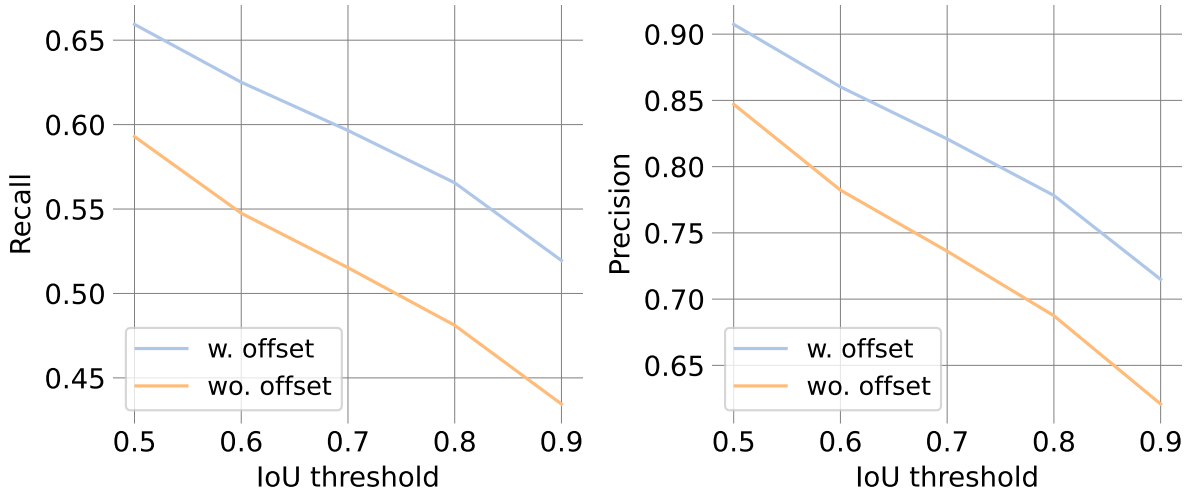

Figure 3.7.: Spatio-temporal instance association performance on *Waymo* with and without offset prediction.

		EPE avg.↓	EPE med.↓	AccS↑	AccR↑	ROutliers↓
Waymo	center	0.265	0.095	36.9	62.9	9.9
	center + ICP	0.212	0.047	61.2	80.0	7.7
	Ours	0.197	0.062	53.3	77.5	5.9
	Ours + ICP	**0.173**	**0.043**	**69.1**	**86.9**	**5.1**
nuScenes	center	0.553	0.258	13.5	32.7	28.2
	center + ICP	0.525	0.179	23.8	43.7	25.5
	Ours	0.301	0.146	26.6	53.4	**12.1**
	Ours + ICP	**0.301**	**0.135**	**32.7**	**56.7**	13.7

Table 3.6.: Comparison to centroid-based motion estimation baseline.

all methods have large standard deviations, which indicates varying difficulty of the scenes, as well as gross errors from challenging samples. Our model still achieves the smallest flow errors and standard deviations under this evaluation setting, for both datasets .

Spatio-temporal instance association. We plot instance association metrics at different thresholds in Fig. 3.7. As can be seen, offset prediction improves association recall and precision by >5%, across a range of thresholds. Such improvement becomes more significant as one increases the *IoU* threshold, reaching ≈10% at $IoU = 0.9$. We conclude that offset prediction is important to retain high-quality spatio-temporal instances, which can subsequently improve the accuracy of motion modelling for the dynamic parts (*AccS* increases by 9.2% in 3.4).

Dynamic object motion modelling.

We additionally compare the proposed TubeNet to two baseline methods. We naively align each frame $\mathbf{X}_k^t$ $(t > 1)$ of an instance $\mathbf{X}_k$ to frame $\mathbf{X}_k^0$ by translating the centroids,

and term this method *center*. For *center+ICP* we refine the simple translational alignment by a subsequent ICP. The detailed comparison is shown in Tab. 3.6. Our learned TubeNet achieves the best performance on both datasets. The improvement is larger on the challenging *nuScenes* data, where the point clouds are sparser and less complete, so centroids computed from partial observations are not an accurate proxy for the object location. Our learned TubeNet can implicitly exploit prior knowledge about object shape and surface-level correspondence, leading to more robust and accurate motion modelling.

3.6.6. Qualitative results

We show additional qualitative results in Fig. 3.8 and Fig. 3.9. Benefiting from the explicit *multi-body* assumption, our model achieves accurate scene flow estimation of both static parts (Fig. 3.8 (1) and Fig. 3.9 (2)) and dynamic parts (Fig. 3.8 (3) and Fig. 3.9(1)). Errors in the automatically generated pseudo-ground truth are shown in Fig. 3.9(3), in this case our model achieves more accurate flow estimation and reconstruction.

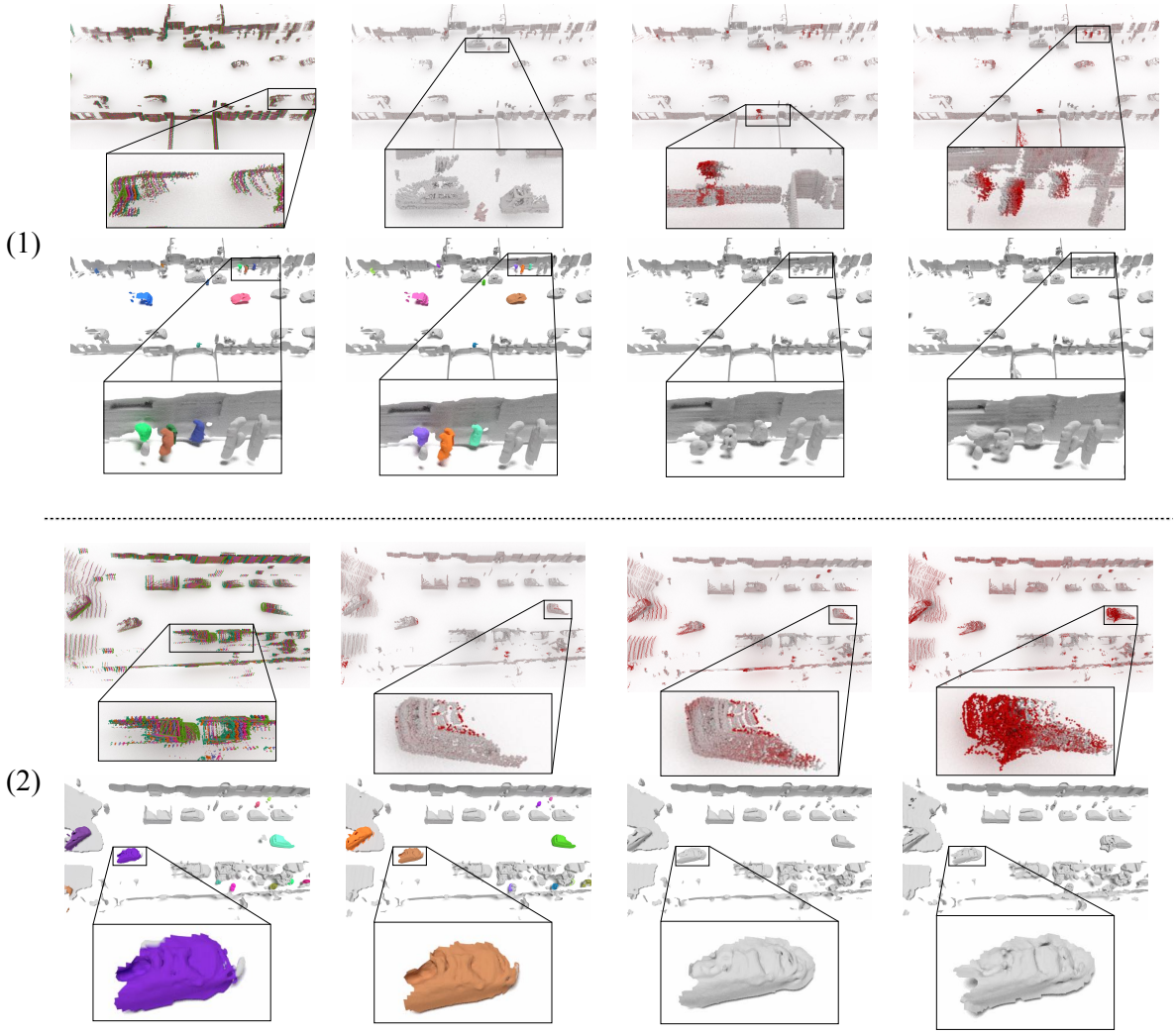

Figure 3.8.: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *Waymo* dataset.

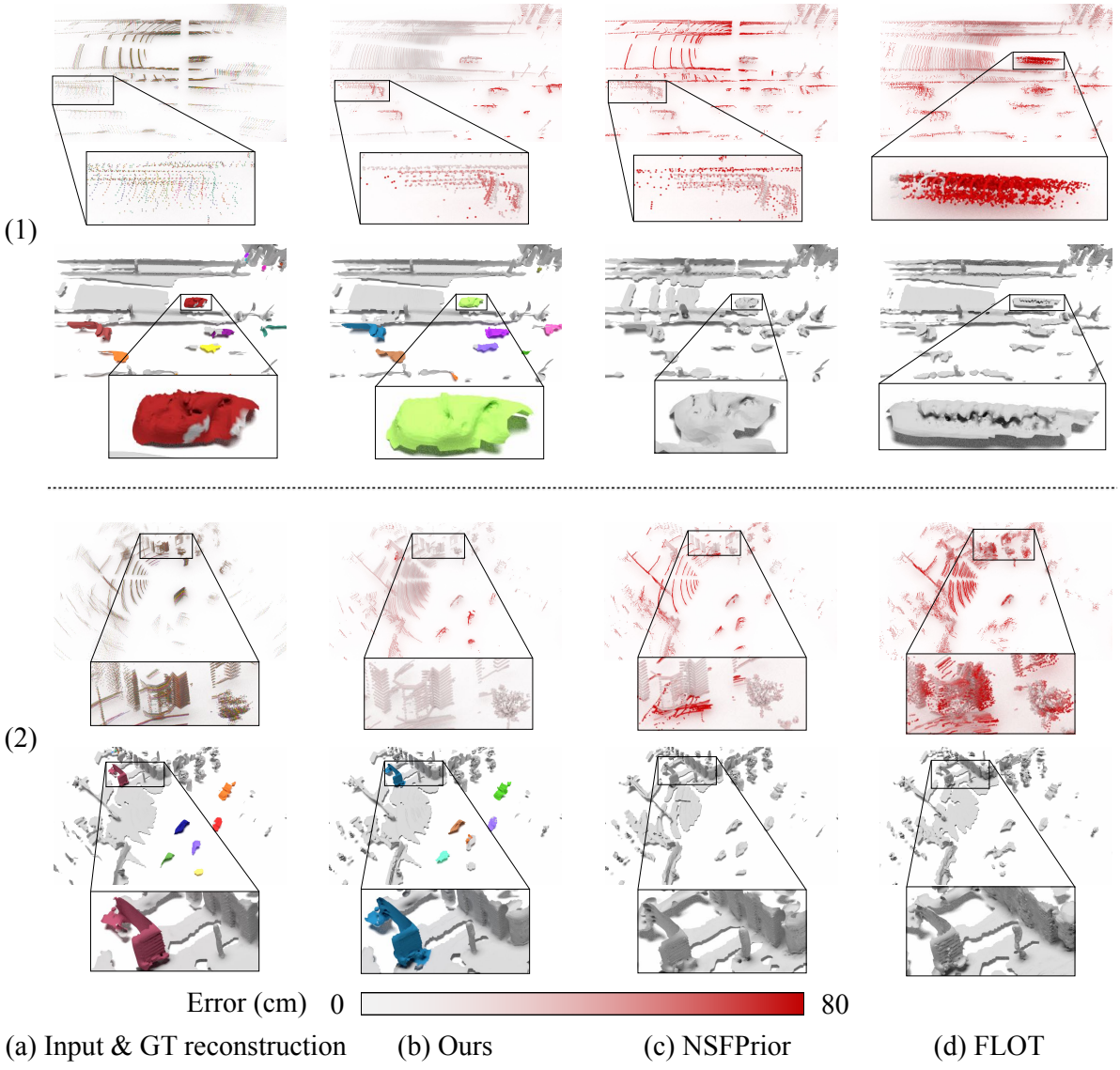

Figure 3.9.: Qualitative results showing scene flow estimation (top) and surface reconstruction (bottom) for three example scenes from the *nuScenes* dataset.

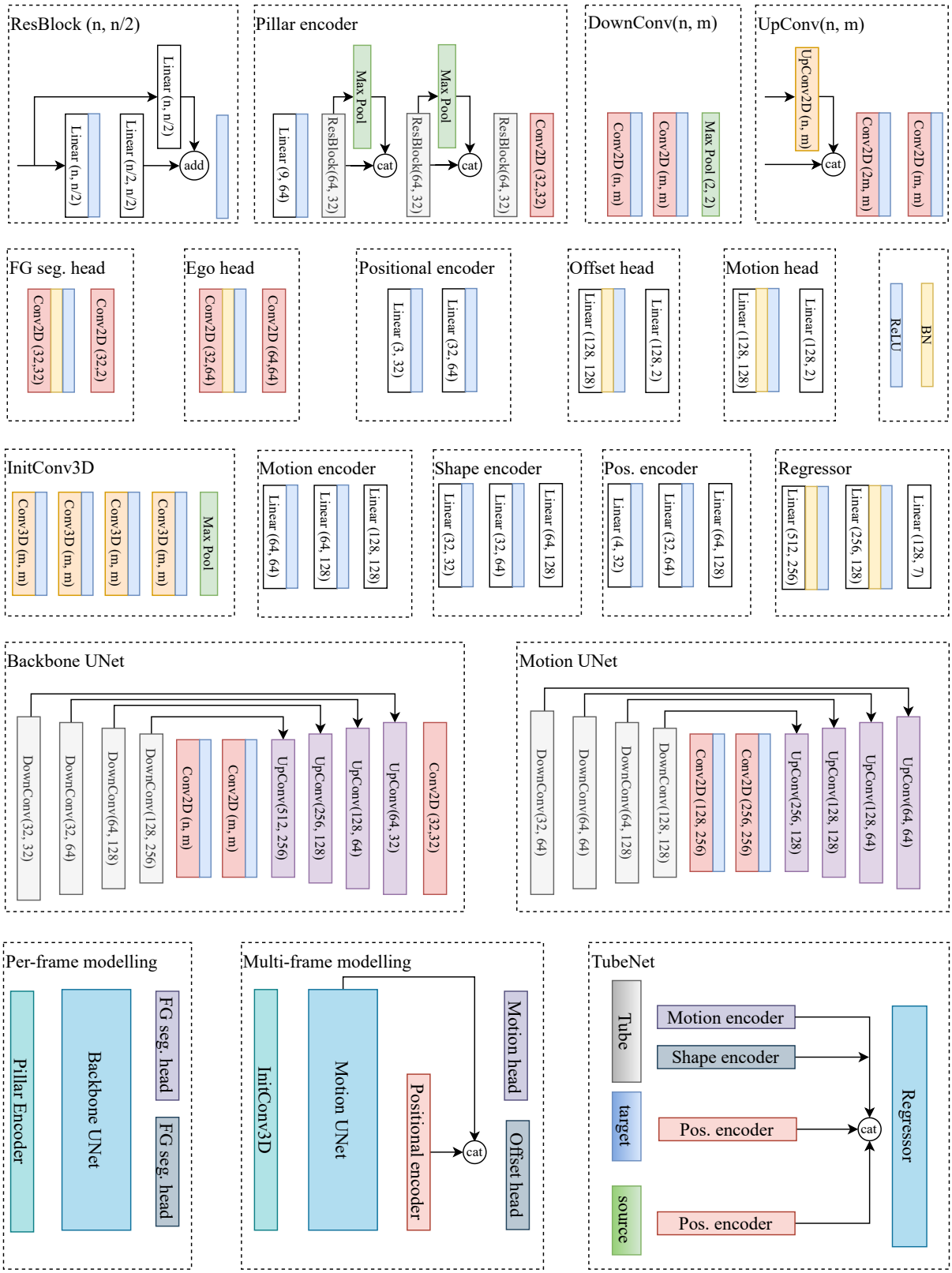

Figure 3.10.: Detailed network architecture. All convolutional layers have kernel size 3×3. (n, m) in Conv, UpConv, DownConv, and Linear layers denote the input and output feature dimensions.

4 Conclusions

placeholder

4.1. Lessons Learned

placeholder

4.2. Limitations

placeholder

4.3. Outlook

placeholder

A Bibliography

[1] Mehmet Aygun, Aljosa Osep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic LiDAR segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[2] Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Bjorn Ommer, and Andreas Geiger. SLIM: Self-supervised LiDAR scene flow and motion segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[3] Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. PointFlowNet: Learning representations for rigid motion estimation from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[4] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[5] Maxim Berman, Amal Rannen Triki, and Matthew B Blaschko. The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[6] 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.

[7] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] Chi Chen and Bisheng Yang. 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.

[9] 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 RA-L*, 2021.

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

[11] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NeurIPS)*, 2013.

[12] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. MOTchallenge: A benchmark for single-camera multiple target tracking. *IJCV*, 2021.

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

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

[15] Hehe Fan, Yi Yang, and Mohan Kankanhalli. Point 4d transformer networks for spatio-temporal modeling in point cloud videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[16] Hehe Fan, Xin Yu, Yuhang Ding, Yi Yang, and Mohan Kankanhalli. PSTNet: Point spatio-temporal convolution on point cloud sequences. In *International Conference on Learning Representations (ICLR)*, 2020.

[17] Artem Filatov, Andrey Rykov, and Viacheslav Murashkin. Any motion detector: Learning class-agnostic scene dynamics from a sequence of lidar point clouds. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[18] Silvio Giancola, Jesus Zarzar, and Bernard Ghanem. Leveraging shape completion for 3d siamese tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] Zan Gojcic, Or Litany, Andreas Wieser, Leonidas J Guibas, and Tolga Birdal. Weakly supervised learning of rigid 3d scene flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[20] Zan Gojcic, Caifa Zhou, Jan D Wegner, Leonidas J Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] Johannes Groß, Aljoša Ošep, and Bastian Leibe. AlignNet-3D: Fast point cloud registration of partially observed objects. In *IEEE International conference on 3D vision (3DV)*, 2019.

[22] Jiayuan Gu, Wei-Chiu Ma, Sivabalan Manivasagam, Wenyuan Zeng, Zihao Wang, Yuwen Xiong, Hao Su, and Raquel Urtasun. Weakly-supervised 3d shape completion in the wild. In *European Conference on Computer Vision (ECCV)*, 2020.

[23] Jiahui Huang, Tolga Birdal, Zan Gojcic, Leonidas J Guibas, and Shi-Min Hu. Multiway non-rigid point cloud registration via learned functional map synchronization. *IEEE T-PAMI*, 2022.

[24] Jiahui Huang, He Wang, Tolga Birdal, Minhyuk Sung, Federica Arrigoni, Shi-Min Hu, and Leonidas J Guibas. MultiBodySync: Multi-body segmentation and motion estimation via 3d scan synchronization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *IEEE International Conference on Machine Learning (ICML)*, 2015.

[26] Philipp Jund, Chris Sweeney, Nichola Abdo, Zhifeng Chen, and Jonathon Shlens. Scalable scene flow from point clouds in the real world. *arXiv preprint arXiv:2103.01306*, 2021.

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

[28] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Eurographics*, 2006.

[29] Giseop Kim and Ayoung Kim. Remove, then revert: Static point cloud map construction using multiresolution range images. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, 2020.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2014.

[31] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

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

[33] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: a point cloud upsampling adversarial network. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

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

[35] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D: Learning scene flow in 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteornet: Deep learning on dynamic 3d point cloud sequences. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[38] Chenxu Luo, Xiaodong Yang, and Alan Yuille. Self-supervised pillar motion learning for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[39] Bojun Ouyang and Dan Raviv. Occlusion guided self-supervised scene flow estimation on 3d point clouds. *arXiv preprint arXiv:2104.04724*, 2021.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[41] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[42] AJ Piergiovanni, Vincent Casser, Michael S Ryoo, and Anelia Angelova. 4d-net for learned multi-modal alignment. *arXiv preprint arXiv:2109.01066*, 2021.

[43] François Pomerleau, Philipp Krüsi, Francis Colas, Paul Furgale, and Roland Siegwart. Long-term 3d map maintenance in dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[44] Gilles Puy, Alexandre Boulch, and Renaud Marlet. FLOT: Scene flow on point clouds guided by optimal transport. In *European Conference on Computer Vision (ECCV)*, 2020.

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

[46] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3d object detection from point cloud sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[47] Davis Rempe, Tolga Birdal, Yongheng Zhao, Zan Gojcic, Srinath Sridhar, and Leonidas J. Guibas. CaSPR: Learning canonical spatiotemporal point cloud representations. 2020.

[48] Johannes Schauer and Andreas Nüchter. 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.

[49] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[50] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[51] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *MLSys*, volume 4, pages 302–315, 2022.

[52] Zachary Teed and Jia Deng. RAFT-3D: Scene flow using rigid-motion embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[54] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. In *IEEE International Conference on Computer Vision (ICCV)*, 1999.

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

[56] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

[57] 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.

[58] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object pose and size estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[59] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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

[61] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems (IROS)*, 2020.

[62] Pengxiang Wu, Siheng Chen, and Dimitris N Metaxas. MotionNet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[63] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision (ECCV)*, 2020.

[64] Bin Yang, Min Bai, Ming Liang, Wenyuan Zeng, and Raquel Urtasun. Auto4d: Learning to label 4d objects from sequential point clouds. *arXiv preprint arXiv:2101.06586*, 2021.

[65] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3D-MAN: 3d multi-frame attention network for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[66] Zi Jian Yew and Gim Hee Lee. RPM-Net: Robust point matching using learned features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[67] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. PCN: Point completion network. In *IEEE International conference on 3D vision (3DV)*, 2018.

[68] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*, 2018.

B List of Publications

1. **S. Huang**, M. Usvyatsov, K. Schindler. ”Indoor Scene Recognition in 3D.” IROS, 2020.
2. **S. Huang***, Z. Gojcic*, M. Usvyatsov, A. Wieser, K. Schindler. ”PREDATOR: Registration of 3D Point Clouds with Low Overlap.” CVPR, 2021. (Oral)
3. **S. Huang**, Z. Gojcic, J. Huang, A. Wieser, K. Schindler ”Dynamic 3D Scene Analysis by Point Cloud Accumulation.” ECCV, 2022.
4. **S. Huang**, Z. Gojcic, Z. Wang, F. William, Y. Kasten, S. Fidler, K. Schindler, O. Litany ”Neural LiDAR Fields for Novel View Synthesis.” ICCV, 2023.
5. H. Wu, X. Zuo, S. Leutenegger, Or Litany, K. Schindle, **S. Huang** ”Dynamic LiDAR Re-simulation using Compositional Neural Fields.” CVPR, 2024. (Hightlight)
6. L. Zhu, **S. Huang**, K. Schindler, I. Armeni. ”Living Scenes: Multi-object Relocalization and Reconstruction in Changing 3D Environments.” CVPR, 2024. (Hightlight)
7. B. Ke, A. Obukhov, **S. Huang**, N. Metzger, R. Daudt, K. Schindler. ”Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation.” CVPR, 2024. (Oral, Best Paper Award Candidate)
8. Y. Jia, L. Hoyer, **S. Huang**, T. Wang, L. Gool, K. Schindler, A. Obukhov. ”DGInStyle: DomainGeneralizable Semantic Segmentation with Image Diffusion Models and Stylized Semantic Control.” ECCV, 2024.
9. Z. Wang, T. Shen, J. Gao, **S. Huang**, J. Munkberg, J. Hasselgren, Z. Gojcic, W. Chen, S. Fidler ”Neural Fields meet Explicit Geometric Representations for Inverse Rendering of Urban Scenes.” CVPR, 2023.
10. L. Zhu, Y. Jia, **S. Huang**, N. Meyer, A. Wieser, K. Schindler, J. Aaron ”DeFlow: Self-supervised 3D Motion Estimation of Debris Flow.” CVPR Workshop, 2023. (Best Paper Award)
11. C. Stucker, B. Ke, Y. Yue, **S. Huang**, I. Armeni, K. Schindler ”ImpliCity: City Modeling from Satellite Images with Deep Implicit Occupancy Fields.” ISPRS Congress, 2022. (Best Young Author Award)
12. T. Sun, Y. Hao, **S. Huang**, S. Savarese, K. Schindler, M. Pollefeys, I. Armeni ”Nothing stands still: A spatiotemporal benchmark on 3d point cloud registration under large geometric and temporal change.” arxiv, 2023

C | Curriculum Vitae

Personal data

Name	**Shengyu Huang**
Date of birth	**17.11.1995**
Nationality	**China**

Education

Oct. 2020 - Nov. 2024	**ETH Zurich, Switzerland** PhD Student
Sep. 2018 - Aug. 2020	**ETH Zurich, Switzerland** M.Sc. in Geomatik
Sep. 2014 - Aug. 2018	**Tongji University, China** BSc. in Surveying and Mapping Engineering

Professional Experience

Oc. 2020 - Nov. 2024	**ETH Zurich, Switzerland** Research and Teaching Assistant
Jul 2023 - Dec 2023	**Google, Munich, Germany** Student Researcher
Apr 2022 - Dec 2022	**NVIDIA, Zurich, Switzerland** Research Scientist Intern